|  |  |
| --- | --- |
| **Comparing the performance of two algorithms for the min-cut problem for weighted graphs.** | Logo  Description automatically generated with medium confidence  **University of Padova**  **Department of Mathematics**  **“Tullio Levi-Civita”** |
| **Syed Riaz Raza[[1]](#footnote-1)  Rana Mandal[[2]](#footnote-2) Muhammad Tabish[[3]](#footnote-3)**  \* **Homework 3 Report for “**[Advanced Algorithms](https://elearning.unipd.it/math/mod/assign/view.php?id=44431)**”** |

|  |  |
| --- | --- |
| **Keywords**  Advance Algorithms,  Stoer and Wagner’s Algorithm,  Karger and stein’s randomized algorithm,  Complexity,  UNIPD,  Padova, Italy | **Abstract:**  In this report we are comparing the performance of two algorithms for the min-cut problem for weighted graphs.   1. **Stoer and Wagner's deterministic algorithm.** 2. **Karger and Stein's randomized algorithm.**   The report summarizes the result in following way:   1. Visualize the result using matplotlib one-by-one 2. Comparing the result of one algorithm with other one and vice versa 3. Conclude the result |

Table of Contents

[1. Introduction: 3](#_Toc106471767)

[A. Definition of Minimum Cut: 3](#_Toc106471768)

[a. Stoer and Wagner's deterministic algorithm: 3](#_Toc106471769)

[b. Karger and Stein's randomized algorithm: 3](#_Toc106471770)

[2. Project Structure: 3](#_Toc106471771)

[A. Implementation: 3](#_Toc106471772)

[B. Output Result: 3](#_Toc106471773)

[a. Stoer and Wagner's deterministic algorithm: 4](#_Toc106471774)

[b. Karger and Stein Algorithm: 4](#_Toc106471775)

[C. Asymptotic Notation & Visualization: 4](#_Toc106471776)

[3. Minimum Cut Solution 5](#_Toc106471777)

[A. Stoer and Wagner's deterministic algorithm: 5](#_Toc106471778)

[a. Pseudocode: 5](#_Toc106471779)

[b. Complexity: 5](#_Toc106471780)

[B. Karger and Stein's randomized algorithm: 6](#_Toc106471781)

[a. Pseudo Code: 6](#_Toc106471782)

[b. Complexity: 6](#_Toc106471783)

[4. Question 1: 7](#_Toc106471784)

[5. Question 2 9](#_Toc106471785)

[6. Question 3: 10](#_Toc106471786)

[7. Dataset Result: 12](#_Toc106471787)

[A. Stoer and Wagner's deterministic algorithm: 12](#_Toc106471788)

[B. Karger and Stein's randomized algorithm: 14](#_Toc106471789)

# Introduction:

## Definition of Minimum Cut:

In Graph Theory, a minimum cut or min-cut of a graph is a cut (a partition of the vertices of a graph into two disjoint subsets) that is minimal in some metric.

Variations of the minimum cut problem consider weighted graphs, directed graphs, terminals, and partitioning the vertices into more than two sets.

### Stoer and Wagner's deterministic algorithm:

In [graph theory](https://handwiki.org/wiki/Graph_theory), the Stoer–Wagner algorithm is a recursive algorithm to solve the [minimum cut](https://handwiki.org/wiki/Minimum_cut) problem in undirected weighted graphs with non-negative weights. It was proposed by Mechthild Stoer and Frank Wagner in 1995. The essential idea of this algorithm is to shrink the graph by merging the most intensive vertices, until the graph only contains two combined vertex sets.[[2]](https://handwiki.org/wiki/Stoer%E2%80%93Wagner_algorithm#cite_note-:0-2) At each phase, the algorithm finds the minimum s-t cut for two vertices s and t chosen at its will. Then the algorithm shrinks the edge between s and t to search for non s-t cuts. The minimum cut found in all phases will be the minimum weighted cut of the graph.

### Karger and Stein's randomized algorithm:

The Karger-Stein random contraction algorithm significantly improves the runtime of the Karger algorithm by decreasing the number of iterations required to produce a minimum cut with a high probability of correctness. The basic concept is that the probability of collapsing an incorrect edge gets higher as the number of edges decreases.

# Project Structure:

## Implementation:

We implemented most of our algorithms using minimum cut problem.

