UNIVERSITY OF SALERNO

DEPARTMENT OF INFORMATION ENGINEERING AND ELECTRICAL AND APPLIED MATHEMATICS

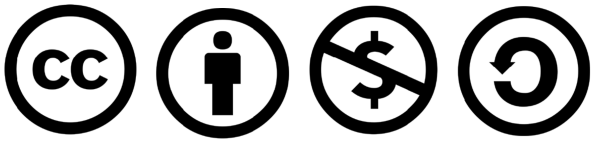

Master's Degree in Computer Engineering

Project report

# Parallel version of the Tarjan's and Kosaraju's algorithm to find strongly connected components in a graph

Group members:

| | | |
|---|---|---|
| Amato Emilio | 0622701903 | e.amato16@studenti.unisa.it |
| Bove Antonio | 0622701898 | a.bove57@studenti.unisa.it |
| De Gregorio Marco | 0622701865 | m.degregorio19@studenti.unisa.it |



ACADEMIC YEAR 2022/2023

# Indice


**1 Problem Description** **5**

1.1 Introduction to the problem and its solution . . . 5

1.2 Execution Example . . . 6

**2 Experimental Setup** **11**

2.1 Hardware . . . 11

2.1.1 CPU . . . 11

2.1.2 RAM . . . 11

2.2 Software . . . 11

**3 Performance, Speed-Up & Efficiency** **12**

3.1 OpenMP & MPI . . . 12

3.2 Graph Density . . . 13

3.3 Analisys . . . 13

3.4 Meaning of the graphs representing the analyzes carried out . . . 14

**4 Tarjan's Algorithm** **15**

4.1 Case Study 1 . . . 16

4.1.1 500 vertices - 25% of density . . . 16

4.1.2 500 vertices - 50% of density . . . 17

4.1.3 500 vertices - 75% of density . . . 18

4.2 Case Study 2 . . . 19

4.2.1 1000 vertices - 25% of density . . . 19

4.2.2 1000 vertices - 50% of density . . . 20

4.2.3 1000 vertices - 75% of density . . . 21

4.2.4 1000 vertices - Special Cases - 0% of density . . . 22

4.2.5 1000 vertices - Special Cases - 100% of density . . . 23

4.3 Case Study 3 . . . 24

4.3.1 2500 vertices - 25% of density . . . 24

4.3.2 2500 vertices - 50% of density . . . 25

4.3.3 2500 vertices - 75% of density . . . 26
4.4 Case Study 4 . . . 27
4.4.1 10000 vertices - 25% of density . . . 27
4.4.2 10000 vertices - 50% of density . . . 28
4.4.3 10000 vertices - 75% of density . . . 29
4.4.4 10000 vertices - Special Cases - 0% of density . . . 30
4.4.5 10000 vertices - Special Cases - 100% of density . . . 31

**5 Kosaraju's algorithm** **32**
5.1 Kosaraju and Tarjan algorithm . . . 32
5.2 Kosaraju's parallel version analysis . . . 32
5.3 Case Study 1 . . . 33
5.3.1 500 vertices - 25% of density . . . 33
5.3.2 500 vertices - 50% of density . . . 34
5.3.3 500 vertices - 75% of density . . . 35
5.4 Case Study 2 . . . 36
5.4.1 1000 vertices - 25% of density . . . 36
5.4.2 1000 vertices - 50% of density . . . 37
5.4.3 1000 vertices - 75% of density . . . 38
5.4.4 1000 vertices - Special Cases - 0% . . . 39
5.4.5 1000 vertices - Special Cases - 100% . . . 40
5.5 Case Study 3 . . . 41
5.5.1 2500 vertices - 25% of density . . . 41
5.5.2 2500 vertices - 50% of density . . . 42
5.5.3 2500 vertices - 75% of density . . . 43
5.6 Case Study 4 . . . 44
5.6.1 10000 vertices - 25% of density . . . 44
5.6.2 10000 vertices - 50% of density . . . 45
5.6.3 10000 vertices - 75% of density . . . 46
5.6.4 10000 vertices - Special Cases - 0% . . . 47

5.6.5 10000 vertices - Special Cases - 100% . . . 48

**6 Cluster** **49**

6.1 Tarjan . . . 50

6.1.1 500 vertices - 25% of density . . . 50

6.1.2 500 vertices - 50% of density . . . 51

6.1.3 500 vertices - 75% of density . . . 52

6.1.4 1000 vertices - 25% of density . . . 53

6.1.5 1000 vertices - 50% of density . . . 54

6.1.6 1000 vertices - 75% of density . . . 55

6.1.7 2500 vertices - 25% of density . . . 56

6.1.8 2500 vertices - 50% of density . . . 57

6.1.9 2500 vertices - 75% of density . . . 58

6.2 Kosaraju . . . 59

6.2.1 500 vertices - 25% of density . . . 59

6.2.2 500 vertices - 50% of density . . . 60

6.2.3 500 vertices - 75% of density . . . 61

6.2.4 1000 vertices - 25% of density . . . 62

6.2.5 1000 vertices - 50% of density . . . 63

6.2.6 1000 vertices - 75% of density . . . 64

6.2.7 2500 vertices - 25% of density . . . 65

6.2.8 2500 vertices - 50% of density . . . 66

6.2.9 2500 vertices - 75% of density . . . 67

**7 Final Considerations** **68**

7.1 Tarjan Implementation . . . 68

7.2 Kosaraju Implementation . . . 69

7.3 Cluster execution . . . 69

**8 Test Case** **69**

**9 Api Documentation** **71**

9.1 include/tarjan.h File Reference . . . . . 71

9.2 include/kosaraju.h File Reference . . . . . 72

9.3 include/Utils.h File Reference . . . . . 74

**10 How To Run** **79**

# 1 Problem Description

Provide a parallel version of the Tarjan's algorithm to find Strongly Connected Components in a graph. The implementation MUST use an hybrid message passing / shared memory paradigm and has to be implemented by using MPI and OpenMP. Students MUST provide parallel processes on different nodes, and each process has to be parallelized by using OpenMP (i.e.: MPI will spawn OPENMP-compiled processes). Students can choose the graph allocation method they prefer. They can eventually produce the graph directly in distributed memory (without store anything). Good graph dimensions are greater than 4GB of data.

*Optional*: use the same parallelization method to parallelize one of the other proposed algorithm in the Wikipedia page. Analyze differences.

## 1.1 Introduction to the problem and its solution

Tarjan's and Kosaraju's, the analayzed algorithm in our solution, are two graph analysis algorithms used to find strongly connected components in an oriented graph. Tarjan's algorithm was first proposed by Robert Tarjan in 1972 and has been used in a variety of applications, such as creating topological maps and analyzing social networks. Kosaraju's algorithm, proposed by Sharir and Kosaraju in 1978, uses a dfs technique similar to Tarjan's algorithm and, like the latter, finds strongly connected components in an oriented graph. Parallelization of Tarjan's and Kosaraju's algorithms is an important problem because both are sequential algorithms and thus can be slow for large graphs. Parallelization of the algorithms allows multiple processors to be used simultaneously to run the algorithm, thus improving performance. There are several ways to parallelize Tarjan and Kosaraju algorithms, including data-based parallelization and task-based parallelization. Data-based parallelization consists of executing the algorithm on different portions of the data in parallel, while task-based parallelization consists of executing different stages of the algorithm in parallel. The following project report will examine in detail the data-based parallelization approach of the Tarjan and Kosaraju algorithms and evaluate their performance in terms of speed and scalability. In particular, we will explore the data-driven approach using OpenMP and MPI. In addition, we will compare the results obtained with those of the original Tarjan and Kosaraju algorithms to evaluate the effectiveness of parallelization. In summary, the goal of this project is to find the best strategy for parallelization of the Tarjan and Kosaraju algorithms and to evaluate the performance of the different approaches through a series of tests on large graphs. We believe that the results obtained in this project can help to develop more efficient and scalable Tarjan and Kosaraju algorithms for the analysis of large-dimensional oriented graphs.

Following an analysis carried out in order to understand what might be a possible solution to the problem under examination, the following solution idea has been reached. Assuming the use of 4 MPI processes, the first step involves reading the entire initial graph by these. At this point, each process, based on its rank, will calculate its own competence interval that we will indicate with $[start; stop[$; in particular, a mechanism of graph division is used in order to distribute the load equally among all processes. Subsequently, each process will perform a specific algorithm for calculating the strongly connected components on its own portion of the graph, storing them in a dynamic array of arrays. At this point, a reduction process begins, starting from the 4 initial processes that will lead to having a single final process that will have the strongly connected components of the initial graph. In particular, a cycle will begin, whose exit condition is given by the fact that only one process remains, in which:

- if the process has an even rank and the process with $rank+1$ exists, then the latter will send it, in addition to the strongly connected components found, a series of information in order to allow the process with $rank+1$ to build a new graph on which to continue execution;
- if the process has an odd rank, on the other hand, it will receive the aforementioned information, store it in appropriate data structures, take care of creating a new graph that will replace the previous one and reapply the algorithm for calculating strongly connected components on this, replacing the previous array.

Subsequently, the even-rank processes that had sent their information, following a split of the communicator, are killed and there will be a new assignment of ranks to the remaining processes (in the case under examination, the processes with rank 0 and 2 after communicating with the processes with rank 1 and 3 are killed and the process with rank 1 is assigned rank 0 and the process with rank 3 is assigned rank 1). The cycle begins again until only one process remains, which will be the owner of the total strongly connected components.

## 1.2 Execution Example

Let's now provide an example of the execution of the previously proposed solution, illustrating all the steps. In particular, we consider having an MPI process number of 4 ($size = 4$) and the following graph made up of 10 nodes:

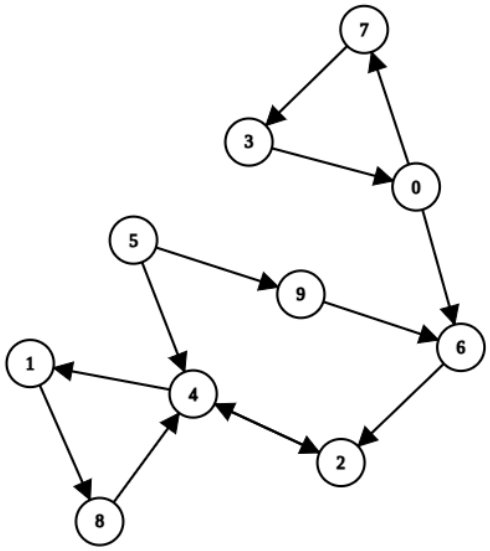
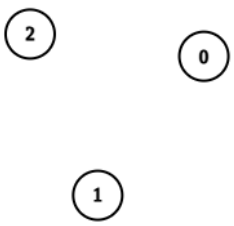



Figura 1: Initial graph

At this point, as previously stated, each process will calculate its own competence range $[start; stop[$ and will execute the algorithm to calculate the strongly connected components, storing them in a dynamic array of arrays that we will call *sccs*.

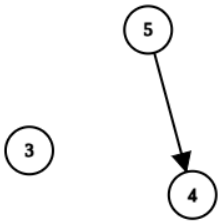



**Process 0**

$start \rightarrow 0$

$stop \rightarrow 3$

**sccs**: $\begin{bmatrix} [0] \rightarrow [0] \\ [1] \rightarrow [1] \\ [2] \rightarrow [2] \end{bmatrix}$




**Process 1**

$start \rightarrow 3$

$stop \rightarrow 6$

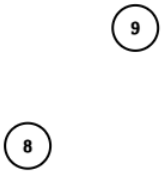
**sccs**: $\begin{bmatrix} [0] \rightarrow [3] \\ [1] \rightarrow [4] \\ [2] \rightarrow [5] \end{bmatrix}$


**Process 2**

$start \rightarrow 6$

$stop \rightarrow 8$

**sccs**: $\begin{bmatrix}[0] \rightarrow [6] \\ [1] \rightarrow [7]\end{bmatrix}$




**Process 3**

$start \rightarrow 8$

$stop \rightarrow 10$

**sccs**: $\begin{bmatrix}[0] \rightarrow [8] \\ [1] \rightarrow [9]\end{bmatrix}$

Subsequently, each process instantiates two hash tables:

- *sccsHT*: which has as key the name of a macronode and as value the array of nodes that belong to it;
- *auxiliaryGraphHT*: which has as key the name of a node and as value the macronode it belongs to.

At this point, the first iteration of the reduction process begins and thus $size = 4$ and $numIteration = 1$. In particular, process 0 sends to process 1 only the strongly connected components found (*sccs*), as initially the hash tables will be empty. Process 1, as a result, will concatenate its *sccs* array with the one received from process 0, populate its hash tables, and create a temporary array (*nodes*) in which it will insert the nodes of the new graph that needs to be constructed, in the following way:

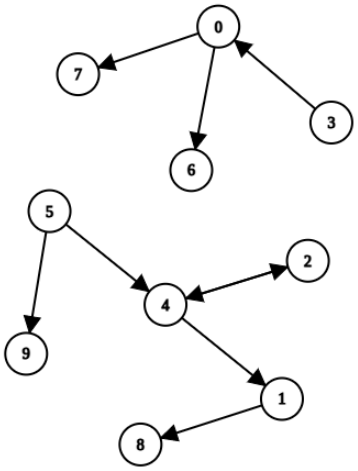
- for each element of the *sccs* array that represents a macronode, it is assigned a new name and added to the temporary array *nodes*; in addition, the hash tables *sccsHT* and *auxiliaryGraphHT* are also populated
- for all other elements, on the other hand, the node in question is only added to the temporary array *nodes*

Therefore, we will have:

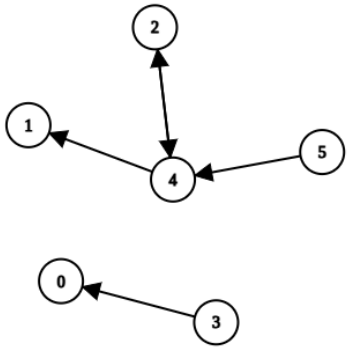
**Process 0 sends to process 1**

**sccs**: $\begin{bmatrix}[0] \rightarrow [0] \\ [1] \rightarrow [1] \\ [2] \rightarrow [2] \\ [3] \rightarrow [3] \\ [4] \rightarrow [4] \\ [5] \rightarrow [5]\end{bmatrix}$ **sccsHT**: *empty* **auxiliaryGraphHT**: *empty* **nodes**: [0 1 2 3 4 5]

At this point, based on the previous information, process 1 will construct a new graph, obtaining:




and, as a result, will calculate the strongly connected components this time considering *start* as the value 0 and *stop* as the number of nodes in the graph. So, again considering only the nodes that are part of the graph, process 1 will calculate the strongly connected components on the following graph:




overwriting *sccs* in the following way:

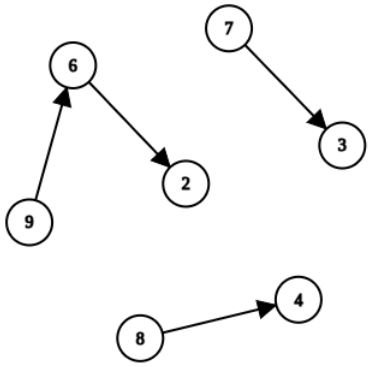
$$\textbf{sccs}: \begin{bmatrix} [0] \rightarrow & [0] \\ [1] \rightarrow & [3] \\ [2] \rightarrow & [1] \\ [3] \rightarrow & [5] \\ [4] \rightarrow & [2\ 4] \end{bmatrix}$$

What has been said for processes 0 and 1 also applies to processes 2 and 3. Therefore:

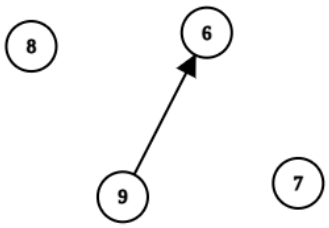
**Process 2 sends to process 3**

$$\textbf{sccs}: \begin{bmatrix} [0] \rightarrow [6] \\ [1] \rightarrow [7] \\ [2] \rightarrow [8] \\ [3] \rightarrow [9] \end{bmatrix} \quad \textbf{sccsHT}: \textit{empty} \quad \textbf{auxiliaryGraphHT}: \textit{empty} \quad \textbf{nodes}: [6\ 7\ 8\ 9]$$

At this point, based on the previous information, process 3 will build a new graph, obtaining:




and, as a result, it will calculate the strongly connected components, this time considering 0 as the value of *start* and the number of nodes in the graph as the value of *stop*. Therefore, taking into account only the nodes that are part of the graph, process 3 will calculate the strongly connected components on the following graph:




overwriting *sccs* in the following way:

$$\textbf{sccs}: \begin{bmatrix} [0] \rightarrow [7] \\ [1] \rightarrow [8] \\ [2] \rightarrow [9] \\ [3] \rightarrow [6] \end{bmatrix}$$

At this point, processes 0 and 2 are killed and processes 3 and 4, following a split operation of the initial communicator, respectively obtain ranks 0 and 1:

**P1** *becomes* **P0**
**P3** *becomes* **P1**

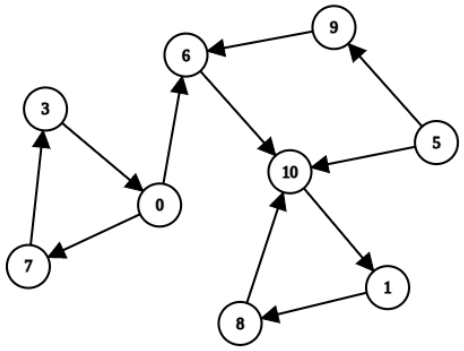
So, the first iteration ends and we have $size = 2$ and $numIteration = 2$. Since $size$ is still greater than 1, the previously described process is repeated again.

**Process 0 sends to process 1**

$$\textbf{sccs}: \begin{bmatrix} [0] \rightarrow & [0] \\ [1] \rightarrow & [3] \\ [2] \rightarrow & [1] \\ [3] \rightarrow & [5] \\ [4] \rightarrow & [2\ 4] \\ [5] \rightarrow & [7] \\ [6] \rightarrow & [8] \\ [7] \rightarrow & [9] \\ [8] \rightarrow & [6] \end{bmatrix} \quad \textbf{sccsHT}: 10 \rightarrow [2\ 4] \quad \textbf{auxiliaryGraphHT}: \begin{bmatrix} 2 \rightarrow 10 \\ 4 \rightarrow 10 \end{bmatrix} \quad \textbf{nodes}: [0\ 3\ 1\ 5\ 10\ 7\ 8\ 9\ 6]$$

We note that in this case, since there is an array of size greater than 1 in *sccs*, a macronode is created and assigned an ID (greater than the number of nodes in the graph minus one, to avoid having nodes with the same name), for example 10.

Starting from the previous information, process 1 will construct a new graph, obtaining:




and, as a result, will calculate the strongly connected components by again examining only the nodes that are part of the graph, overwriting *sccs* in the following way:

$$\textbf{sccs}: \begin{bmatrix} [0] \rightarrow & [3\ 0\ 7] \\ [1] \rightarrow & [10\ 1\ 8] \\ [2] \rightarrow & [5] \\ [3] \rightarrow & [6] \\ [4] \rightarrow & [9] \end{bmatrix}$$

At this point, process 0 is killed and process 1, following the split of the communicator again, obtains rank 0:

**P1** *becomes* **P0**

The second iteration ends this way and we have $size = 1$ and $numIteration = 3$. Since $size = 1$ the loop ends and the process with rank 0 (the last remaining one) has in *sccs* the strongly connected components. In particular, by expanding the macronodes through the $sccsHT$, we find that the final strongly connected components are 5:

$$\begin{bmatrix} SCC\ 1 \rightarrow & [3\ 0\ 7] \\ SCC\ 2 \rightarrow & [2\ 4\ 1\ 8] \\ SCC\ 3 \rightarrow & [5] \\ SCC\ 4 \rightarrow & [6] \\ SCC\ 5 \rightarrow & [9] \end{bmatrix}$$

# 2 Experimental Setup

## 2.1 Hardware

### 2.1.1 CPU

Model Name: AMD Ryzen 9 3900X
Socket: AMD Socket AM4
Foundry: TSMC
Process Size: 7 nm
Transistors: 7,600 million
Die Size: 2x 74 $mm^2$
I/O Process Size: 12 nm
I/O Die Size: 124 $mm^2$
Package: µOPGA-1331
tCaseMax: 95°C
Frequency: 3.8 GHz
Turbo Clock: up to 4.6 GHz
Base Clock: 100 MHz
Multiplier: 38.0x
Multiplier Unlocked: Yes
TDP: 105 W
FP32: 2,649.6 GFLOPS
Number of Cores: 12
Number of Threads: 24
SMP # CPUs: 1
Integrated Graphics: N/A
Cache L1: 64K (per core)
Cache L2: 512K (per core)
Cache L3: 64MB
Memory Support: DDR4-3200 MHz
Dual-channel
ECC Memory: No
PCI-Express: Gen 4, 24 Lanes (CPU only)
Generation: Ryzen 9 (Zen 2 (Matisse))

### 2.1.2 RAM

Memory Series: VENGEANCE LPX
SPD Latency:  CL16 (16-18-18-36)
SPD Speed: 2666MHz
SPD Voltage: 1.2V
Speed Rating: PC4-25600 (3200MHz)
Heat Spreader: Anodized Aluminum
Package Memory Format: DIMM senza buffer
Performance Profile: XMP 2.0
Package Memory Pin: 288
Densità: 16GB (2 x 8GB)
Tipo: DRAM kit di memoria
Tecnologia: DDR4 SDRAM
Data Integrity Check: Non ECC

## 2.2 Software

- Ubuntu 22.10
- GCC 11.3.0
- Visual Studio 1.74.2
- CMake 3.22.1
- Python 3.10.6

# 3 Performance, Speed-Up & Efficiency

In the field of high performance computing, the relationship between speed-up and efficiency is a topic of great importance. Speed-up is a measure of the increase in speed obtained by using multiple processors compared to a single processor, while efficiency is a measure of the amount of work actually performed by the processors compared to the maximum theoretical amount of work.
Speed-up is typically expressed as the ratio of the execution time on a single processor to the execution time on multiple processors:

$$S_n = \frac{T_1}{T_p(n)}$$

where $T_1$ is the time taken by the best known serial algorithm, while $T_p(n)$ is the execution time of the parallel algorithm on $n$ processors.
For example, if the execution time of a certain application is 100 seconds on a single processor and 50 seconds on 4 processors, the speed-up is 2. In theory, the use of multiple processors should allow for linear speed-up relative to the number of processors used. However, in practice this is not always possible due to issues of communication and synchronization between processors.

Efficiency, particularly relative efficiency, is given by the ratio of the time to execute an algorithm on a single processor to $n$ multiplied by the time to execute a parallel algorithm on $n$ processors:

$$E_r = \frac{T_1}{n \cdot T_n}$$

Using the previous example, relative efficiency would be $100/(4 \cdot 50)$ on 4 processors. In theory, efficiency should always be greater than or equal to 1, as more processors are used to perform the work. However, in practice there may be losses in efficiency due to issues of communication and synchronization between processors.

## 3.1 OpenMP & MPI

MPI, or Message Passing Interface, is a widely-used library for parallel computing on distributed memory systems. It provides a set of functions for inter-process communication, allowing multiple processes to work together and solve a problem faster by dividing the workload among them. MPI is a standard, meaning that it is not tied to any particular system or architecture. This makes it portable and allows it to be used on a wide range of computers, from small clusters to large supercomputers. One of the main advantages of MPI is that it allows for a high degree of flexibility in designing parallel algorithms. The library provides a variety of communication operations, including point-to-point communication (sending messages between two processes), collective communication (sending messages between all processes), and non-blocking communication (allowing processes to continue executing while waiting for a message to arrive). Despite its advantages, MPI also has some limitations. One limitation is that it does not provide any support for shared memory programming, which can be useful for certain types of algorithms.

On the other hand, OpenMP is a useful solution for parallelizing existing code that has been written to run on a single core. However, it is not always possible to gain benefits from using OpenMP. There are some reasons why OpenMP might not be suitable for a certain code or why significant benefits may not be gained from parallelization. One of the reasons why OpenMP might not be suitable for a certain code is that the code might already be optimized for execution on a single core. In this case, parallelizing the code may not lead to a significant increase in performance. Furthermore, OpenMP is mainly designed to parallelize code that is heavily dependent on loops, such as the code used to solve scientific computing problems. If the code is not heavily dependent on loops, it may not be possible to gain benefits from parallelization. Another reason why OpenMP might not be suitable for a certain code is that it might require a high level of communication between cores. If the code requires a high level of communication between cores, it might be necessary to use another parallel library, such as MPI, which is designed to handle a high level of communication between cores.

## 3.2 Graph Density

The density of a graph is a measure of how "populated" the graph is with edges compared to the maximum possible number of edges. In other words, it indicates how "complete" the graph is in relation to its size.
The density of a graph is calculated as the ratio of the number of edges present in the graph to the maximum number of edges possible for that graph. In the case of an undirected graph with n vertices, the maximum number of possible edges is:

$$\frac{n(n-1)}{2}$$

In the case of a directed graph with n vertices, the maximum number of possible edges is $n(n-1)$.
The density of a graph can vary from 0 to 1. A graph with density 0 is an empty graph, meaning a graph without edges. A graph with density 1 is a complete graph, meaning a graph in which every vertex is connected to all other vertices.
Graphs with densities greater than 0.5 are considered dense graphs. Graphs with densities less than 0.5 are considered sparse graphs.
The density of a graph can be used to determine the properties of the graph itself and to compare graphs of different sizes. For example, dense graphs are often more similar to each other than sparse graphs, regardless of their sizes. Furthermore, the density of a graph can be used to determine the complexity of algorithms that operate on that graph. For example, algorithms that explore a complete graph have a higher complexity than algorithms that explore a sparse graph.

## 3.3 Analisys

For the analysis of Performance, Speed-up and Efficiency, the measurements that have been conducted are based on the following criterion: given the sequential version of the assigned algorithm and given its parallel implementation, for all three levels of optimization each of the latter was tested, for each optimization level set, on three types of graph. Given a graph with a certain number of vertices, three possible scenarios were considered: one where the graph is sparse, one where it is moderately dense, and one where it is very dense. In addition, two special cases were also considered, one where the graph has a density of 0 and one where it has a density of 1.
For each graph tested in the different optimization levels set, the variability between the different versions of the parallelized algorithm was managed by spreading the number of MPI processes and the number of OpenMP threads with different combinations.
The comparison between the two versions was carried out, once the number of measurements for each version was fixed, on the following input graphs and, for each of it, on how many levels of density the graph has, represented by min number - max number of edges per vertex:

- graph with a number of vertices equal to 500:
    1. **low density 25%**: 0 - 250
    2. **medium density 50%**: 250 - 500
    3. **high density 75%**: 500 - 750
- graph with a number of vertices equal to 1000:
    1. **low density 25%**: 0 - 250
    2. **medium density 50%**: 250 - 500
    3. **high density 75%**: 500 - 750

- graph with a number of vertices equal to 10000:
  1. **low density 25%**: 0 - 2500
  2. **medium density 50%**: 2500 - 5000
  3. **high density 75%**: 5000 - 7500
- special case graph with a number of vertices equal to 1000
  1. **zero density 0%**: 0 - 0
  2. **maximum density 100%**: 999 - 999
- special case graph with a number of vertices equal to 10000
  1. **zero density 0%**: 0 - 0
  2. **maximum density 100%**: 9999 - 9999

For each density level of each graph on which the execution of the algorithm version has been defined, as regards the management of MPI processes and OpenMP threads, the latter has been structured by assigning the following resources to each of prefixed density level:

- 0 MPI process - No OpenMP threads (*Sequential version*)
- 1 MPI process - {1, 2, 4, 8, 16} OpenMP threads (*Parallel version*)
- 2 MPI processes - {1, 2, 4, 8, 16} OpenMP threads (*Parallel version*)
- 4 MPI processes - {1, 2, 4, 8, 16} OpenMP threads (*Parallel version*)
- 8 MPI processes - {1, 2, 4, 8, 16} OpenMP threads (*Parallel version*)

Therefore, from the considerations made, we can define the case studies analysed:

1. **Case Study 1**: the sequential program and the parallel program are compliled with a graph of 500 vertices.
2. **Case Study 2**: the sequential program and the parallel program are compiled with a graph of 1000 vertices.
3. **Case Study 3**: the sequential program and the parallel program are compiled with a graph of 2500 vertices.
4. **Case Study 4**: the sequential program and the parallel program are compiled with a graph of 10000 vertices.

## 3.4 Meaning of the graphs representing the analyzes carried out

$x$ and $y$ *axis* represents the number of MPI processes with which the evaluation of the speedup was computed.

In particular, the meaning of the colors are the following ones:

- **red** representes the evaulation of Speedup with 1 OpenMP thread.
- **green** representes the evaulation of Speedup with 2 OpenMP threads.
- **blue** representes the evaulation of Speedup with 4 OpenMP threads.
- **yellow** representes the evaulation of Speedup with 8 OpenMP threads.
- **cyan** representes the evaulation of Speedup with 16 OpenMP threads.

# 4 Tarjan's Algorithm

The Tarjan algorithm is a well-known algorithm used for finding strongly connected components in a graph. It was first proposed by Robert Tarjan in 1972 and since then it has become a widely used method in graph theory and computer science. The algorithm is based on depth-first search (DFS) and uses a shared stack to keep track of visited vertices. Now, we will discuss the Tarjan algorithm, including its working principle and time complexity.

The algorithm proceeds as follows:

1. initialize an empty stack and an array of flags to mark visited vertices
2. for each unvisited vertex in the graph, perform a DFS starting from that vertex
3. during the DFS, for each vertex v:
   - if the vertex has not been visited yet, mark it as visited and push it onto the stack
   - if the vertex has already been visited and it is still on the stack, then it is a vertex of a strongly connected component. In this case, keep removing vertices from the top of the stack until v is encountered. The removed vertices form a strongly connected component

The Tarjan algorithm runs in time $O(V + E)$ where V is the number of vertices and E the number of edges in the graph. The time complexity of the algorithm is determined by the number of vertices and edges in the graph, and it is considered very efficient. Furthermore, it's widely used in different applications like compilers, operating systems, and databases, and it's an algorithm that still being used and studied today.

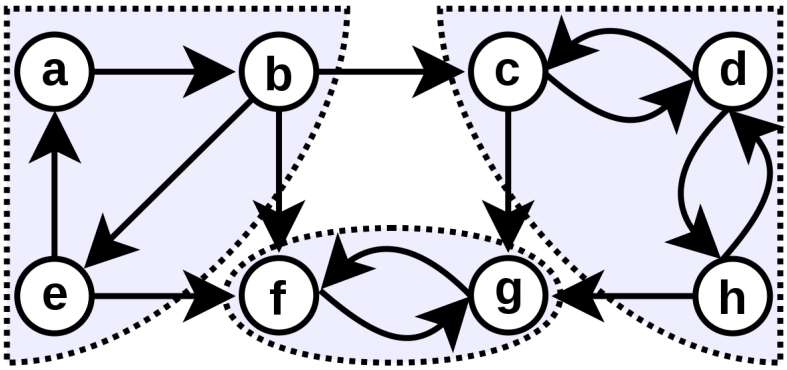
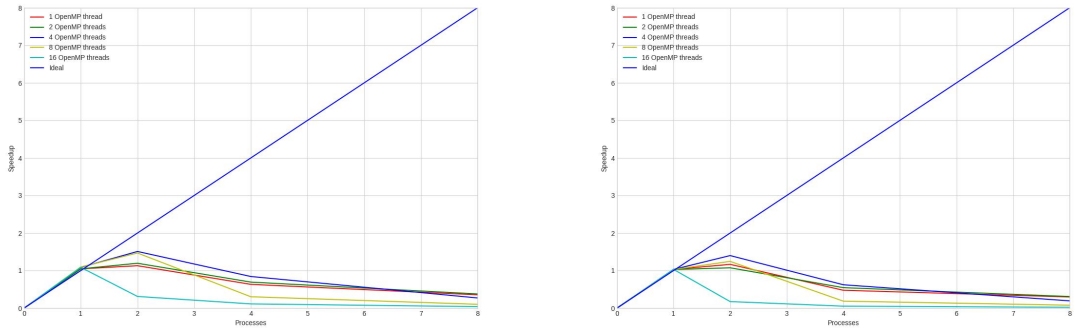



Figura 2: Strongly connected components on a graph

We will then show the results obtained by implementing the solution described in paragraphs 1.1 and 1.2 and using Tarjan as the algorithm for calculating the strongly connected components.

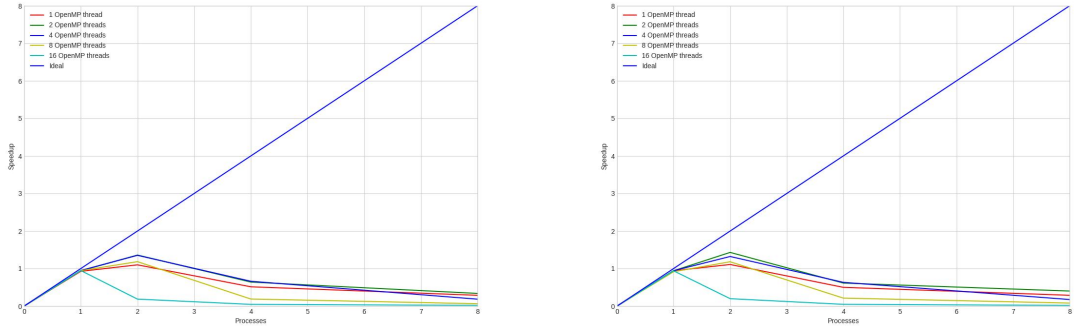
## 4.1 Case Study 1

### 4.1.1 500 vertices - 25% of density




(a) (b)

Figura 3: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



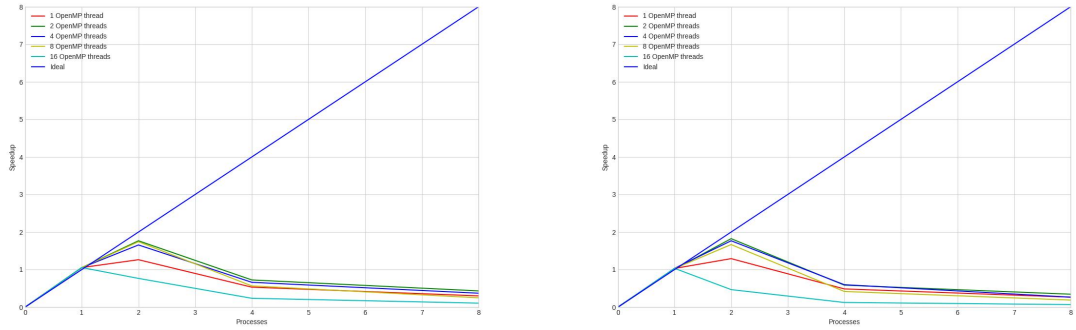

(a) (b)

Figura 4: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.002801 | 0.0 | 0.002801 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0026780000000000003 | 0.0 | 0.0026780000000000003 | 1.0459297983569829 | 1.0459297983569829 |
| **Parallel** | 1 | 2 | 0.0026780000000000003 | 0.0 | 0.0026780000000000003 | 1.0459297983569829 | 1.0459297983569829 |
| **Parallel** | 1 | 4 | 0.002582 | 0.0 | 0.002582 | 1.084817970565453 | 1.084817970565453 |
| **Parallel** | 1 | 8 | 0.0025568 | 0.0 | 0.0025568 | 1.0955100125156445 | 1.0955100125156445 |
| **Parallel** | 1 | 16 | 0.002582 | 0.0 | 0.002582 | 1.084817970565453 | 1.084817970565453 |
| **Parallel** | 2 | 1 | 0.000924 | 0.0015563999999999999 | 0.0024804000000000002 | 1.1292533462344783 | 0.5646266731172391 |
| **Parallel** | 2 | 2 | 0.0008238000000000001 | 0.0015204 | 0.0023444 | 1.1947619860092136 | 0.5973809930046068 |
| **Parallel** | 2 | 4 | 0.0009254 | 0.0009310000000000001 | 0.0018564 | 1.50883430295195 | 0.754417151475975 |
| **Parallel** | 2 | 8 | 0.000822 | 0.0010866 | 0.0019088 | 1.4674140821458508 | 0.7337070410729254 |
| **Parallel** | 2 | 16 | 0.0008265999999999999 | 0.0081986 | 0.0090248 | 0.3103669887421328 | 0.1551834943710664 |
| **Parallel** | 4 | 1 | 0.0004456 | 0.003982 | 0.004428 | 0.63256549232159 | 0.1581413730803975 |
| **Parallel** | 4 | 2 | 0.0005086 | 0.0035364 | 0.0040447999999999994 | 0.6924940664556963 | 0.17312351661392408 |
| **Parallel** | 4 | 4 | 0.0005088 | 0.0028187999999999998 | 0.0033276 | 0.84174780622671 | 0.2104369515566775 |
| **Parallel** | 4 | 8 | 0.0005154 | 0.0087472 | 0.0092626 | 0.3023988944788721 | 0.07559972361971802 |
| **Parallel** | 4 | 16 | 0.0004827999999999997 | 0.024621999999999998 | 0.0251048 | 0.11157228896466015 | 0.027893072241165037 |
| **Parallel** | 8 | 1 | 0.00039900000000000005 | 0.0075142 | 0.0079134 | 0.35395657997826474 | 0.04424457249728309 |
| **Parallel** | 8 | 2 | 0.0004672000000000001 | 0.006951400000000001 | 0.007418800000000001 | 0.37755432145360435 | 0.04719429018170054 |
| **Parallel** | 8 | 4 | 0.00045240000000000005 | 0.009898 | 0.010350400000000001 | 0.2706175606739836 | 0.03382719508424795 |
| **Parallel** | 8 | 8 | 0.0004408 | 0.0272402 | 0.027681 | 0.10118854087641342 | 0.012648567609551678 |
| **Parallel** | 8 | 16 | 0.0004058 | 0.071011 | 0.07141700000000001 | 0.03922035369729896 | 0.00490254421216237 |

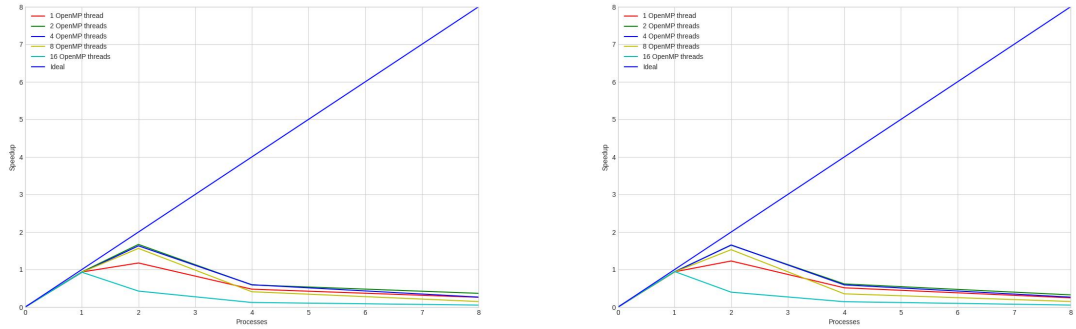
Tabella 1: Measurement with O0 optimization

### 4.1.2 500 vertices - 50% of density




(a) (b)

Figura 5: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



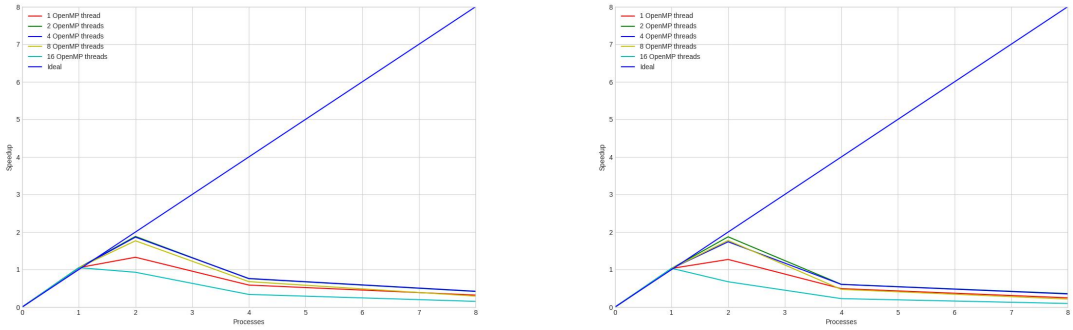

(a) (b)

Figura 6: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.007753 | 0.0 | 0.007753 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0073644 | 0.0 | 0.0073644 | 1.0527673673347455 | 1.0527673673347455 |
| **Parallel** | 1 | 2 | 0.0073706 | 0.0 | 0.0073706 | 1.051881800667517 | 1.051881800667517 |
| **Parallel** | 1 | 4 | 0.007377399999999999 | 0.0 | 0.007377399999999999 | 1.0509122455065472 | 1.0509122455065472 |
| **Parallel** | 1 | 8 | 0.0073690000000000005 | 0.0 | 0.0073690000000000005 | 1.0521101913421087 | 1.0521101913421087 |
| **Parallel** | 1 | 16 | 0.0073606 | 0.0 | 0.0073606 | 1.0533108713963535 | 1.0533108713963535 |
| **Parallel** | 2 | 1 | 0.0023594 | 0.0037822000000000003 | 0.0061414 | 1.2624157358257075 | 0.6312078679128538 |
| **Parallel** | 2 | 2 | 0.0023593999999999998 | 0.0020282 | 0.0043874 | 1.7671058029812647 | 0.8835529014906324 |
| **Parallel** | 2 | 4 | 0.0025060000000000004 | 0.002177 | 0.0046832 | 1.655491971301674 | 0.827745985650837 |
| **Parallel** | 2 | 8 | 0.0022159999999999997 | 0.002233 | 0.004449 | 1.7426387952348843 | 0.8713193976174421 |
| **Parallel** | 2 | 16 | 0.0023628000000000004 | 0.007773999999999999 | 0.010140199999999999 | 0.7645805802646892 | 0.3822902901323446 |
| **Parallel** | 4 | 1 | 0.0012334000000000002 | 0.013376599999999999 | 0.014609800000000001 | 0.530671193308601 | 0.13266779832715025 |
| **Parallel** | 4 | 2 | 0.0010719999999999998 | 0.009644 | 0.0107164 | 0.7234705684744878 | 0.18086764211862194 |
| **Parallel** | 4 | 4 | 0.0013166 | 0.0104204 | 0.011737000000000001 | 0.6605606202607139 | 0.16514015506517848 |
| **Parallel** | 4 | 8 | 0.0011576 | 0.0125832 | 0.013740800000000001 | 0.5642320680018631 | 0.14105801700046577 |
| **Parallel** | 4 | 16 | 0.0013014 | 0.0317456 | 0.033047 | 0.23460525917632463 | 0.05865131479408116 |
| **Parallel** | 8 | 1 | 0.0007768 | 0.0251106 | 0.0258878 | 0.29948469935645367 | 0.03743558741955671 |
| **Parallel** | 8 | 2 | 0.0006975999999999999 | 0.0172658 | 0.0179634 | 0.4315998084995045 | 0.053949976062438065 |
| **Parallel** | 8 | 4 | 0.0006960000000000001 | 0.020251799999999997 | 0.0209478 | 0.3701104650607701 | 0.046263808132596264 |
| **Parallel** | 8 | 8 | 0.0007078 | 0.030530800000000004 | 0.0312388 | 0.24818494948589576 | 0.03102311868573697 |
| **Parallel** | 8 | 16 | 0.0009471999999999999 | 0.07242180000000001 | 0.07336899999999999 | 0.1056713325791547 | 0.013208916572394338 |

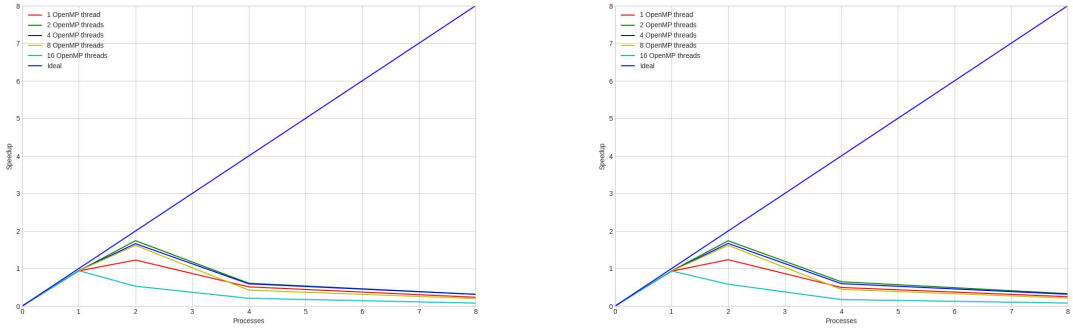
Tabella 2: Measurement with O0 optimization

### 4.1.3 500 vertices - 75% of density




(a) (b)

Figura 7: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization




(a) (b)

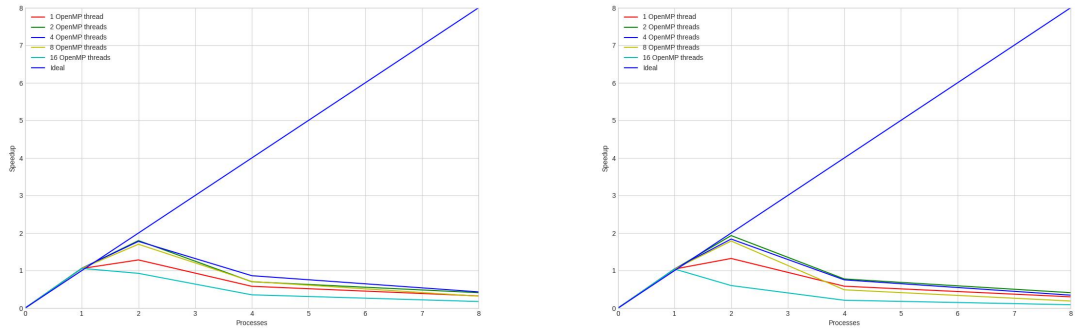
Figura 8: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.012997 | 0.0 | 0.012997 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0123714 | 0.0 | 0.0123714 | 1.050568246116042 | 1.050568246116042 |
| **Parallel** | 1 | 2 | 0.012401599999999999 | 0.0 | 0.012401599999999999 | 1.0480099342020386 | 1.0480099342020386 |
| **Parallel** | 1 | 4 | 0.0124882 | 0.0 | 0.0124882 | 1.0407424608830735 | 1.0407424608830735 |
| **Parallel** | 1 | 8 | 0.0123866 | 0.0 | 0.0123866 | 1.0492790596289538 | 1.0492790596289538 |
| **Parallel** | 1 | 16 | 0.012365600000000001 | 0.0 | 0.012365600000000001 | 1.0510610079575595 | 1.0510610079575595 |
| **Parallel** | 2 | 1 | 0.0037416 | 0.006045 | 0.0097866 | 1.3280403817464697 | 0.6640201908732348 |
| **Parallel** | 2 | 2 | 0.0037504 | 0.0031518000000000006 | 0.006902 | 1.8830773688785858 | 0.9415386844392929 |
| **Parallel** | 2 | 4 | 0.003766 | 0.0032080000000000003 | 0.0069742 | 1.863582919904792 | 0.931791459952396 |
| **Parallel** | 2 | 8 | 0.0037917999999999993 | 0.003565 | 0.007357 | 1.7666168275112137 | 0.8833084137556069 |
| **Parallel** | 2 | 16 | 0.0037421999999999998 | 0.010264199999999998 | 0.0140062 | 0.9279461952563864 | 0.4639730976281932 |
| **Parallel** | 4 | 1 | 0.0017048 | 0.020432200000000005 | 0.022137399999999995 | 0.5871059835391691 | 0.14677649588479227 |
| **Parallel** | 4 | 2 | 0.0018276 | 0.015358599999999997 | 0.0171864 | 0.7562374901084578 | 0.18905937252711444 |
| **Parallel** | 4 | 4 | 0.0015803999999999998 | 0.0154654 | 0.0170458 | 0.7624752138356663 | 0.19061880345891657 |
| **Parallel** | 4 | 8 | 0.0018470000000000001 | 0.0173048 | 0.019151599999999998 | 0.678637816161574 | 0.1696594540403935 |
| **Parallel** | 4 | 16 | 0.0017576000000000002 | 0.036733400000000006 | 0.0384908 | 0.33766510438858116 | 0.08441627609714529 |
| **Parallel** | 8 | 1 | 0.0010358000000000001 | 0.0395396 | 0.0405754 | 0.3203172365521967 | 0.040039654569024585 |
| **Parallel** | 8 | 2 | 0.0012642 | 0.0299956 | 0.031259800000000004 | 0.4157736133948393 | 0.051971701674354914 |
| **Parallel** | 8 | 4 | 0.0011126 | 0.0296396 | 0.030752400000000003 | 0.4226336806233009 | 0.052829210077912615 |
| **Parallel** | 8 | 8 | 0.0012716000000000001 | 0.043118000000000004 | 0.0443894 | 0.2927951267644978 | 0.03659939084556223 |
| **Parallel** | 8 | 16 | 0.0012720000000000001 | 0.0836908 | 0.0849626 | 0.15297319055678615 | 0.01912164881959827 |

Tabella 3: Measurement with O0 optimization

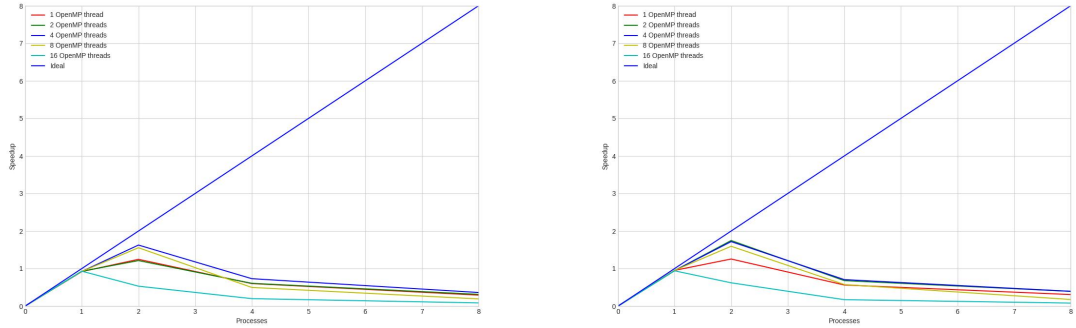
## 4.2 Case Study 2

### 4.2.1 1000 vertices - 25% of density




(a) (b)

Figura 9: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



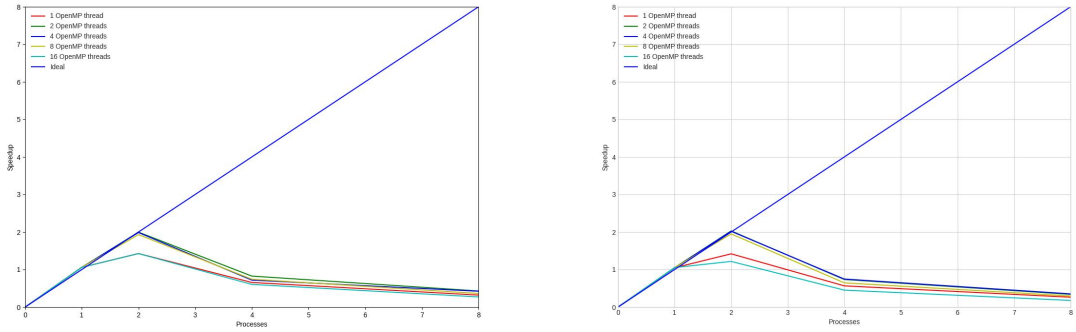

(a) (b)

Figura 10: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.0062416 | 0.0 | 0.0062416 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.005996400000000001 | 0.0 | 0.005996400000000001 | 1.0408912013874991 | 1.0408912013874991 |
| **Parallel** | 1 | 2 | 0.0060056 | 0.0 | 0.0060056 | 1.0392966564539763 | 1.0392966564539763 |
| **Parallel** | 1 | 4 | 0.0059784 | 0.0 | 0.0059784 | 1.0440251572327044 | 1.0440251572327044 |
| **Parallel** | 1 | 8 | 0.006006 | 0.0 | 0.006006 | 1.0392274392274392 | 1.0392274392274392 |
| **Parallel** | 1 | 16 | 0.0059976000000000005 | 0.0 | 0.0059976000000000005 | 1.0406829398426036 | 1.0406829398426036 |
| **Parallel** | 2 | 1 | 0.0017104000000000002 | 0.0030099999999999997 | 0.0047204 | 1.3222608253537838 | 0.6611304126768919 |
| **Parallel** | 2 | 2 | 0.001604 | 0.0016262 | 0.0032300000000000002 | 1.9323839009287924 | 0.9661919504643962 |
| **Parallel** | 2 | 4 | 0.001678 | 0.0017142000000000001 | 0.0033922 | 1.839985849890926 | 0.919992924945463 |
| **Parallel** | 2 | 8 | 0.0016158 | 0.0018766000000000002 | 0.0034926 | 1.787092710301781 | 0.8935463551508905 |
| **Parallel** | 2 | 16 | 0.0016847999999999998 | 0.008705599999999999 | 0.010390199999999999 | 0.600719909145156 | 0.300359954572578 |
| **Parallel** | 4 | 1 | 0.0006928 | 0.010013000000000001 | 0.010706 | 0.583000186811134 | 0.1457500467027835 |
| **Parallel** | 4 | 2 | 0.0007056 | 0.0073324 | 0.008038 | 0.7765115700422991 | 0.19412789251057477 |
| **Parallel** | 4 | 4 | 0.0007256000000000001 | 0.007575 | 0.0083004 | 0.751963760782613 | 0.18799094019565324 |
| **Parallel** | 4 | 8 | 0.000841 | 0.011969400000000002 | 0.012810199999999999 | 0.4872367332282088 | 0.1218091833070522 |
| **Parallel** | 4 | 16 | 0.000759 | 0.029092600000000003 | 0.0298518 | 0.20908621925646023 | 0.05227155481411506 |
| **Parallel** | 8 | 1 | 0.000602 | 0.020151400000000003 | 0.020753599999999997 | 0.3007478220646057 | 0.03759347775807571 |
| **Parallel** | 8 | 2 | 0.0006404 | 0.0145428 | 0.015183199999999999 | 0.41108593708836083 | 0.051385742136045104 |
| **Parallel** | 8 | 4 | 0.0006164 | 0.0174754 | 0.0180918 | 0.3449960755701478 | 0.04312450944626847 |
| **Parallel** | 8 | 8 | 0.0005352 | 0.0325566 | 0.033091999999999996 | 0.1886135621902575 | 0.023576695273782186 |
| **Parallel** | 8 | 16 | 0.0005718 | 0.06932419999999999 | 0.069896 | 0.08929838617374385 | 0.01116229827171798 |

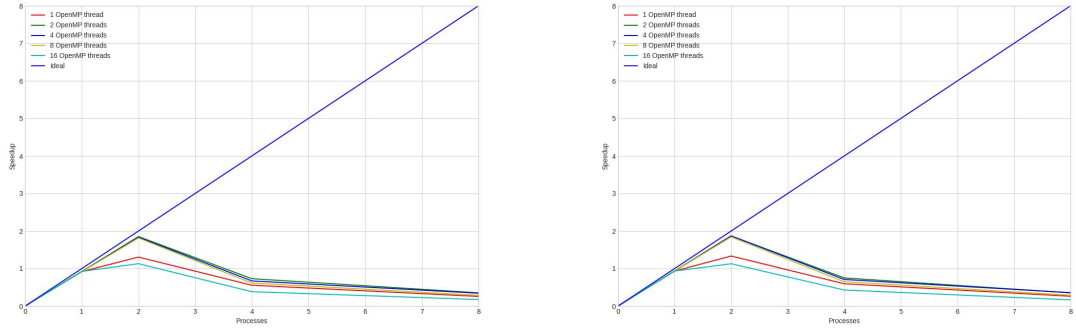
Tabella 4: Measurement with O1 optimization

### 4.2.2 1000 vertices - 50% of density




(a) (b)

Figura 11: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



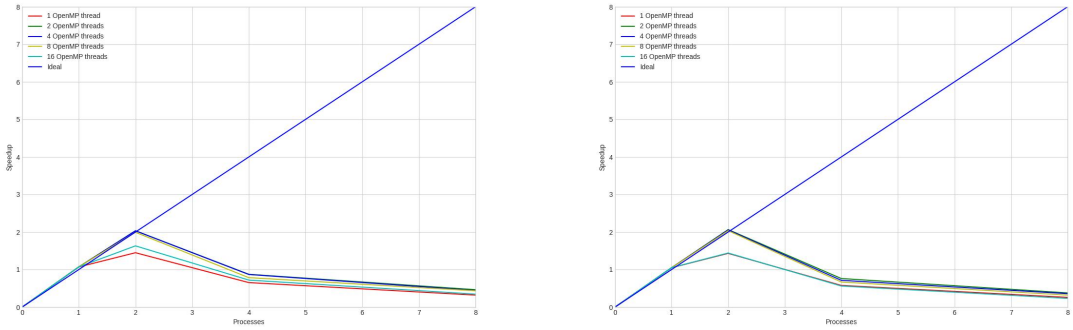

(a) (b)

Figura 12: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.0182018 | 0.0 | 0.0182018 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0173662 | 0.0 | 0.0173662 | 1.0481164561043868 | 1.0481164561043868 |
| **Parallel** | 1 | 2 | 0.017338799999999998 | 0.0 | 0.017338799999999998 | 1.049772763974439 | 1.049772763974439 |
| **Parallel** | 1 | 4 | 0.0173888 | 0.0 | 0.0173888 | 1.0467542326094958 | 1.0467542326094958 |
| **Parallel** | 1 | 8 | 0.0173158 | 0.0 | 0.0173158 | 1.051167142147634 | 1.051167142147634 |
| **Parallel** | 1 | 16 | 0.01728 | 0.0 | 0.01728 | 1.0533449074074075 | 1.0533449074074075 |
| **Parallel** | 2 | 1 | 0.0047852 | 0.008040799999999999 | 0.012826 | 1.4191330110712614 | 0.7095665055356307 |
| **Parallel** | 2 | 2 | 0.0047798 | 0.004186199999999999 | 0.0089664 | 2.030000892219843 | 1.0150004461099216 |
| **Parallel** | 2 | 4 | 0.0047786 | 0.0042382 | 0.009017 | 2.018609293556615 | 1.0093046467783076 |
| **Parallel** | 2 | 8 | 0.004892199999999999 | 0.0044402 | 0.009332199999999999 | 1.9504296950343973 | 0.9752148475171987 |
| **Parallel** | 2 | 16 | 0.0048162000000000005 | 0.010138600000000001 | 0.014954600000000002 | 1.2171372019311784 | 0.6085686009655892 |
| **Parallel** | 4 | 1 | 0.0019213999999999998 | 0.030237000000000003 | 0.0321578 | 0.5660150880968225 | 0.14150377202420564 |
| **Parallel** | 4 | 2 | 0.0017691999999999999 | 0.0230006 | 0.024769599999999996 | 0.7348443253019832 | 0.1837110813254958 |
| **Parallel** | 4 | 4 | 0.0018144 | 0.0225084 | 0.024323 | 0.7483369650125395 | 0.1870842412531349 |
| **Parallel** | 4 | 8 | 0.002053 | 0.0261378 | 0.0281908 | 0.645664543042411 | 0.16141613576060276 |
| **Parallel** | 4 | 16 | 0.002034 | 0.03825440000000001 | 0.0402888 | 0.4517831258314966 | 0.11294578145787415 |
| **Parallel** | 8 | 1 | 0.0010294000000000002 | 0.0680304 | 0.0690596 | 0.2635665425226906 | 0.03294581781533632 |
| **Parallel** | 8 | 2 | 0.0011766 | 0.0524874 | 0.0536638 | 0.3391820929565182 | 0.042397761619564774 |
| **Parallel** | 8 | 4 | 0.0010342 | 0.0506008 | 0.051635 | 0.3525089571027404 | 0.04406361963784255 |
| **Parallel** | 8 | 8 | 0.0010764 | 0.06043860000000001 | 0.061515 | 0.2958920588474356 | 0.03698650735592945 |
| **Parallel** | 8 | 16 | 0.0011661999999999998 | 0.1013552 | 0.10252159999999999 | 0.17754112304138836 | 0.022192640380173545 |

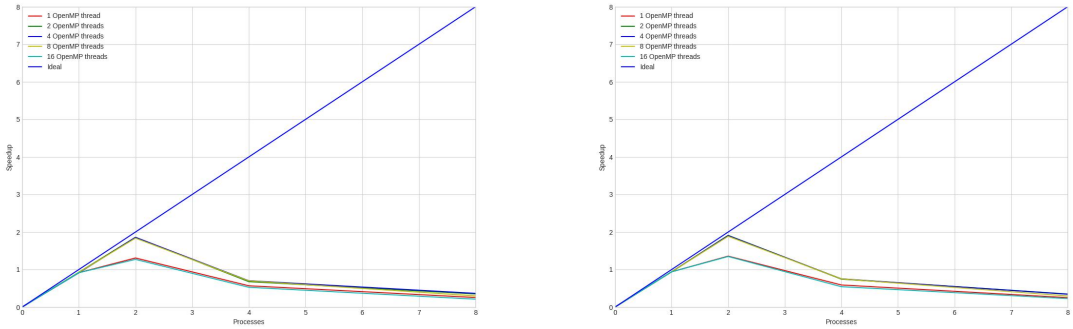
Tabella 5: Measurement with O1 optimization

### 4.2.3 1000 vertices - 75% of density




(a) (b)

Figura 13: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



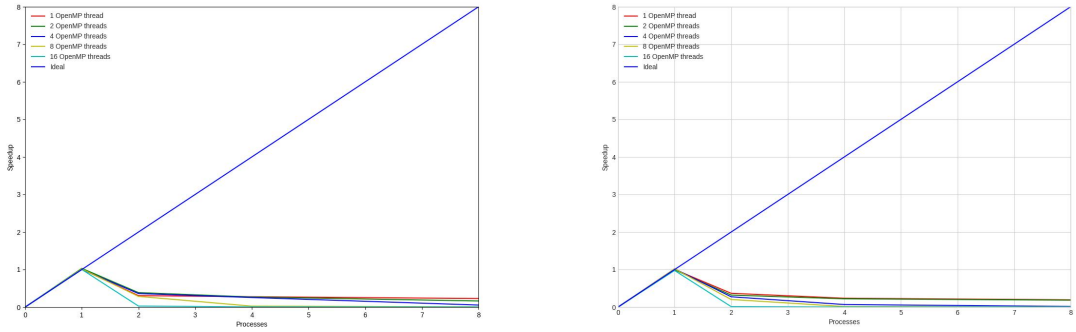

(a) (b)

Figura 14: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.030330600000000003 | 0.0 | 0.030330600000000003 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.028916800000000003 | 0.0 | 0.028916800000000003 | 1.0488919935815857 | 1.0488919935815857 |
| **Parallel** | 1 | 2 | 0.0289202 | 0.0 | 0.0289202 | 1.048768680714518 | 1.048768680714518 |
| **Parallel** | 1 | 4 | 0.0288968 | 0.0 | 0.0288968 | 1.0496179507765566 | 1.0496179507765566 |
| **Parallel** | 1 | 8 | 0.028938399999999996 | 0.0 | 0.028938399999999996 | 1.0481090868880105 | 1.0481090868880105 |
| **Parallel** | 1 | 16 | 0.028862 | 0.0 | 0.028862 | 1.0508835146559492 | 1.0508835146559492 |
| **Parallel** | 2 | 1 | 0.007817000000000001 | 0.013381600000000002 | 0.0211986 | 1.4307831649259857 | 0.7153915824629928 |
| **Parallel** | 2 | 2 | 0.007829 | 0.0068596 | 0.014688399999999999 | 2.0649355954358546 | 1.0324677977179273 |
| **Parallel** | 2 | 4 | 0.0078662 | 0.0068909999999999996 | 0.014757600000000001 | 2.0552528866482356 | 1.0276264433241178 |
| **Parallel** | 2 | 8 | 0.0078308 | 0.007063799999999999 | 0.0148946 | 2.03634874384005 | 1.018174371920025 |
| **Parallel** | 2 | 16 | 0.0078954 | 0.0131566 | 0.0210522 | 1.4407330350272183 | 0.7203665175136091 |
| **Parallel** | 4 | 1 | 0.002909 | 0.049592800000000006 | 0.0525018 | 0.577705907226038 | 0.1444264768065095 |
| **Parallel** | 4 | 2 | 0.0029782 | 0.036861200000000004 | 0.039839799999999995 | 0.7613140628216006 | 0.19032851570540016 |
| **Parallel** | 4 | 4 | 0.0029094 | 0.0396074 | 0.0425168 | 0.7133791818763408 | 0.1783447954690852 |
| **Parallel** | 4 | 8 | 0.0030949999999999997 | 0.04226220000000001 | 0.045357 | 0.6687082478999934 | 0.16717706197499835 |
| **Parallel** | 4 | 16 | 0.0030223999999999997 | 0.051002000000000006 | 0.0540238 | 0.5614303325571323 | 0.14035758313928307 |
| **Parallel** | 8 | 1 | 0.001638 | 0.11556380000000002 | 0.1172018 | 0.25878954077497107 | 0.032348692596871384 |
| **Parallel** | 8 | 2 | 0.0014425999999999998 | 0.07875499999999999 | 0.0801974 | 0.37819929324392065 | 0.04727491165549008 |
| **Parallel** | 8 | 4 | 0.0015255999999999998 | 0.0831544 | 0.0846802 | 0.3581781809679241 | 0.04477227262099051 |
| **Parallel** | 8 | 8 | 0.0017163999999999999 | 0.0928762 | 0.09459200000000001 | 0.32064656630581867 | 0.040080820788227334 |
| **Parallel** | 8 | 16 | 0.0016976 | 0.12905419999999998 | 0.1307512 | 0.2319718671798041 | 0.02899648339747551 |

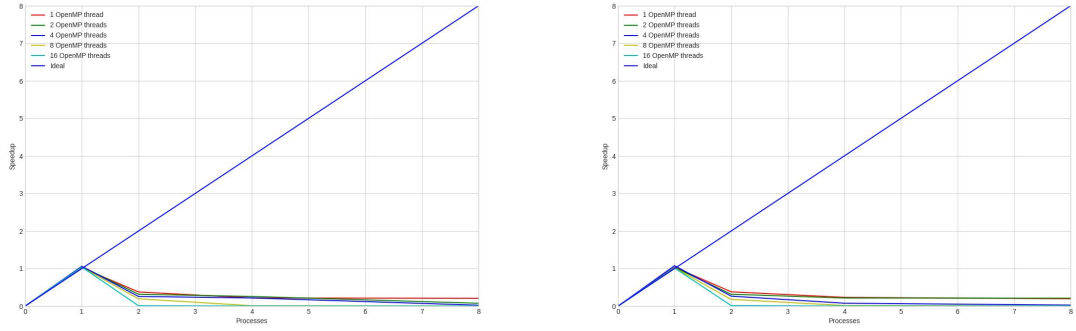
Tabella 6: Measurement with O1 optimization

### 4.2.4 1000 vertices - Special Cases - 0% of density




(a) (b)

Figura 15: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



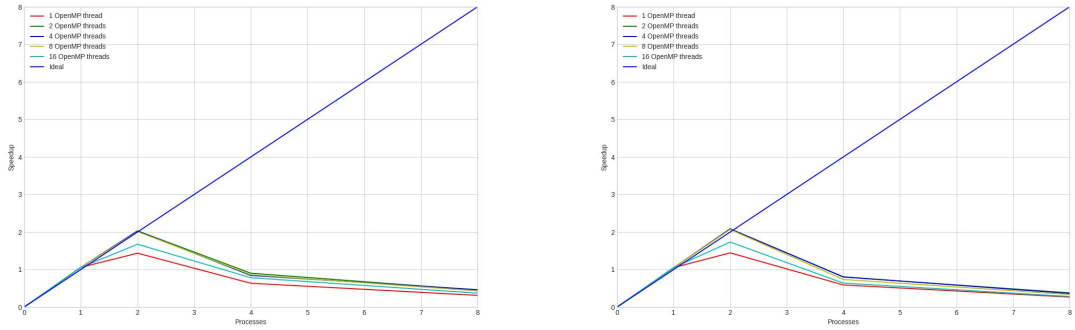

(a) (b)

Figura 16: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.0001796 | 0.0 | 0.0001796 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0001778 | 0.0 | 0.0001778 | 1.0101237345331833 | 1.0101237345331833 |
| **Parallel** | 1 | 2 | 0.000174 | 0.0 | 0.000174 | 1.032183908045977 | 1.032183908045977 |
| **Parallel** | 1 | 4 | 0.0001788 | 0.0 | 0.0001788 | 1.0044742729306488 | 1.0044742729306488 |
| **Parallel** | 1 | 8 | 0.0001762 | 0.0 | 0.0001762 | 1.0192962542565267 | 1.0192962542565267 |
| **Parallel** | 1 | 16 | 0.000179 | 0.0 | 0.000179 | 1.0033519553072627 | 1.0033519553072627 |
| **Parallel** | 2 | 1 | 0.00031099999999999997 | 0.0002768 | 0.000588 | 0.3054421768707483 | 0.15272108843537416 |
| **Parallel** | 2 | 2 | 0.00029239999999999995 | 0.0001712199999999999 8 | 0.0004648 | 0.3864027538726334 | 0.1932013769363167 |
| **Parallel** | 2 | 4 | 0.00029059999999999996 | 0.000207 | 0.0004971999999999999 | 0.3612228479485117 | 0.18061142397425586 |
| **Parallel** | 2 | 8 | 0.00032720000000000004 | 0.0003016 | 0.0006288 | 0.2856234096692112 | 0.1428117048346056 |
| **Parallel** | 2 | 16 | 0.0003516 | 0.006207799999999999 | 0.0065594 | 0.027385530993684 5 | 0.013690276549684422 |
| **Parallel** | 4 | 1 | 0.0003886 | 0.00026199999999999997 | 0.0006512 | 0.2757985257985258 | 0.06894963144963145 |
| **Parallel** | 4 | 2 | 0.0004142 | 0.0002536 | 0.000668 | 0.2688622754491018 | 0.06721556886227545 |
| **Parallel** | 4 | 4 | 0.0004482 | 0.00025560000000000003 | 0.000704 | 0.2551136363636364 | 0.0637784090909091 |
| **Parallel** | 4 | 8 | 0.0004188 | 0.006672 | 0.0070907999999999995 | 0.025328594798894342 | 0.006332148699723586 |
| **Parallel** | 4 | 16 | 0.0004046 | 0.0321834 | 0.032588 | 0.005511231128022585 | 0.0013778077820056462 |
| **Parallel** | 8 | 1 | 0.00044860000000000006 | 0.00033539999999999997 | 0.0007842 | 0.22902320836521298 | 0.028627901045651622 |
| **Parallel** | 8 | 2 | 0.0004548 | 0.0006468 | 0.0011014 | 0.1630651897584892 | 0.02038314871981115 |
| **Parallel** | 8 | 4 | 0.00048279999999999997 | 0.002801 | 0.0032842 | 0.05468607271177151 | 0.006835759088971439 |
| **Parallel** | 8 | 8 | 0.00044740000000000003 | 0.029304000000000004 | 0.0297516 | 0.006036650129740921 | 0.0007545812662176152 |
| **Parallel** | 8 | 16 | 0.00044780000000000004 | 0.069749 | 0.07019700000000001 | 0.0025585138966052676 | 0.0003198142370756584 5 |

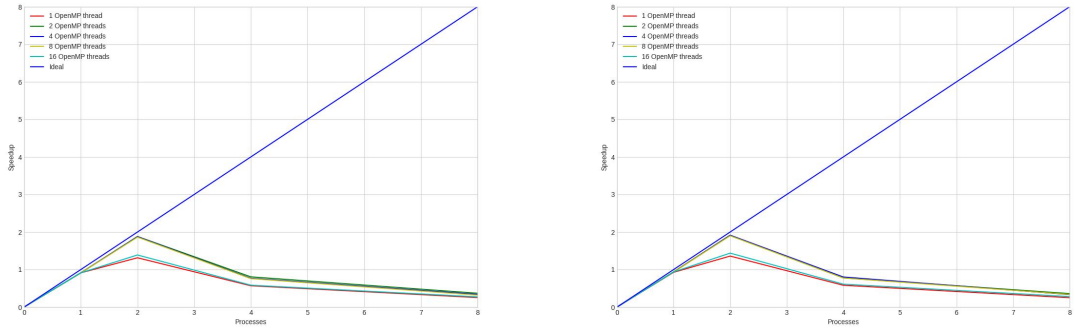
Tabella 7: Measurement with O0 optimization

### 4.2.5 1000 vertices - Special Cases - 100% of density




(a) (b)

Figura 17: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization




(a) (b)

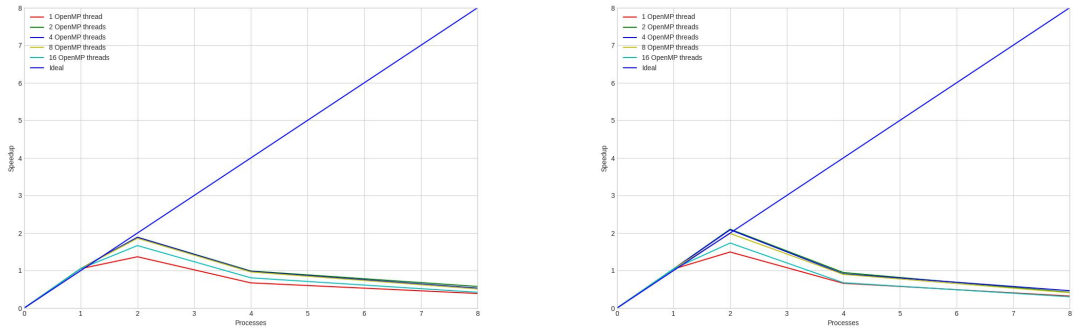
Figura 18: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.048830200000000004 | 0.0 | 0.048830200000000004 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0464392 | 0.0 | 0.0464392 | 1.0514866750503886 | 1.0514866750503886 |
| **Parallel** | 1 | 2 | 0.04677560000000001 | 0.0 | 0.04677560000000001 | 1.0439246102668913 | 1.0439246102668913 |
| **Parallel** | 1 | 4 | 0.046619799999999996 | 0.0 | 0.046619799999999996 | 1.04741333081652 | 1.04741333081652 |
| **Parallel** | 1 | 8 | 0.0465936 | 0.0 | 0.0465936 | 1.0480023007451669 | 1.0480023007451669 |
| **Parallel** | 1 | 16 | 0.046505200000000003 | 0.0 | 0.046505200000000003 | 1.0499944092273552 | 1.0499944092273552 |
| **Parallel** | 2 | 1 | 0.012528000000000001 | 0.0212566 | 0.0337846 | 1.4453390006097455 | 0.7226695003048728 |
| **Parallel** | 2 | 2 | 0.012598599999999998 | 0.0108266 | 0.0234252 | 2.0845158205693015 | 1.0422579102846508 |
| **Parallel** | 2 | 4 | 0.012596400000000002 | 0.0108528 | 0.023449 | 2.0824001023497805 | 1.0412000511748902 |
| **Parallel** | 2 | 8 | 0.012605999999999997 | 0.011015399999999998 | 0.023621200000000002 | 2.06721927759809 | 1.033609638799045 |
| **Parallel** | 2 | 16 | 0.012717799999999998 | 0.015476800000000002 | 0.028194599999999997 | 1.7318990161236552 | 0.8659495080618276 |
| **Parallel** | 4 | 1 | 0.0043132000000000005 | 0.0784266 | 0.0827398 | 0.5901657968716386 | 0.14754144921790965 |
| **Parallel** | 4 | 2 | 0.0043308 | 0.056724399999999994 | 0.0610554 | 0.7997687346246196 | 0.1999421836561549 |
| **Parallel** | 4 | 4 | 0.0044066 | 0.056159 | 0.0605658 | 0.8062338811672595 | 0.20155847029181487 |
| **Parallel** | 4 | 8 | 0.0044513999999999995 | 0.06190659999999999 | 0.066358 | 0.7358600319479189 | 0.18396500798697973 |
| **Parallel** | 4 | 16 | 0.004565 | 0.0718104 | 0.0763752 | 0.6393462799442752 | 0.1598365699860688 |
| **Parallel** | 8 | 1 | 0.0021868 | 0.1804298 | 0.1826168 | 0.26739160909620585 | 0.03342395113702573 |
| **Parallel** | 8 | 2 | 0.0022412 | 0.12609720000000002 | 0.1283378 | 0.38048182219112375 | 0.04756022777389047 |
| **Parallel** | 8 | 4 | 0.0025296000000000003 | 0.13233299999999998 | 0.1348626 | 0.362073695746634 | 0.04525921196832925 |
| **Parallel** | 8 | 8 | 0.0025503999999999995 | 0.1429974 | 0.145548 | 0.33549207134416137 | 0.04193650891802017 |
| **Parallel** | 8 | 16 | 0.0054926 | 0.1672506 | 0.1727436 | 0.2826744377215712 | 0.0353343047151964 |

Tabella 8: Measurement with O1 optimization

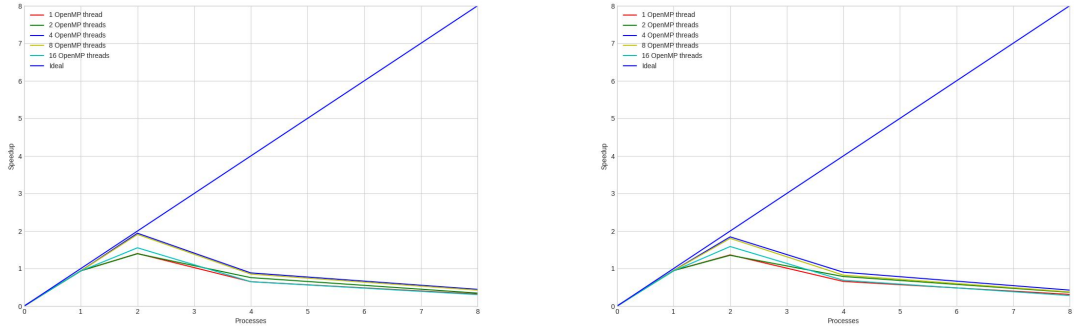
## 4.3 Case Study 3

### 4.3.1 2500 vertices - 25% of density




(a) (b)

Figura 19: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



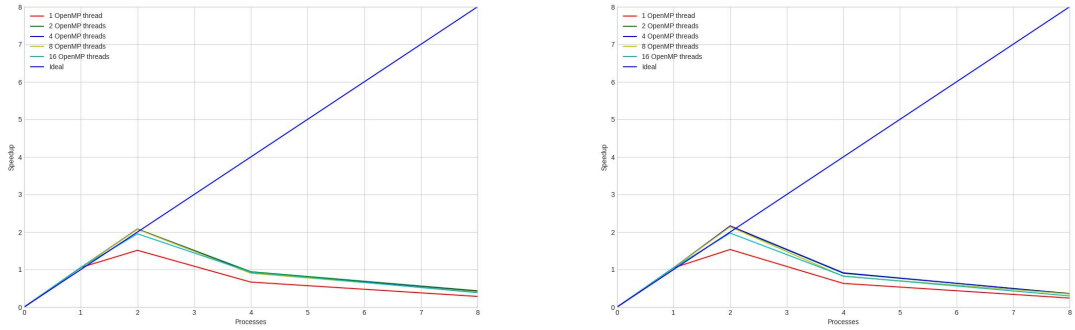

(a) (b)