Other than the core data structures (for each algorithm), some functions are the same in all project files like:

**Core Program Units:**

* class KargerGraph (diff for each algo)
* class StoerWagner (diff for each algo)

## Output Result:

The result created as output for each algorithm implementation is in .csv format, and all the project files implementing the algorithm are using the same format to export the data.

### Stoer and Wagner's deterministic algorithm:

* dataset number
* n vertex
* n edges
* nano seconds time
* seconds time
* result
* exe times

### Karger and Stein Algorithm:

* dataset number
* n vertex
* n edges
* nano seconds time
* seconds time
* result
* discovery time
* rep times
* K
* k\_min
* is\_treshold\_activated

## Asymptotic Notation & Visualization:

To compute Error Ratio and Asymptotic complexity for each algorithm the following parameters were created for each graph in dataset (for each algorithm implementation):

**Note:** Our work here to solve an intractable problem and to compare the execution times and Computing Ration and Constant:

**Note:** n here is the number of a graph in a dataset

* ratios= Minimumcut [estimated\_time][n+1] **/** Minimumcut [estimated\_time][n]
* constant= Minimumcut [estimated\_time] [n] / Minimumcut [num\_nodes][n]
* reference = avg(constant)\* Minimumcut [n][num\_nodes])

We have created a separate file system which import the computational results of algorithm as mentioned in above paragraph and visualize the result of algorithms in following category

* Computational complexity of the algorithm
* Theoretical(C) computational complexity of the algorithm (reference variable)

# Minimum Cut Solution

## Stoer and Wagner's deterministic algorithm:

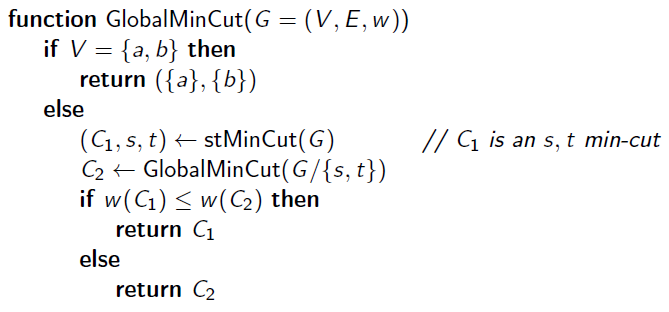
In [graph theory](https://handwiki.org/wiki/Graph_theory), the Stoer–Wagner algorithm is a recursive algorithm to solve the [minimum cut](https://handwiki.org/wiki/Minimum_cut) problem in undirected weighted graphs with non-negative weights. It was proposed by Mechthild Stoer and Frank Wagner in 1995. The essential idea of this algorithm is to shrink the graph by merging the most intensive vertices, until the graph only contains two combined vertex sets. At each phase, the algorithm finds the minimum s-t cut for two vertices s and t chosen at its will. Then the algorithm shrinks the edge between s and t to search for non s-t cuts. The minimum cut found in all phases will be the minimum weighted cut of the graph.

### Pseudocode:

A screenshot of a computer

Description automatically generated with medium confidence

**General Structure:**



### Complexity:

**Given a graph G with n vertices and m edges:**

* The execution time of stMinCut where Q is implemented with a MaxHeap:
* The execution time of GlobalMinCut implemented with a MaxHeap:

## Karger and Stein's randomized algorithm:

The Karger-Stein random contraction algorithm significantly improves the runtime of the Karger algorithm by decreasing the number of iterations required to produce a minimum cut with a high probability of correctness. The basic concept is that the probability of collapsing an incorrect edge gets higher as the number of edges decreases.

We can implement Full\_Contraction with O(n2) running time

* Pick an edge
* Contraction

### Pseudo Code:

Text

Description automatically generated Text, letter

Description automatically generated

Text, letter

Description automatically generated

### Complexity:

Recursive\_Contract has running time

Recursive\_Contract finds a minimum cut with probability **1/ log n**

by repeating Recursive\_Contracttimes, the error probability became less or equal to **1/n**

The total running time of the algorithm is

# Question 1:

Run the two algorithms you have implemented on the graphs of the dataset. For the Karger e Stein algorithm, use several repetitions that guarantees a probability to obtain a global min-cut for at least 1−1/n.