Figura 20: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.0443628 | 0.0 | 0.0443628 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0425736 | 0.0 | 0.0425736 | 1.0420260443091494 | 1.0420260443091494 |
| **Parallel** | 1 | 2 | 0.042465800000000005 | 0.0 | 0.042465800000000005 | 1.044671241328316 | 1.044671241328316 |
| **Parallel** | 1 | 4 | 0.042559599999999996 | 0.0 | 0.042559599999999996 | 1.0423688192558203 | 1.0423688192558203 |
| **Parallel** | 1 | 8 | 0.0424726 | 0.0 | 0.0424726 | 1.0445039860992735 | 1.0445039860992735 |
| **Parallel** | 1 | 16 | 0.042426399999999996 | 0.0 | 0.042426399999999996 | 1.0456413930948656 | 1.0456413930948656 |
| **Parallel** | 2 | 1 | 0.0122476 | 0.0174326 | 0.0296802 | 1.4946934319849596 | 0.7473467159924798 |
| **Parallel** | 2 | 2 | 0.012052799999999999 | 0.0090528 | 0.0211056 | 2.101944507618831 | 1.0509722538094155 |
| **Parallel** | 2 | 4 | 0.012117 | 0.009154399999999998 | 0.021271400000000003 | 2.085560893970307 | 1.0427804469851536 |
| **Parallel** | 2 | 8 | 0.0121748 | 0.0101434 | 0.022318199999999996 | 1.987740946850553 | 0.9938704734252765 |
| **Parallel** | 2 | 16 | 0.012333 | 0.0132554 | 0.0255884 | 1.7337074611933532 | 0.8668537305966766 |
| **Parallel** | 4 | 1 | 0.0044134000000000005 | 0.0627094 | 0.0671228 | 0.6609199854594863 | 0.16522999636487157 |
| **Parallel** | 4 | 2 | 0.004571 | 0.042232000000000006 | 0.046803 | 0.947862316518172 | 0.236965579129543 |
| **Parallel** | 4 | 4 | 0.0044735999999999995 | 0.044119599999999995 | 0.0485932 | 0.9129425516327387 | 0.22823563790818469 |
| **Parallel** | 4 | 8 | 0.004968599999999999 | 0.0444124 | 0.049381400000000006 | 0.8983706415775979 | 0.22459266039439948 |
| **Parallel** | 4 | 16 | 0.0046974 | 0.060938599999999996 | 0.0656364 | 0.6758871601733185 | 0.16897179004332963 |
| **Parallel** | 8 | 1 | 0.0024067999999999997 | 0.1355576 | 0.137964 | 0.32155344872575453 | 0.040194181090719316 |
| **Parallel** | 8 | 2 | 0.0023882 | 0.1030402 | 0.10542880000000002 | 0.4207844535838404 | 0.05259805669798005 |
| **Parallel** | 8 | 4 | 0.002622 | 0.09305260000000001 | 0.0956744 | 0.46368516551972105 | 0.05796064568996513 |
| **Parallel** | 8 | 8 | 0.0027224 | 0.105266 | 0.10798840000000001 | 0.41081079078864025 | 0.05135134884858003 |
| **Parallel** | 8 | 16 | 0.0029555999999999996 | 0.14400439999999998 | 0.1469596 | 0.3018707182109913 | 0.037733839776373915 |

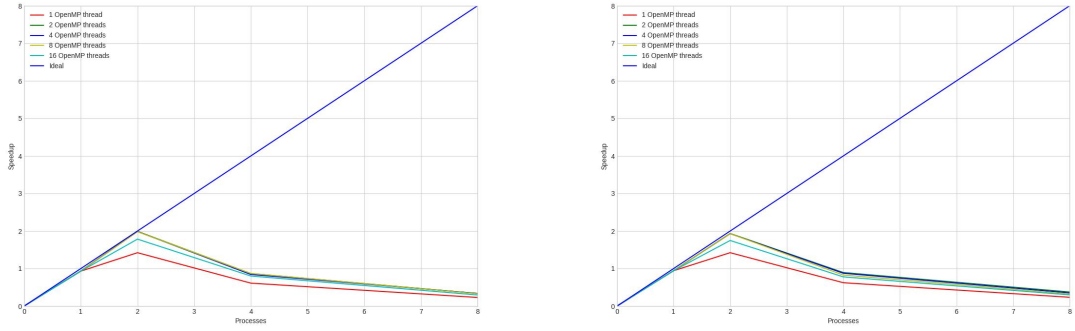
Tabella 9: Measurement with O1 optimization

### 4.3.2 2500 vertices - 50% of density




(a) (b)

Figura 21: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



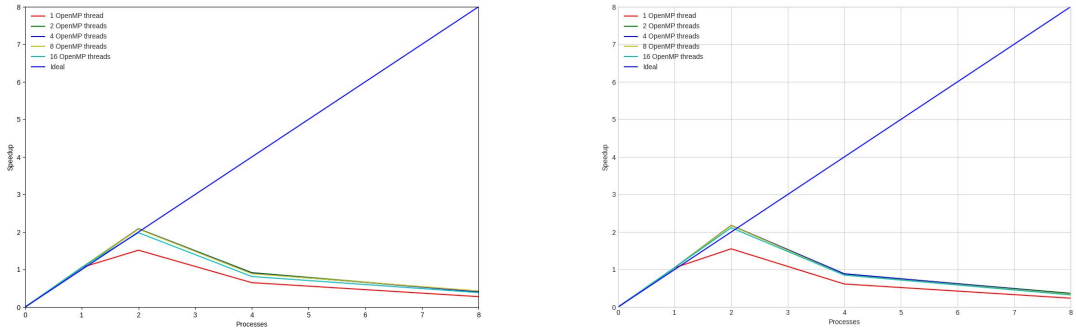

(a) (b)

Figura 22: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.13916080000000003 | 0.0 | 0.13916080000000003 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.13302740000000002 | 0.0 | 0.13302740000000002 | 1.0461062908844345 | 1.0461062908844345 |
| **Parallel** | 1 | 2 | 0.13307000000000002 | 0.0 | 0.13307000000000002 | 1.0457713985120614 | 1.0457713985120614 |
| **Parallel** | 1 | 4 | 0.1330088 | 0.0 | 0.1330088 | 1.0462525787767427 | 1.0462525787767427 |
| **Parallel** | 1 | 8 | 0.1331484 | 0.0 | 0.1331484 | 1.045155630860003 | 1.045155630860003 |
| **Parallel** | 1 | 16 | 0.13308 | 0.0 | 0.13308 | 1.0456928163510673 | 1.0456928163510673 |
| **Parallel** | 2 | 1 | 0.03763180000000001 | 0.053057 | 0.09068860000000001 | 1.5344905533881879 | 0.7672452766940939 |
| **Parallel** | 2 | 2 | 0.0377744 | 0.026872 | 0.06464620000000001 | 2.1526524374209157 | 1.0763262187104579 |
| **Parallel** | 2 | 4 | 0.0374056 | 0.026938200000000002 | 0.06434360000000001 | 2.162776095835483 | 1.0813880479177416 |
| **Parallel** | 2 | 8 | 0.037787 | 0.027248 | 0.06503519999999999 | 2.1397766132801936 | 1.0698883066400968 |
| **Parallel** | 2 | 16 | 0.037883 | 0.0327006 | 0.07058339999999999 | 1.971579719877479 | 0.9857898599387395 |
| **Parallel** | 4 | 1 | 0.0126972 | 0.20786360000000004 | 0.220561 | 0.6309401934158805 | 0.15773504835397012 |
| **Parallel** | 4 | 2 | 0.012660200000000002 | 0.1416944 | 0.1543548 | 0.9015644476232683 | 0.22539111190581707 |
| **Parallel** | 4 | 4 | 0.0126626 | 0.1396828 | 0.152345 | 0.9134582690603565 | 0.22836456726508914 |
| **Parallel** | 4 | 8 | 0.012995200000000002 | 0.1568528 | 0.16984819999999998 | 0.819324549803884 | 0.204831137450971 |
| **Parallel** | 4 | 16 | 0.0131714 | 0.1551554 | 0.1683266 | 0.8267308910178192 | 0.2066827227544548 |
| **Parallel** | 8 | 1 | 0.0062284 | 0.5711672 | 0.5773954 | 0.24101473617559133 | 0.030126842021948916 |
| **Parallel** | 8 | 2 | 0.0062346 | 0.37756059999999997 | 0.383795 | 0.3625914876431429 | 0.045323935955392865 |
| **Parallel** | 8 | 4 | 0.0066832 | 0.3848771999999999 | 0.39156040000000003 | 0.3554005972003298 | 0.044425074650041224 |
| **Parallel** | 8 | 8 | 0.0068606 | 0.39002220000000004 | 0.3968826 | 0.3506346713108613 | 0.04382933391385766 |
| **Parallel** | 8 | 16 | 0.0070842000000000006 | 0.4594347999999999 | 0.46651879999999996 | 0.29829623157737706 | 0.03728702894717213 |

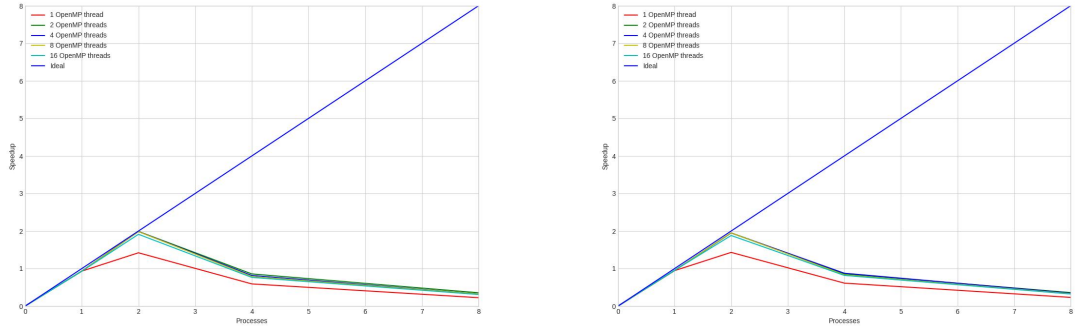
Tabella 10: Measurement with O1 optimization

### 4.3.3 2500 vertices - 75% of density




(a) (b)

Figura 23: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization




(a) (b)

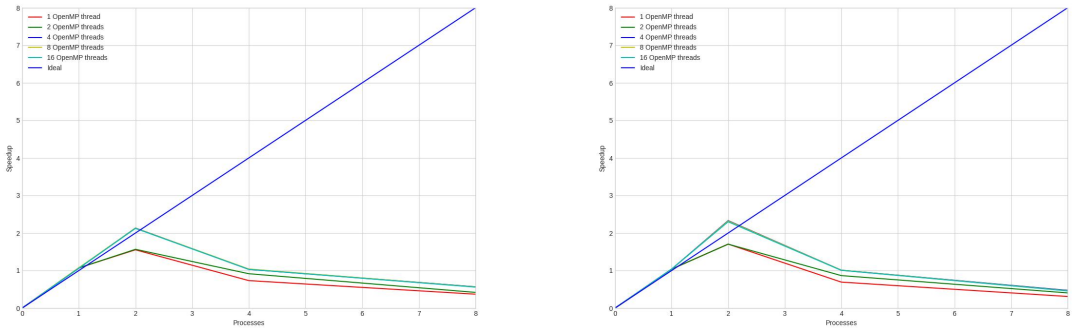
Figura 24: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.42665120000000006 | 0.0 | 0.42665120000000006 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.40412860000000006 | 0.0 | 0.40412860000000006 | 1.0557312696997936 | 1.0557312696997936 |
| **Parallel** | 1 | 2 | 0.404099 | 0.0 | 0.404099 | 1.0558086013575883 | 1.0558086013575883 |
| **Parallel** | 1 | 4 | 0.41049420000000003 | 0.0 | 0.41049420000000003 | 1.0393598740250167 | 1.0393598740250167 |
| **Parallel** | 1 | 8 | 0.40713039999999995 | 0.0 | 0.40713039999999995 | 1.04794729158029 | 1.04794729158029 |
| **Parallel** | 1 | 16 | 0.40456380000000003 | 0.0 | 0.40456380000000003 | 1.0545955915976666 | 1.0545955915976666 |
| **Parallel** | 2 | 1 | 0.1265274 | 0.1548254 | 0.28135260000000006 | 1.5164288511995268 | 0.7582144255997634 |
| **Parallel** | 2 | 2 | 0.12657639999999998 | 0.07782039999999998 | 0.2043968 | 2.0873673169051576 | 1.0436836584525788 |
| **Parallel** | 2 | 4 | 0.1266986 | 0.07795579999999999 | 0.20465460000000002 | 2.084737894970355 | 1.0423689474851776 |
| **Parallel** | 2 | 8 | 0.12664340000000002 | 0.0786228 | 0.20526619999999998 | 2.0785263233791054 | 1.0392631616895527 |
| **Parallel** | 2 | 16 | 0.12647919999999999 | 0.0887808 | 0.21525999999999995 | 1.9820273158041446 | 0.9910136579020723 |
| **Parallel** | 4 | 1 | 0.0522686 | 0.6028165999999999 | 0.6550848 | 0.6512915579784481 | 0.16282288949461202 |
| **Parallel** | 4 | 2 | 0.052274 | 0.41169399999999995 | 0.46396740000000003 | 0.9195715043772472 | 0.2298928760943118 |
| **Parallel** | 4 | 4 | 0.0522532 | 0.4202108 | 0.4724642 | 0.9030339229935307 | 0.22575848074838267 |
| **Parallel** | 4 | 8 | 0.052374199999999996 | 0.4262898 | 0.4786639999999999 | 0.891337556198085 | 0.22283438904952124 |
| **Parallel** | 4 | 16 | 0.052535399999999996 | 0.4710888 | 0.5236240000000001 | 0.814804516217744 | 0.203701129054436 |
| **Parallel** | 8 | 1 | 0.030617 | 1.4972408 | 1.5278576 | 0.2792480136892339 | 0.03490600171115424 |
| **Parallel** | 8 | 2 | 0.0305944 | 1.0106538 | 1.0412482 | 0.4097497599515659 | 0.051218719993945734 |
| **Parallel** | 8 | 4 | 0.030820800000000002 | 1.0092431999999998 | 1.0400639999999999 | 0.4102162943818843 | 0.051277036797735535 |
| **Parallel** | 8 | 8 | 0.0312626 | 0.9612740000000001 | 0.9925362 | 0.4298595859778213 | 0.053732448247227664 |
| **Parallel** | 8 | 16 | 0.03252240000000001 | 1.0702375999999998 | 1.1027602 | 0.3868939049486915 | 0.04836173811858644 |

Tabella 11: Measurement with O0 optimization

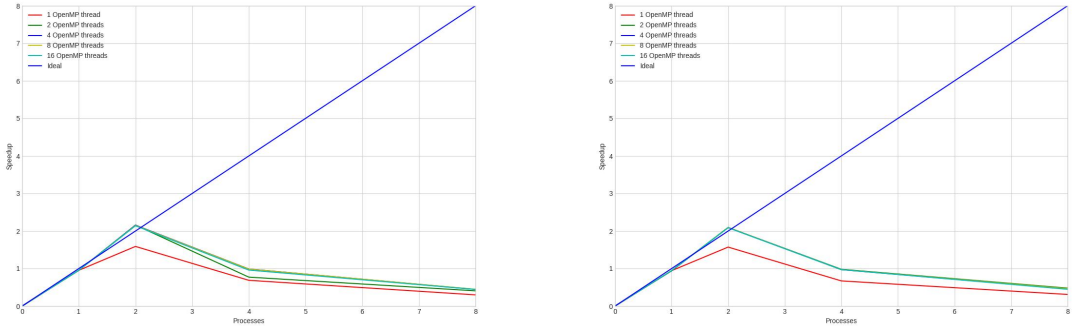
## 4.4 Case Study 4

### 4.4.1 10000 vertices - 25% of density




(a) (b)

Figura 25: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



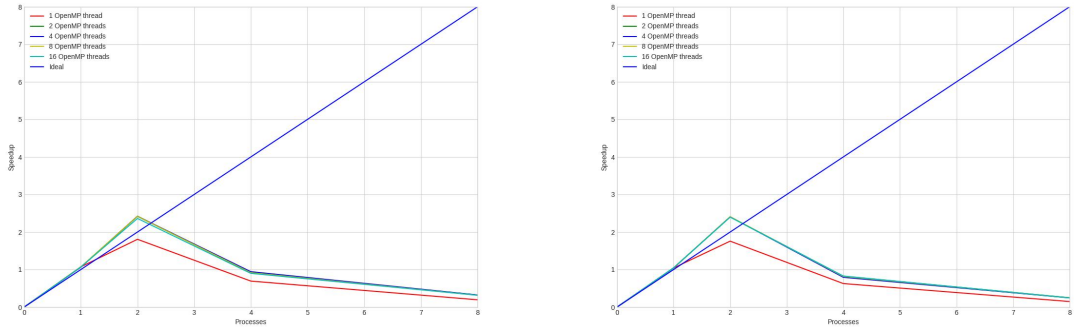

(a) (b)

Figura 26: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.8900881999999999 | 0.0 | 0.8900881999999999 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.858418 | 0.0 | 0.858418 | 1.0368936811669838 | 1.0368936811669838 |
| **Parallel** | 1 | 2 | 0.8573674 | 0.0 | 0.8573674 | 1.038164268900357 | 1.038164268900357 |
| **Parallel** | 1 | 4 | 0.8581804 | 0.0 | 0.8581804 | 1.0371807605953247 | 1.0371807605953247 |
| **Parallel** | 1 | 8 | 0.8603012000000001 | 0.0 | 0.8603012000000001 | 1.0346239200875227 | 1.0346239200875227 |
| **Parallel** | 1 | 16 | 0.8575512 | 0.0 | 0.8575512 | 1.0379417578798793 | 1.0379417578798793 |
| **Parallel** | 2 | 1 | 0.23273939999999999 | 0.2892004 | 0.5219396 | 1.7053471321202682 | 0.8526735660601341 |
| **Parallel** | 2 | 2 | 0.23270439999999998 | 0.28963 | 0.5223342 | 1.7040588190472687 | 0.8520294095236344 |
| **Parallel** | 2 | 4 | 0.2337522 | 0.14815839999999997 | 0.3819106 | 2.3306192601095646 | 1.1653096300547823 |
| **Parallel** | 2 | 8 | 0.23270200000000002 | 0.15071020000000002 | 0.383412 | 2.3214928066935827 | 1.1607464033467914 |
| **Parallel** | 2 | 16 | 0.23347859999999998 | 0.15354360000000003 | 0.3870226 | 2.2998352034222287 | 1.1499176017111143 |
| **Parallel** | 4 | 1 | 0.0888838 | 1.1964009999999998 | 1.2852846000000002 | 0.6925222631625709 | 0.17313056579064273 |
| **Parallel** | 4 | 2 | 0.089077 | 0.9389668 | 1.028044 | 0.8658074946208527 | 0.21645187365521318 |
| **Parallel** | 4 | 4 | 0.0891376 | 0.7915987999999999 | 0.8807362 | 1.0106183894791652 | 0.2526545973697913 |
| **Parallel** | 4 | 8 | 0.09004960000000001 | 0.7893361999999999 | 0.8793858 | 1.0121703125067518 | 0.25304257812668796 |
| **Parallel** | 4 | 16 | 0.09007659999999999 | 0.7944167999999999 | 0.8844933999999999 | 1.0063254287708647 | 0.2515813571927162 |
| **Parallel** | 8 | 1 | 0.0486286 | 2.8275306000000002 | 2.8761592 | 0.3094711168978407 | 0.038683889612230085 |
| **Parallel** | 8 | 2 | 0.0483292 | 2.1480826 | 2.1964118000000004 | 0.40524650250012306 | 0.05065581281251538 |
| **Parallel** | 8 | 4 | 0.051574600000000005 | 1.8299406 | 1.8815154 | 0.473069845721167 | 0.059133730715145875 |
| **Parallel** | 8 | 8 | 0.053162799999999996 | 1.8656082 | 1.9187708 | 0.4638845869449336 | 0.0579855733681167 |
| **Parallel** | 8 | 16 | 0.0538454 | 1.8911652 | 1.9450105999999998 | 0.45762640059648 | 0.05720330007456 |

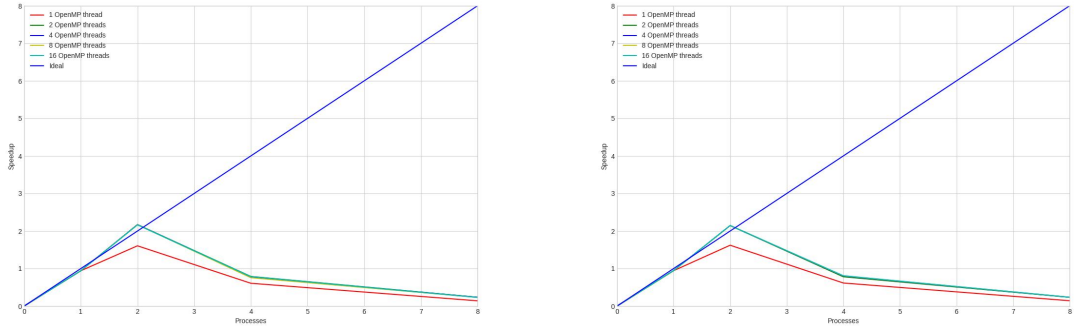
Tabella 12: Measurement with O1 optimization

### 4.4.2 10000 vertices - 50% of density




(a) (b)

Figura 27: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



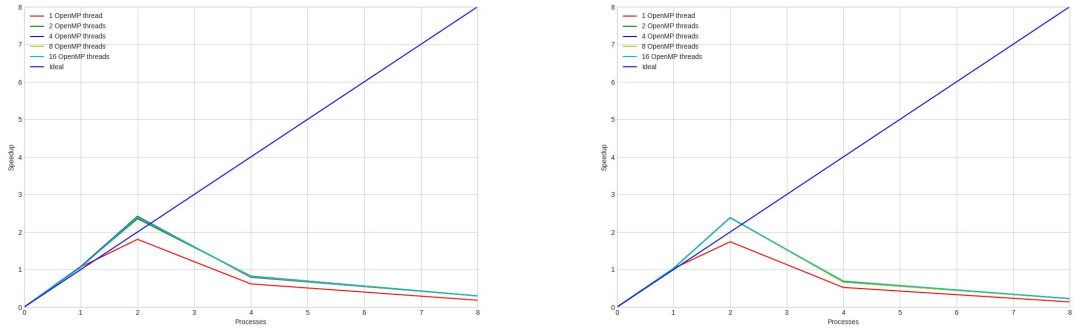

(a) (b)

Figura 28: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 5.228125799999999 | 0.0 | 5.228125799999999 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 4.8749970000000005 | 0.0 | 4.8749970000000005 | 1.0724367214995205 | 1.0724367214995205 |
| **Parallel** | 1 | 2 | 4.900415000000001 | 0.0 | 4.900415000000001 | 1.0668740912759427 | 1.0668740912759427 |
| **Parallel** | 1 | 4 | 4.920706600000001 | 0.0 | 4.920706600000001 | 1.0624746047650957 | 1.0624746047650957 |
| **Parallel** | 1 | 8 | 4.8850076 | 0.0 | 4.8850076 | 1.070239030948488 | 1.070239030948488 |
| **Parallel** | 1 | 16 | 4.909088 | 0.0 | 4.909088 | 1.0649892199936117 | 1.0649892199936117 |
| **Parallel** | 2 | 1 | 1.4243508000000003 | 1.470082 | 2.8944327999999997 | 1.806269539234077 | 0.9031347696170385 |
| **Parallel** | 2 | 2 | 1.425722 | 0.736356 | 2.1620777999999996 | 2.418102530815496 | 1.209051265407748 |
| **Parallel** | 2 | 4 | 1.4233513999999998 | 0.7365878 | 2.1599394000000003 | 2.4204965194856847 | 1.2102482597428423 |
| **Parallel** | 2 | 8 | 1.4242178 | 0.7375038 | 2.161722 | 2.4185005287451387 | 1.2092502643725693 |
| **Parallel** | 2 | 16 | 1.4690277999999999 | 0.7448782 | 2.2139062000000003 | 2.3614938157723206 | 1.1807469078861603 |
| **Parallel** | 4 | 1 | 0.5742687999999999 | 6.970635400000001 | 7.544904 | 0.692934701356041 | 0.17323367533901024 |
| **Parallel** | 4 | 2 | 0.5746612 | 5.109519199999999 | 5.684180400000001 | 0.9197677469912809 | 0.22994193674782024 |
| **Parallel** | 4 | 4 | 0.5742339999999999 | 4.964993 | 5.539227 | 0.9438367122343964 | 0.2359591780585991 |
| **Parallel** | 4 | 8 | 0.5741222 | 5.1240908 | 5.6982132000000005 | 0.9175026655724287 | 0.22937566639310716 |
| **Parallel** | 4 | 16 | 0.5747658 | 5.2438034 | 5.818569200000001 | 0.8985242970041498 | 0.22463107425103745 |
| **Parallel** | 8 | 1 | 0.3206894 | 26.510825600000004 | 26.831514799999997 | 0.19485019161124661 | 0.024356273951405827 |
| **Parallel** | 8 | 2 | 0.3200786 | 16.0177628 | 16.3378412 | 0.32000101702543166 | 0.04000012712817896 |
| **Parallel** | 8 | 4 | 0.3212148 | 15.810775800000002 | 16.1319906 | 0.32408435695468346 | 0.04051054461933543 |
| **Parallel** | 8 | 8 | 0.3217104 | 16.1981328 | 16.519843399999996 | 0.3164755060571579 | 0.03955943825714474 |
| **Parallel** | 8 | 16 | 0.32662979999999997 | 16.293736399999997 | 16.620366 | 0.31456141218550776 | 0.03932017652318847 |

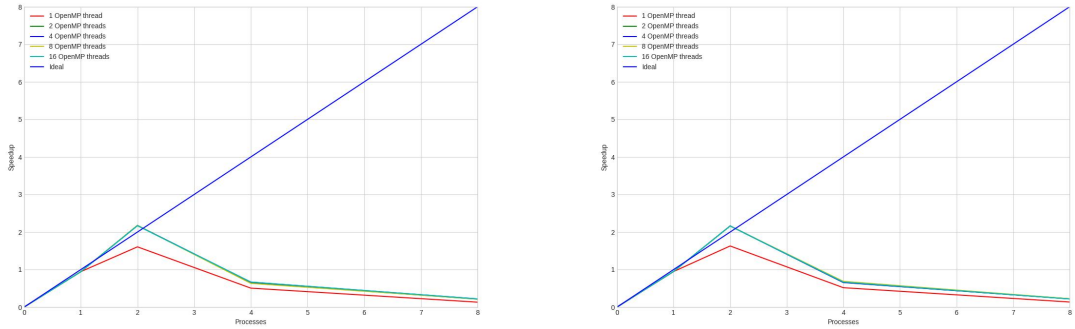
Tabella 13: Measurement with O0 optimization

### 4.4.3 10000 vertices - 75% of density




(a) (b)

Figura 29: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



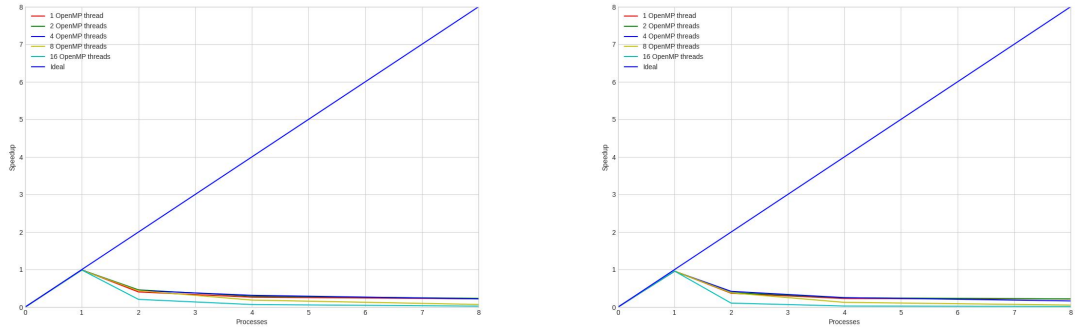

(a) (b)

Figura 30: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 8.7069948 | 0.0 | 8.7069948 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 8.149559400000001 | 0.0 | 8.149559400000001 | 1.0684006794281418 | 1.0684006794281418 |
| **Parallel** | 1 | 2 | 8.1039906 | 0.0 | 8.1039906 | 1.0744083044716268 | 1.0744083044716268 |
| **Parallel** | 1 | 4 | 8.049288 | 0.0 | 8.049288 | 1.081709935089911 | 1.081709935089911 |
| **Parallel** | 1 | 8 | 8.2708258 | 0.0 | 8.2708258 | 1.052735846522121 | 1.052735846522121 |
| **Parallel** | 1 | 16 | 8.187621799999999 | 0.0 | 8.187621799999999 | 1.0634339265646102 | 1.0634339265646102 |
| **Parallel** | 2 | 1 | 2.3697942 | 2.4507336 | 4.8205278 | 1.806232670206777 | 0.9031163351033885 |
| **Parallel** | 2 | 2 | 2.4678876 | 1.2275548 | 3.6954424 | 2.356144097929926 | 1.178072048964963 |
| **Parallel** | 2 | 4 | 2.370035 | 1.2281488 | 3.5981838 | 2.4198304711393566 | 1.2099152355696783 |
| **Parallel** | 2 | 8 | 2.3960332 | 1.2287218000000002 | 3.6247548000000003 | 2.402092080821577 | 1.2010460404107886 |
| **Parallel** | 2 | 16 | 2.4062744 | 1.2383694 | 3.6446438 | 2.3889837465049397 | 1.1944918732524699 |
| **Parallel** | 4 | 1 | 0.9543166 | 13.135392 | 14.0897088 | 0.6179683997443581 | 0.15449209993608953 |
| **Parallel** | 4 | 2 | 0.9545667999999999 | 9.563544799999999 | 10.518111600000001 | 0.8278096992239558 | 0.20695242480598894 |
| **Parallel** | 4 | 4 | 0.9542000000000002 | 9.979996600000002 | 10.934196600000002 | 0.7963086012190415 | 0.19907715030476036 |
| **Parallel** | 4 | 8 | 0.9551701999999999 | 9.796744 | 10.7519142 | 0.8098088059519671 | 0.20245220148799178 |
| **Parallel** | 4 | 16 | 0.9568540000000001 | 9.5875586 | 10.544412600000001 | 0.8257448878660154 | 0.20643622196650385 |
| **Parallel** | 8 | 1 | 0.5322442000000001 | 47.077196400000005 | 47.609440400000004 | 0.18288378789682225 | 0.02286047348710278 |
| **Parallel** | 8 | 2 | 0.5302334 | 28.635598599999998 | 29.1658318 | 0.2985340812395414 | 0.037316760154942674 |
| **Parallel** | 8 | 4 | 0.5317035999999999 | 28.653636 | 29.185339399999997 | 0.29833453984091757 | 0.037291817480114696 |
| **Parallel** | 8 | 8 | 0.534897 | 28.8670814 | 29.401978200000002 | 0.2961363599677793 | 0.037017044995972415 |
| **Parallel** | 8 | 16 | 0.5385164 | 28.762609799999996 | 29.301126000000004 | 0.29715563831915537 | 0.03714445478989442 |

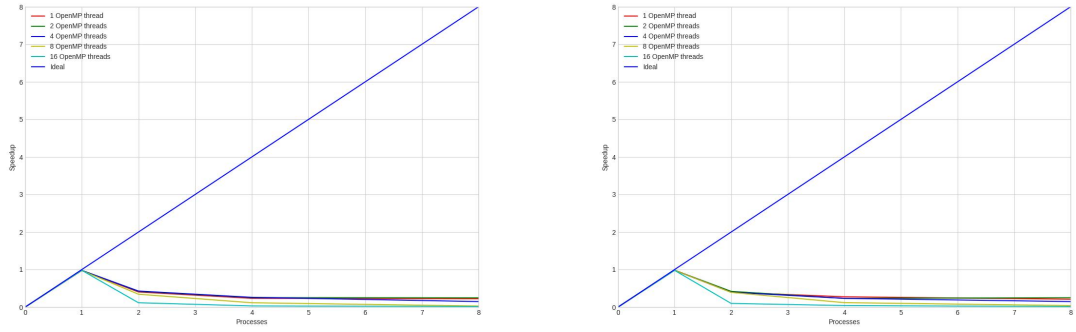
Tabella 14: Measurement with O0 optimization

### 4.4.4 10000 vertices - Special Cases - 0% of density




(a) (b)

Figura 31: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



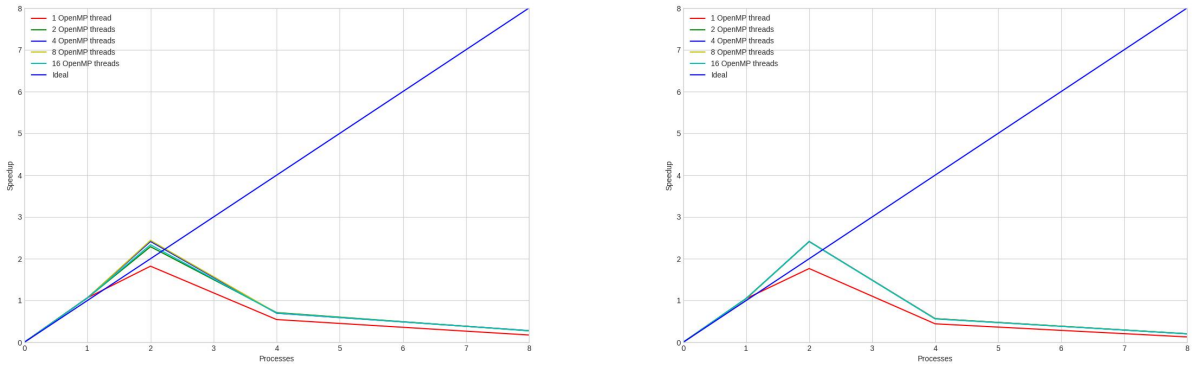

(a) (b)

Figura 32: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.0019538 | 0.0 | 0.0019538 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0019708000000000004 | 0.0 | 0.0019708000000000004 | 0.9913740612949054 | 0.9913740612949054 |
| **Parallel** | 1 | 2 | 0.0019758 | 0.0 | 0.0019758 | 0.988865269764146 | 0.988865269764146 |
| **Parallel** | 1 | 4 | 0.0019784 | 0.0 | 0.0019784 | 0.9875657096643752 | 0.9875657096643752 |
| **Parallel** | 1 | 8 | 0.0019844 | 0.0 | 0.0019844 | 0.9845797218302762 | 0.9845797218302762 |
| **Parallel** | 1 | 16 | 0.0019768 | 0.0 | 0.0019768 | 0.9883650343990287 | 0.9883650343990287 |
| **Parallel** | 2 | 1 | 0.0033044000000000003 | 0.0015406 | 0.0048444 | 0.4033110395508216 | 0.2016555197754108 |
| **Parallel** | 2 | 2 | 0.0031212 | 0.0011252 | 0.0042458 | 0.4601724056714871 | 0.23008620283574355 |
| **Parallel** | 2 | 4 | 0.0033998 | 0.0010236 | 0.0044234999999999995 | 0.4417164044130946 | 0.2208582022065473 |
| **Parallel** | 2 | 8 | 0.0034608 | 0.0009824 | 0.004443000000000001 | 0.43974791807337377 | 0.21987395903668688 |
| **Parallel** | 2 | 16 | 0.0035496 | 0.005952000000000001 | 0.009501200000000001 | 0.20563718267166248 | 0.10281859133583124 |
| **Parallel** | 4 | 1 | 0.004377799999999999 | 0.0030162 | 0.007394200000000001 | 0.2642341294528143 | 0.06605853236320358 |
| **Parallel** | 4 | 2 | 0.004398 | 0.0023737999999999997 | 0.0067716 | 0.288528560458385 | 0.07213214011459625 |
| **Parallel** | 4 | 4 | 0.003975 | 0.0022454000000000003 | 0.006220399999999999 | 0.31409555655584853 | 0.07852388913896213 |
| **Parallel** | 4 | 8 | 0.0046246 | 0.0058328 | 0.010457600000000001 | 0.18683063035495714 | 0.046707657588739286 |
| **Parallel** | 4 | 16 | 0.0051588 | 0.0235674 | 0.0287262 | 0.0680145651008487 | 0.017003641275212176 |
| **Parallel** | 8 | 1 | 0.0050378 | 0.0038033999999999997 | 0.008840999999999998 | 0.22099310032801722 | 0.027624137541002153 |
| **Parallel** | 8 | 2 | 0.004713400000000001 | 0.0036331999999999996 | 0.008346799999999998 | 0.23407773038769353 | 0.02925971629846169 |
| **Parallel** | 8 | 4 | 0.004716 | 0.0042224 | 0.008938199999999999 | 0.2185898726813005 | 0.027323734085162563 |
| **Parallel** | 8 | 8 | 0.0044928 | 0.0235066 | 0.0279994 | 0.06978006671571534 | 0.008722508339464417 |
| **Parallel** | 8 | 16 | 0.0048454 | 0.0709326 | 0.075778 | 0.025783208846894876 | 0.0032229011058618594 |

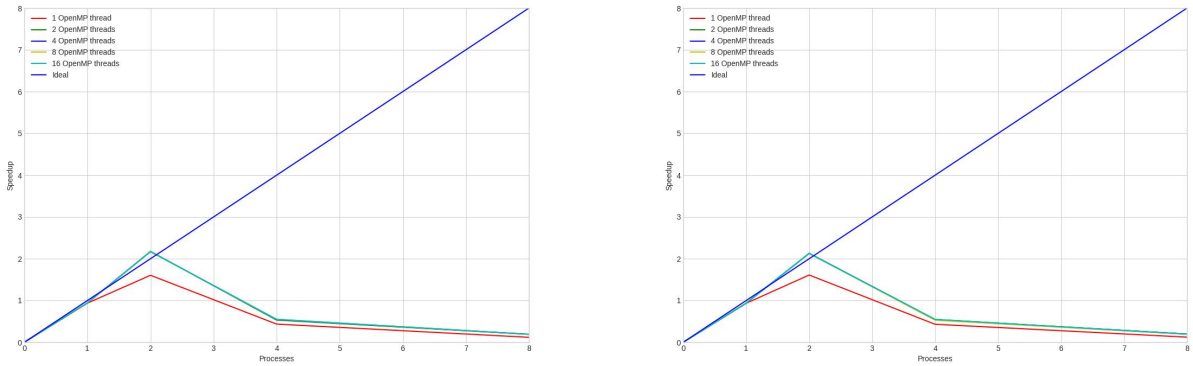
Tabella 15: Measurement with O0 optimization

### 4.4.5 10000 vertices - Special Cases - 100% of density




(a) (b)

Figura 33: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization




(a) (b)

Figura 34: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 7.1541885999999995 | 0.0 | 7.1541885999999995 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 6.8340534 | 0.0 | 6.8340534 | 1.0468441174310987 | 1.0468441174310987 |
| **Parallel** | 1 | 2 | 6.8174104 | 0.0 | 6.8174104 | 1.0493997251507698 | 1.0493997251507698 |
| **Parallel** | 1 | 4 | 6.846560800000001 | 0.0 | 6.846560800000001 | 1.0449317268897982 | 1.0449317268897982 |
| **Parallel** | 1 | 8 | 6.881190599999999 | 0.0 | 6.881190599999999 | 1.0396730763423412 | 1.0396730763423412 |
| **Parallel** | 1 | 16 | 6.819358800000001 | 0.0 | 6.819358800000001 | 1.0490998948464185 | 1.0490998948464185 |
| **Parallel** | 2 | 1 | 1.869538 | 2.1838623999999998 | 4.053400399999999 | 1.7649844313431264 | 0.8824922156715632 |
| **Parallel** | 2 | 2 | 1.8661496 | 1.0983245999999998 | 2.9644742 | 2.413307762975302 | 1.206653881487651 |
| **Parallel** | 2 | 4 | 1.8681933999999998 | 1.0978906 | 2.9660842 | 2.4119978117950933 | 1.2059989058975467 |
| **Parallel** | 2 | 8 | 1.8709226 | 1.0960744 | 2.9669972000000002 | 2.4112555953878214 | 1.2056277976939107 |
| **Parallel** | 2 | 16 | 1.8695856 | 1.0993586 | 2.9689438 | 2.409674645912799 | 1.2048373229563996 |
| **Parallel** | 4 | 1 | 0.6471376 | 15.5539132 | 16.201051200000002 | 0.4415879260970423 | 0.11039698152426057 |
| **Parallel** | 4 | 2 | 0.6464604 | 12.0673058 | 12.713766 | 0.5627119926542615 | 0.14067799816356538 |
| **Parallel** | 4 | 4 | 0.6509688 | 11.957168 | 12.6081368 | 0.5674263147271689 | 0.14185657868179222 |
| **Parallel** | 4 | 8 | 0.6478364000000001 | 11.983995799999999 | 12.631832 | 0.566361918049575 | 0.14159047951239376 |
| **Parallel** | 4 | 16 | 0.6472222 | 12.151599399999998 | 12.7988216 | 0.5589724447756971 | 0.13974311119392427 |
| **Parallel** | 8 | 1 | 0.2934982 | 55.7621624 | 56.05566040000001 | 0.12762651530549088 | 0.01595331441318636 |
| **Parallel** | 8 | 2 | 0.2933952 | 34.648206599999995 | 34.941602 | 0.20474701188571717 | 0.025593376485714647 |
| **Parallel** | 8 | 4 | 0.298612 | 35.0996636 | 35.3982758 | 0.2021055669609761 | 0.02526319587012201 |
| **Parallel** | 8 | 8 | 0.2993162 | 34.5301598 | 34.829476 | 0.20540615081317903 | 0.02567576885164738 |
| **Parallel** | 8 | 16 | 0.3087523999999999 | 35.0826502 | 35.391402400000004 | 0.20214481808723123 | 0.025268102260903903 |

Tabella 16: Measurement with O1 optimization

# 5 Kosaraju's algorithm

Using the same parallelization method, we decide to parallelize one of the other proposed algorithm for finding the strongly connected components, and in particular the selected algorithm is the Kosaraju's algorithm.