Measure the execution times of the algorithms and create a graph showing the increase of execution times as the number of vertices in the graph increases. Compare the measured times with the asymptotic complexity of the algorithms. For each problem instance, report the weight of the minimum cut obtains by your code.

**Solution:** The results of weight of the minimum cut, the data has been inserted in the table

Chart, line chart

Description automatically generated

**Figure 1.1:** Stoer & Wagner complexity with for each graph with equal number of nodes.

The graph just illustrated (fig. 1.1) shows the expected (in yellow) and effective (in blue) computational complexity for the Stoer & Wagner algorithm with more executions of the algorithm. As can be seen from the image, the effective complexity curve remains below the theoretical curve(C). With the yellow line having linear representation and blue line showing quadratic representation.

Chart

Description automatically generated

**Figure 1.2:** Karger & Stein complexity with for each graph with equal number of nodes.

The graph just illustrated (fig. 1.2) shows the expected (in yellow) and effective (in blue) computational complexity for the Karger & Stein algorithm with more executions of the algorithm. As can be seen from the image, the effective complexity curve remains slightly below the theoretical curve(C) and showing quadratic representation, and therefore we can say two complexities are almost comparable.

Text

Description automatically generated

**Figure 1.3:** Total run time of algorithms.

# Question 2

Measure the *discovery time* of the Karger and Stein algorithm. The discovery time is the instant (in seconds) when the algorithm finds the minimum cost cut. Compare the discovery time with the overall execution time for each of the graphs in the dataset.

**Figure 2.1:** Comparison of discovery time and execution time of each graph in a dataset as the number of vertices are increasing.

The illustration (fig. 2) shows the comparisons between the Discovery and Execution time of each graph in a dataset as the number of vertices are increasing while executing Karger and Stein Algorithm.

As shown, we can observe in the start, discovery time is miniature as compared to the execution time of whole graph, the difference can be seen clearly as the number of vertices increased to greater than 300 and the results of discovery time is 20%-40% of the total execution time.

**Note:** A graph with difference between (Discovery\_time – Execution\_time) was created but it didn’t illustrated the results very well, and was removed.

# Question 3:

Comment on the results you have obtained: how do the algorithms behave with respect to the various instances? There is an algorithm that is always better than the other? Which algorithm is more efficient?

A picture containing shape

Description automatically generated

**Figure 3.1:** Comparing the execution time of Karger Stein and Stoer Wagner.

The graph just illustrated (fig. 3) the comparison between the execution time of Karger Stein (in blue) and execution time of Stoer Wagner as the number of vertices are being increased.

In the case of the execution time, it can be clearly seen that Karger Stein algorithm turns out to be more efficient than Stoer & Wagner algorithm.

In case of complexity and simplicity Karger and Stein is a little complex, whereas Stoer Wagner is also referred as “Simple Min-Cut Algorithm”. For Karger Stein the room for improvement is increasing.

We compared the error of min-cut(result) by comparing the result of both algorithms, giving a proof that both algorithms are working fine, and giving the right result

While executing the Karger Stein algorithm we analyzed the result for asymptotic complexity (repetition times) where the algorithm achieved the result of being executed only 1 num\_call time when it reached the vertex# 60, as compared to the Stoer Wagner which reached the same result of being executed 1 num\_call time when the program reached the vertex# 300.