Kosaraju's algorithm is a linear time algorithm, that makes use of the fact that the transpose graph (the same graph with the direction of every edge reversed) has exactly the same strongly connected components as the original graph. In this case, where strongly connected components are to be represented by appointing a separate root vertex for each component, and assigning to each vertex the root vertex of its component, then Kosaraju's algorithm can be stated as follows:

1. perform a depth-first search (DFS) on the graph, marking the finish times of each vertex as it is visited. This is done by starting at an arbitrary vertex and visiting all its unvisited neighbors recursively. The DFS should be done on the original graph, not its transpose
2. create a new graph that is the transpose of the original graph, where all edges are reversed. Perform another DFS on this new graph, but this time visit the vertices in decreasing order of their finish times from the first DFS. When visiting a vertex, add it to the current strongly connected component
3. repeat step 2 for all vertices that have not been visited yet
4. the strongly connected components are the subgraphs of connected vertices that have been found

Kosaraju's algorithm is efficient because it takes advantage of the fact that the strongly connected components in a graph are the strongly connected components in the graph's transpose. The first pass of DFS finds all the strongly connected components in the transpose of the original graph. The second pass of DFS uses the order of the stack from the first pass to find the strongly connected components in the original graph.

## 5.1 Kosaraju and Tarjan algorithm

The main difference between the two algorithms is how they find the strongly connected components.

Kosaraju's algorithm works as mentioned above. Tarjan's algorithm, on the other hand, uses a single depth-first search (DFS) to find the strongly connected components. It uses a stack to keep track of the nodes being visited and a low-link value for each node to determine whether a component has been found. The low-link value of a node is the smallest index of any node known to be reachable from that node, including the node itself.
Both algorithms are linear time complexity and have O(V+E) space complexity, where V is the number of nodes and E is the number of edges in the graph.

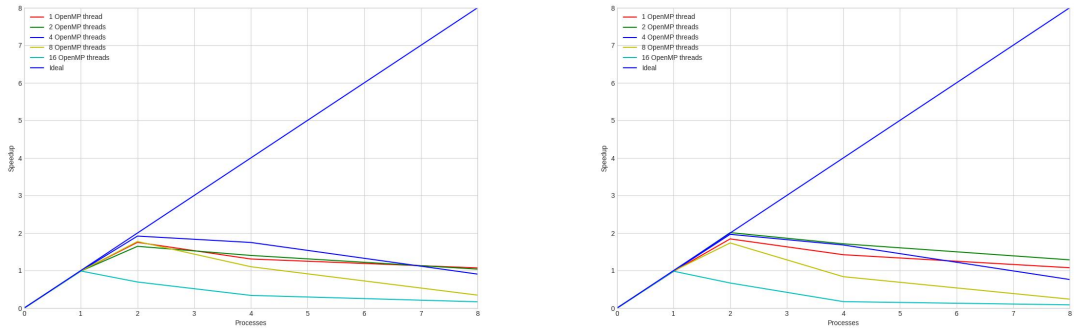
Another difference is that Kosaraju's algorithm is a two-pass algorithm and Tarjan's algorithm is a single-pass algorithm, which means Kosaraju's algorithm needs more memory space than Tarjan's algorithm. Furthermore, Kosaraju's algorithm is also more robust to graphs with negative cycles, while Tarjan's algorithm can fail in those cases.

## 5.2 Kosaraju's parallel version analysis

Also for Kosaraju, as done for the Tarjan algorithm, the analysis was conducted following the same structuring as the previously proposed case studies. So, subsequent analyses will be shown referring to the structuring given previously to those of Tarjan's algorithm. Therefore, we will also have here the 4 proposed case studies, which will now be presented.

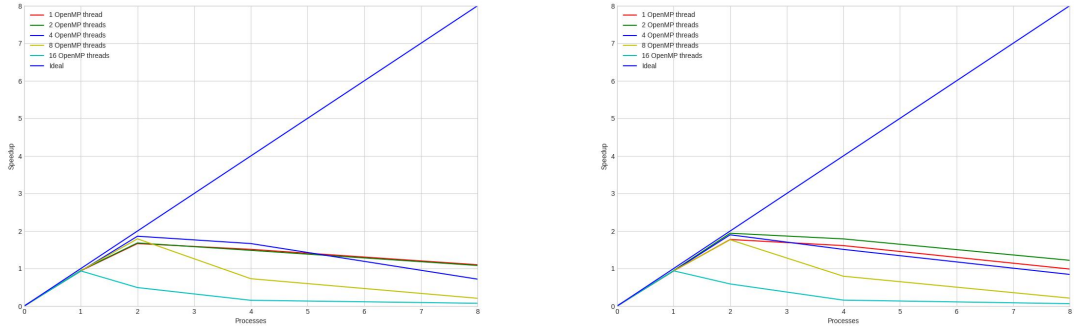
## 5.3 Case Study 1

### 5.3.1 500 vertices - 25% of density




(a) (b)

Figura 35: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



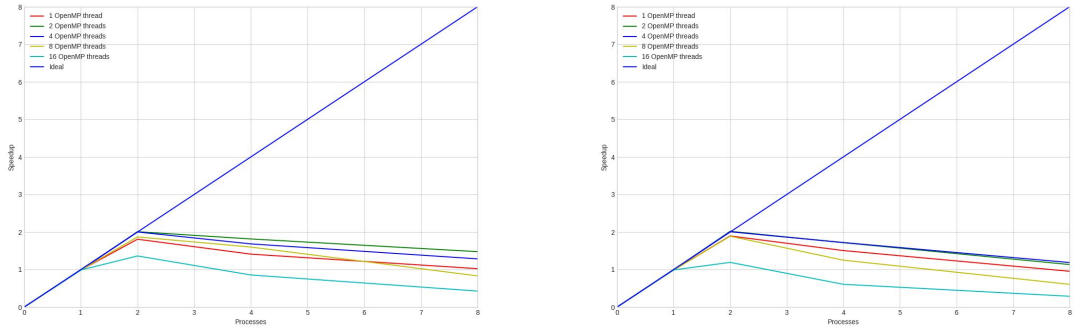

(a) (b)

Figura 36: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.0060886000000000004 | 0.0 | 0.0060886000000000004 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0062234000000000005 | 0.0 | 0.0062234000000000005 | 0.9783398142494456 | 0.9783398142494456 |
| **Parallel** | 1 | 2 | 0.0061952 | 0.0 | 0.0061952 | 0.9827931301652894 | 0.9827931301652894 |
| **Parallel** | 1 | 4 | 0.006185 | 0.0 | 0.006185 | 0.9844139046079224 | 0.9844139046079224 |
| **Parallel** | 1 | 8 | 0.006147000000000006 | 0.0 | 0.006147000000000006 | 0.9904994306165609 | 0.9904994306165609 |
| **Parallel** | 1 | 16 | 0.006196599999999999 | 0.0 | 0.006196599999999999 | 0.9825710873704937 | 0.9825710873704937 |
| **Parallel** | 2 | 1 | 0.0025136 | 0.0007882 | 0.0033021999999999995 | 1.84380110229544 | 0.92190055114772 |
| **Parallel** | 2 | 2 | 0.0025189999999999995 | 0.0005088 | 0.0030278 | 2.010899002576128 | 1.005449501288064 |
| **Parallel** | 2 | 4 | 0.0025296 | 0.000568 | 0.003097600000000003 | 1.9655862603305785 | 0.9827931301652892 |
| **Parallel** | 2 | 8 | 0.002648 | 0.0008596 | 0.003508 | 1.7356328392246296 | 0.8678164196123148 |
| **Parallel** | 2 | 16 | 0.002639 | 0.006472799999999999 | 0.0091118 | 0.6682104523804299 | 0.33410522619021493 |
| **Parallel** | 4 | 1 | 0.0015974 | 0.0026842000000000003 | 0.0042816 | 1.422038490284006 | 0.3555096225710015 |
| **Parallel** | 4 | 2 | 0.0014303999999999999 | 0.002124 | 0.0035546 | 1.712879086254431 | 0.42821977156360774 |
| **Parallel** | 4 | 4 | 0.0015991999999999998 | 0.002022 | 0.003621400000000003 | 1.6812834815264814 | 0.42032087038162036 |
| **Parallel** | 4 | 8 | 0.0016691999999999998 | 0.0056045999999999995 | 0.0072737999999999995 | 0.8370590337925157 | 0.20926475844812892 |
| **Parallel** | 4 | 16 | 0.0015696000000000002 | 0.033775999999999994 | 0.0353458 | 0.17225809007010737 | 0.04306452251752684 |
| **Parallel** | 8 | 1 | 0.0009876 | 0.0046728 | 0.0056602 | 1.0756863715063074 | 0.13446079643828843 |
| **Parallel** | 8 | 2 | 0.0009914 | 0.0037433999999999996 | 0.0047348 | 1.285925487876996 | 0.1607406859846245 |
| **Parallel** | 8 | 4 | 0.0010523999999999998 | 0.006940399999999999 | 0.007992599999999999 | 0.7617796461727099 | 0.09522245577158873 |
| **Parallel** | 8 | 8 | 0.0010221999999999998 | 0.0244956 | 0.0255178 | 0.23860207384649149 | 0.029825259230811436 |
| **Parallel** | 8 | 16 | 0.0009768 | 0.06928419999999999 | 0.07026099999999999 | 0.08665689358249956 | 0.010832111697812445 |

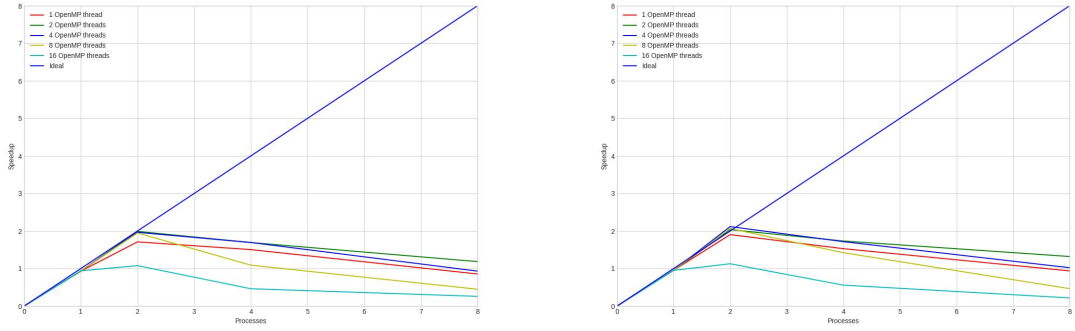
Tabella 17: Measurement with O1 optimization

### 5.3.2 500 vertices - 50% of density




(a) (b)

Figura 37: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



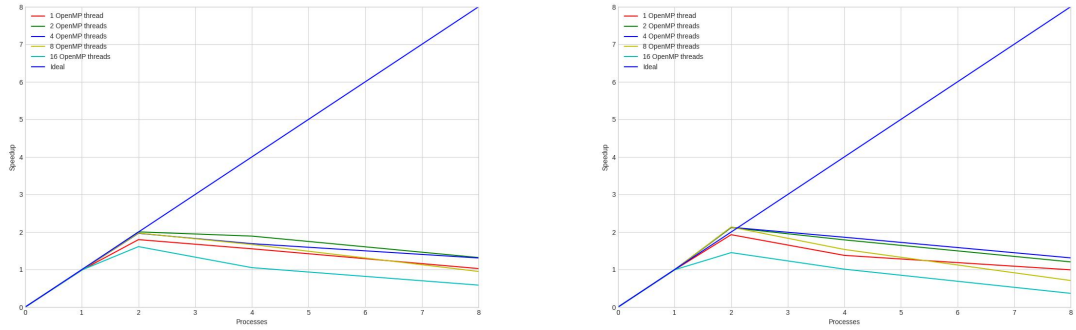

(a) (b)

Figura 38: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.0314408 | 0.0 | 0.0314408 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.031837199999999996 | 0.0 | 0.031837199999999996 | 0.9875491563328433 | 0.9875491563328433 |
| **Parallel** | 1 | 2 | 0.031842999999999996 | 0.0 | 0.031842999999999996 | 0.9873692805326132 | 0.9873692805326132 |
| **Parallel** | 1 | 4 | 0.0318494 | 0.0 | 0.0318494 | 0.987170872920683 | 0.987170872920683 |
| **Parallel** | 1 | 8 | 0.031863800000000005 | 0.0 | 0.031863800000000005 | 0.9867247472052924 | 0.9867247472052924 |
| **Parallel** | 1 | 16 | 0.03185060000000001 | 0.0 | 0.03185060000000001 | 0.9871336803702282 | 0.9871336803702282 |
| **Parallel** | 2 | 1 | 0.0136764 | 0.0037102000000000003 | 0.0173866 | 1.8083351546593354 | 0.9041675773296677 |
| **Parallel** | 2 | 2 | 0.013685600000000001 | 0.0019702000000000005 | 0.0156556 | 2.0082781879966274 | 1.0041390939983137 |
| **Parallel** | 2 | 4 | 0.0136778 | 0.0020312000000000004 | 0.0157088 | 2.0014768792014666 | 1.0007384396007333 |
| **Parallel** | 2 | 8 | 0.014525399999999999 | 0.0022898000000000003 | 0.016815200000000002 | 1.8697844807079305 | 0.9348922403539652 |
| **Parallel** | 2 | 16 | 0.014316599999999999 | 0.0087574 | 0.023073999999999997 | 1.362607263586721 | 0.6813036317933605 |
| **Parallel** | 4 | 1 | 0.0090662 | 0.0132152 | 0.0222812 | 1.4110909645800045 | 0.3527727411450011 |
| **Parallel** | 4 | 2 | 0.008273399999999998 | 0.0090424 | 0.017315999999999998 | 1.8157080157080159 | 0.45392700392700397 |
| **Parallel** | 4 | 4 | 0.009501800000000001 | 0.009170599999999998 | 0.0186724 | 1.683811400784045 | 0.42095285019601125 |
| **Parallel** | 4 | 8 | 0.008631400000000001 | 0.0110262 | 0.019657600000000004 | 1.59942210646264 | 0.39985552661566 |
| **Parallel** | 4 | 16 | 0.0086538 | 0.0280568 | 0.0367108 | 0.8564455146714317 | 0.21411137866785793 |
| **Parallel** | 8 | 1 | 0.0051524000000000005 | 0.0255768 | 0.030708999999999997 | 1.0238301475137581 | 0.12797876843921976 |
| **Parallel** | 8 | 2 | 0.0047246 | 0.016563199999999997 | 0.021287400000000005 | 1.4769675958548245 | 0.18462094948185306 |
| **Parallel** | 8 | 4 | 0.0055 | 0.018961600000000002 | 0.0244618 | 1.2853019810480013 | 0.16066274763100016 |
| **Parallel** | 8 | 8 | 0.005852 | 0.031959400000000006 | 0.03781140000000001 | 0.8315164209735686 | 0.10393955262169607 |
| **Parallel** | 8 | 16 | 0.0048398 | 0.06882479999999999 | 0.0736646 | 0.4268101639050507 | 0.05335127048813134 |

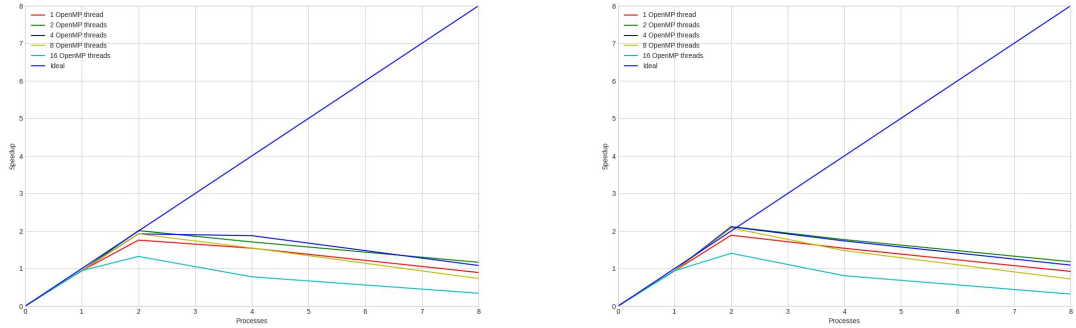
Tabella 18: Measurement with O0 optimization

### 5.3.3 500 vertices - 75% of density




(a) (b)

Figura 39: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization




(a) (b)

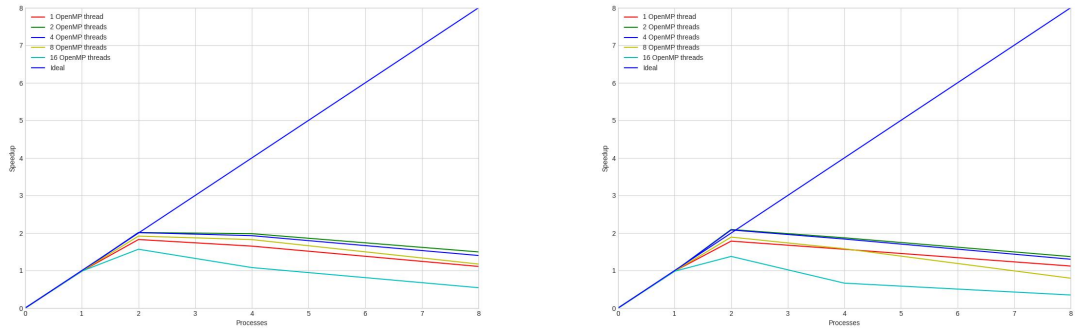
Figura 40: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.0285124 | 0.0 | 0.0285124 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.028688200000000004 | 0.0 | 0.028688200000000004 | 0.9938720449522799 | 0.9938720449522799 |
| **Parallel** | 1 | 2 | 0.028771800000000004 | 0.0 | 0.028771800000000004 | 0.9909842276117586 | 0.9909842276117586 |
| **Parallel** | 1 | 4 | 0.0286378 | 0.0 | 0.0286378 | 0.9956211720174035 | 0.9956211720174035 |
| **Parallel** | 1 | 8 | 0.028728999999999998 | 0.0 | 0.028728999999999998 | 0.9924605799018414 | 0.9924605799018414 |
| **Parallel** | 1 | 16 | 0.0286668 | 0.0 | 0.0286668 | 0.9946139785396347 | 0.9946139785396347 |
| **Parallel** | 2 | 1 | 0.011406 | 0.003373 | 0.014779200000000001 | 1.9292248565551584 | 0.9646124282775792 |
| **Parallel** | 2 | 2 | 0.0116682 | 0.0018 | 0.0134682 | 2.1170163793231462 | 1.0585081896615731 |
| **Parallel** | 2 | 4 | 0.011565599999999999 | 0.0018482 | 0.013413999999999999 | 2.1255702996868946 | 1.0627851498434473 |
| **Parallel** | 2 | 8 | 0.0114146 | 0.001961 | 0.013375599999999998 | 2.131672597864769 | 1.0658362989323844 |
| **Parallel** | 2 | 16 | 0.011308599999999999 | 0.0083376 | 0.019646000000000004 | 1.4513081543316704 | 0.7256540771658352 |
| **Parallel** | 4 | 1 | 0.007363 | 0.013323199999999999 | 0.020686 | 1.3783428405685005 | 0.34458571014212513 |
| **Parallel** | 4 | 2 | 0.0060934000000000006 | 0.009825200000000001 | 0.015918599999999998 | 1.7911374115814207 | 0.4477843528953552 |
| **Parallel** | 4 | 4 | 0.005730600000000001 | 0.009589199999999999 | 0.015319600000000003 | 1.8611713099558733 | 0.46529282748896833 |
| **Parallel** | 4 | 8 | 0.0064860000000000004 | 0.012080200000000003 | 0.018566399999999997 | 1.5356988969320926 | 0.38392472423302315 |
| **Parallel** | 4 | 16 | 0.0062342000000000005 | 0.0219888 | 0.0282228 | 1.0102612072508752 | 0.2525653018127188 |
| **Parallel** | 8 | 1 | 0.0033542000000000003 | 0.025358600000000002 | 0.028713 | 0.9930136175251628 | 0.12412670219064535 |
| **Parallel** | 8 | 2 | 0.004 | 0.0196726 | 0.0236724 | 1.2044575117013907 | 0.15055718896267384 |
| **Parallel** | 8 | 4 | 0.0033686000000000002 | 0.018365199999999998 | 0.021733799999999998 | 1.3118920759370198 | 0.16398650949212748 |
| **Parallel** | 8 | 8 | 0.0036606 | 0.0365864 | 0.04024679999999999 | 0.7084389317908506 | 0.08855486647385633 |
| **Parallel** | 8 | 16 | 0.004942 | 0.0728564 | 0.07779820000000001 | 0.3664917697324616 | 0.0458114712165577 |

Tabella 19: Measurement with O1 optimization

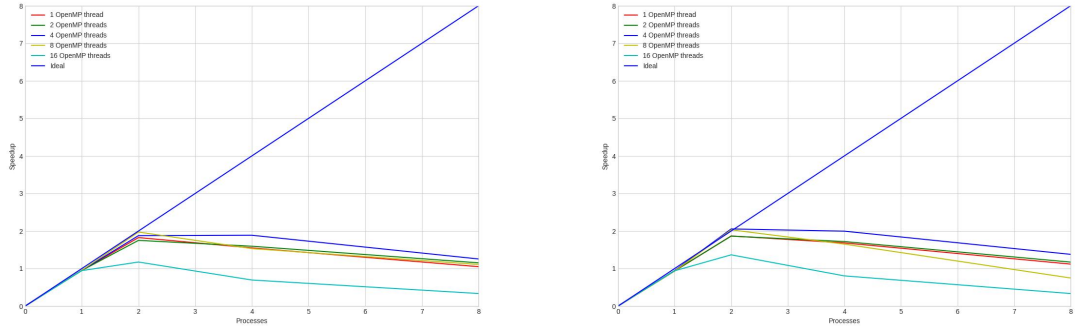
## 5.4 Case Study 2

### 5.4.1 1000 vertices - 25% of density




(a) (b)

Figura 41: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



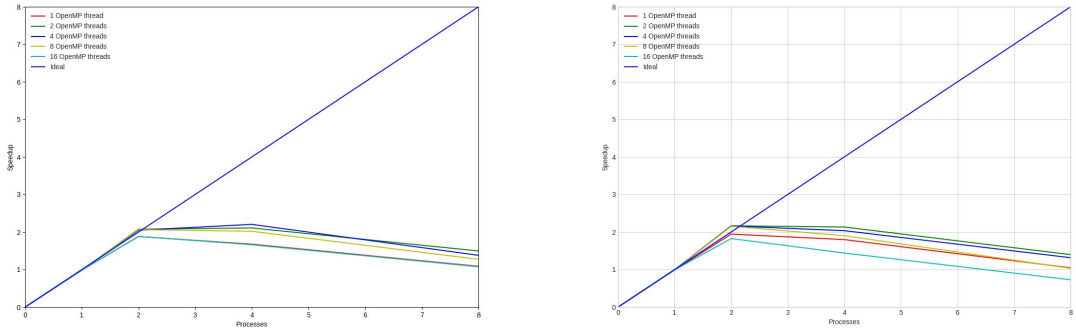

(a) (b)

Figura 42: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.025509999999999998 | 0.0 | 0.025509999999999998 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0258718 | 0.0 | 0.0258718 | 0.9860156618403048 | 0.9860156618403048 |
| **Parallel** | 1 | 2 | 0.0258802 | 0.0 | 0.0258802 | 0.9856956283181737 | 0.9856956283181737 |
| **Parallel** | 1 | 4 | 0.0259158 | 0.0 | 0.0259158 | 0.9843415985614953 | 0.9843415985614953 |
| **Parallel** | 1 | 8 | 0.0259386 | 0.0 | 0.0259386 | 0.9834763634120577 | 0.9834763634120577 |
| **Parallel** | 1 | 16 | 0.025985600000000005 | 0.0 | 0.025985600000000005 | 0.9816975555692381 | 0.9816975555692381 |
| **Parallel** | 2 | 1 | 0.0113484 | 0.002946 | 0.014294199999999998 | 1.7846399238852122 | 0.8923199619426061 |
| **Parallel** | 2 | 2 | 0.0105888 | 0.0015926 | 0.012181399999999998 | 2.094176367248428 | 1.047088183624214 |
| **Parallel** | 2 | 4 | 0.0106054 | 0.001649 | 0.0122546 | 2.0816672922820816 | 1.0408336461410408 |
| **Parallel** | 2 | 8 | 0.0111108 | 0.0023740000000000002 | 0.013484399999999999 | 1.8918157278039809 | 0.9459078639019904 |
| **Parallel** | 2 | 16 | 0.011133399999999998 | 0.0073924 | 0.018525999999999997 | 1.3769836985857715 | 0.6884918492928858 |
| **Parallel** | 4 | 1 | 0.0061832 | 0.0100936 | 0.0162768 | 1.5672613781578686 | 0.39181534453946715 |
| **Parallel** | 4 | 2 | 0.006161399999999999 | 0.0074576 | 0.013619000000000001 | 1.8731184374770538 | 0.46827960936926344 |
| **Parallel** | 4 | 4 | 0.0064834 | 0.0073812 | 0.0138646 | 1.8399376830200653 | 0.45998442075501633 |
| **Parallel** | 4 | 8 | 0.0060988 | 0.010029399999999999 | 0.016128200000000002 | 1.5817016158033752 | 0.3954254039508438 |
| **Parallel** | 4 | 16 | 0.0057426 | 0.032633199999999994 | 0.038376 | 0.6647383781530123 | 0.16618459453825307 |
| **Parallel** | 8 | 1 | 0.0036035999999999998 | 0.019167 | 0.022770999999999996 | 1.1202845724825437 | 0.14003557156031796 |
| **Parallel** | 8 | 2 | 0.0039162 | 0.014674199999999998 | 0.018590600000000002 | 1.372198853183867 | 0.17152485664798336 |
| **Parallel** | 8 | 4 | 0.0036851999999999996 | 0.0159148 | 0.019600199999999998 | 1.3015173314558015 | 0.1626896664319752 |
| **Parallel** | 8 | 8 | 0.0042654 | 0.027778800000000003 | 0.032044199999999995 | 0.7960879035831758 | 0.09951098794789698 |
| **Parallel** | 8 | 16 | 0.0047254 | 0.06827659999999999 | 0.0730018 | 0.34944343838097136 | 0.04368042979762142 |

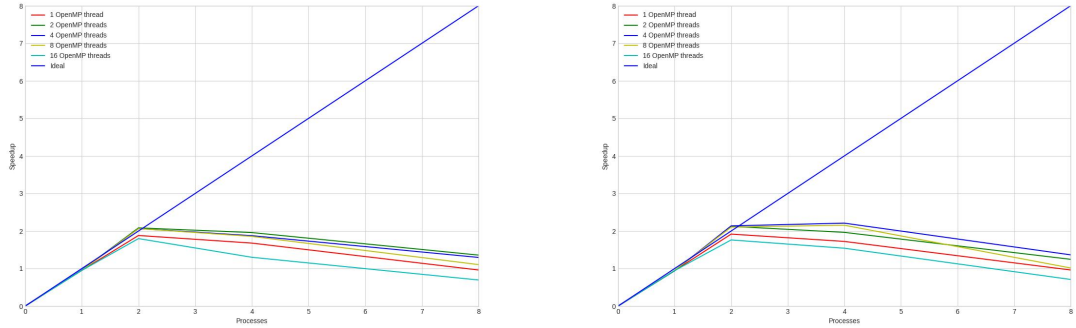
Tabella 20: Measurement with O1 optimization

### 5.4.2 1000 vertices - 50% of density




(a) (b)

Figura 43: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



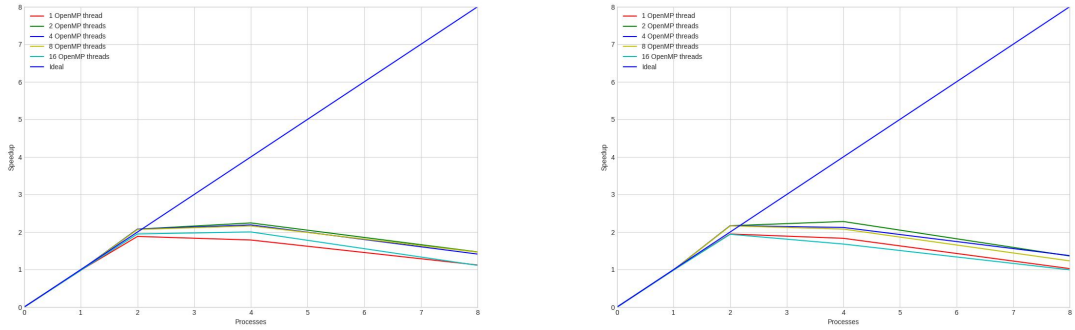

(a) (b)

Figura 44: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.07762240000000001 | 0.0 | 0.07762240000000001 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.07864420000000001 | 0.0 | 0.07864420000000001 | 0.9870073063239246 | 0.9870073063239246 |
| **Parallel** | 1 | 2 | 0.07837000000000001 | 0.0 | 0.07837000000000001 | 0.9904606354472375 | 0.9904606354472375 |
| **Parallel** | 1 | 4 | 0.0782384 | 0.0 | 0.0782384 | 0.9921266283564082 | 0.9921266283564082 |
| **Parallel** | 1 | 8 | 0.07872660000000001 | 0.0 | 0.07872660000000001 | 0.9859742450455119 | 0.9859742450455119 |
| **Parallel** | 1 | 16 | 0.078431 | 0.0 | 0.078431 | 0.98969030102893 | 0.98969030102893 |
| **Parallel** | 2 | 1 | 0.031869400000000006 | 0.008047799999999999 | 0.0399172 | 1.944585291553516 | 0.972292645776758 |
| **Parallel** | 2 | 2 | 0.031666 | 0.0041818 | 0.0358478 | 2.165332321648749 | 1.0826661608243744 |
| **Parallel** | 2 | 4 | 0.03168220000000001 | 0.0042282000000000005 | 0.035910399999999995 | 2.1615576546070225 | 1.0807788273035113 |
| **Parallel** | 2 | 8 | 0.03170920000000001 | 0.0043289999999999995 | 0.036038600000000004 | 2.1538683522667363 | 1.0769341761333682 |
| **Parallel** | 2 | 16 | 0.0317238 | 0.010787 | 0.042511200000000006 | 1.8259282259733904 | 0.9129641129866952 |
| **Parallel** | 4 | 1 | 0.015341000000000002 | 0.027827599999999997 | 0.043168200000000004 | 1.7981384445031297 | 0.4495346111257824 |
| **Parallel** | 4 | 2 | 0.0142398 | 0.0221474 | 0.036386999999999996 | 2.1332453898370303 | 0.5333113474592576 |
| **Parallel** | 4 | 4 | 0.0154258 | 0.0227044 | 0.038130199999999996 | 2.035719718228596 | 0.508929929557149 |
| **Parallel** | 4 | 8 | 0.0150528 | 0.025766 | 0.040818600000000003 | 1.9016428784916681 | 0.47541071962291703 |
| **Parallel** | 4 | 16 | 0.0154336 | 0.0384456 | 0.0538794 | 1.4406693467262073 | 0.3601673366815518 |
| **Parallel** | 8 | 1 | 0.008030200000000001 | 0.0661454 | 0.0741756 | 1.0464681108073277 | 0.13080851385091596 |
| **Parallel** | 8 | 2 | 0.009586 | 0.0459496 | 0.0555358 | 1.397700222199014 | 0.17471252777487675 |
| **Parallel** | 8 | 4 | 0.010313 | 0.048668600000000006 | 0.05898159999999999 | 1.3160443256880117 | 0.16450554071100146 |
| **Parallel** | 8 | 8 | 0.0103296 | 0.06484340000000001 | 0.075173 | 1.0325835073763188 | 0.12907293842203985 |
| **Parallel** | 8 | 16 | 0.010220199999999999 | 0.0963814 | 0.1066018 | 0.7281528079263203 | 0.09101910099079004 |

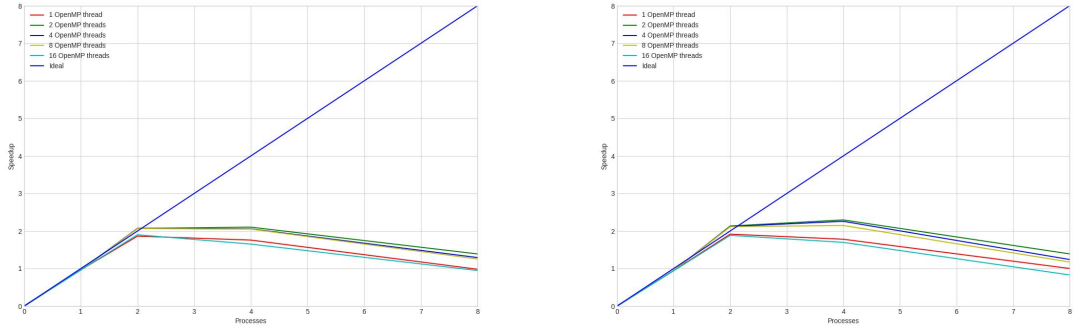
Tabella 21: Measurement with O1 optimization

### 5.4.3 1000 vertices - 75% of density




(a) (b)

Figura 45: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



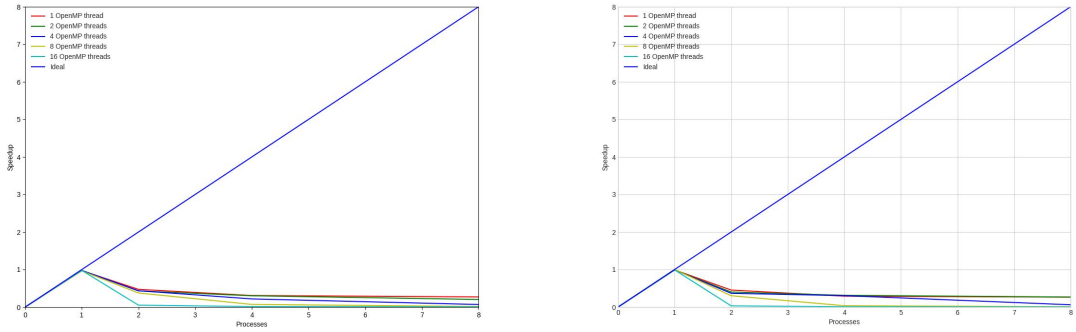

(a) (b)

Figura 46: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.2386202 | 0.0 | 0.2386202 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.2427738 | 0.0 | 0.2427738 | 0.982891069794187 | 0.982891069794187 |
| **Parallel** | 1 | 2 | 0.24257420000000002 | 0.0 | 0.24257420000000002 | 0.9836998328758787 | 0.9836998328758787 |
| **Parallel** | 1 | 4 | 0.24308760000000001 | 0.0 | 0.24308760000000001 | 0.9816222629208565 | 0.9816222629208565 |
| **Parallel** | 1 | 8 | 0.2428168 | 0.0 | 0.2428168 | 0.9827170113435314 | 0.9827170113435314 |
| **Parallel** | 1 | 16 | 0.24290859999999997 | 0.0 | 0.24290859999999997 | 0.982345623003879 | 0.982345623003879 |
| **Parallel** | 2 | 1 | 0.10291059999999999 | 0.0238158 | 0.1267262 | 1.8829586936245226 | 0.9414793468122613 |
| **Parallel** | 2 | 2 | 0.10277500000000002 | 0.0121438 | 0.114919 | 2.0764207833343487 | 1.0382103916671743 |
| **Parallel** | 2 | 4 | 0.1027398 | 0.0121748 | 0.11491480000000001 | 2.0764966740576494 | 1.0382483370288247 |
| **Parallel** | 2 | 8 | 0.10272060000000001 | 0.012703399999999998 | 0.1154238 | 2.067339664783173 | 1.0336698323915865 |
| **Parallel** | 2 | 16 | 0.1026314 | 0.0197984 | 0.1224296 | 1.9490401014133838 | 0.9745200507066919 |
| **Parallel** | 4 | 1 | 0.052272 | 0.0812218 | 0.133494 | 1.7874975654336525 | 0.4468743913584131 |
| **Parallel** | 4 | 2 | 0.052300400000000004 | 0.0541194 | 0.10642 | 2.242249577147153 | 0.5605623942867882 |
| **Parallel** | 4 | 4 | 0.052181200000000004 | 0.05682280000000001 | 0.10900399999999999 | 2.1890958129976883 | 0.5472739532494221 |
| **Parallel** | 4 | 8 | 0.05245299999999999 | 0.0579074 | 0.11036019999999999 | 2.162194341800758 | 0.5405485854501895 |
| **Parallel** | 4 | 16 | 0.052837800000000004 | 0.066348 | 0.11918579999999998 | 2.0020858189482307 | 0.5005214547370577 |
| **Parallel** | 8 | 1 | 0.0329756 | 0.17973060000000002 | 0.2127062 | 1.1218300171786248 | 0.1402287521473281 |
| **Parallel** | 8 | 2 | 0.03771819999999999 | 0.12458559999999999 | 0.1623038 | 1.4702071054405381 | 0.18377588818006727 |
| **Parallel** | 8 | 4 | 0.0385352 | 0.1303324 | 0.1688674 | 1.4130625567753161 | 0.17663281959691451 |
| **Parallel** | 8 | 8 | 0.032504599999999995 | 0.1304674 | 0.16297199999999998 | 1.4641791227941 | 0.1830223903492625 |
| **Parallel** | 8 | 16 | 0.03300379999999999 | 0.1816046 | 0.21460819999999997 | 1.1118876165961973 | 0.13898595207452466 |

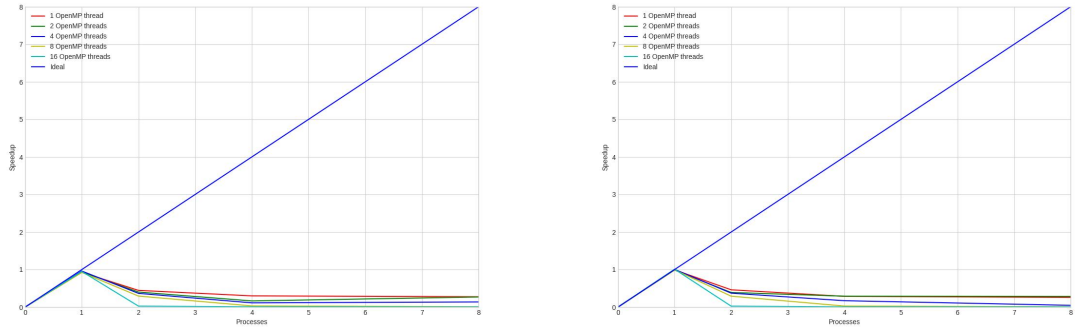
Tabella 22: Measurement with O0 optimization

### 5.4.4 1000 vertices - Special Cases - 0%




(a) (b)

Figura 47: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



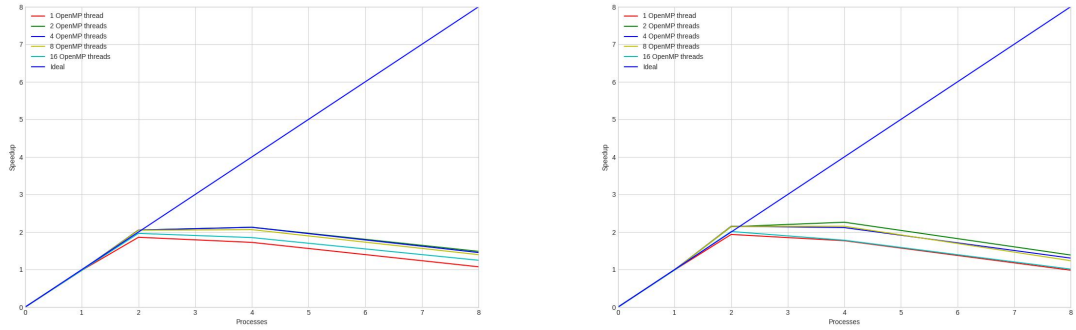

(a) (b)

Figura 48: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.000192 | 0.0 | 0.000192 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0001936 | 0.0 | 0.0001936 | 0.9917355371900827 | 0.9917355371900827 |
| **Parallel** | 1 | 2 | 0.0001944 | 0.0 | 0.0001944 | 0.9876543209876543 | 0.9876543209876543 |
| **Parallel** | 1 | 4 | 0.00019099999999999998 | 0.0 | 0.00019099999999999998 | 1.005235602094241 | 1.005235602094241 |
| **Parallel** | 1 | 8 | 0.0001934 | 0.0 | 0.0001934 | 0.9927611168562565 | 0.9927611168562565 |
| **Parallel** | 1 | 16 | 0.00019020000000000002 | 0.0 | 0.00019020000000000002 | 1.0094637223974763 | 1.0094637223974763 |
| **Parallel** | 2 | 1 | 0.0003372 | 7.92e-05 | 0.0004168 | 0.46065259117082535 | 0.23032629558541268 |
| **Parallel** | 2 | 2 | 0.000363 | 0.00012880000000000001 | 0.0004921999999999999 | 0.39008533116619265 | 0.19504266558309633 |
| **Parallel** | 2 | 4 | 0.0003464 | 0.00017040000000000002 | 0.0005170000000000001 | 0.37137330754352027 | 0.18568665377176014 |
| **Parallel** | 2 | 8 | 0.0003788 | 0.00028059999999999994 | 0.0006594 | 0.29117379435850776 | 0.14558689717925388 |
| **Parallel** | 2 | 16 | 0.00036839999999999996 | 0.0071778 | 0.007546199999999999 | 0.025443269460125633 | 0.012721634730062817 |
| **Parallel** | 4 | 1 | 0.0005068 | 0.00016340000000000001 | 0.0006704 | 0.2863961813842482 | 0.07159904534606205 |
| **Parallel** | 4 | 2 | 0.0004921999999999999 | 0.0001686 | 0.0006606 | 0.29064486830154407 | 0.07266121707538602 |
| **Parallel** | 4 | 4 | 0.00046540000000000004 | 0.0006612 | 0.0011268 | 0.1703940362087327 | 0.042598509052183174 |
| **Parallel** | 4 | 8 | 0.00045620000000000003 | 0.006476999999999999 | 0.0069320000000000005 | 0.027692840246927825 | 0.006923210061731956 |
| **Parallel** | 4 | 16 | 0.0005022 | 0.034721 | 0.0352234 | 0.005450921830374126 | 0.0013627304575935316 |
| **Parallel** | 8 | 1 | 0.0005362 | 0.0002056 | 0.0007417999999999999 | 0.2588298732812079 | 0.032353734160150985 |
| **Parallel** | 8 | 2 | 0.0004816 | 0.0001944 | 0.0006760000000000001 | 0.28402366863905326 | 0.03550295857988166 |
| **Parallel** | 8 | 4 | 0.0005158000000000001 | 0.0035435999999999996 | 0.0040592 | 0.047299960583366184 | 0.005912495072920773 |
| **Parallel** | 8 | 8 | 0.000475 | 0.0271712 | 0.027646399999999998 | 0.006944846345274611 | 0.0008681057931593264 |
| **Parallel** | 8 | 16 | 0.0004778 | 0.0603056 | 0.060783599999999999 | 0.003158746767220106 | 0.0003948434590251326 |

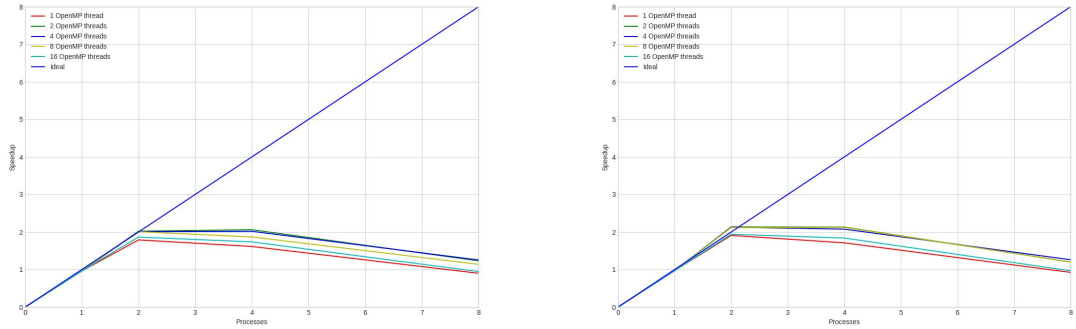
Tabella 23: Measurement with O1 optimization

### 5.4.5 1000 vertices - Special Cases - 100%




(a) (b)

Figura 49: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization




(a) (b)

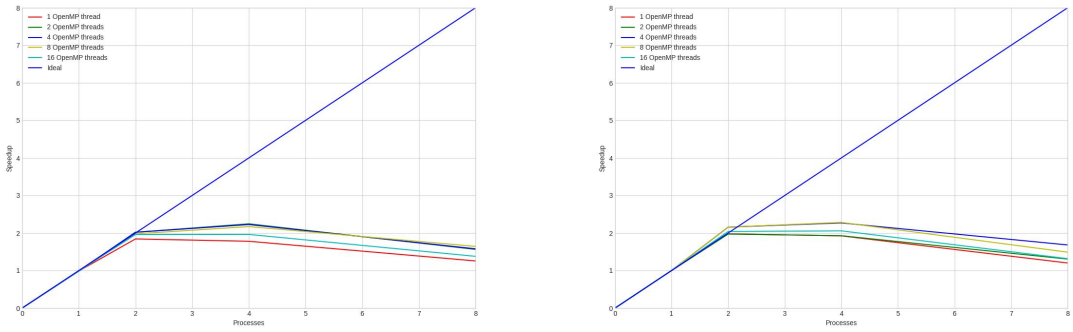
Figura 50: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.20019 | 0.0 | 0.20019 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.20231719999999997 | 0.0 | 0.20231719999999997 | 0.9894858173205245 | 0.9894858173205245 |
| **Parallel** | 1 | 2 | 0.201967 | 0.0 | 0.201967 | 0.9912015329236955 | 0.9912015329236955 |
| **Parallel** | 1 | 4 | 0.2016072 | 0.0 | 0.2016072 | 0.9929704891491972 | 0.9929704891491972 |
| **Parallel** | 1 | 8 | 0.2019534 | 0.0 | 0.2019534 | 0.9912682826830348 | 0.9912682826830348 |
| **Parallel** | 1 | 16 | 0.20206280000000004 | 0.0 | 0.20206280000000004 | 0.9907315943360181 | 0.9907315943360181 |
| **Parallel** | 2 | 1 | 0.08221120000000001 | 0.0212174 | 0.1034284 | 1.9355418821136168 | 0.9677709410568084 |
| **Parallel** | 2 | 2 | 0.0826932 | 0.010768999999999999 | 0.09346220000000001 | 2.141935456259322 | 1.070967728129661 |
| **Parallel** | 2 | 4 | 0.0820808 | 0.0108172 | 0.0928976 | 2.1549534110676705 | 1.0774767055338352 |
| **Parallel** | 2 | 8 | 0.0822054 | 0.010928000000000002 | 0.09313339999999999 | 2.149497387618191 | 1.0747486938090955 |
| **Parallel** | 2 | 16 | 0.08263020000000001 | 0.0167954 | 0.09942619999999999 | 2.0134531944296374 | 1.0067265972148187 |
| **Parallel** | 4 | 1 | 0.03675 | 0.076424 | 0.11317379999999999 | 1.7688723008328784 | 0.4422180752082196 |
| **Parallel** | 4 | 2 | 0.036967 | 0.0515588 | 0.0885256 | 2.261379759075341 | 0.5653449397688353 |
| **Parallel** | 4 | 4 | 0.0369992 | 0.057527400000000006 | 0.0945268 | 2.1178120913857237 | 0.5294530228464309 |
| **Parallel** | 4 | 8 | 0.0372234 | 0.0557446 | 0.09296800000000001 | 2.15332157301437 | 0.5383303932535926 |
| **Parallel** | 4 | 16 | 0.0373812 | 0.07500359999999999 | 0.11238479999999999 | 1.7812907083520193 | 0.4453226770880048 |
| **Parallel** | 8 | 1 | 0.022938399999999998 | 0.18067060000000001 | 0.20360899999999998 | 0.9832080114336794 | 0.12290100142920993 |
| **Parallel** | 8 | 2 | 0.0211616 | 0.1231074 | 0.1442686 | 1.3876200365152225 | 0.1734525045644028 |
| **Parallel** | 8 | 4 | 0.0221332 | 0.1311654 | 0.1532986 | 1.3058827673573015 | 0.16323534591966268 |
| **Parallel** | 8 | 8 | 0.0223742 | 0.13951980000000003 | 0.161894 | 1.236549841254154 | 0.15456873015676925 |
| **Parallel** | 8 | 16 | 0.0239474 | 0.17379280000000003 | 0.1977404 | 1.0123879591626193 | 0.1265484948953274 |

Tabella 24: Measurement with O1 optimization

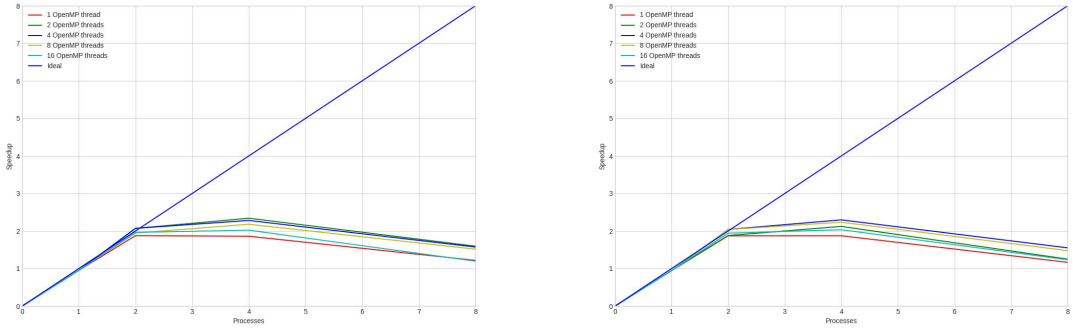
## 5.5 Case Study 3

### 5.5.1 2500 vertices - 25% of density




(a) (b)

Figura 51: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



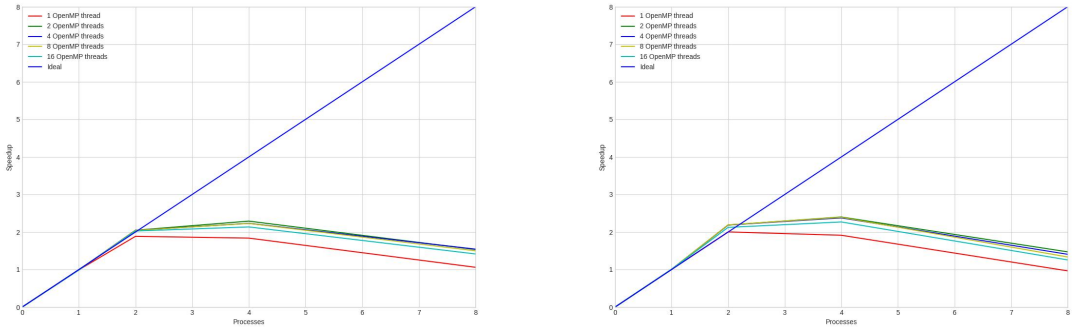

(a) (b)

Figura 52: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.18749880000000002 | 0.0 | 0.18749880000000002 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.18718059999999997 | 0.0 | 0.18718059999999997 | 1.0016999624961136 | 1.0016999624961136 |
| **Parallel** | 1 | 2 | 0.187158 | 0.0 | 0.187158 | 1.0018209213605618 | 1.0018209213605618 |
| **Parallel** | 1 | 4 | 0.1870102 | 0.0 | 0.1870102 | 1.002612691714142 | 1.002612691714142 |
| **Parallel** | 1 | 8 | 0.18710180000000004 | 0.0 | 0.18710180000000004 | 1.0021218395547236 | 1.0021218395547236 |
| **Parallel** | 1 | 16 | 0.18725679999999997 | 0.0 | 0.18725679999999997 | 1.0012923429215925 | 1.0012923429215925 |
| **Parallel** | 2 | 1 | 0.07759 | 0.0171842 | 0.0947742 | 1.9783738612407176 | 0.9891869306203588 |
| **Parallel** | 2 | 2 | 0.0778402 | 0.0172546 | 0.09509480000000001 | 1.971704025877335 | 0.9858520129386675 |
| **Parallel** | 2 | 4 | 0.07800860000000001 | 0.0089186 | 0.08692720000000001 | 2.156963528101676 | 1.078481764050838 |
| **Parallel** | 2 | 8 | 0.0777244 | 0.0091698 | 0.0868944 | 2.157777716400597 | 1.0788888582002985 |
| **Parallel** | 2 | 16 | 0.0777668 | 0.014113599999999999 | 0.0918804 | 2.0406833231026424 | 1.0203416615513212 |
| **Parallel** | 4 | 1 | 0.0376676 | 0.05976240000000001 | 0.09742999999999999 | 1.9244462691162891 | 0.4811115672790723 |
| **Parallel** | 4 | 2 | 0.037361000000000005 | 0.0598828 | 0.09724000000000001 | 1.9281271852247954 | 0.48203179630619886 |
| **Parallel** | 4 | 4 | 0.0376352 | 0.0449994 | 0.08263480000000001 | 2.2690053101114787 | 0.5672513275278697 |
| **Parallel** | 4 | 8 | 0.037739 | 0.0443926 | 0.0821316 | 2.282906944464737 | 0.5707267361161843 |
| **Parallel** | 4 | 16 | 0.0389298 | 0.0521706 | 0.0911006 | 2.0581510988950678 | 0.5145377747237669 |
| **Parallel** | 8 | 1 | 0.0234846 | 0.1325586 | 0.1560434 | 1.201581098591802 | 0.15019763732397526 |
| **Parallel** | 8 | 2 | 0.0217792 | 0.12176959999999999 | 0.1435488 | 1.3061676586638133 | 0.16327095733297667 |
| **Parallel** | 8 | 4 | 0.02288 | 0.08858239999999999 | 0.11146260000000001 | 1.6821678302856744 | 0.2102709787857093 |
| **Parallel** | 8 | 8 | 0.024355599999999998 | 0.1014478 | 0.1258032 | 1.4904135983822353 | 0.1863016997977794 |
| **Parallel** | 8 | 16 | 0.0236308 | 0.1187376 | 0.1423682 | 1.3169991613295666 | 0.16462489516619583 |

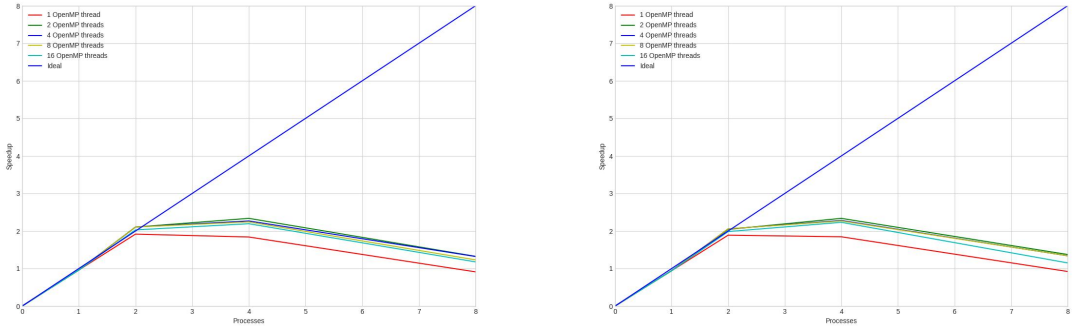
Tabella 25: Measurement with O1 optimization

### 5.5.2 2500 vertices - 50% of density




(a) (b)

Figura 53: (c) Speedup with no optimization - (d) Speedup with $O1$ optimization



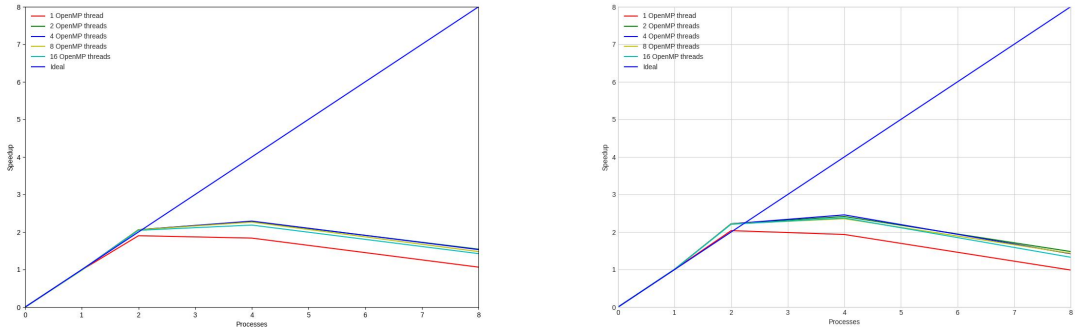

(a) (b)

Figura 54: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.6142852000000001 | 0.0 | 0.6142852000000001 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.6102293999999999 | 0.0 | 0.6102293999999999 | 1.0066463529944643 | 1.0066463529944643 |
| **Parallel** | 1 | 2 | 0.610177 | 0.0 | 0.610177 | 1.0067328004824831 | 1.0067328004824831 |
| **Parallel** | 1 | 4 | 0.6103080000000001 | 0.0 | 0.6103080000000001 | 1.0065167095958107 | 1.0065167095958107 |
| **Parallel** | 1 | 8 | 0.6089992 | 0.0 | 0.6089992 | 1.0086798143577203 | 1.0086798143577203 |
| **Parallel** | 1 | 16 | 0.6101135999999999 | 0.0 | 0.6101135999999999 | 1.006837415196121 | 1.006837415196121 |
| **Parallel** | 2 | 1 | 0.2541324 | 0.052635999999999995 | 0.3067686 | 2.0024383199584315 | 1.0012191599792157 |
| **Parallel** | 2 | 2 | 0.2541982 | 0.026652600000000005 | 0.2808502 | 2.1872343334631776 | 1.0936171667315888 |
| **Parallel** | 2 | 4 | 0.25422979999999995 | 0.0266568 | 0.2808864 | 2.1869524476799165 | 1.0934762238399582 |
| **Parallel** | 2 | 8 | 0.253667 | 0.0267882 | 0.2804554 | 2.1903133261117453 | 1.0951566630558727 |
| **Parallel** | 2 | 16 | 0.2542532 | 0.0350288 | 0.28928200000000004 | 2.12348227680948 | 1.06174113840474 |
| **Parallel** | 4 | 1 | 0.1193104 | 0.20163540000000002 | 0.3209458 | 1.913984230359145 | 0.47849605758978625 |
| **Parallel** | 4 | 2 | 0.11922239999999999 | 0.1367314 | 0.2559538 | 2.3999846847360735 | 0.5999961711840184 |
| **Parallel** | 4 | 4 | 0.11897139999999999 | 0.1394936 | 0.2584652 | 2.376665021055059 | 0.5941662552637648 |
| **Parallel** | 4 | 8 | 0.11962460000000001 | 0.137184 | 0.2568086 | 2.3919962181951853 | 0.5979990545487963 |
| **Parallel** | 4 | 16 | 0.1200026 | 0.15079720000000002 | 0.2707998 | 2.2684108333905715 | 0.5671027083476429 |
| **Parallel** | 8 | 1 | 0.0652122 | 0.570825 | 0.6360372 | 0.9658007424722959 | 0.12072509280903698 |
| **Parallel** | 8 | 2 | 0.065706 | 0.35167580000000004 | 0.41738220000000004 | 1.4717570610342272 | 0.1839696326292784 |
| **Parallel** | 8 | 4 | 0.06618539999999999 | 0.3704338 | 0.4366194 | 1.406912290200573 | 0.17586403627507163 |
| **Parallel** | 8 | 8 | 0.06685699999999999 | 0.3931410000000001 | 0.4599978 | 1.3354089954343262 | 0.16692612442929078 |
| **Parallel** | 8 | 16 | 0.066733 | 0.4220766 | 0.4888096 | 1.2566962678310738 | 0.15708703347888422 |

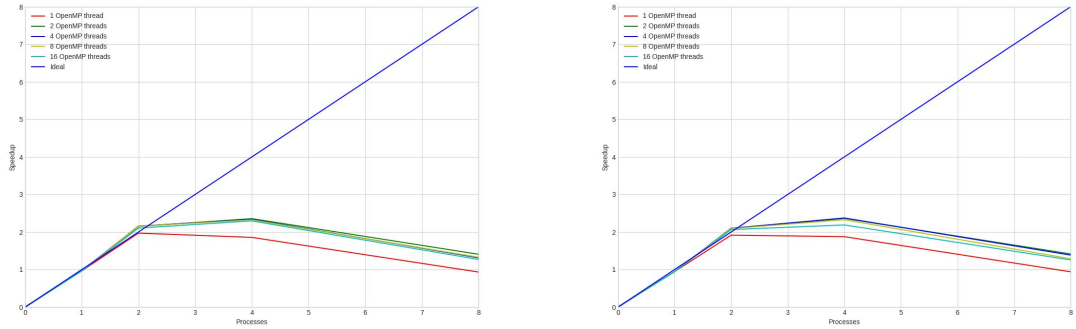
Tabella 26: Measurement with O1 optimization

### 5.5.3 2500 vertices - 75% of density




(a) (b)

Figura 55: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization




(a) (b)

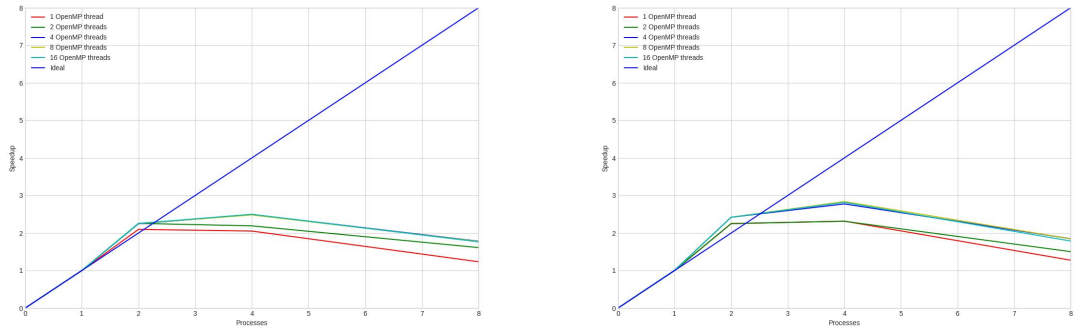
Figura 56: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 1.051844 | 0.0 | 1.051844 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 1.0457075999999998 | 0.0 | 1.0457075999999998 | 1.0058681795943725 | 1.0058681795943725 |
| **Parallel** | 1 | 2 | 1.0439520000000002 | 0.0 | 1.0439520000000002 | 1.007559734547182 | 1.007559734547182 |
| **Parallel** | 1 | 4 | 1.0454591999999998 | 0.0 | 1.0454591999999998 | 1.0061071728098048 | 1.0061071728098048 |
| **Parallel** | 1 | 8 | 1.0454478 | 0.0 | 1.0454478 | 1.0061181438231541 | 1.0061181438231541 |
| **Parallel** | 1 | 16 | 1.0455804 | 0.0 | 1.0455804 | 1.0059905484073726 | 1.0059905484073726 |
| **Parallel** | 2 | 1 | 0.42989639999999996 | 0.08637639999999999 | 0.5162726000000001 | 2.0373810270000767 | 1.0186905135000384 |
| **Parallel** | 2 | 2 | 0.43011879999999997 | 0.04342939999999999 | 0.47354799999999997 | 2.221198273459079 | 1.1105991367295396 |
| **Parallel** | 2 | 4 | 0.4304928 | 0.043543799999999994 | 0.4740366 | 2.218908835309341 | 1.1094544176546706 |
| **Parallel** | 2 | 8 | 0.4307708 | 0.043637999999999996 | 0.47440899999999997 | 2.2171670436269126 | 1.1085835218134563 |
| **Parallel** | 2 | 16 | 0.4299099999999999 | 0.0468074 | 0.47671740000000007 | 2.2064308959563883 | 1.1032154479781942 |
| **Parallel** | 4 | 1 | 0.2009918 | 0.34272539999999996 | 0.5437166 | 1.9345445770829877 | 0.4836361442707469 |
| **Parallel** | 4 | 2 | 0.2010844 | 0.23456459999999998 | 0.43564879999999995 | 2.4144310738374584 | 0.6036077684593646 |
| **Parallel** | 4 | 4 | 0.2011178 | 0.22696 | 0.42807780000000006 | 2.4571327922167416 | 0.6142831980541854 |
| **Parallel** | 4 | 8 | 0.20150659999999995 | 0.2452874 | 0.44679440000000004 | 2.354201395541215 | 0.5885503488853038 |
| **Parallel** | 4 | 16 | 0.2013352 | 0.24146600000000001 | 0.4428012 | 2.3754316835636398 | 0.5938579208909099 |
| **Parallel** | 8 | 1 | 0.10956740000000001 | 0.9555200000000001 | 1.0650870000000001 | 0.9875662739287964 | 0.12344578424109955 |
| **Parallel** | 8 | 2 | 0.1100904 | 0.6003244 | 0.710415 | 1.4806049984868 | 0.18507562481085 |
| **Parallel** | 8 | 4 | 0.11018359999999998 | 0.6297222 | 0.7399060000000002 | 1.4215913913388996 | 0.17769892391736244 |
| **Parallel** | 8 | 8 | 0.11137920000000001 | 0.6226374 | 0.7340166 | 1.432997564360261 | 0.17912469554503263 |
| **Parallel** | 8 | 16 | 0.1109378 | 0.68105 | 0.7919879999999999 | 1.3281059814037588 | 0.16601324767546985 |

Tabella 27: Measurement with O1 optimization

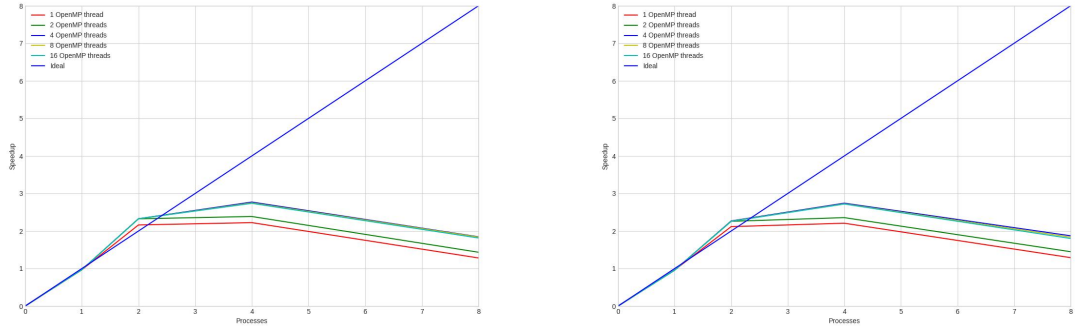
## 5.6 Case Study 4

### 5.6.1 10000 vertices - 25% of density




(a) (b)

Figura 57: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



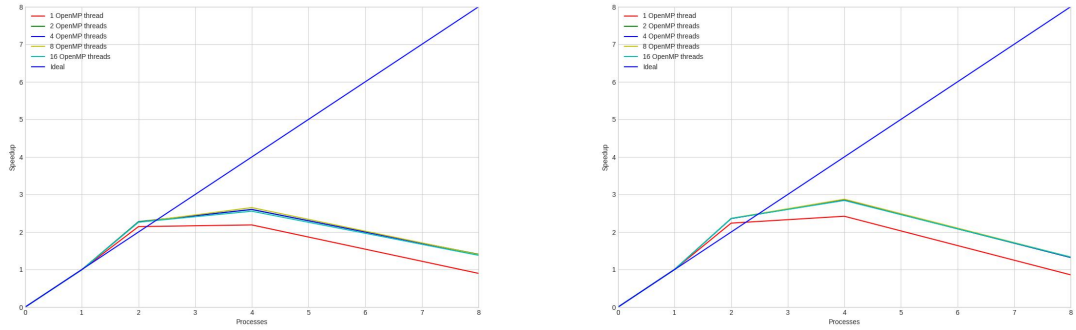

(a) (b)

Figura 58: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 4.6047184 | 0.0 | 4.6047184 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 4.5881896 | 0.0 | 4.5881896 | 1.0036024666461039 | 1.0036024666461039 |
| **Parallel** | 1 | 2 | 4.5959942 | 0.0 | 4.5959942 | 1.0018982182353495 | 1.0018982182353495 |
| **Parallel** | 1 | 4 | 4.5764958 | 0.0 | 4.5764958 | 1.0061668580576433 | 1.0061668580576433 |
| **Parallel** | 1 | 8 | 4.593551400000001 | 0.0 | 4.593551400000001 | 1.0024310166639256 | 1.0024310166639256 |
| **Parallel** | 1 | 16 | 4.5711714 | 0.0 | 4.5711714 | 1.0073388191044423 | 1.0073388191044423 |
| **Parallel** | 2 | 1 | 1.7510961999999999 | 0.2956798 | 2.046776 | 2.2497422287539037 | 1.1248711143769519 |
| **Parallel** | 2 | 2 | 1.7502232 | 0.29434079999999996 | 2.044564 | 2.252176209695564 | 1.126088104847782 |
| **Parallel** | 2 | 4 | 1.7537002000000002 | 0.14888099999999999 | 1.9025811999999998 | 2.4202480293613755 | 1.2101240146806878 |
| **Parallel** | 2 | 8 | 1.7525688 | 0.14921040000000002 | 1.901779 | 2.4212689276724584 | 1.2106344638362292 |
| **Parallel** | 2 | 16 | 1.751189 | 0.1518658 | 1.9030550000000002 | 2.419645464792137 | 1.2098227323960684 |
| **Parallel** | 4 | 1 | 0.8126104 | 1.1748825999999999 | 1.9874934 | 2.31684714022195 | 0.5792117850554875 |
| **Parallel** | 4 | 2 | 0.8128476000000001 | 1.177126 | 1.9899738 | 2.3139593094140234 | 0.5784898273535058 |
| **Parallel** | 4 | 4 | 0.8135258000000001 | 0.8456706 | 1.6591957999999998 | 2.7752712488785236 | 0.6938178122196309 |
| **Parallel** | 4 | 8 | 0.813063 | 0.8094612 | 1.6225239999999999 | 2.8379970958827117 | 0.7094992739706779 |
| **Parallel** | 4 | 16 | 0.8173378 | 0.8175595999999998 | 1.6348976 | 2.8165179274836545 | 0.7041294818709136 |
| **Parallel** | 8 | 1 | 0.46393059999999997 | 3.1437794000000006 | 3.6077099999999995 | 1.276354917662451 | 0.15954436470780636 |
| **Parallel** | 8 | 2 | 0.4641844 | 2.59724 | 3.0614244 | 1.5041097862811834 | 0.18801372328514793 |
| **Parallel** | 8 | 4 | 0.4650944 | 2.0274072 | 2.4925018000000003 | 1.8474283147960013 | 0.23092853934950017 |
| **Parallel** | 8 | 8 | 0.46750780000000003 | 2.0295244 | 2.4970322000000005 | 1.8440765000947923 | 0.23050956251184904 |
| **Parallel** | 8 | 16 | 0.47304840000000004 | 2.1033192 | 2.5763672 | 1.7872911904793698 | 0.22341139880992122 |

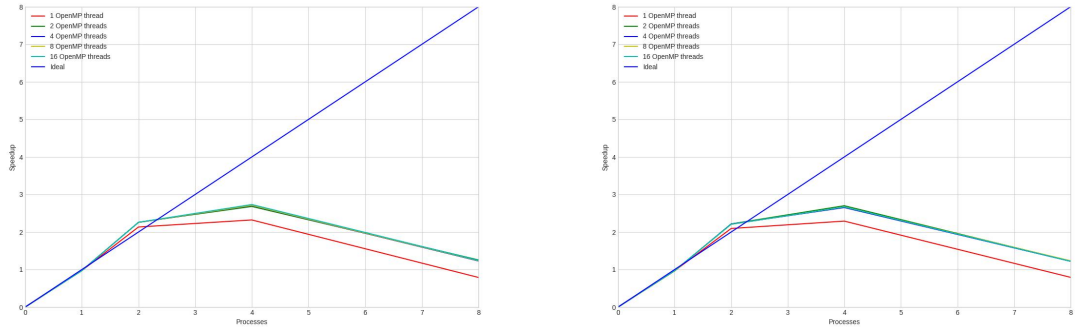
Tabella 28: Measurement with O1 optimization

### 5.6.2 10000 vertices - 50% of density




(a) (b)

Figura 59: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



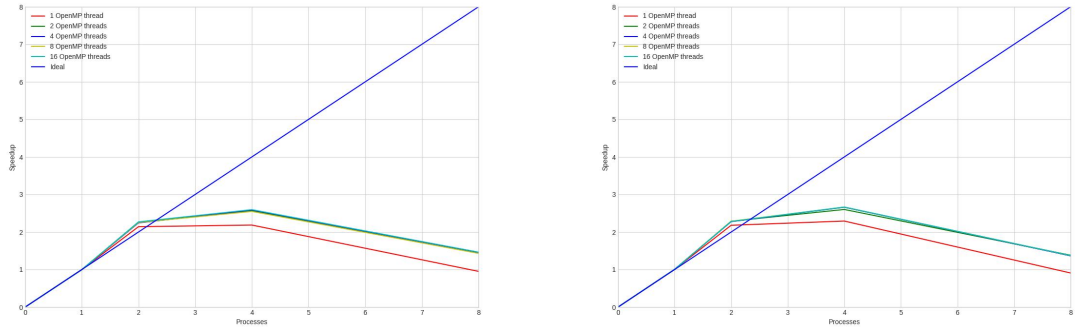

(a) (b)

Figura 60: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 16.796227800000004 | 0.0 | 16.796227800000004 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 16.688307799999997 | 0.0 | 16.688307799999997 | 1.0064668030631607 | 1.0064668030631607 |
| **Parallel** | 1 | 2 | 16.7258816 | 0.0 | 16.7258816 | 1.00420582912652 | 1.00420582912652 |
| **Parallel** | 1 | 4 | 16.7055284 | 0.0 | 16.7055284 | 1.0054293044690525 | 1.0054293044690525 |
| **Parallel** | 1 | 8 | 16.7320922 | 0.0 | 16.7320922 | 1.0038330890861338 | 1.0038330890861338 |
| **Parallel** | 1 | 16 | 16.737076000000002 | 0.0 | 16.737076000000002 | 1.0035341776544482 | 1.0035341776544482 |
| **Parallel** | 2 | 1 | 6.689685599999999 | 0.8183674 | 7.5080528 | 2.237095056124273 | 1.1185475280621364 |
| **Parallel** | 2 | 2 | 6.7069632 | 0.4113576 | 7.118320799999999 | 2.3595772474879197 | 1.1797886237439599 |
| **Parallel** | 2 | 4 | 6.714285000000001 | 0.41060679999999994 | 7.1248917999999986 | 2.3574011046736185 | 1.1787005523368093 |
| **Parallel** | 2 | 8 | 6.691253000000001 | 0.41090299999999996 | 7.102156199999999 | 2.3649476760311194 | 1.1824738380155597 |
| **Parallel** | 2 | 16 | 6.6935270000000004 | 0.41639 | 7.109916799999999 | 2.3623662937940435 | 1.1811831468970218 |
| **Parallel** | 4 | 1 | 2.9452784 | 3.994055800000001 | 6.9393344 | 2.4204378736957834 | 0.6051094684239459 |
| **Parallel** | 4 | 2 | 2.9535919999999996 | 2.9381614 | 5.891753400000001 | 2.850802920570301 | 0.7127007301425753 |
| **Parallel** | 4 | 4 | 2.9451928 | 2.91772 | 5.8629128 | 2.8648264732847473 | 0.7162066183211868 |
| **Parallel** | 4 | 8 | 2.9491666000000003 | 2.894377 | 5.843543599999999 | 2.874322320449531 | 0.7185805801123828 |
| **Parallel** | 4 | 16 | 2.9319711999999996 | 2.9815496 | 5.913520799999999 | 2.8403092452131067 | 0.7100773113032767 |
| **Parallel** | 8 | 1 | 1.6777440000000001 | 17.8527728 | 19.5305168 | 0.8599991475904009 | 0.10749989344880011 |
| **Parallel** | 8 | 2 | 1.7062198 | 11.0147128 | 12.7209326 | 1.3203613546384174 | 0.16504516932980218 |
| **Parallel** | 8 | 4 | 1.711169 | 10.991712 | 12.702880799999999 | 1.3222376927287238 | 0.16527971159109048 |
| **Parallel** | 8 | 8 | 1.7067909999999997 | 10.8748392 | 12.581630400000002 | 1.3349802264100845 | 0.16687252830126056 |
| **Parallel** | 8 | 16 | 1.7055281999999998 | 10.878062400000001 | 12.5835908 | 1.3347722495871372 | 0.16684653119839216 |

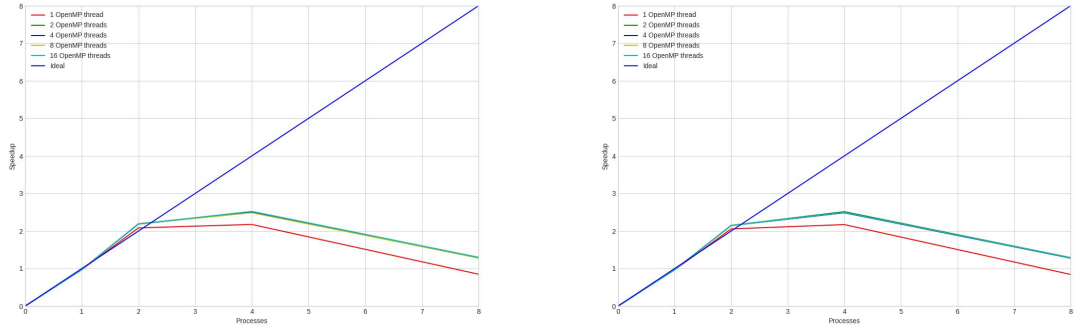
Tabella 29: Measurement with O1 optimization