# Dataset Result:

## Stoer and Wagner's deterministic algorithm:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | n\_vertex | n\_edges | Time (ns) | Time (s) | Result | exe\_times |
| 1 | 10 | 14 | 2030945 | 0.0020309 | 3056 | 460 |
| 2 | 10 | 10 | 1188435.484 | 0.0011884 | 223 | 682 |
| 3 | 10 | 12 | 1118221.473 | 0.0011182 | 2302 | 638 |
| 4 | 10 | 11 | 1011846.728 | 0.0010118 | 4974 | 871 |
| 5 | 20 | 24 | 2889937.017 | 0.0028899 | 1526 | 181 |
| 6 | 20 | 24 | 4522151.412 | 0.0045222 | 1684 | 354 |
| 7 | 20 | 27 | 9477300.658 | 0.0094773 | 522 | 152 |
| 8 | 20 | 25 | 10172838.65 | 0.0101728 | 2866 | 207 |
| 9 | 40 | 52 | 20713347.73 | 0.0207133 | 2137 | 44 |
| 10 | 40 | 54 | 14491207.95 | 0.0144912 | 1446 | 88 |
| 11 | 40 | 51 | 14193219.05 | 0.0141932 | 648 | 63 |
| 12 | 40 | 50 | 14145845.35 | 0.0141458 | 2486 | 86 |
| 13 | 60 | 82 | 29707856.25 | 0.0297079 | 1282 | 32 |
| 14 | 60 | 72 | 29829734.38 | 0.0298297 | 299 | 32 |
| 15 | 60 | 83 | 32099238.46 | 0.0320992 | 2113 | 26 |
| 16 | 60 | 79 | 31184308 | 0.0311843 | 159 | 25 |
| 17 | 80 | 101 | 50028043.75 | 0.050028 | 969 | 16 |
| 18 | 80 | 105 | 43484717.39 | 0.0434847 | 1756 | 23 |
| 19 | 80 | 108 | 38275737.04 | 0.0382757 | 714 | 27 |
| 20 | 80 | 108 | 41880884 | 0.0418809 | 2610 | 25 |
| 21 | 100 | 128 | 63953193.75 | 0.0639532 | 341 | 16 |
| 22 | 100 | 120 | 56473123.53 | 0.0564731 | 890 | 17 |
| 23 | 100 | 125 | 57774294.12 | 0.0577743 | 772 | 17 |
| 24 | 100 | 133 | 64013520 | 0.0640135 | 1561 | 15 |
| 25 | 150 | 197 | 140655271.4 | 0.1406553 | 951 | 7 |
| 26 | 150 | 206 | 154352450 | 0.1543525 | 424 | 6 |
| 27 | 150 | 195 | 145157100 | 0.1451571 | 1153 | 6 |
| 28 | 150 | 198 | 168361450 | 0.1683614 | 707 | 6 |
| 29 | 200 | 276 | 308606000 | 0.308606 | 484 | 3 |
| 30 | 200 | 260 | 279913000 | 0.279913 | 850 | 3 |
| 31 | 200 | 269 | 279464500 | 0.2794645 | 1382 | 3 |
| 32 | 200 | 274 | 287958100 | 0.2879581 | 1102 | 3 |
| 33 | 250 | 317 | 444742300 | 0.4447423 | 346 | 2 |
| 34 | 250 | 322 | 442721400 | 0.4427214 | 381 | 2 |
| 35 | 250 | 338 | 456653500 | 0.4566535 | 129 | 2 |
| 36 | 250 | 326 | 449405900 | 0.4494059 | 670 | 2 |
| 37 | 300 | 403 | 671063800 | 0.6710638 | 1137 | 1 |
| 38 | 300 | 393 | 677553200 | 0.6775532 | 869 | 1 |
| 39 | 300 | 408 | 700373000 | 0.700373 | 868 | 1 |
| 40 | 300 | 411 | 690243800 | 0.6902438 | 1148 | 1 |
| 41 | 350 | 468 | 955078000 | 0.955078 | 676 | 1 |
| 42 | 350 | 475 | 965932000 | 0.965932 | 290 | 1 |
| 43 | 350 | 462 | 1053498600 | 1.0534986 | 818 | 1 |
| 44 | 350 | 474 | 961604900 | 0.9616049 | 175 | 1 |
| 45 | 400 | 543 | 1275305800 | 1.2753058 | 508 | 1 |
| 46 | 400 | 527 | 1271454000 | 1.271454 | 904 | 1 |
| 47 | 400 | 526 | 1251964800 | 1.2519648 | 362 | 1 |
| 48 | 400 | 525 | 1270084300 | 1.2700843 | 509 | 1 |
| 49 | 450 | 595 | 1777374200 | 1.7773742 | 400 | 1 |
| 50 | 450 | 602 | 1721216200 | 1.7212162 | 364 | 1 |
| 51 | 450 | 593 | 1741665700 | 1.7416657 | 336 | 1 |
| 52 | 450 | 594 | 1629809100 | 1.6298091 | 639 | 1 |
| 53 | 500 | 670 | 2150573900 | 2.1505739 | 43 | 1 |
| 54 | 500 | 671 | 2140404900 | 2.1404049 | 805 | 1 |
| 55 | 500 | 670 | 2957780700 | 2.9577807 | 363 | 1 |
| 56 | 500 | 666 | 2106707100 | 2.1067071 | 584 | 1 |