### 5.6.3 10000 vertices - 75% of density




(a) (b)

Figura 61: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



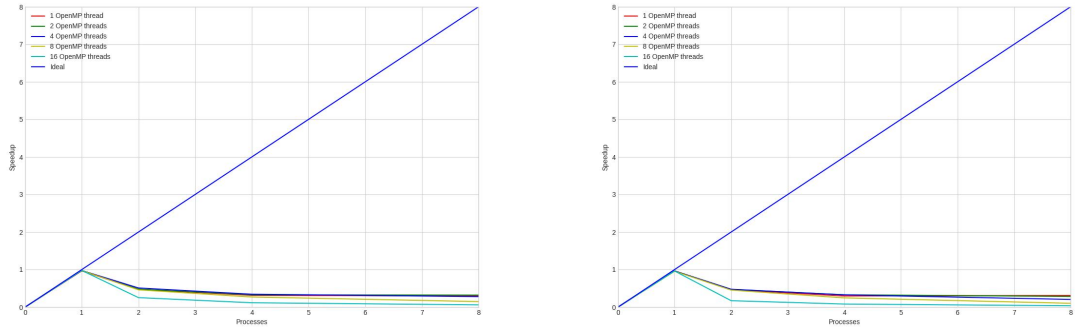

(a) (b)

Figura 62: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 32.46680680000001 | 0.0 | 32.46680680000001 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 32.27933899999999 | 0.0 | 32.27933899999999 | 1.0058076715883189 | 1.0058076715883189 |
| **Parallel** | 1 | 2 | 32.32407499999999 | 0.0 | 32.32407499999999 | 1.0044156499451264 | 1.0044156499451264 |
| **Parallel** | 1 | 4 | 32.2727116 | 0.0 | 32.2727116 | 1.0060142203854976 | 1.0060142203854976 |
| **Parallel** | 1 | 8 | 32.3266016 | 0.0 | 32.3266016 | 1.004337146283883 | 1.004337146283883 |
| **Parallel** | 1 | 16 | 32.285578400000006 | 0.0 | 32.285578400000006 | 1.0056132926520531 | 1.0056132926520531 |
| **Parallel** | 2 | 1 | 13.532039600000001 | 1.3597875999999998 | 14.8918272 | 2.180176170725377 | 1.0900880853626884 |
| **Parallel** | 2 | 2 | 13.516655200000002 | 0.6807154 | 14.1973704 | 2.286818325173795 | 1.1434091625868974 |
| **Parallel** | 2 | 4 | 13.5817914 | 0.6805277999999999 | 14.262319 | 2.276404475317093 | 1.1382022376585466 |
| **Parallel** | 2 | 8 | 13.512113199999998 | 0.6798088 | 14.1919218 | 2.2876962864888397 | 1.1438481432444199 |
| **Parallel** | 2 | 16 | 13.5506124 | 0.6889174 | 14.239530000000002 | 2.2800476420218927 | 1.1400238210109463 |
| **Parallel** | 4 | 1 | 6.3275904 | 7.834450800000001 | 14.1620408 | 2.292523179286421 | 0.5731307948216052 |
| **Parallel** | 4 | 2 | 6.3424792 | 6.1403266 | 12.4828056 | 2.600922247799806 | 0.6502305619499515 |
| **Parallel** | 4 | 4 | 6.3196682 | 5.8868562 | 12.206524400000001 | 2.659791250652807 | 0.6649478126632018 |
| **Parallel** | 4 | 8 | 6.339520200000001 | 5.852125 | 12.191645600000001 | 2.663037285138932 | 0.665759321284733 |
| **Parallel** | 4 | 16 | 6.310885600000001 | 5.867402 | 12.1782872 | 2.6659583787776007 | 0.6664895946944002 |
| **Parallel** | 8 | 1 | 4.2317986 | 31.5099036 | 35.741702000000004 | 0.9083732722073505 | 0.11354665902591882 |
| **Parallel** | 8 | 2 | 4.1967884 | 19.2950466 | 23.491834800000003 | 1.382046446197553 | 0.17275580577469413 |
| **Parallel** | 8 | 4 | 4.1966854 | 19.600255399999998 | 23.7969408 | 1.3643269138191076 | 0.17054086422738846 |
| **Parallel** | 8 | 8 | 4.2021982 | 19.4705906 | 23.6727888 | 1.3714821297269382 | 0.17143526621586727 |
| **Parallel** | 8 | 16 | 4.1932804 | 19.450278000000004 | 23.6435578 | 1.3731777203175406 | 0.17164721503969257 |

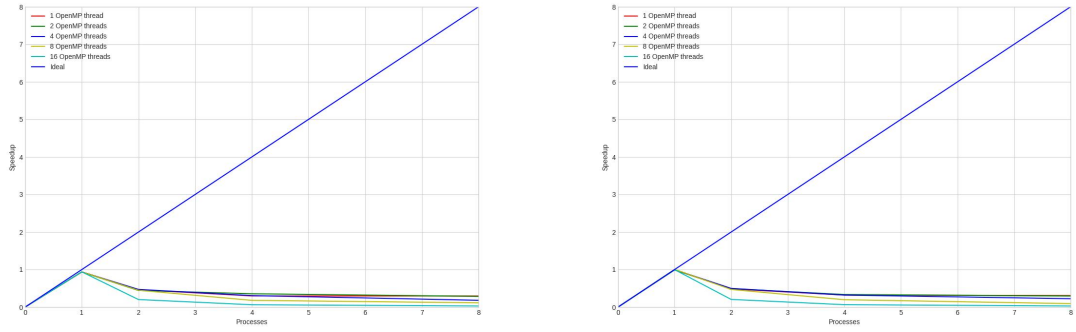
Tabella 30: Measurement with O1 optimization

### 5.6.4 10000 vertices - Special Cases - 0%




(a) (b)

Figura 63: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



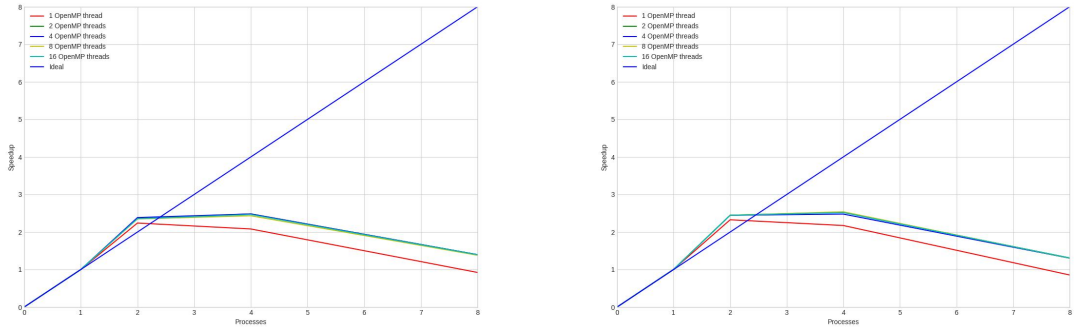

(a) (b)

Figura 64: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.0021524 | 0.0 | 0.0021524 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0021578 | 0.0 | 0.0021578 | 0.9974974511076096 | 0.9974974511076096 |
| **Parallel** | 1 | 2 | 0.0021742000000000003 | 0.0 | 0.0021742000000000003 | 0.9899733235212951 | 0.9899733235212951 |
| **Parallel** | 1 | 4 | 0.0021578 | 0.0 | 0.0021578 | 0.9974974511076096 | 0.9974974511076096 |
| **Parallel** | 1 | 8 | 0.0021398 | 0.0 | 0.0021398 | 1.00588840078512 | 1.00588840078512 |
| **Parallel** | 1 | 16 | 0.002156 | 0.0 | 0.002156 | 0.9983302411873841 | 0.9983302411873841 |
| **Parallel** | 2 | 1 | 0.0036842 | 0.0007821999999999999 | 0.004466200000000001 | 0.48193094800949343 | 0.24096547400474672 |
| **Parallel** | 2 | 2 | 0.0037302000000000004 | 0.0006566 | 0.0043866 | 0.4906761500934665 | 0.24533807504673324 |
| **Parallel** | 2 | 4 | 0.0036974 | 0.0006236 | 0.0043214 | 0.49807932614430506 | 0.24903966307215253 |
| **Parallel** | 2 | 8 | 0.0038788000000000004 | 0.0007015999999999999 | 0.0045806 | 0.46989477361044407 | 0.23494738680522204 |
| **Parallel** | 2 | 16 | 0.0038772 | 0.0066327999999999995 | 0.0105102 | 0.20479153584137313 | 0.10239576792068657 |
| **Parallel** | 4 | 1 | 0.0051284 | 0.0015951999999999997 | 0.0067234 | 0.3201356456554719 | 0.08003391141386798 |
| **Parallel** | 4 | 2 | 0.005094 | 0.0013160000000000001 | 0.0064098 | 0.3357983088395894 | 0.08394957720989735 |
| **Parallel** | 4 | 4 | 0.005466 | 0.0012482 | 0.006714199999999999 | 0.3205743052038963 | 0.08014357630097407 |
| **Parallel** | 4 | 8 | 0.005079800000000001 | 0.0058788 | 0.010959 | 0.19640478145816226 | 0.049101195364540565 |
| **Parallel** | 4 | 16 | 0.0048696 | 0.028484200000000005 | 0.0333538 | 0.06453237712044804 | 0.01613309428011201 |
| **Parallel** | 8 | 1 | 0.0051804 | 0.0018315999999999999 | 0.0070118 | 0.30696825351550244 | 0.038371031689437805 |
| **Parallel** | 8 | 2 | 0.0052239999999999995 | 0.0020876000000000002 | 0.0073116 | 0.29438153071831064 | 0.03679769133978883 |
| **Parallel** | 8 | 4 | 0.005865400000000001 | 0.0037776000000000003 | 0.009643200000000001 | 0.2232039157126265 | 0.02790048946407831 |
| **Parallel** | 8 | 8 | 0.0052266 | 0.0179794 | 0.023205999999999997 | 0.0927518745152116 | 0.011593984314401 45 |
| **Parallel** | 8 | 16 | 0.0053796 | 0.0661244 | 0.0715038 | 0.03010189668241408 | 0.00376273708530176 |

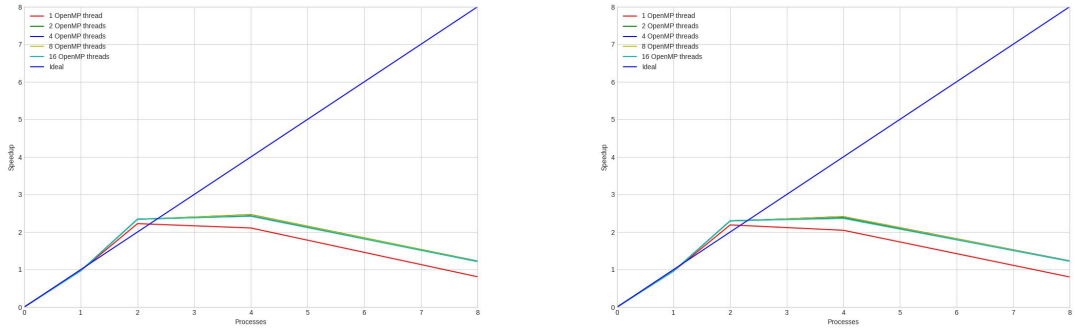
Tabella 31: Measurement with O2 optimization

### 5.6.5 10000 vertices - Special Cases - 100%




(a) (b)

Figura 65: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization




(a) (b)

Figura 66: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 51.669504800000006 | 0.0 | 51.669504800000006 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 51.390501199999996 | 0.0 | 51.390501199999996 | 1.005429088907193 | 1.005429088907193 |
| **Parallel** | 1 | 2 | 51.468928000000005 | 0.0 | 51.468928000000005 | 1.0038970463888426 | 1.0038970463888426 |
| **Parallel** | 1 | 4 | 51.354802799999995 | 0.0 | 51.354802799999995 | 1.006127995491008 | 1.006127995491008 |
| **Parallel** | 1 | 8 | 51.327782 | 0.0 | 51.327782 | 1.006657657640457 | 1.006657657640457 |
| **Parallel** | 1 | 16 | 51.410499800000004 | 0.0 | 51.410499800000004 | 1.005037978642643 | 1.005037978642643 |
| **Parallel** | 2 | 1 | 20.0413236 | 2.1745039999999998 | 22.2158276 | 2.3257969826881446 | 1.1628984913440723 |
| **Parallel** | 2 | 2 | 20.016664400000003 | 1.0888214 | 21.1054856 | 2.448155222735079 | 1.2240776113675396 |
| **Parallel** | 2 | 4 | 20.036047399999998 | 1.0889616 | 21.1250092 | 2.4458926531497087 | 1.2229463265748544 |
| **Parallel** | 2 | 8 | 20.0166854 | 1.0897078 | 21.106393399999998 | 2.448049925952769 | 1.2240249629763844 |
| **Parallel** | 2 | 16 | 20.032993400000002 | 1.0900614000000002 | 21.1230552 | 2.446118911813477 | 1.2230594559067385 |
| **Parallel** | 4 | 1 | 8.604989 | 15.163447 | 23.7684358 | 2.173870642341555 | 0.5434676605853888 |
| **Parallel** | 4 | 2 | 8.5861154 | 11.8555806 | 20.441696 | 2.527652539202227 | 0.6319131348005568 |
| **Parallel** | 4 | 4 | 8.610945000000001 | 12.2414702 | 20.8524156 | 2.47786663143238 | 0.619466657858095 |
| **Parallel** | 4 | 8 | 8.593002 | 11.755037999999999 | 20.348039800000002 | 2.539286599980014 | 0.6348216499950035 |
| **Parallel** | 4 | 16 | 8.602756600000001 | 11.9277536 | 20.53051 | 2.516718035742902 | 0.6291795089357255 |
| **Parallel** | 8 | 1 | 4.7579116 | 55.642517 | 60.4004288 | 0.8554493043599055 | 0.10693116304498819 |
| **Parallel** | 8 | 2 | 4.752538799999999 | 34.686370800000006 | 39.43891 | 1.3101149296468895 | 0.16376436620586118 |
| **Parallel** | 8 | 4 | 4.774870399999999 | 34.888955 | 39.663825599999996 | 1.3026858609422691 | 0.16283573261778364 |
| **Parallel** | 8 | 8 | 4.773937 | 34.8914938 | 39.6654312 | 1.3026331300792717 | 0.16282914125990897 |
| **Parallel** | 8 | 16 | 4.7803746 | 34.6970002 | 39.4773748 | 1.3088384185059845 | 0.16360480231324806 |

Tabella 32: Measurement with O1 optimization

# 6 Cluster

In addition to the measurements taken on our machine, we also tried running our program on a raspberry pi cluster, specifically, the front-end node we connected to is a raspberry *pi*3, while the worker nodes are raspberry *pi*4. In order to make a fair comparison with the measurements taken on our machine, on the cluster the program was launched using the same graphs except for the one with 10000 nodes.
Therefore, the following graphs were considered:

- graph with a number of vertices equal to 500:
  1. **low density 25%**: 0 - 125
  2. **medium density 50%**: 125 - 250
  3. **high density 75%**: 250 - 375
- graph with a number of vertices equal to 1000:
  1. **low density 25%**: 0 - 250
  2. **medium density 50%**: 250 - 500
  3. **high density 75%**: 500 - 750
- graph with a number of vertices equal to 2500:
  1. **low density 25%**: 0 - 625
  2. **medium density 50%**: 625 - 1350
  3. **high density 75%**: 1350 - 1875

For each density level of each graph on which the execution of the algorithm version has been defined, as regards the management of MPI processes and OpenMP threads, the latter has been structured by assigning the following resources to each of prefixed density level:

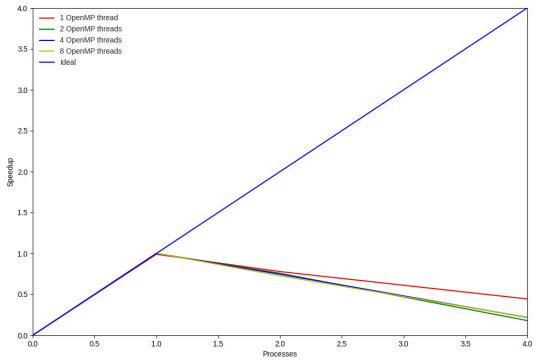
- 0 MPI process - {1, 2, 4, 8} OpenMP threads (*Sequential version*)
- 1 MPI process - {1, 2, 4, 8} OpenMP threads (*Parallel version*)
- 2 MPI process - {1, 2, 4, 8} OpenMP threads (*Parallel version*)
- 4 MPI process - {1, 2, 4, 8} OpenMP threads (*Parallel version*)
- 8 MPI process - {1, 2, 4, 8} OpenMP threads (*Parallel version*)

Therefore, from the considerations made, we can define the case studies analysed:

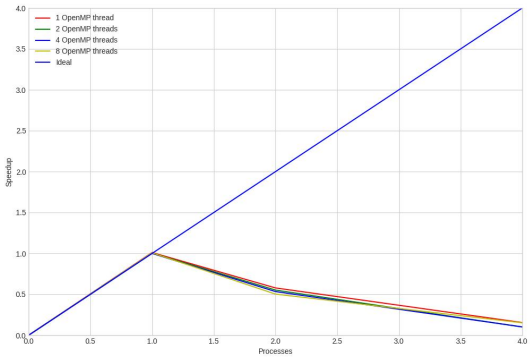
- **Case Study 1**: the sequential program and the parallel program are compliled on a graph with a number of vertices equal to 500
- **Case Study 2**: the sequential program and the parallel program are compiled on a graph with a number of vertices equal to 1000
- **Case Study 3**: the sequential program and the parallel program are compiled on a graph with a number of vertices equal to 2500

## 6.1 Tarjan

### 6.1.1 500 vertices - 25% of density

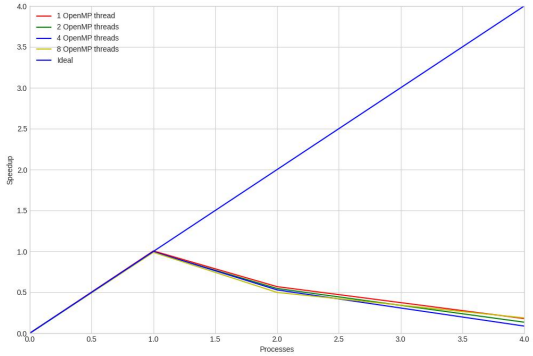



(a)

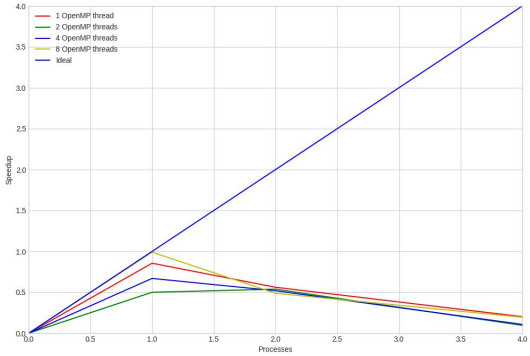



(b)

Figura 67: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization




(a)



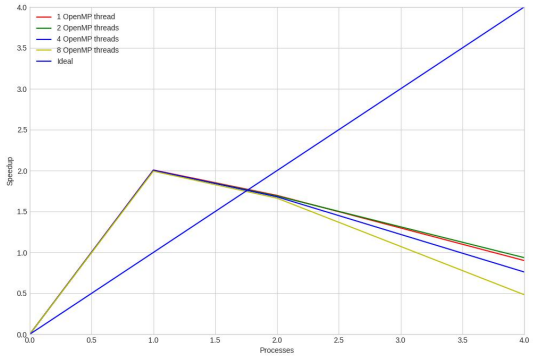

(b)

Figura 68: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

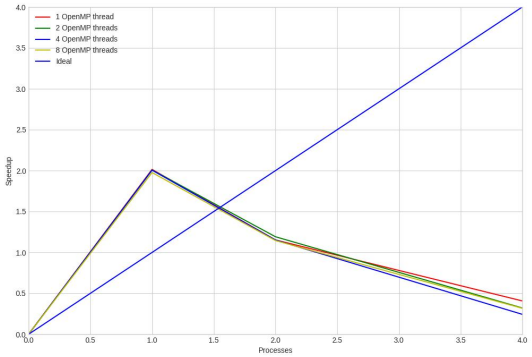
| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.006798 | 0.0 | 0.006798 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0067220000000000005 | 0.0 | 0.0067220000000000005 | 1.0113061588812853 | 1.0113061588812853 |
| **Parallel** | 1 | 2 | 0.006764 | 0.0 | 0.006764 | 1.0050266114725015 | 1.0050266114725015 |
| **Parallel** | 1 | 4 | 0.006796 | 0.0 | 0.006796 | 1.0002942907592702 | 1.0002942907592702 |
| **Parallel** | 1 | 8 | 0.0067475 | 0.0 | 0.0067475 | 1.0074842534271953 | 1.0074842534271953 |
| **Parallel** | 2 | 1 | 0.0026815 | 0.009071 | 0.011752499999999999 | 0.5784301212507977 | 0.28921506062539887 |
| **Parallel** | 2 | 2 | 0.0027004999999999998 | 0.009611999999999999 | 0.0123125 | 0.5521218274111676 | 0.2760609137055838 |
| **Parallel** | 2 | 4 | 0.0026725 | 0.0100635 | 0.012736 | 0.5337625628140703 | 0.26688128140703515 |
| **Parallel** | 2 | 8 | 0.002679 | 0.0108295 | 0.0135085 | 0.5032387015582781 | 0.25161935077913905 |
| **Parallel** | 4 | 1 | 0.0032985 | 0.040594000000000005 | 0.0438925 | 0.15487839608133508 | 0.03871959902033377 |
| **Parallel** | 4 | 2 | 0.0034384999999999997 | 0.063927 | 0.06736500000000001 | 0.10091293698508126 | 0.025228234246270315 |
| **Parallel** | 4 | 4 | 0.00244 | 0.06442200000000001 | 0.066862 | 0.10167210074481768 | 0.02541802518620442 |
| **Parallel** | 4 | 8 | 0.003229 | 0.0415465 | 0.044775 | 0.15182579564489113 | 0.03795644891122278 |

Tabella 33: Measurement with O1 optimization

### 6.1.2 500 vertices - 50% of density

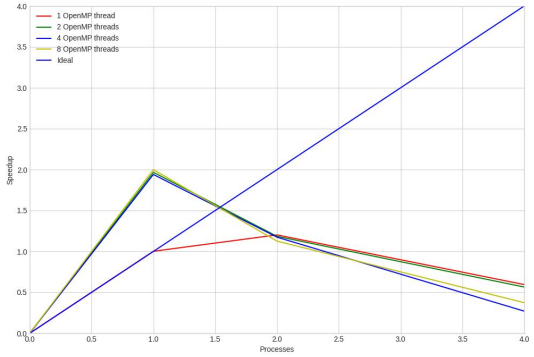



(a)

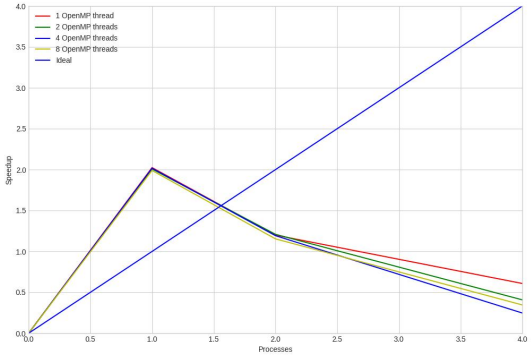



(b)

Figura 69: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization




(a)



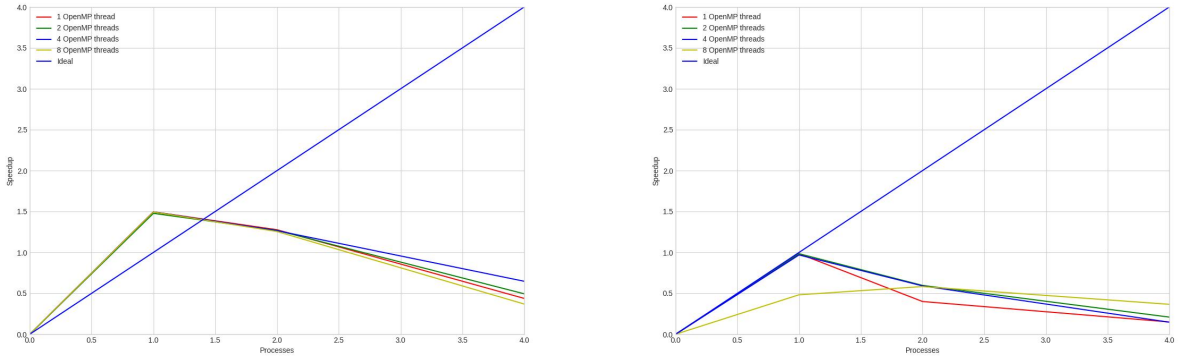

(b)

Figura 70: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.1119405 | 0.0 | 0.1119405 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0557535 | 0.0 | 0.0557535 | 2.0077752966181497 | 2.0077752966181497 |
| **Parallel** | 1 | 2 | 0.055941 | 0.0 | 0.055941 | 2.0010457446238 | 2.0010457446238 |
| **Parallel** | 1 | 4 | 0.055836 | 0.0 | 0.055836 | 2.0048087255534064 | 2.0048087255534064 |
| **Parallel** | 1 | 8 | 0.056162000000000004 | 0.0 | 0.056162000000000004 | 1.9931715394750897 | 1.9931715394750897 |
| **Parallel** | 2 | 1 | 0.022352499999999997 | 0.043597 | 0.0659495 | 1.6973669246923784 | 0.8486834623461892 |
| **Parallel** | 2 | 2 | 0.022383 | 0.04384 | 0.0662225 | 1.6903695873758917 | 0.8451847936879459 |
| **Parallel** | 2 | 4 | 0.0223725 | 0.044256500000000004 | 0.066629 | 1.6800567320536104 | 0.8400283660268052 |
| **Parallel** | 2 | 8 | 0.022381 | 0.044908000000000003 | 0.0672885 | 1.6635903609086247 | 0.8317951804543123 |
| **Parallel** | 4 | 1 | 0.01176 | 0.112472 | 0.124232 | 0.9010601133363385 | 0.22526502833408463 |
| **Parallel** | 4 | 2 | 0.011737000000000001 | 0.1077975 | 0.119534 | 0.936474141248515 | 0.23411853531212876 |
| **Parallel** | 4 | 4 | 0.0117725 | 0.1353025 | 0.147075 | 0.7611116777154512 | 0.1902779194288628 |
| **Parallel** | 4 | 8 | 0.023448999999999998 | 0.208146 | 0.231595 | 0.48334592715732205 | 0.12083648178933051 |

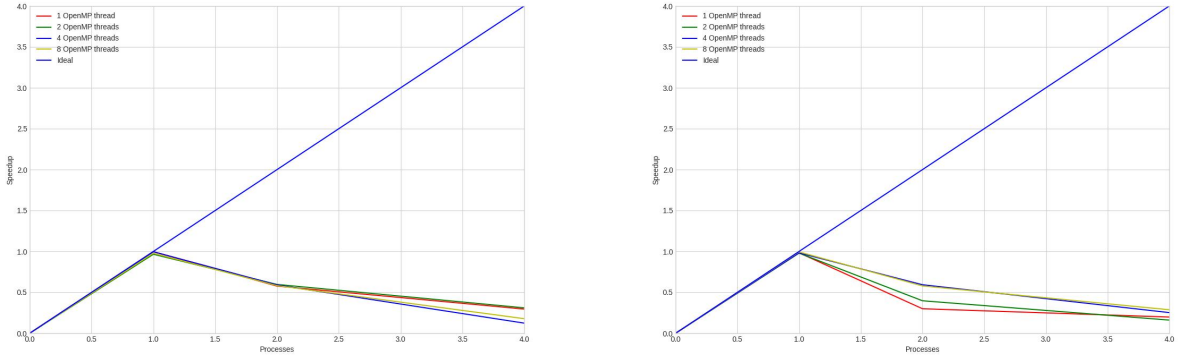
Tabella 34: Measurement with O0 optimization

### 6.1.3 500 vertices - 75% of density




(a) (b)

Figura 71: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



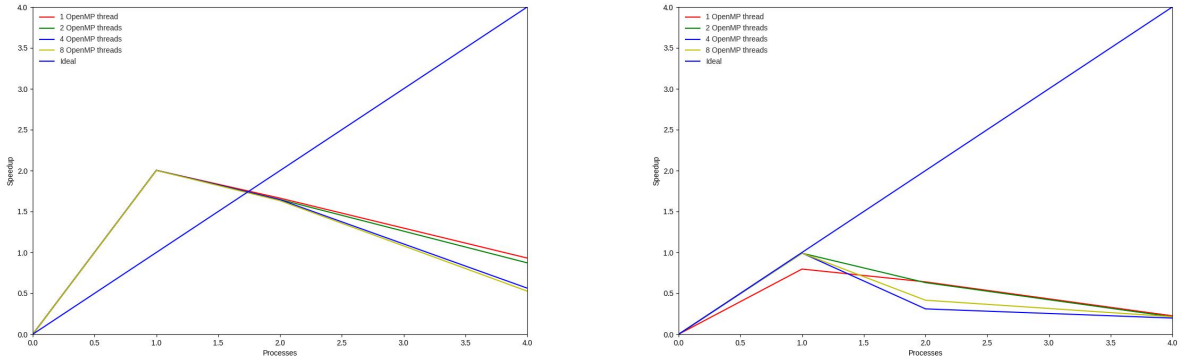

(a) (b)

Figura 72: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.1370005 | 0.0 | 0.1370005 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0915655 | 0.0 | 0.0915655 | 1.4962021722155179 | 1.4962021722155179 |
| **Parallel** | 1 | 2 | 0.0927035 | 0.0 | 0.0927035 | 1.4778352489388211 | 1.4778352489388211 |
| **Parallel** | 1 | 4 | 0.09162000000000001 | 0.0 | 0.09162000000000001 | 1.4953121589172669 | 1.4953121589172669 |
| **Parallel** | 1 | 8 | 0.091675 | 0.0 | 0.091675 | 1.4944150531769838 | 1.4944150531769838 |
| **Parallel** | 2 | 1 | 0.0363815 | 0.070842 | 0.107224 | 1.277703685741998 | 0.638851842870999 |
| **Parallel** | 2 | 2 | 0.036406999999999995 | 0.0714655 | 0.1078725 | 1.2700224802428794 | 0.6350112401214397 |
| **Parallel** | 2 | 4 | 0.036431000000000005 | 0.07176199999999999 | 0.10819300000000001 | 1.2662602941040546 | 0.6331301470520273 |
| **Parallel** | 2 | 8 | 0.036420999999999995 | 0.0726845 | 0.109106 | 1.2556642164500578 | 0.6278321082250289 |
| **Parallel** | 4 | 1 | 0.028665999999999997 | 0.2840405 | 0.312706 | 0.43811279604484726 | 0.10952819901121182 |
| **Parallel** | 4 | 2 | 0.0284525 | 0.248655 | 0.277107 | 0.4943956666558405 | 0.12359891666396013 |
| **Parallel** | 4 | 4 | 0.018923000000000002 | 0.1925965 | 0.2115195 | 0.647696784457225 | 0.16192419611430625 |
| **Parallel** | 4 | 8 | 0.038179000000000005 | 0.3329475 | 0.3711265 | 0.3691477164794214 | 0.09228692911985535 |

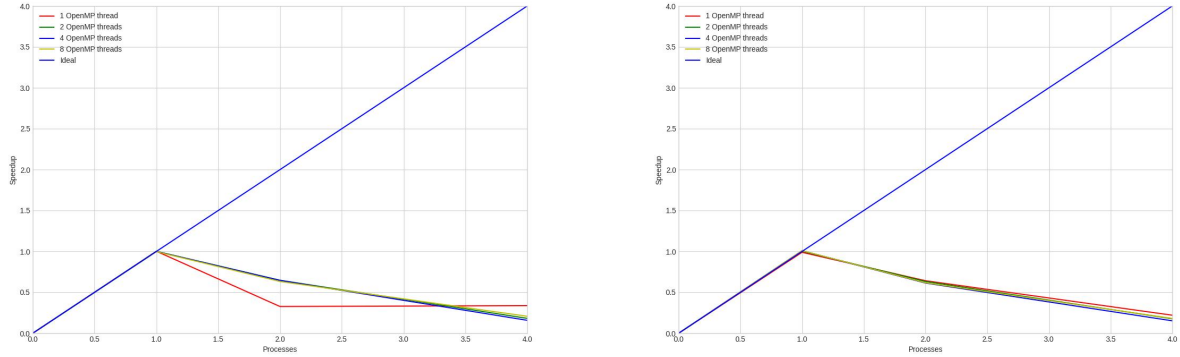
Tabella 35: Measurement with O0 optimization

### 6.1.4 1000 vertices - 25% of density




(a) (b)

Figura 73: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



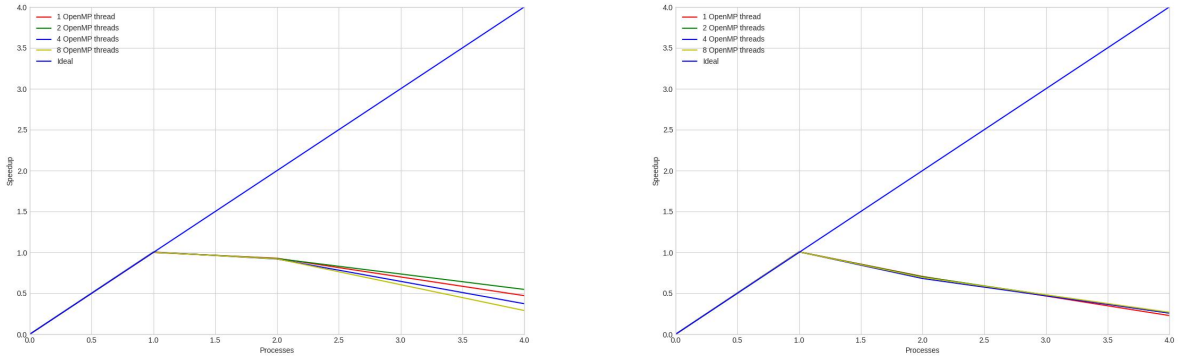

(a) (b)

Figura 74: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.1677215 | 0.0 | 0.1677215 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.08372299999999999 | 0.0 | 0.08372299999999999 | 2.0032906130931765 | 2.0032906130931765 |
| **Parallel** | 1 | 2 | 0.0837155 | 0.0 | 0.0837155 | 2.0034700861847567 | 2.0034700861847567 |
| **Parallel** | 1 | 4 | 0.083701 | 0.0 | 0.083701 | 2.003817158695834 | 2.003817158695834 |
| **Parallel** | 1 | 8 | 0.08375450000000001 | 0.0 | 0.08375450000000001 | 2.0025371771069014 | 2.0025371771069014 |
| **Parallel** | 2 | 1 | 0.0333405 | 0.06747 | 0.10081 | 1.6637387163971828 | 0.8318693581985914 |
| **Parallel** | 2 | 2 | 0.0332655 | 0.0685635 | 0.1018285 | 1.6470978164266388 | 0.8235489082133194 |
| **Parallel** | 2 | 4 | 0.0333025 | 0.06868650000000001 | 0.101989 | 1.644505780035102 | 0.822252890017551 |
| **Parallel** | 2 | 8 | 0.033261 | 0.069466 | 0.102727 | 1.6326915027208038 | 0.8163457513604019 |
| **Parallel** | 4 | 1 | 0.0215405 | 0.15852 | 0.1800605 | 0.9314730326751286 | 0.23286825816878215 |
| **Parallel** | 4 | 2 | 0.021466 | 0.170658 | 0.1921245 | 0.8729834039906414 | 0.21824585099766036 |
| **Parallel** | 4 | 4 | 0.0321735 | 0.2653735 | 0.297547 | 0.5636806958228447 | 0.1409201739557112 |
| **Parallel** | 4 | 8 | 0.043039999999999995 | 0.27527749999999995 | 0.31831750000000003 | 0.5269000290590369 | 0.13172500726475922 |

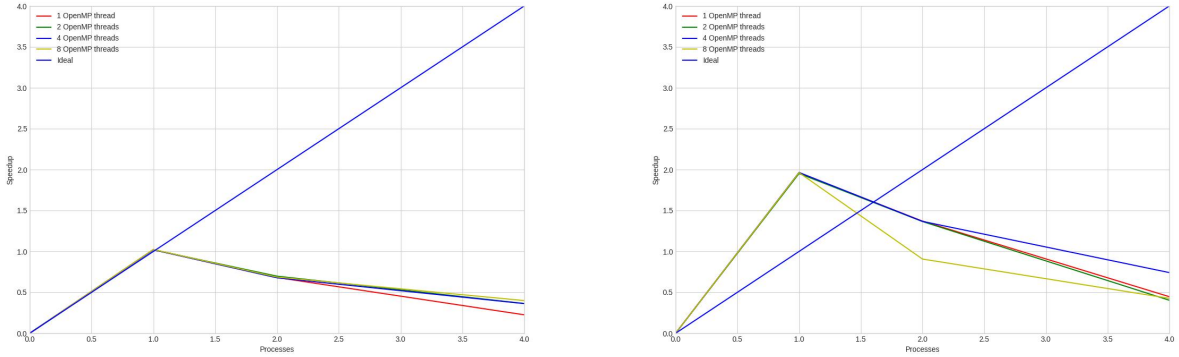
Tabella 36: Measurement with O0 optimization

### 6.1.5 1000 vertices - 50% of density




(a) (b)

Figura 75: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



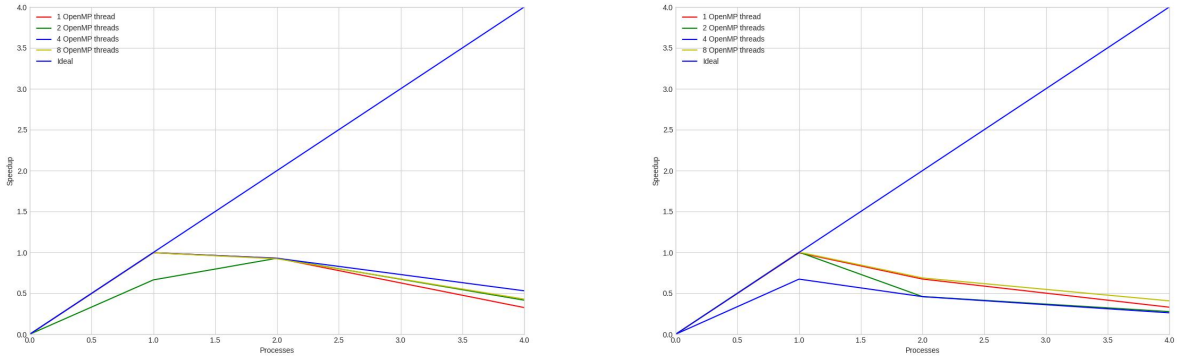

(a) (b)