## Karger and Stein's randomized algorithm:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **n\_vertex** | **n\_edges** | **Time\_ns** | **Time\_s** | **Result** | **discover(ns)** | **Repetition** | **k** | **K min** |
| 1 | 10 | 14 | 4971423 | 0.004971 | 3056 | 1301693.878 | 98 | 11 | 2 |
| 2 | 10 | 10 | 4863303 | 0.004863 | 223 | 502751 | 100 | 11 | 1 |
| 3 | 10 | 12 | 4616924 | 0.004617 | 2302 | 1185666 | 225 | 11 | 2 |
| 4 | 10 | 11 | 4581093 | 0.004581 | 4974 | 1802942 | 231 | 11 | 2 |
| 5 | 20 | 24 | 30318700 | 0.030319 | 1526 | 7192174 | 31 | 19 | 10 |
| 6 | 20 | 24 | 29636026 | 0.029636 | 1684 | 4711981 | 27 | 19 | 2 |
| 7 | 20 | 27 | 31384614 | 0.031385 | 522 | 3167014 | 28 | 19 | 1 |
| 8 | 20 | 25 | 29409726 | 0.02941 | 2866 | 7910079 | 34 | 19 | 1 |
| 9 | 40 | 52 | 2.19E+08 | 0.219051 | 2137 | 59035825 | 4 | 28 | 6 |
| 10 | 40 | 54 | 2.32E+08 | 0.232152 | 1446 | 22588275 | 4 | 28 | 2 |
| 11 | 40 | 51 | 2.14E+08 | 0.214365 | 648 | 22948025 | 4 | 28 | 7 |
| 12 | 40 | 50 | 2.21E+08 | 0.221243 | 2486 | 50220300 | 4 | 28 | 3 |
| 13 | 60 | 82 | 7.92E+08 | 0.792061 | 1282 | 4.09E+08 | 1 | 35 | 18 |
| 14 | 60 | 72 | 7.96E+08 | 0.795775 | 299 | 1.18E+08 | 1 | 35 | 5 |
| 15 | 60 | 83 | 7.46E+08 | 0.745547 | 2113 | 42686000 | 1 | 35 | 2 |
| 16 | 60 | 79 | 7.57E+08 | 0.756588 | 159 | 28053800 | 1 | 35 | 1 |
| 17 | 80 | 101 | 1.85E+09 | 1.854154 | 969 | 54226300 | 1 | 40 | 1 |
| 18 | 80 | 105 | 1.85E+09 | 1.85311 | 1756 | 1.44E+08 | 1 | 40 | 3 |
| 19 | 80 | 108 | 1.88E+09 | 1.877795 | 714 | 43785800 | 1 | 40 | 1 |
| 20 | 80 | 108 | 1.89E+09 | 1.890357 | 0 | 6.57E+08 | 1 | 40 | 14 |
| 21 | 100 | 128 | 3.82E+09 | 3.82248 | 341 | 98751800 | 1 | 44 | 1 |
| 22 | 100 | 120 | 3.9E+09 | 3.904697 | 890 | 1.7E+08 | 1 | 44 | 2 |
| 23 | 100 | 125 | 3.8E+09 | 3.799816 | 772 | 79470200 | 1 | 44 | 1 |
| 24 | 100 | 133 | 3.69E+09 | 3.690227 | 1561 | 1.74E+08 | 1 | 44 | 2 |
| 25 | 150 | 197 | 1.47E+10 | 14.68531 | 951 | 2.68E+08 | 1 | 52 | 1 |
| 26 | 150 | 206 | 1.44E+10 | 14.35232 | 424 | 2.71E+08 | 1 | 52 | 1 |
| 27 | 150 | 195 | 1.41E+10 | 14.12117 | 1153 | 1.62E+09 | 1 | 52 | 6 |
| 28 | 150 | 198 | 1.41E+10 | 14.14129 | 707 | 8.26E+08 | 1 | 52 | 3 |
| 29 | 200 | 276 | 3.63E+10 | 36.26252 | 484 | 4.38E+09 | 1 | 58 | 7 |
| 30 | 200 | 260 | 3.55E+10 | 35.54782 | 850 | 7.96E+09 | 1 | 58 | 13 |
| 31 | 200 | 269 | 3.7E+10 | 37.00348 | 1382 | 2.52E+09 | 1 | 58 | 4 |
| 32 | 200 | 274 | 3.58E+10 | 35.76403 | 1102 | 2.44E+09 | 1 | 58 | 4 |
| 33 | 250 | 317 | 7.4E+10 | 73.97771 | 346 | 1.21E+09 | 1 | 63 | 1 |
| 34 | 250 | 322 | 7.49E+10 | 74.94908 | 0 | 9.39E+09 | 1 | 63 | 8 |
| 35 | 250 | 338 | 7.47E+10 | 74.67894 | 129 | 1.23E+09 | 1 | 63 | 1 |
| 36 | 250 | 326 | 7.53E+10 | 75.2985 | 670 | 1.16E+09 | 1 | 63 | 1 |
| 37 | 300 | 403 | 1.40E+11 | 139.5622 | 1137 | 2.04E+09 | 1 | 68 | 1 |
| 38 | 300 | 393 | 1.55E+11 | 154.5544 | 869 | 2.07E+10 | 1 | 68 | 10 |
| 39 | 300 | 408 | 1.44E+11 | 144.3123 | 868 | 1.7E+10 | 1 | 68 | 8 |
| 40 | 300 | 411 | 1.37E+11 | 137.4575 | 1148 | 2.03E+10 | 1 | 68 | 10 |
| 41 | 350 | 468 | 2.30E+11 | 229.5759 | 676 | 6.44E+09 | 1 | 71 | 2 |
| 42 | 350 | 475 | 2.32E+11 | 232.0913 | 0 | 6.44E+10 | 1 | 71 | 20 |
| 43 | 350 | 462 | 2.28E+11 | 227.6666 | 818 | 1.63E+10 | 1 | 71 | 5 |
| 44 | 350 | 474 | 2.28E+11 | 228.06 | 175 | 6.55E+09 | 1 | 71 | 2 |
| 45 | 400 | 543 | 3.60E+11 | 360.4575 | 508 | 9.51E+09 | 1 | 75 | 2 |
| 46 | 400 | 527 | 3.58E+11 | 357.6185 | 904 | 9.56E+09 | 1 | 75 | 2 |
| 47 | 400 | 526 | 3.60E+11 | 360.2154 | 362 | 4.98E+09 | 1 | 75 | 1 |
| 48 | 400 | 525 | 3.68E+11 | 367.5047 | 509 | 3.9E+10 | 1 | 75 | 8 |
| 49 | 450 | 595 | 5.47E+11 | 546.7849 | 400 | 6.97E+09 | 1 | 78 | 1 |
| 50 | 450 | 602 | 5.47E+11 | 546.8291 | 364 | 6.97E+09 | 1 | 78 | 1 |
| 51 | 450 | 593 | 5.18E+11 | 517.8779 | 336 | 2.65E+10 | 1 | 78 | 4 |
| 52 | 450 | 594 | 6.15E+11 | 614.9765 | 639 | 1.35E+10 | 1 | 78 | 2 |
| 53 | 500 | 670 | 9.67E+11 | 966.7981 | 43 | 2.33E+10 | 1 | 80 | 2 |
| 54 | 500 | 671 | 1.01E+12 | 1005.748 | 805 | 2.46E+11 | 1 | 80 | 21 |
| 55 | 500 | 670 | 8.66E+11 | 866.3529 | 363 | 1.14E+10 | 1 | 80 | 1 |
| 56 | 500 | 666 | 8.48E+11 | 847.6328 | 584 | 4.54E+10 | 1 | 80 | 4 |

**EOF**

1. **Syed Riaz Raza; E-mail:** [syedriaz.raza@studenti.unipd.it](mailto:syedriaz.raza@studenti.unipd.it); **Portfolio**: [riazraza.me](https://riazraza.me/) [↑](#footnote-ref-1)
2. **Rana Mandal; E-mail:** [rana.mandal@studenti.unipd.it](mailto:rana.mandal@studenti.unipd.it); [↑](#footnote-ref-2)
3. **Muhammad Tabish; E-mail:** [muhammad.tabish@studenti.unipd.it](mailto:muhammad.tabish@studenti.unipd.it); [↑](#footnote-ref-3)