Figura 76: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.173553 | 0.0 | 0.173553 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.0885625 | 0.0 | 0.0885625 | 1.959666901905434 | 1.959666901905434 |
| **Parallel** | 1 | 2 | 0.08890100000000001 | 0.0 | 0.08890100000000001 | 1.952205262033048 | 1.952205262033048 |
| **Parallel** | 1 | 4 | 0.0882955 | 0.0 | 0.0882955 | 1.9655928105056317 | 1.9655928105056317 |
| **Parallel** | 1 | 8 | 0.0884035 | 0.0 | 0.0884035 | 1.9631915025988793 | 1.9631915025988793 |
| **Parallel** | 2 | 1 | 0.030814 | 0.095799 | 0.12661299999999998 | 1.370736022367372 | 0.685368011183686 |
| **Parallel** | 2 | 2 | 0.0307515 | 0.096322 | 0.1270735 | 1.365768629966122 | 0.682884314983061 |
| **Parallel** | 2 | 4 | 0.030677 | 0.0962355 | 0.12691249999999998 | 1.3675012311632033 | 0.6837506155816017 |
| **Parallel** | 2 | 8 | 0.045978 | 0.14536300000000002 | 0.191341 | 0.9070350839600504 | 0.4535175419800252 |
| **Parallel** | 4 | 1 | 0.022692999999999998 | 0.365568 | 0.3882605 | 0.447001433316034 | 0.1117503583290085 |
| **Parallel** | 4 | 2 | 0.029369 | 0.400739 | 0.4301075 | 0.40351075021942195 | 0.10087768755485549 |
| **Parallel** | 4 | 4 | 0.0151135 | 0.2189355 | 0.234049 | 0.741524210742195 | 0.18538105268554875 |
| **Parallel** | 4 | 8 | 0.0244915 | 0.38227900000000004 | 0.40677050000000003 | 0.4266607337552748 | 0.1066651834388187 |

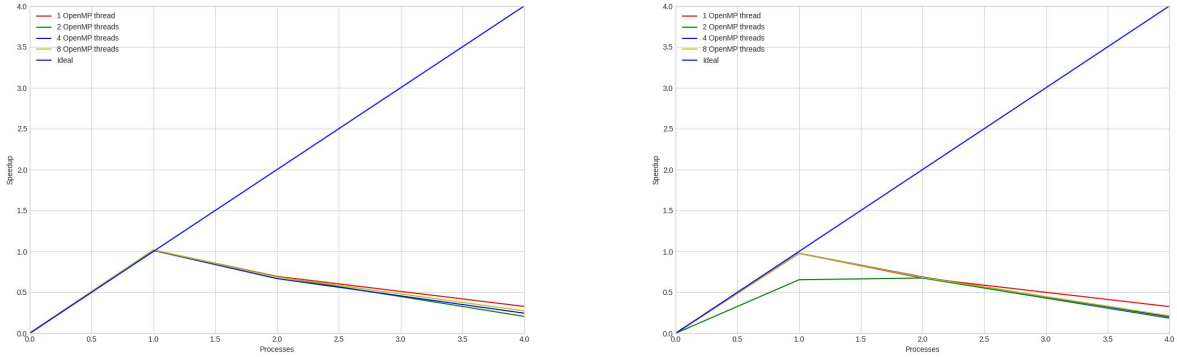
Tabella 37: Measurement with O3 optimization

### 6.1.6 1000 vertices - 75% of density




(a) (b)

Figura 77: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



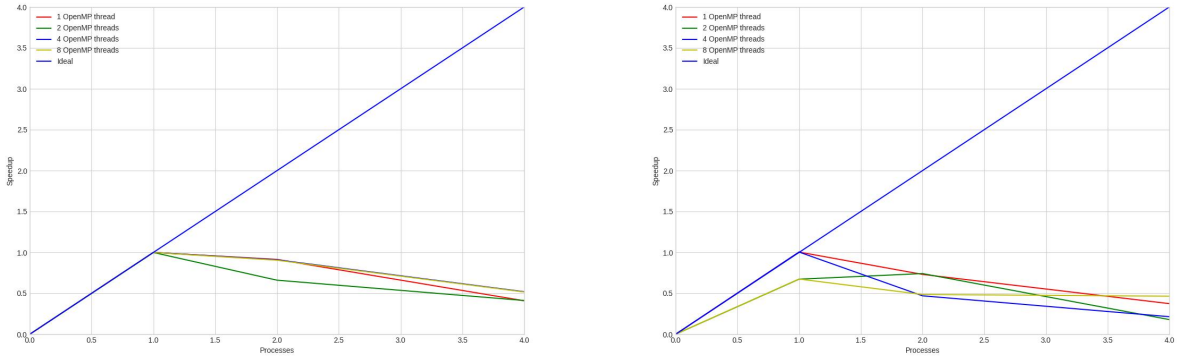

(a) (b)

Figura 78: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.1428705 | 0.0 | 0.1428705 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.1408595 | 0.0 | 0.1408595 | 1.0142766373585028 | 1.0142766373585028 |
| **Parallel** | 1 | 2 | 0.14056 | 0.0 | 0.14056 | 1.0164378201479796 | 1.0164378201479796 |
| **Parallel** | 1 | 4 | 0.14120349999999998 | 0.0 | 0.14120349999999998 | 1.0118056563753735 | 1.0118056563753735 |
| **Parallel** | 1 | 8 | 0.140536 | 0.0 | 0.140536 | 1.016611402060682 | 1.016611402060682 |
| **Parallel** | 2 | 1 | 0.049229499999999995 | 0.156385 | 0.2056145 | 0.6948464237687517 | 0.34742321188437586 |
| **Parallel** | 2 | 2 | 0.049263 | 0.15758450000000002 | 0.20684750000000002 | 0.6907045045262814 | 0.3453522522631407 |
| **Parallel** | 2 | 4 | 0.049218 | 0.16457650000000001 | 0.2137935 | 0.668264002413544 | 0.334132001206772 |
| **Parallel** | 2 | 8 | 0.049678 | 0.158081 | 0.207759 | 0.6876741801799201 | 0.34383709008996005 |
| **Parallel** | 4 | 1 | 0.0248445 | 0.41063700000000003 | 0.43548149999999997 | 0.32807478618494706 | 0.08201869654623677 |
| **Parallel** | 4 | 2 | 0.049817 | 0.642997 | 0.692814 | 0.20621768613220864 | 0.05155442153305216 |
| **Parallel** | 4 | 4 | 0.047091499999999994 | 0.535216 | 0.5823075 | 0.24535232673458612 | 0.06133808168364653 |
| **Parallel** | 4 | 8 | 0.0421485 | 0.4804425 | 0.522591 | 0.2733887495192225 | 0.06834718737980562 |

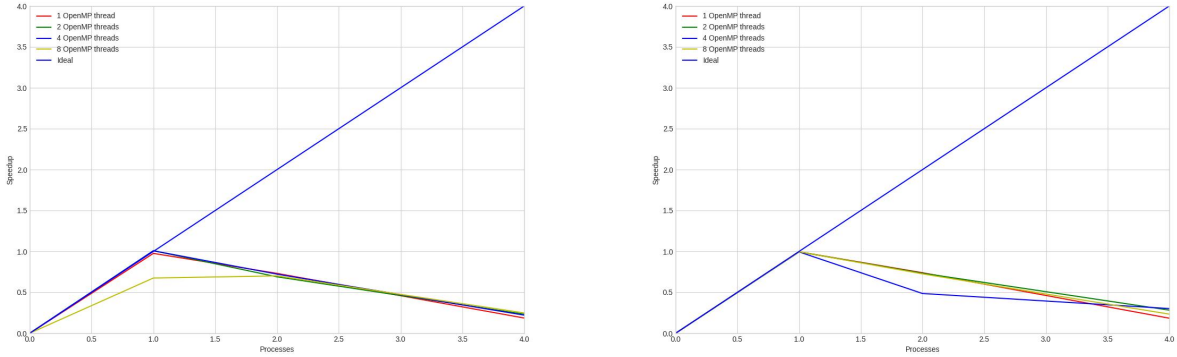
Tabella 38: Measurement with O2 optimization

### 6.1.7 2500 vertices - 25% of density




(a) (b)

Figura 79: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



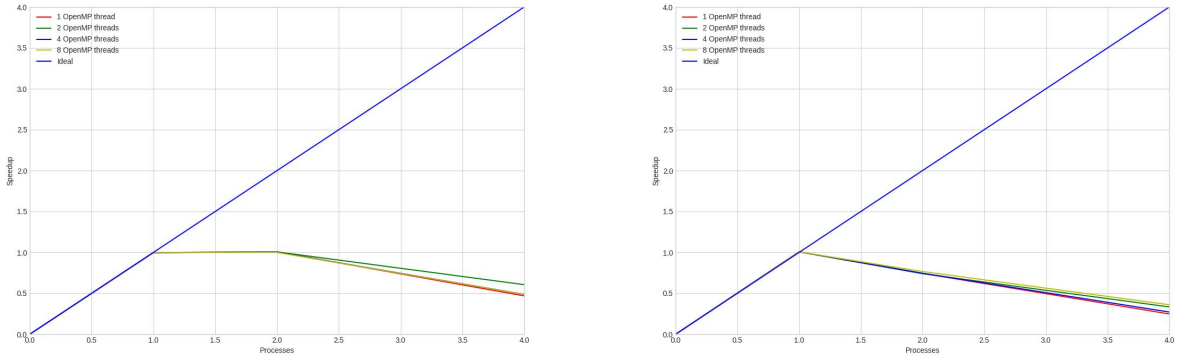

(a) (b)

Figura 80: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.6152565 | 0.0 | 0.6152565 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.615105 | 0.0 | 0.615105 | 1.0002462994122954 | 1.0002462994122954 |
| **Parallel** | 1 | 2 | 0.6156505 | 0.0 | 0.6156505 | 0.9993600265085466 | 0.9993600265085466 |
| **Parallel** | 1 | 4 | 0.6148480000000001 | 0.0 | 0.6148480000000001 | 1.000664391849693 | 1.000664391849693 |
| **Parallel** | 1 | 8 | 0.6158144999999999 | 0.0 | 0.6158144999999999 | 0.9990938829793713 | 0.9990938829793713 |
| **Parallel** | 2 | 1 | 0.2494575 | 0.42329150000000004 | 0.6727485 | 0.9145416154774035 | 0.45727080773870177 |
| **Parallel** | 2 | 2 | 0.373166 | 0.557903 | 0.9310689999999999 | 0.6608065567643214 | 0.3304032783821607 |
| **Parallel** | 2 | 4 | 0.24908 | 0.426975 | 0.6760550000000001 | 0.91006870742765 | 0.455034353713825 |
| **Parallel** | 2 | 8 | 0.2491355 | 0.43075949999999996 | 0.679895 | 0.904928702226079 | 0.4524643511130395 |
| **Parallel** | 4 | 1 | 0.21075349999999998 | 1.293708 | 1.5044615000000001 | 0.40895463260442355 | 0.10223865815110589 |
| **Parallel** | 4 | 2 | 0.220294 | 1.271547 | 1.491841 | 0.4124142586240759 | 0.10310356465601897 |
| **Parallel** | 4 | 4 | 0.2185295 | 0.965278 | 1.1838085 | 0.5197263746627938 | 0.12993159366569845 |
| **Parallel** | 4 | 8 | 0.22529349999999998 | 0.9666995 | 1.191993 | 0.5161578130072911 | 0.12903945325182278 |

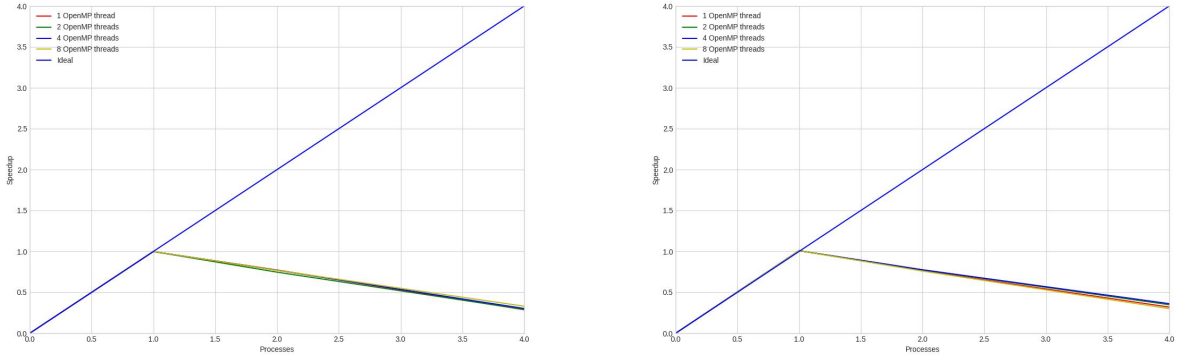
Tabella 39: Measurement with O0 optimization

### 6.1.8 2500 vertices - 50% of density




(a) (b)

Figura 81: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



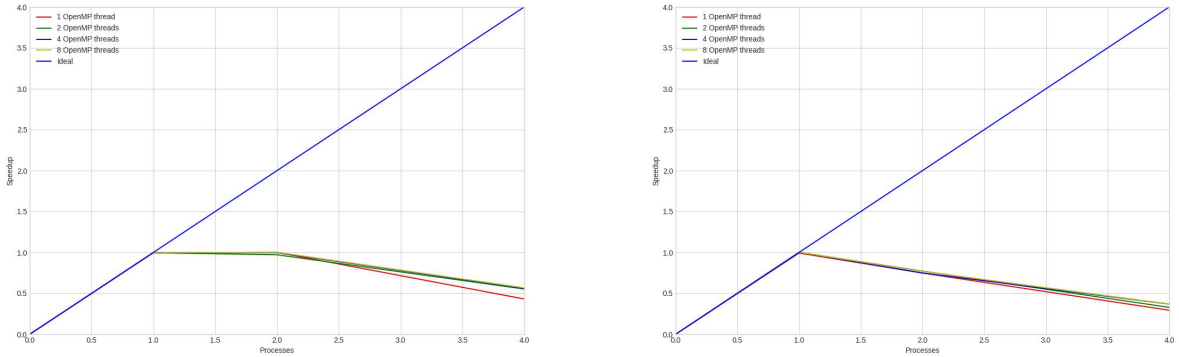

(a) (b)

Figura 82: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 1.905571 | 0.0 | 1.905571 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 1.9094579999999999 | 0.0 | 1.9094579999999999 | 0.9979643438085573 | 0.9979643438085573 |
| **Parallel** | 1 | 2 | 1.9098525 | 0.0 | 1.9098525 | 0.9977582038403489 | 0.9977582038403489 |
| **Parallel** | 1 | 4 | 1.9089645000000002 | 0.0 | 1.9089645000000002 | 0.9982223346741125 | 0.9982223346741125 |
| **Parallel** | 1 | 8 | 1.909032 | 0.0 | 1.909032 | 0.9981870392953077 | 0.9981870392953077 |
| **Parallel** | 2 | 1 | 0.760822 | 1.1317865 | 1.892609 | 1.0068487468885543 | 0.5034243734442772 |
| **Parallel** | 2 | 2 | 0.7601035 | 1.1329395 | 1.893043 | 1.0066179162332816 | 0.5033089581166408 |
| **Parallel** | 2 | 4 | 0.7624025000000001 | 1.137478 | 1.89988 | 1.0029954523443585 | 0.5014977261721792 |
| **Parallel** | 2 | 8 | 0.762183 | 1.1435655 | 1.905748 | 0.9999071230823802 | 0.4999535615411901 |
| **Parallel** | 4 | 1 | 0.38238150000000004 | 3.679025 | 4.0614065 | 0.46918992225969003 | 0.11729748056492251 |
| **Parallel** | 4 | 2 | 0.387339 | 2.7564960000000003 | 3.1438355 | 0.6061293601398674 | 0.15153234003496685 |
| **Parallel** | 4 | 4 | 0.578028 | 3.331068 | 3.9090955000000003 | 0.48747107866768663 | 0.12186776966692166 |
| **Parallel** | 4 | 8 | 0.5860529999999999 | 3.3269539999999997 | 3.9130075 | 0.48698373310043486 | 0.12174593327510871 |

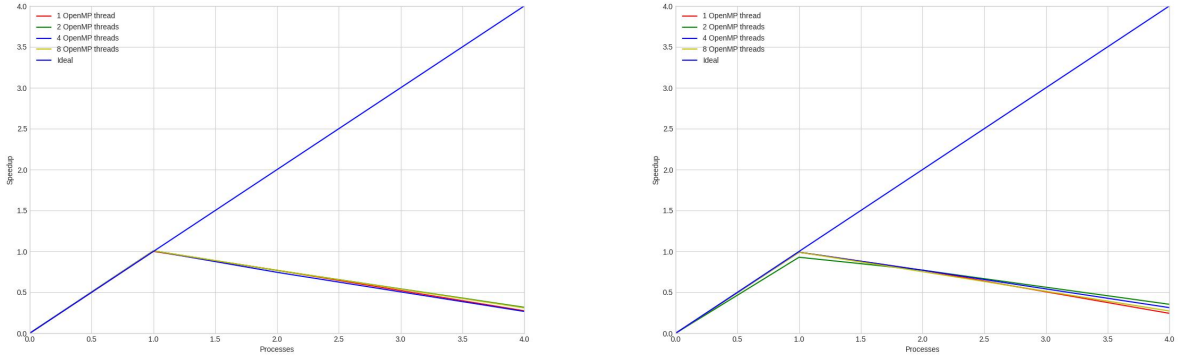
Tabella 40: Measurement with O0 optimization

### 6.1.9 2500 vertices - 75% of density




(a) (b)

Figura 83: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization




(a) (b)

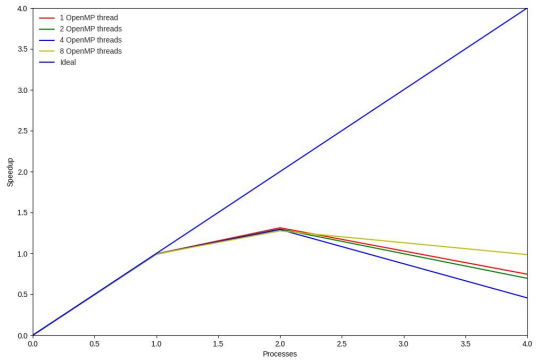
Figura 84: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Tarjan | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 3.085816 | 0.0 | 3.085816 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 3.0966775 | 0.0 | 3.0966775 | 0.9964925311079373 | 0.9964925311079373 |
| **Parallel** | 1 | 2 | 3.097115 | 0.0 | 3.097115 | 0.9963517660790767 | 0.9963517660790767 |
| **Parallel** | 1 | 4 | 3.09511 | 0.0 | 3.09511 | 0.9969971988071505 | 0.9969971988071505 |
| **Parallel** | 1 | 8 | 3.095537 | 0.0 | 3.095537 | 0.9968596724897811 | 0.9968596724897811 |
| **Parallel** | 2 | 1 | 1.229075 | 1.8531795 | 3.082255 | 1.0011553229697088 | 0.5005776614848544 |
| **Parallel** | 2 | 2 | 1.228714 | 1.9479135 | 3.176627 | 0.9714127595087494 | 0.4857063797543747 |
| **Parallel** | 2 | 4 | 1.229393 | 1.8656139999999999 | 3.095007 | 0.997030378283474 | 0.498515189141737 |
| **Parallel** | 2 | 8 | 1.2298225 | 1.8503345 | 3.080157 | 1.0018372440106138 | 0.5009186220053069 |
| **Parallel** | 4 | 1 | 0.8727885 | 6.2820225 | 7.1548110000000005 | 0.4312924548251519 | 0.10782311370628797 |
| **Parallel** | 4 | 2 | 0.6274660000000001 | 4.9531155 | 5.580582 | 0.5529559461719226 | 0.13823898654298064 |
| **Parallel** | 4 | 4 | 0.6245455 | 4.8829815 | 5.5075275 | 0.5602906204281322 | 0.14007265510703304 |
| **Parallel** | 4 | 8 | 0.6212105 | 4.819912 | 5.4411225000000005 | 0.5671285658428017 | 0.14178214146070042 |

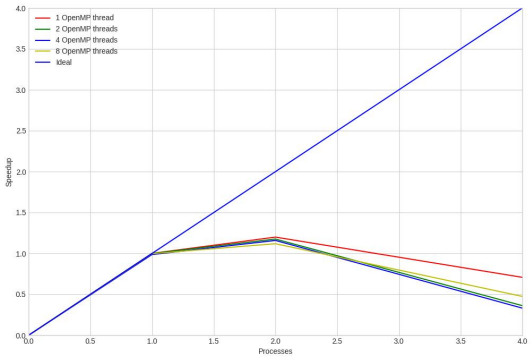
Tabella 41: Measurement with O0 optimization

## 6.2 Kosaraju

### 6.2.1 500 vertices - 25% of density

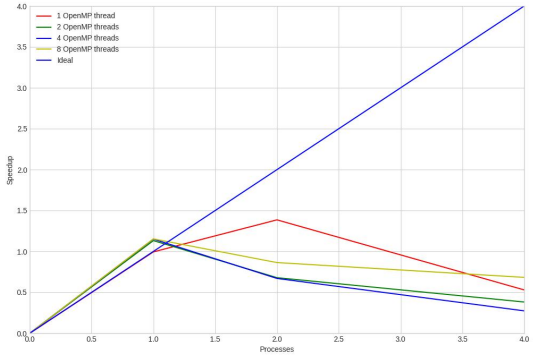



(a)

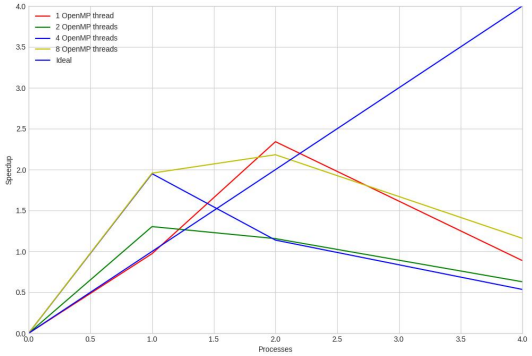



(b)

Figura 85: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization




(a)



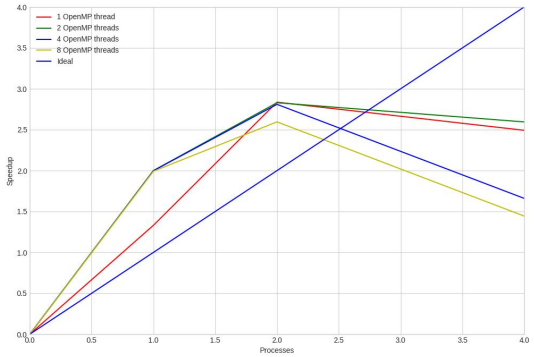

(b)

caption(a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

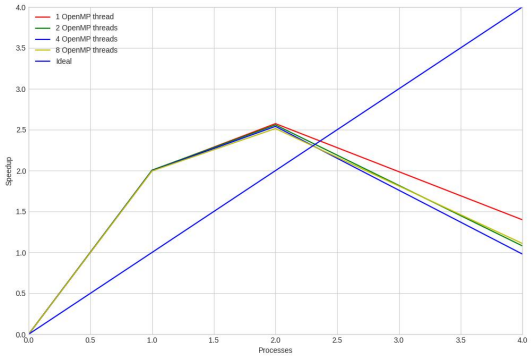
| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.053664500000000004 | 0.0 | 0.053664500000000004 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.055174 | 0.0 | 0.055174 | 0.9726410990684018 | 0.9726410990684018 |
| **Parallel** | 1 | 2 | 0.0411665 | 0.0 | 0.0411665 | 1.3035963708355094 | 1.3035963708355094 |
| **Parallel** | 1 | 4 | 0.027512500000000002 | 0.0 | 0.027512500000000002 | 1.9505497501135847 | 1.9505497501135847 |
| **Parallel** | 1 | 8 | 0.0274215 | 0.0 | 0.0274215 | 1.9570227741006145 | 1.9570227741006145 |
| **Parallel** | 2 | 1 | 0.0139325 | 0.008985 | 0.0229175 | 2.3416384858732413 | 1.1708192429366207 |
| **Parallel** | 2 | 2 | 0.027561000000000002 | 0.018835499999999998 | 0.0463965 | 1.1566497472869721 | 0.5783248736434861 |
| **Parallel** | 2 | 4 | 0.0276155 | 0.019523 | 0.0471385 | 1.1384430985288034 | 0.5692215492644017 |
| **Parallel** | 2 | 8 | 0.0138595 | 0.010735 | 0.024593999999999998 | 2.1820159388468734 | 1.0910079694234367 |
| **Parallel** | 4 | 1 | 0.0182365 | 0.0421695 | 0.060406 | 0.8883968479952323 | 0.22209921199880808 |
| **Parallel** | 4 | 2 | 0.0183415 | 0.0669215 | 0.085263 | 0.6293996223449797 | 0.15734990558624493 |
| **Parallel** | 4 | 4 | 0.0181445 | 0.08203450000000001 | 0.10017899999999999 | 0.5356861218419031 | 0.13392153046047578 |
| **Parallel** | 4 | 8 | 0.0135335 | 0.0327075 | 0.046241000000000004 | 1.160539348197487 | 0.29013483704937176 |

Tabella 42: Measurement with O3 optimization

### 6.2.2 500 vertices - 50% of density

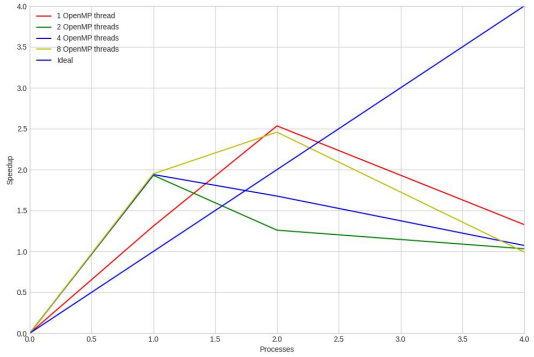



(c)

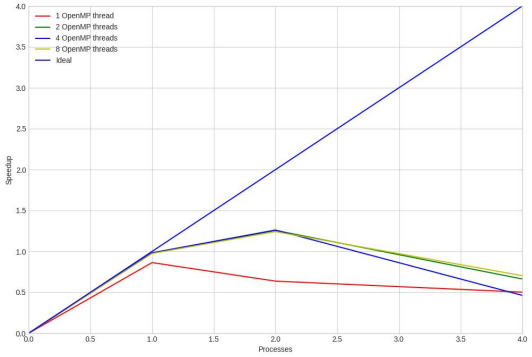



(d)

Figura 86: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization




(a)



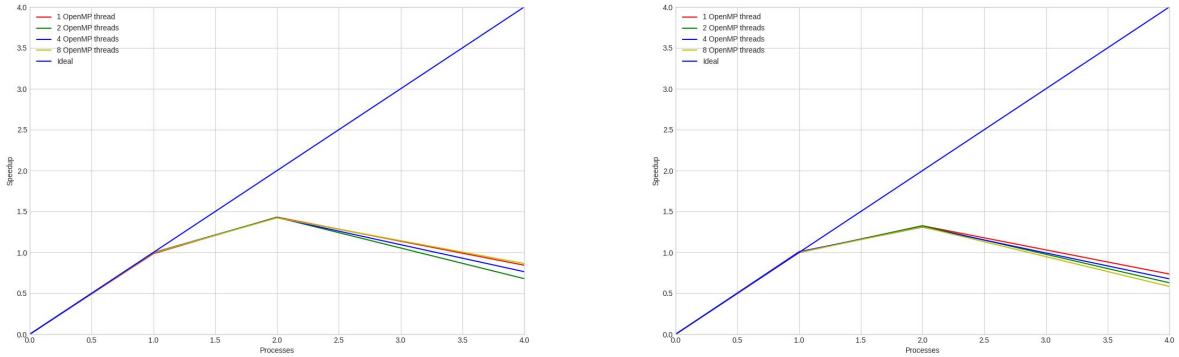

(b)

Figura 87: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.471223 | 0.0 | 0.471223 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.3541485 | 0.0 | 0.3541485 | 1.3305802509399307 | 1.3305802509399307 |
| **Parallel** | 1 | 2 | 0.235648 | 0.0 | 0.235648 | 1.9996902159152634 | 1.9996902159152634 |
| **Parallel** | 1 | 4 | 0.2357785 | 0.0 | 0.2357785 | 1.9985834162147949 | 1.9985834162147949 |
| **Parallel** | 1 | 8 | 0.236559 | 0.0 | 0.236559 | 1.9919893134482307 | 1.9919893134482307 |
| **Parallel** | 2 | 1 | 0.123045 | 0.043080499999999994 | 0.16612500000000002 | 2.8365568096313014 | 1.4182784048156507 |
| **Parallel** | 2 | 2 | 0.12282599999999999 | 0.043676 | 0.1665025 | 2.830125673788682 | 1.415062836894341 |
| **Parallel** | 2 | 4 | 0.1234895 | 0.044186 | 0.16767549999999998 | 2.810327090123483 | 1.4051635450617415 |
| **Parallel** | 2 | 8 | 0.131428 | 0.050095 | 0.1815235 | 2.595933859803276 | 1.297966929901638 |
| **Parallel** | 4 | 1 | 0.0762455 | 0.11279549999999999 | 0.18904100000000001 | 2.492702641225977 | 0.6231756603064943 |
| **Parallel** | 4 | 2 | 0.0762235 | 0.10535749999999999 | 0.181581 | 2.595111823373591 | 0.6487779558433977 |
| **Parallel** | 4 | 4 | 0.113893000000000001 | 0.169828 | 0.2837215 | 1.6608646154767968 | 0.4152161538691992 |
| **Parallel** | 4 | 8 | 0.1516865 | 0.174523 | 0.32621 | 1.4445387940283867 | 0.3611346985070967 |

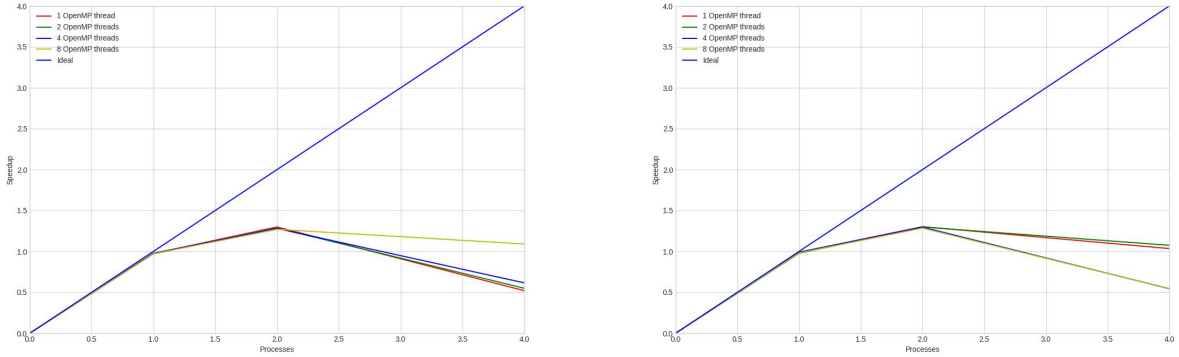
Tabella 43: Measurement with O0 optimization

### 6.2.3 500 vertices - 75% of density




(a) (b)

Figura 88: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



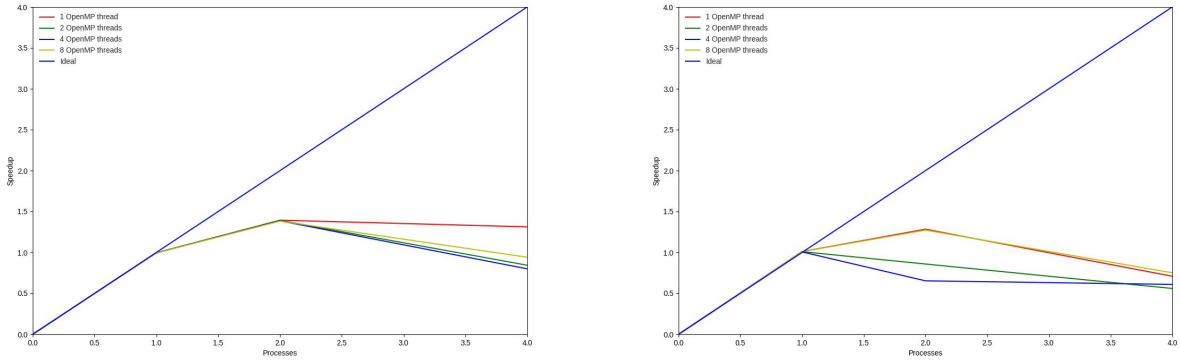

(a) (b)

Figura 89: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.151996 | 0.0 | 0.151996 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.152389 | 0.0 | 0.152389 | 0.9974210736995452 | 0.9974210736995452 |
| **Parallel** | 1 | 2 | 0.151681 | 0.0 | 0.151681 | 1.0020767268148283 | 1.0020767268148283 |
| **Parallel** | 1 | 4 | 0.150752 | 0.0 | 0.150752 | 1.008251963489705 | 1.008251963489705 |
| **Parallel** | 1 | 8 | 0.1521115 | 0.0 | 0.1521115 | 0.9992406885738421 | 0.9992406885738421 |
| **Parallel** | 2 | 1 | 0.07499349999999999 | 0.0396055 | 0.1145985 | 1.3263349869326386 | 0.6631674934663193 |
| **Parallel** | 2 | 2 | 0.074418 | 0.040293499999999996 | 0.1147115 | 1.3250284409148168 | 0.6625142204574084 |
| **Parallel** | 2 | 4 | 0.07475499999999999 | 0.041286 | 0.116041 | 1.3098473815289422 | 0.6549236907644711 |
| **Parallel** | 2 | 8 | 0.07450999999999999 | 0.0413035 | 0.1158135 | 1.312420400039719 | 0.6562102000198595 |
| **Parallel** | 4 | 1 | 0.064689 | 0.14184449999999998 | 0.20653299999999997 | 0.7359405034546538 | 0.18398512586366345 |
| **Parallel** | 4 | 2 | 0.0659215 | 0.175603 | 0.24152500000000002 | 0.6293178759962736 | 0.1573294689990684 |
| **Parallel** | 4 | 4 | 0.065265 | 0.1591 | 0.2243655 | 0.6774481816500308 | 0.1693620454125077 |
| **Parallel** | 4 | 8 | 0.085883 | 0.174058 | 0.2599405 | 0.584733814084377 | 0.14618345352109424 |

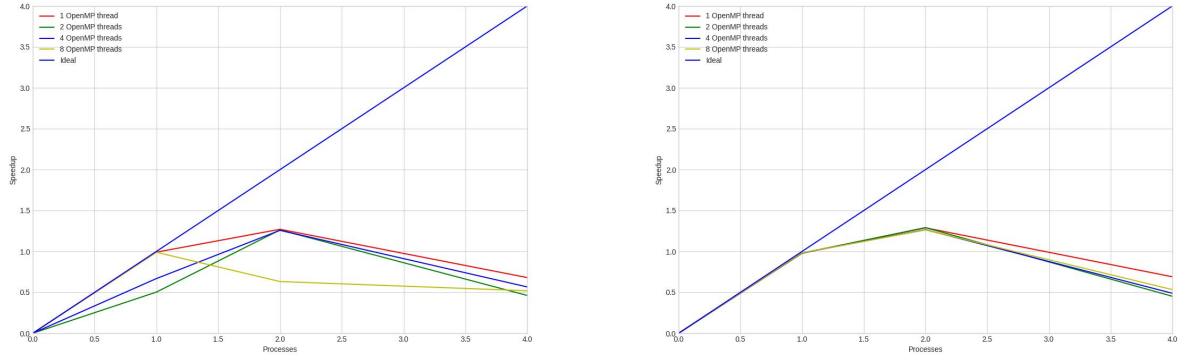
Tabella 44: Measurement with O1 optimization

### 6.2.4 1000 vertices - 25% of density




(a) (b)

Figura 90: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



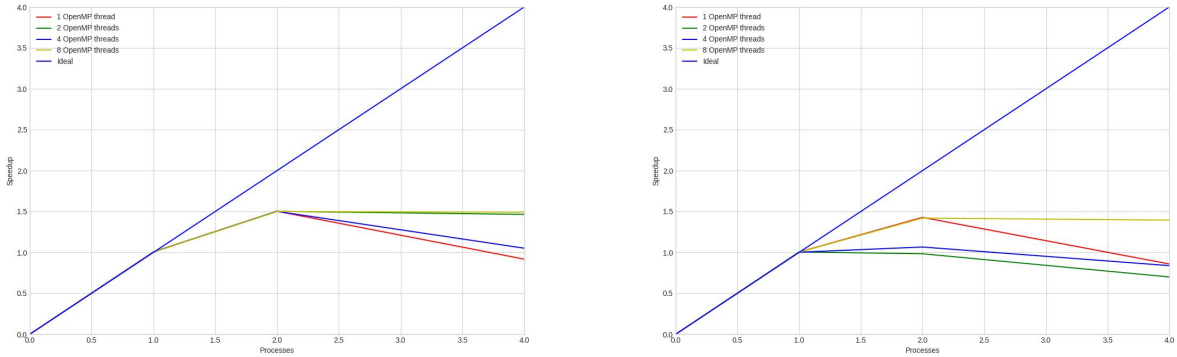

(a) (b)

Figura 91: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.1289945 | 0.0 | 0.1289945 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.1276515 | 0.0 | 0.1276515 | 1.0105208321092978 | 1.0105208321092978 |
| **Parallel** | 1 | 2 | 0.12778099999999998 | 0.0 | 0.12778099999999998 | 1.0094967170393097 | 1.0094967170393097 |
| **Parallel** | 1 | 4 | 0.1281655 | 0.0 | 0.1281655 | 1.00646819932041 | 1.00646819932041 |
| **Parallel** | 1 | 8 | 0.1277915 | 0.0 | 0.1277915 | 1.0094137716514793 | 1.0094137716514793 |
| **Parallel** | 2 | 1 | 0.06372900000000001 | 0.0367405 | 0.10047 | 1.2839106200855979 | 0.6419553100427989 |
| **Parallel** | 2 | 2 | 0.0943875 | 0.0558795 | 0.1502665 | 0.8584381748426962 | 0.4292190874213481 |
| **Parallel** | 2 | 4 | 0.1256485 | 0.071918 | 0.197566 | 0.6529185183685453 | 0.32645925918427265 |
| **Parallel** | 2 | 8 | 0.0639415 | 0.0374 | 0.10134199999999999 | 1.272863176175722 | 0.636431588087861 |
| **Parallel** | 4 | 1 | 0.0601815 | 0.12174399999999999 | 0.181926 | 0.7090492837747217 | 0.1772623209436804 |
| **Parallel** | 4 | 2 | 0.0802545 | 0.15014850000000002 | 0.2304025 | 0.5598658868718873 | 0.13996647171797183 |
| **Parallel** | 4 | 4 | 0.0698685 | 0.14183199999999999 | 0.2117005 | 0.6093254385322662 | 0.15233135963306654 |
| **Parallel** | 4 | 8 | 0.0661435 | 0.1055565 | 0.1717005 | 0.7512762047868237 | 0.18781905119670592 |

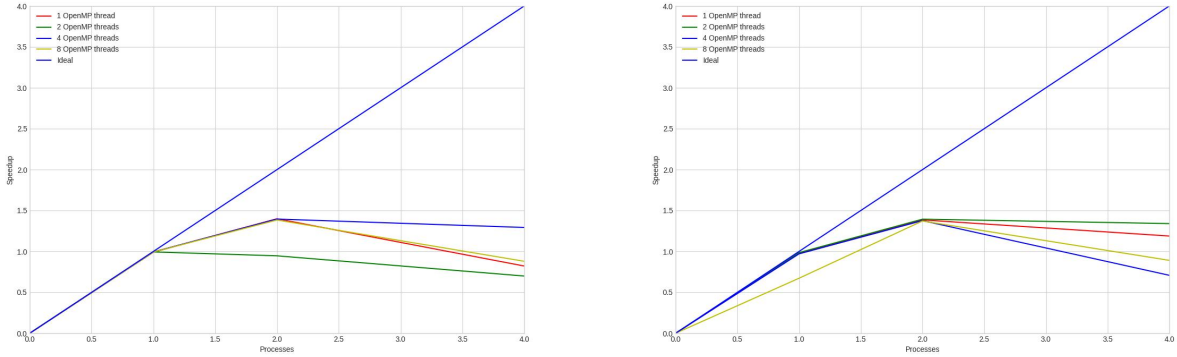
Tabella 45: Measurement with O1 optimization

### 6.2.5 1000 vertices - 50% of density




(a) (b)

Figura 92: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



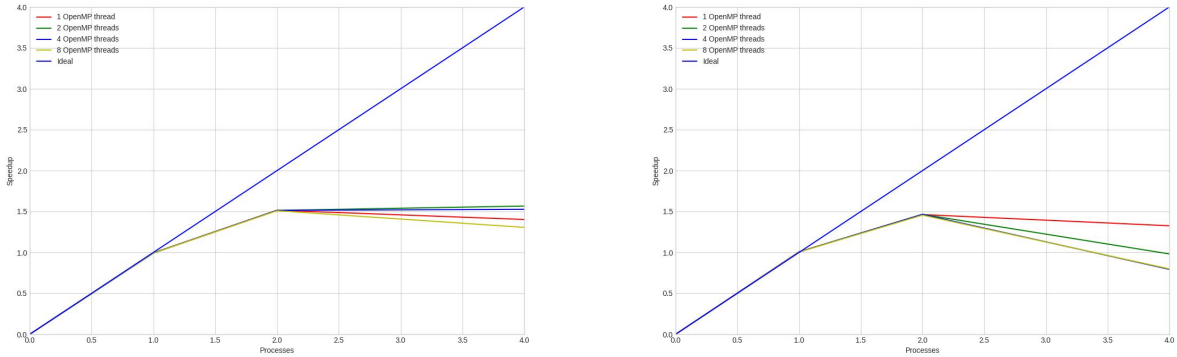

(a) (b)

Figura 93: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 1.178648 | 0.0 | 1.178648 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 1.1710034999999999 | 0.0 | 1.1710034999999999 | 1.0065281615298332 | 1.0065281615298332 |
| **Parallel** | 1 | 2 | 1.172132 | 0.0 | 1.172132 | 1.0055591008521225 | 1.0055591008521225 |
| **Parallel** | 1 | 4 | 1.1711385 | 0.0 | 1.1711385 | 1.0064121365662557 | 1.0064121365662557 |
| **Parallel** | 1 | 8 | 1.172515 | 0.0 | 1.172515 | 1.0052306367082724 | 1.0052306367082724 |
| **Parallel** | 2 | 1 | 0.6073105000000001 | 0.1765045 | 0.7838145000000001 | 1.5037333450708041 | 0.7518666725354021 |
| **Parallel** | 2 | 2 | 0.6066925 | 0.1774705 | 0.7841629999999999 | 1.5030650515262771 | 0.7515325257631386 |
| **Parallel** | 2 | 4 | 0.6069125 | 0.1776935 | 0.784606 | 1.5022163990588906 | 0.7511081995294453 |
| **Parallel** | 2 | 8 | 0.6070655 | 0.1782845 | 0.7853495 | 1.500794232376795 | 0.7503971161883976 |
| **Parallel** | 4 | 1 | 0.552286 | 0.731128 | 1.2834135 | 0.9183696447014154 | 0.22959241117535384 |
| **Parallel** | 4 | 2 | 0.381012 | 0.4236565 | 0.8046679999999999 | 1.464763107269085 | 0.36619077681727125 |
| **Parallel** | 4 | 4 | 0.555158 | 0.5655195 | 1.1206775 | 1.0517280841276817 | 0.2629320210319204 |
| **Parallel** | 4 | 8 | 0.369267 | 0.4210115 | 0.790278 | 1.4914346597020287 | 0.3728586649255072 |

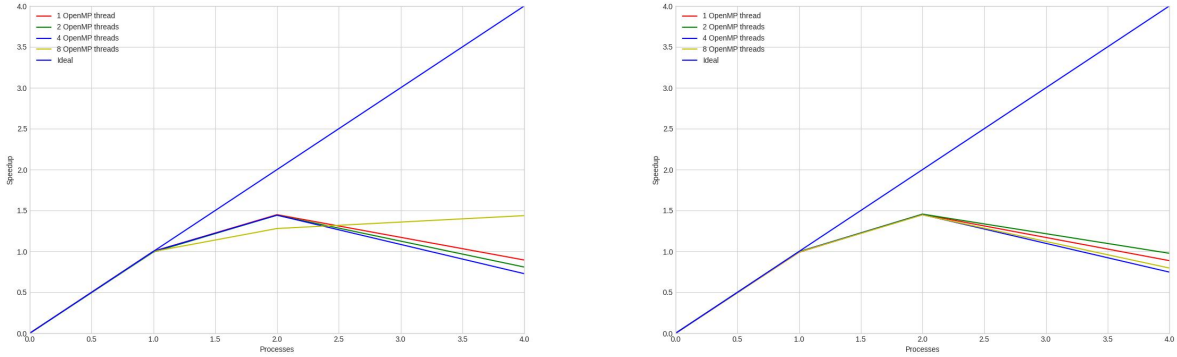
Tabella 46: Measurement with O0 optimization

### 6.2.6 1000 vertices - 75% of density




(a) (b)

Figura 94: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



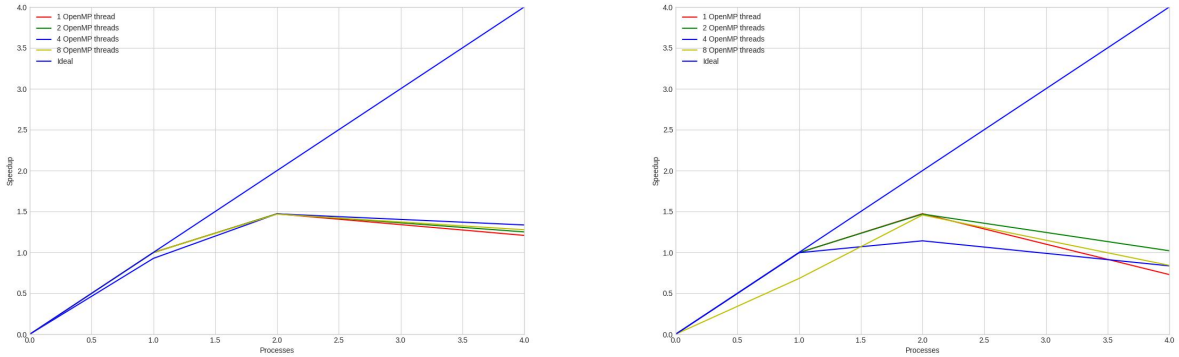

(a) (b)

Figura 95: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 0.916936 | 0.0 | 0.916936 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 0.9102635 | 0.0 | 0.9102635 | 1.0073302950189698 | 1.0073302950189698 |
| **Parallel** | 1 | 2 | 0.9110935 | 0.0 | 0.9110935 | 1.0064126239513287 | 1.0064126239513287 |
| **Parallel** | 1 | 4 | 0.9105989999999999 | 0.0 | 0.9105989999999999 | 1.0069591554570123 | 1.0069591554570123 |
| **Parallel** | 1 | 8 | 0.911638 | 0.0 | 0.911638 | 1.0058115172908546 | 1.0058115172908546 |
| **Parallel** | 2 | 1 | 0.463241 | 0.163566 | 0.6268075 | 1.46286698866877 | 0.731433494334385 |
| **Parallel** | 2 | 2 | 0.4628645 | 0.1628425 | 0.625707 | 1.4654398943914644 | 0.7327199471957322 |
| **Parallel** | 2 | 4 | 0.462677 | 0.1630565 | 0.6257334999999999 | 1.4653778325756892 | 0.7326889162878446 |
| **Parallel** | 2 | 8 | 0.4664355 | 0.16329749999999998 | 0.6297335 | 1.4560699089376699 | 0.7280349544688349 |
| **Parallel** | 4 | 1 | 0.28684350000000003 | 0.40466199999999997 | 0.691505 | 1.3260005350648223 | 0.3315001337662056 |
| **Parallel** | 4 | 2 | 0.3973905 | 0.5364445 | 0.9338345 | 0.9819041810941874 | 0.24547604527354686 |
| **Parallel** | 4 | 4 | 0.49897650000000004 | 0.6567805 | 1.155757 | 0.793364002986787 | 0.19834100074669675 |
| **Parallel** | 4 | 8 | 0.48768100000000003 | 0.6594599999999999 | 1.1471405 | 0.7993231866541196 | 0.1998307966635299 |

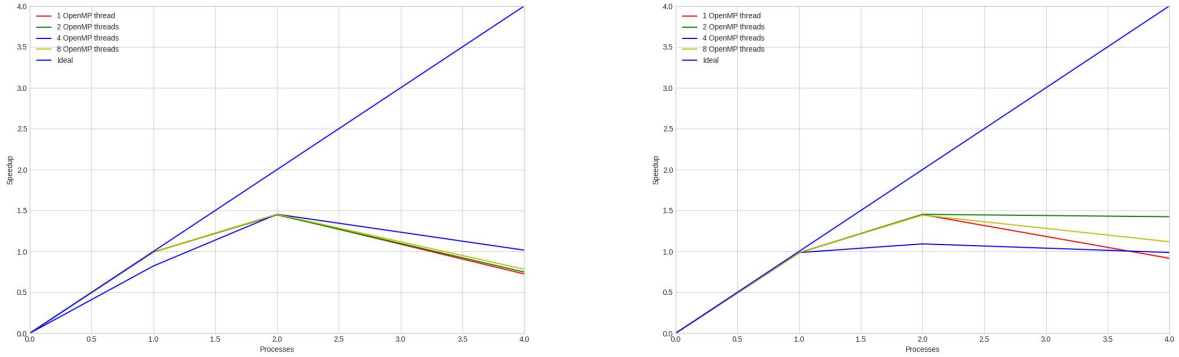
Tabella 47: Measurement with O1 optimization

### 6.2.7 2500 vertices - 25% of density




(a) (b)

Figura 96: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



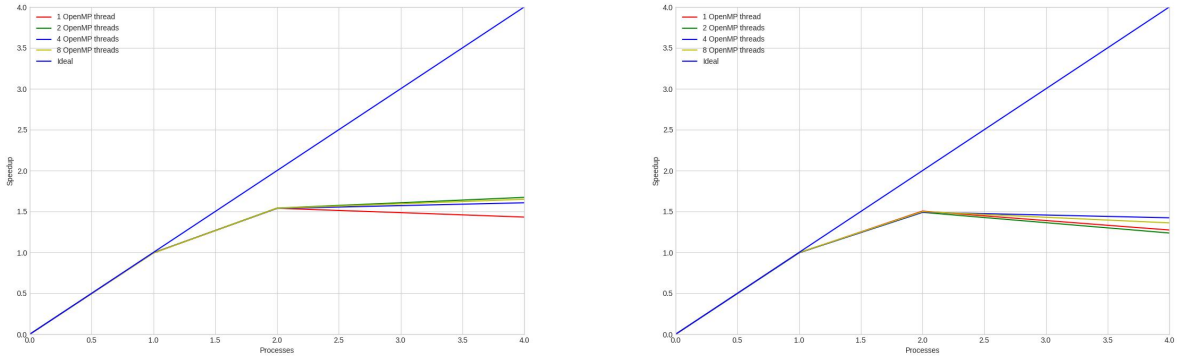

(a) (b)

Figura 97: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 1.262461 | 0.0 | 1.262461 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 1.266 | 0.0 | 1.266 | 0.9972045813586098 | 0.9972045813586098 |
| **Parallel** | 1 | 2 | 1.265112 | 0.0 | 1.265112 | 0.9979045333535688 | 0.9979045333535688 |
| **Parallel** | 1 | 4 | 1.266654 | 0.0 | 1.266654 | 0.9966897037391428 | 0.9966897037391428 |
| **Parallel** | 1 | 8 | 1.8521855 | 0.0 | 1.8521855 | 0.681606135022653 | 0.681606135022653 |
| **Parallel** | 2 | 1 | 0.6389245 | 0.217843 | 0.8567675 | 1.473516444076135 | 0.7367582220380675 |
| **Parallel** | 2 | 2 | 0.6406229999999999 | 0.2188435 | 0.8594665 | 1.4688891306409266 | 0.7344445653204633 |
| **Parallel** | 2 | 4 | 0.8784595 | 0.22688950000000002 | 1.105349 | 1.1421379130030427 | 0.5710689565015213 |
| **Parallel** | 2 | 8 | 0.6418265000000001 | 0.22433999999999998 | 0.8661665000000001 | 1.4575269304458207 | 0.7287634652229104 |
| **Parallel** | 4 | 1 | 0.6949375 | 1.035975 | 1.730913 | 0.7293613254970066 | 0.18234033137425165 |
| **Parallel** | 4 | 2 | 0.563077 | 0.673924 | 1.2370014999999999 | 1.0205816241936652 | 0.2551454060484163 |
| **Parallel** | 4 | 4 | 0.7204385 | 0.7900445 | 1.510483 | 0.8357995422656197 | 0.20894988556640492 |
| **Parallel** | 4 | 8 | 0.7134395 | 0.7856515 | 1.4990915 | 0.8421507292917078 | 0.21053768232292694 |

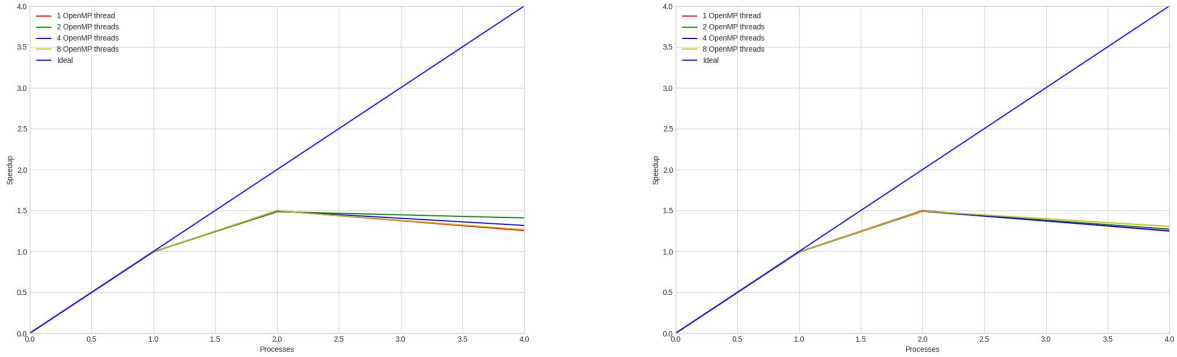
Tabella 48: Measurement with O1 optimization

### 6.2.8 2500 vertices - 50% of density




(a) (b)

Figura 98: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization



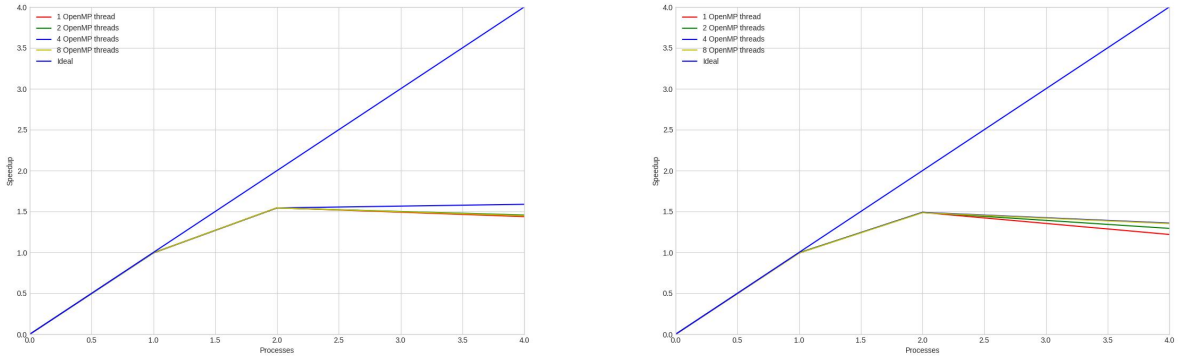

(a) (b)

Figura 99: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 10.194613499999999 | 0.0 | 10.194613499999999 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 10.258692 | 0.0 | 10.258692 | 0.9937537358563839 | 0.9937537358563839 |
| **Parallel** | 1 | 2 | 10.247829 | 0.0 | 10.247829 | 0.9948071440302136 | 0.9948071440302136 |
| **Parallel** | 1 | 4 | 10.2489585 | 0.0 | 10.2489585 | 0.9946975099957717 | 0.9946975099957717 |
| **Parallel** | 1 | 8 | 10.2670505 | 0.0 | 10.2670505 | 0.9929447118235173 | 0.9929447118235173 |
| **Parallel** | 2 | 1 | 5.4724725 | 1.146126 | 6.618598 | 1.5402980359284546 | 0.7701490179642273 |
| **Parallel** | 2 | 2 | 5.4775955 | 1.135443 | 6.6130385 | 1.541592945512112 | 0.770796472756056 |
| **Parallel** | 2 | 4 | 5.482314000000001 | 1.1479835 | 6.6302975 | 1.5375801010437313 | 0.7687900505218657 |
| **Parallel** | 2 | 8 | 5.478479 | 1.1426150000000002 | 6.6210945 | 1.5397172627576905 | 0.7698586313788452 |
| **Parallel** | 4 | 1 | 3.4221494999999997 | 3.6953110000000002 | 7.1174605 | 1.432338612908354 | 0.3580846532270885 |
| **Parallel** | 4 | 2 | 3.4129625 | 2.680319 | 6.0932815 | 1.6730908460408402 | 0.41827271151021006 |
| **Parallel** | 4 | 4 | 3.4194295 | 2.9244665000000003 | 6.343896 | 1.6069956853012721 | 0.40174892132531803 |
| **Parallel** | 4 | 8 | 3.4281699999999997 | 2.749977 | 6.178146 | 1.6501088676117397 | 0.4125272169029349 |

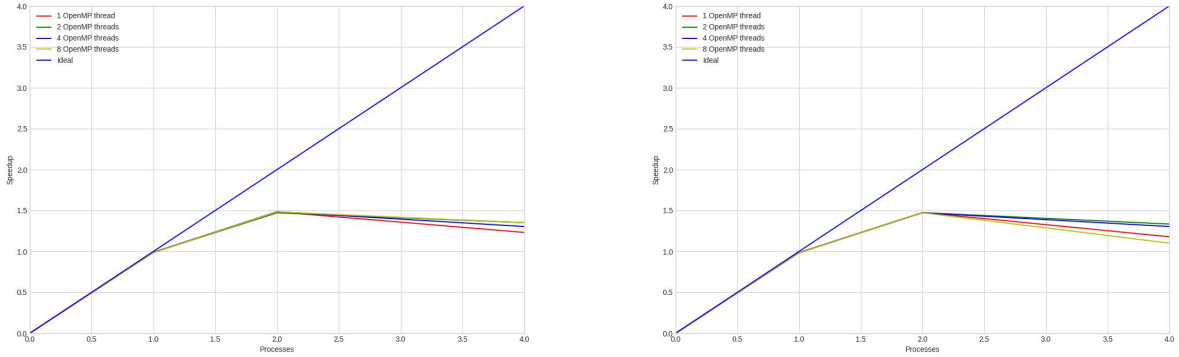
Tabella 49: Measurement with O0 optimization

### 6.2.9 2500 vertices - 75% of density




(a) (b)

Figura 100: (a) Speedup with no optimization - (b) Speedup with $O1$ optimization




(a) (b)

Figura 101: (a) Speedup with $O2$ optimization - (b) Speedup with $O3$ optimization

| Version | Processes | OMP | Kosaraju | Communication | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|
| **Serial** | 1 | 0 | 18.0724475 | 0.0 | 18.0724475 | 1.0 | 1.0 |
| **Parallel** | 1 | 1 | 18.170600999999998 | 0.0 | 18.170600999999998 | 0.9945982249018621 | 0.9945982249018621 |
| **Parallel** | 1 | 2 | 18.165741 | 0.0 | 18.165741 | 0.9948643162973643 | 0.9948643162973643 |
| **Parallel** | 1 | 4 | 18.18484 | 0.0 | 18.18484 | 0.9938194397091202 | 0.9938194397091202 |
| **Parallel** | 1 | 8 | 18.171785999999997 | 0.0 | 18.171785999999997 | 0.9945333661754547 | 0.9945333661754547 |
| **Parallel** | 2 | 1 | 9.831790999999999 | 1.861803 | 11.6935945 | 1.5454997605740475 | 0.7727498802870237 |
| **Parallel** | 2 | 2 | 9.825023 | 1.8783055000000002 | 11.703328500000001 | 1.5442143233012726 | 0.7721071616506363 |
| **Parallel** | 2 | 4 | 9.834575000000001 | 1.869847 | 11.704422000000001 | 1.5440700531816092 | 0.7720350265908046 |
| **Parallel** | 2 | 8 | 9.8256115 | 1.868599 | 11.69421 | 1.545418416464216 | 0.772709208232108 |
| **Parallel** | 4 | 1 | 6.213698 | 6.3526495 | 12.5663475 | 1.4381623220271444 | 0.3595405805067861 |
| **Parallel** | 4 | 2 | 6.7782915 | 5.627601500000001 | 12.405892999999999 | 1.4567631286196003 | 0.3641907821549001 |
| **Parallel** | 4 | 4 | 6.5868285 | 4.7915735 | 11.378401499999999 | 1.5883116358655476 | 0.3970779089663869 |
| **Parallel** | 4 | 8 | 6.222115 | 6.2490665 | 12.471181000000001 | 1.4491368139071992 | 0.3622842034767998 |

Tabella 50: Measurement with O0 optimization

# 7 Final Considerations

In this section we will analyze what were the results following the various measurements made. In particular, as per trace, both MPI and OpenMP have been used in the development. As far as MPI is concerned, this has been used in order to divide the work among several processes with the aim of improving performance compared to the serial algorithm, thus exploiting data-parallelism. With regard to OpenMP, on the other hand, during the writing and testing of the code's performance, following careful analyses, some portions of the code have been identified where this has brought considerable benefits, but also others where the use of this approach does not it did nothing but slow down the execution. In particular, for the two algorithms that compute the strictly connected components used (Tarjan and Kosaraju) the use of OpenMP was not advantageous given their recursive nature. The problem here is that both algorithms perform one or more dfs and the entire body of the dfs is one critical section, which means that even if there are 1000 recursive calls in parallel, they will execute one after another sequentially and not in parallel. It will even be slower than the sequential version because of the constant cache invalidation and the added OpenMP overhead. Conversely, advantages have been obtained by parallelizing the code snippet for the creation of the new graph by a process following receipt of the necessary information from the previous process, as well as the functions for serializing the data structures before being transmitted to the other processes.

## 7.1 Tarjan Implementation

Considering the results previously obtained, when it comes to parallelizing Tarjan's algorithm, the number of processes used plays a crucial role in the overall performance of the algorithm. As the number of processes increases, the overall execution time of the algorithm decreases. However, as the number of processes grows up, the communication overhead also increases. This is because each process has to communicate with the other processes to share information about his graph and the strongly connected components found.

This phenomenon can be observed particularly when the number of vertices and the density of the graph increase. In sparse graphs, the algorithm would have to spend most of its time waiting for the next non-trivial component to be processed, which could lead to increased overhead and synchronization costs when using MPI. Additionally, if the graph is very sparse, the overhead of parallelism and communication might be higher than the overhead of the serial version, leading to slower performance.

In particular, the best performance is achieved when using two processes. This is because the communication overhead is minimized while still achieving a significant speedup. Consequently, as the number of nodes in the graph increases, better and better performance will be obtained using two MPI processes.

However, when the graph has a density of 0%, the parallel version of the algorithm may not be as efficient as the serial version. This is because, in this type of graph, the number of strongly connected components is equal to the number of vertices, and there are no macro-components to be found. This means that the graph never decreases in size and the last iteration of the parallel algorithm will perform Tarjan on the exact same graph as the initial one, leading to no speedup compared to the serial version.

Let's consider an example of a graph with 100 vertices and a density of 0%. In this case, there are no edges between the vertices, meaning that each vertex is a separate strongly connected component.

- ***Serial version***: The algorithm will start from one vertex and perform the depth-first search to find the strongly connected components. Since there are no edges between the vertices, the algorithm will find that each vertex is a separate strongly connected component. The total execution time of the algorithm will be the time it takes to perform the Tarjan's algorithm.
- ***Parallel version with 4 processes***: The graph is divided among the 4 processes and each process performs the Tarjan's algorithm on its part of the graph. However, since the graph is sparse and there are no edges between the vertices, each process will find that each vertex of its part of the graph is a separate strongly connected component. At this point, when the processes will exchange the various information, the remaining processes will run Tarjan on a graph that never tends to get smaller, up to the last iteration, in which the last process will find itself at run Tarjan on the starting graph. Therefore, in this case, the execution time will be greater since the time required for the last process to execute Tarjan on the final graph (which is therefore

equivalent to the initial one) will be added to the Tarjan execution times of the other processes up to that moment.

## 7.2 Kosaraju Implementation

Same as Tarjan's algorithm, when it comes to parallelizing Kosaraju's algorithm, the number of processes used plays a crucial role in the overall performance of the algorithm. Kosaraju's algorithm benefits a lot from parallelism and it scales well with the number of processes used. As the number of processes increases, the overall execution time of the algorithm decreases significantly.

The communication overhead does increase as the number of processes increases, but the speedup that the algorithm achieves by using more processes compensates for this overhead. This is not the case with Tarjan's algorithm, where the parallel version does not have the same scaling as the serial version, and the communication overhead is not compensated in the same way.

In contrast to Tarjan, as the number of nodes increases in Kosaraju, performance is also observed to improve with an increase in the number of processes.

Regarding optimization levels, as the algorithm is already sufficiently optimized, no significant benefits are observed even when compiled with different levels of optimization.

## 7.3 Cluster execution

As already discussed in chapter 6, in addition to the measurements taken on our machine, the same measurements were taken on a Raspberry Pi cluster, where the front-end node is represented by a Raspberry Pi 3, while the worker nodes are Raspberry Pi 4s.

Initially, we were unaware that the sequential program was run on the cluster frontend, which is a Raspberry Pi 3. In contrast, the parallel algorithm was executed on the cluster workers, which are Raspberry Pi 4. This is the reason why in some measurements taken on the cluster, an abnormal behavior is observed in which some speedup curves are above the ideal line, as the sequential algorithm was run on an older device. Upon examining a helpful table, we discovered that the Raspberry Pi 3 contains a Broadcom BCM2837B0 quad core Cortex A53 processor with a clocking speed of 1.4GHz whereas on the other hand the Raspberry Pi 4 model has a Broadcom BCM2711 quad core Cortex-A72 processor with the clock speeds at 1.5GHz. Therefore, Raspberry Pi 4 has 10% faster processing performance in comparison to Raspberry Pi 3.

Despite the presence of this inconsistency, since the time available for using the cluster had expired, it was not possible to repeat the measurements in a uniform manner.

# 8 Test Case

The code provided in the *test* folder is a test program that compares the correctness of two different algorithms for finding strongly connected components (SCCs) in a directed graph. Six test cases have been carried out, which consider two different levels of number of vertices (100 and 1000), each with an increasing level of edges.

The program first includes the "TestUtil.h" header file, which likely contains utility functions used in the testing process. Then, the program defines some macro constants, such as the number of vertices in the test graph, and the lower and upper bounds for the edges.

The main function first allocates memory for a Results struct and uses the generateGraph function to create a graph with the specified number of vertices, lower and upper bounds, and writes it to a file. Next, the program runs the Tarjan algorithm in both sequential and parallel versions using the "system" function, which allows for

execution of command-line commands. The program then reads the results of the sequential and parallel versions of the algorithm and stores them in the Results struct.

The program then calls the compareResults function to compare the results of the two versions of the algorithm. The function begins by asserting that the number of SCCs found by the sequential and parallel versions of the algorithm is the same. This ensures that both versions of the algorithm have found the same number of SCCs in the graph, which is an important condition for the validity of the comparison. Then the function iterates over the SCCs found by the sequential version of the algorithm and for each SCC it asserts that the length of the SCC found by the parallel version is the same. It also iterates over the items of each SCC, and for each item, it asserts that the item found by the parallel version is the same as the item found by the sequential version.

This comparison ensures that the SCCs found by the parallel version of the algorithm are the same as the SCCs found by the sequential version of the algorithm. This is important because it shows that the parallel version of the algorithm is producing the same results as the sequential version, which means that it is working correctly. After that, the program frees the memory allocated for the Results struct, removes the output files, and repeats the process for the Kosaraju algorithm.

# 9 Api Documentation

## 9.1 include/tarjan.h File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <omp.h>

#include "../include/Tarjan.h"

#define NOT_INITIALIZED -1
#define PARAMETERS_NUM 3
```

Functions:

- `void sccUtil(TGraph, int, THTAuxiliarySCC*, TStack*, int*, int, int, int*, TArray*);`
- `TArray* scc(TGraph, int, int, int*);`

---

### *Function documentation:*

```
void sccUtil(TGraph graph, int u, THTAuxiliarySCC *auxiliaryHT, TStack *st, int *time, int
start, int stop, int *sccCount, TArray* sccs)
```

This is a helper function for the SCC() function. It performs the recursive DFS traversal of the graph to find the SCCs.

### *Parameters:*

| | |
|---|---|
| graph | Pointer to the graph structure |
| u | The current node being visited |
| auxiliaryHT | Pointer to a hash table that stores auxiliary information for each node |
| st | Pointer to a stack used to track nodes during the SCC search |
| time | Pointer to a variable that tracks the current time |
| start | The index of the first node in the subgraph being searched |
| stop | The index of the last node in the subgraph being searched |
| sccCount | Pointer to a variable that tracks the number of SCCs found |
| sccs | Pointer to an array of nodes representing the SCCs found |

### *Function documentation:*

```
TArray* scc(TGraph graph, int start, int stop, int *sccCount)
```

This function finds the strongly connected components (SCCs) in a graph using the Tarjan's algorithm. It returns an array of nodes, where each node represents a SCC and its children represent the nodes in the SCC.

***Parameters:***

| | |
|---|---|
| graph | a graph data structure represented as an adjacency list |
| start | the index of the first vertex in the graph |
| stop | he index of the last vertex in the graph |
| sccCount | a pointer to a variable that stores the number of SCCs found |

***Returns:***

an array of nodes representing the SCCs found in the graph.

## 9.2 include/kosaraju.h File Reference

```
#include <stdbool.h>

#include "TGraph.h"
#include "TArray.h"
#include "THTAuxiliarySCC.h"
#include "Utils.h"

#define NOT_INITIALIZED -1
#define DELTA 10
```

Functions:

- `TGraph transposeGraph(TGraph*, int, int);`
- `void dfs1(int, TGraph*, THTAuxiliarySCC*, TStack*, int, int);`
- `void dfs2(int, TGraph*, THTAuxiliarySCC*, int, int, int *, TArray *);`
- `TArray* kosaraju(TGraph , int , int , int *);`

### *Function documentation:*

```
TGraph transposeGraph(TGraph* graph, int start, int stop)
```

This function finds the strongly connected components (SCCs) in a graph using the Tarjan's algorithm. It returns an array of nodes, where each node represents a SCC and its children represent the nodes in the SCC.

***Parameters:***

| | |
|---|---|
| graph | The original graph |
| start | The start index of the cut |
| stop | The stop index of the cut |
| sccCount | a pointer to a variable that stores the number of SCCs found |

***Returns:***
The transposed graph.

---

## *Function documentation:*

```
void dfs1(int node, TGraph* graph, THTAuxiliarySCC* auxiliaryHT, TStack* stack, int start, int stop)
```

This function finds the strongly connected components (SCCs) in a graph using the Tarjan's algorithm. It returns an array of nodes, where each node represents a SCC and its children represent the nodes in the SCC.
***Parameters:***

| | |
|---|---|
| node | The starting node of the search |
| graph | The graph to be searched |
| auxiliaryHT | A hash table used for auxiliary data |
| stack | A stack used for storing visited nodes |
| start | The start of the cut range |
| stop | The stop of the cut range |

---

## *Function documentation:*

```
void dfs2(int node, TGraph* transpose, THTAuxiliarySCC* auxiliaryHT, int start,
int stop, int *sccCount, TArray *sccs)
```

This function finds the strongly connected components (SCCs) in a graph using the Tarjan's algorithm. It returns an array of nodes, where each node represents a SCC and its children represent the nodes in the SCC.
***Parameters:***

| | |
|---|---|
| node | The starting node of the dfs |
| transpose | A pointer to the transposed graph on which to perform the DFS |
| auxiliaryHT | A pointer to the auxiliary hash table used to store information of the nodes |
| start | The lower bound of the range in which to search for SCCs |
| stop | The upper bound of the range in which to search for SCCs |
| sccCount | A pointer to the variable that keeps track of the number of SCCs found so far |
| sccs | A pointer to the array that stores the SCCs found |

---

## *Function documentation:*

```
TArray* kosaraju(TGraph graph, int start, int stop, int *sccCount)
```

The code is an implementation of the Kosaraju's algorithm to find the strongly connected components (SCCs) in a given graph.
***Parameters: Returns:***

| | |
|---|---|
| graph | The graph on which to perform the Kosaraju algorithm |
| start | The lower bound of the range in which to search for SCCs |
| stop | The upper bound of the range in which to search for SCCs |
| sccCount | A pointer to the variable that keeps track of the number of SCCs found so far |

An array of SCCs.

---

## 9.3 include/Utils.h File Reference

```
#include <stdio.h>

#include "TGraph.h"
#include "THTSCCs.h"
#include "THTAuxiliaryGraph.h"
```

Functions:

- `int calculateId(int, int, int, int*);`
- `bool isInCut(TGraph*, int, int, int);`
- `int min(int, int);`
- `void createNewGraph(TGraph *, TGraph*, TGraph*, TGraph*, THTSCCs*, THTAuxiliaryGraph*, int*, int, int);`
- `int* serializeSCCs(TArray*, int, int*);`
- `void deserializeSCCs(TArray*, int*, int, int*);`
- `int* serializeSCCsHT(THTSCCs*, int*);`
- `void deserializeSCCsHT(THTSCCs*, int*) ;`
- `int *serializeAuxiliaryGraphHT(THTAuxiliaryGraph*, int*);`
- `void deserializeAuxiliaryGraphHT(THTAuxiliaryGraph*, int*);`
- `int* serializeGraph(TGraph*, int*);`
- `TGraph deserializeGraph(int*, int);`
- `TArray* getAdjacencyList(int, TGraph*, TGraph*, TGraph*, int);`

---

### *Function documentation:*

```
int calculateId(int rank, int numItem, int vertex, int* offset)
```

This function calculates an id for a given vertex, rank, number of items, and offset.

### *Parameters:*

| | |
|---|---|
| rank | The rank of the item |
| numItem | The number of items |
| vertex | The vertex of the item |
| offset | Pointer to the current offset value |

### *Returns:*

The calculated ID

---

### *Function documentation:*

```
bool isInCut(TGraph* graph, int vertex, int start, int stop)
```

This function checks if a given vertex is in a cut of a given graph.

### *Parameters:*

| | |
|---|---|
| graph | Pointer to the graph data structure |
| vertex | The vertex to check for |
| start | The starting index of the cut |
| stop | The ending index of the cut |

### *Returns:*

True if the vertex is within the cut, false otherwise.

---

### *Function documentation:*

```
int minParallel(int a, int b)
```

This function is used to compare 2 int values and after return the minimum value beetwen.

### *Parameters:*

| | |
|---|---|
| a | the first int value |
| b | the second int value |

### *Returns:*

The minimum of $a$ and $b$

---

### *Function documentation:*

```
void createNewGraph(TGraph* graph, TGraph* oldGraph, TGraph* newGraph, TGraph* rvdGraph,
THTSCCs* sccsHT, THTAuxiliaryGraph* auxiliaryGraphHT, int* nodes, int sccCount, int numIteration)
```

The function uses OpenMP to parallelize the creation of the new graph.

### *Parameters:*

| | |
|---|---|
| graph | Pointer to the original graph |
| oldGraph | Pointer to the previous iteration graph |
| newGraph | Pointer to the new graph to be created |
| rvdGraph | Pointer to the received graph |
| sscsHT | Pointer to the hash table for strongly connected components |
| auxiliaryGraphHT | Pointer to the hash table for auxiliary graph |
| nodes | Array containing the ids of the new graph's vertices |
| sccCount | Number of strongly connected components |
| numIteration | The number of iteration |

### *Function documentation:*

```
int *serializeSCCs(TArray *sccs, int sccCount, int *size)
```

This function serializes an array of strongly connected components (sccs) into a one-dimensional array of integers (buf).

### *Parameters: Returns:*

| | |
|---|---|
| graph | Pointer to the graph data structure |
| vertex | The vertex to check for |
| start | The starting index of the cut |
| stop | The ending index of the cut |

True if the vertex is within the cut, false otherwise.

---

### *Function documentation:*

```
void deserializeSCCs(TArray *sccs, int *buf, int bufSize, int* sccCount)
```

This function deserializes an array of strongly connected components (SCCs) that were previously serialized.

### *Parameters:*

| | |
|---|---|
| sccs | Array of strongly connected components |
| buf | Buffer containing the serialized SCCs |
| bufSize | Size of the buffer |
| sccCount | Pointer to the number of strongly connected components |

---

### *Function documentation:*

```
int *serializeSCCsHT(THTSCCs* sccsHT, int *size)
```

This function serializes an hash table of strongly connected components (sccs) into a one-dimensional array of integers.

### *Parameters:*

| | |
|---|---|
| sccsHT | Pointer to a THTSCCs struct |
| size | Pointer to an integer which will be used to store the size of the serialized data |

### *Returns:*

Pointer to the serialized data

### *Function documentation:*

```
void deserializeSCCsHT(THTSCCs* sccsHT, int *buf)
```

Deserialization function for a hash table of strongly connected components (THTSCCs).

***Parameters:***

| | |
|---|---|
| sccsHT | Pointer to a THTSCCs struct |
| buf | Pointer to an integer array which contains the serialized data |

### *Function documentation:*

```
int *serializeAuxiliaryGraphHT(THTAuxiliaryGraph* auxGraphHT, int* size)
```

This function serializes a THTAuxiliaryGraph data structure into an array of integers.

***Parameters: Returns:***

| | |
|---|---|
| auxGraphHT | Pointer to a THTAuxiliaryGraph struct |
| size | Pointer to an integer variable that stores the size of the serialized data array |

Pointer to an integer array which contains the serialized data.

### *Function documentation:*

```
void deserializeAuxiliaryGraphHT(THTAuxiliaryGraph* auxGraphHT, int* buf)
```

This function is used to deserialize an auxiliary graph hash table from a buffer of integers.

***Parameters:***

| | |
|---|---|
| auxGraphHT | Pointer to a THTAuxiliaryGraph struct |
| buf | Pointer to an integer array which contains the serialized data |

### *Function documentation:*

```
int* serializeGraph(TGraph* graph, int* size)
```

This function serializes a graph into an array of integers.

### *Parameters:*

| | |
|---|---|
| graph | Pointer to a TGraph struct |
| size | Pointer to an integer variable which will store the size of the serialized data array |

### *Returns:*

Pointer to an integer array which contains the serialized data.

---

### *Function documentation:*

```
int* serializeGraph(TGraph* graph, int* size)
```

This function deserializes a graph from an array of integers.

### *Parameters:*

| | |
|---|---|
| buf | Pointer to an integer array which contains the serialized data |
| bufSize | Size of the serialized data array |

### *Returns:*

Pointer to a TGraph struct.

---

### *Function documentation:*

```
TArray* getAdjacencyList(int numIteration, TGraph* graph, TGraph* oldGraph, TGraph* rvdGraph, int vertex
```

This function returns the adjacency list of a given vertex in a given graph.

### *Parameters:*

| | |
|---|---|
| numIteration | The number of iteration |
| graph | Pointer to the original graph |
| oldGraph | Pointer to the previous iteration graph |
| rvdGraph | Pointer to the received graph |
| vertex | The vertex |

### *Returns:*

The adjacency list of the vertex.

---

# 10 How To Run

Starting from the main directory of our project, we can compile and build the program with few and simple terminal commands:

1. Create a build directory and launch cmake

   ```
   mkdir build
   cd build
   cmake ..
   ```

2. Generate executables with `make`
3. To generate measures (It can take a long time!) run `make generate_measures`
4. To extract mean times and speedup curves from them run `make extract_measures`

Results can be found in the *measures*, divided into *Tarjan* and *Kosaraju* directories with all the optimization version (O0, O1, O2, O3).
Finally, if it is necessary to remove all executable files from a binary program and coding directory there is a simple command: `make clean`