

**Capstone Project Phase 2**

Project Name:

**Online algorithms in optical networks –   
design, analysis and simulation.**

Project Number:   
  
**21-1-R-21**

Name: **Amir Amara** E-Mail: **amiramara11@gmail.com**

Name: **Fida Khoury** E-Mail: **fidakory96@gmail.com**

Supervisor(s):

**Professor Shmuel Zaks**

**Table of content**

1 Introduction

2 Project Review and Process Description

2.1 Background

2.1.1 Online Algorithm

2.1.2 Competitive Analysis

2.1.3 Competitive Ratio

2.1.4 Optical Networks

2.1.4.1 Why do we need optical fiber networks

2.1.4.2 Elements involved in optical networks

2.1.5 Add/Drop Multiplexers

2.2 MinADM Algorithm

2.2.1 Preliminaries

2.2.2 How the algorithm works?

2.2.3 Expected results in line topology

2.2.4 Expected results in ring topology

2.3 Process description

2.4 Testing Process Description

2.4.1 Unit Testing

2.4.2 Functionality Testing

2.5 Result and conclusion

3 User documentation

3.1 User’s Guide

3.2 Maintenance Guide

4 UML Diagrams

4.1 Use-Case Diagram

5 References

1. **Introduction**

Our project conducts research on On-Line Algorithms which have been an active area of research within theoretical computer science. The theoretical study of on-line algorithms and the growing interest in it is due to several factors, including new applications, online model extensions, new performance measures and constraints. Our research is in the area of on-line algorithm in optical network its aim is to reduce resources in optical fiber networks, ADMs specifically. Our research handles the design, analysis and simulation of on-line algorithms that help reduce these resources (ADMs).

The algorithm is minADM which helps optimizing the total number of ADMs by reducing the assigned ADMs to lightpaths with the same color at the conjunction point between them.

1. **Project Review and process description** 
   1. **Background** 
      1. **Online Algorithms**

Online algorithm is the process which at each point in time decides what to do based only on past information and with no (or inexact) knowledge about the future. An online algorithm is one that can process its input piece-by-piece in a serial fashion, i.e., in the order that the input is fed to the algorithm, without having the entire input available from the start.

This algorithm needs to make decisions without full knowledge of the input. It has full knowledge of the past but no (or the partial) knowledge of the future. Online Algorithm needs to respond to the request sequence that has been shown, not knowing next items in the sequence. For any input, the cost of our online algorithm is never worse than c times the cost of the optimal offline algorithm. (**See [3]).** For this type of problem, we will attempt to design algorithms that are competitive with the optimum offline algorithm, the algorithm that has perfect knowledge of the future.

* + 1. **Competitive Analysis**

Competitive analysis is a method of analyzing online algorithms, in which performance of an algorithm is compared performance of offline algorithm, considering all possible request sequences. (**See [4])**

The competitive ratio of an algorithm, is defined as the worst-case ratio of its cost divided by the optimal cost, overall possible inputs. (**See [4]).**

**Defined as:**

Figure 2: Competitive Ratio

*Where I stand for the iteration of the domain of the lightpaths the algorithm is calculating.*

* + 1. **Optical Networks**

An optical network is a communication network used for sharing information at high speed and long distance between two ends through an optical fiber cable. It is the latest and most advance form of data communication.

* + - 1. **Why do we need optical fiber cable?**

The major reason for using these optical fiber cables is that signals can be transferred in the form of light pulses. In modern era fast internet is a basic need and fiber cables have the capability to transfer large amount of data chunks at high speed and cover long distances. Thus, these are the reasons why fiber cables are critical aspect of modern data communication.

* + - 1. **Elements involved in optical network.**

An optical network consists of the following elements:

* Trunk
* Nodes
* Topology
* Routers
* ADM
* Regenerator
* Lightpath

**Stations:** Stations are the points of the information transmitted from or received to. These are basically the communication devices used by an average user (Laptops, Mobiles etc.).

**Trunks:** Trunks are transmission lines used for transferring and receiving data on both ends. An optical network contains multiple trunks to transmit data over large areas.

**Nodes:** Nodes are just hubs in multi transmission lines in a network. In single transmission hubs are not required but in multiple transmission hub play essential role in connecting stations at both transmitting and receiving end.

**Topology:** Multiple optical fiber cables are connected in network through the help of nodes. The way in which in these optical fiber cables are connected is known as topology of a network.  
**Router:** A router is a device that is placed with in a network to provide suitable path for signal transmission.

**ADM:** An add-drop multiplexer (ADM) is an important element of an optical fiber network. A multiplexer combines, or multiplexes, several lower-bandwidth streams of data into a single beam of light. An add-drop multiplexer also has the capability to add one or more lower-bandwidth signals to an existing high-bandwidth data stream, and at the same time can extract or drop other low-bandwidth signals, removing them from the stream and redirecting them to some other network path.

**Regenerator:** also known as an optical regenerator that helps our signals inside optical fiber networks to travel long distances without any distortions that can replace the optical amplifiers that regenerate signal amplitudes over long distances.

**Lightpath:** a lightpath in optical networks is a path that’s established between two node endpoints that are called source and destination which the light passes through in an unmodified way and is only optical, and each lightpath is assigned with a wavelength of its own.

Although an optical network is constructed using various elements, our research conducts optical networks work related specifically to ADMs in depth.

* + 1. **Add/Drop Multiplexers**

An add/drop multiplexer (ADM) is a critical element of an optical fiber network. It can combine (i.e., multiplex) several low-bandwidth streams of data into a single light beam; and simultaneously, it can drop or remove other low-bandwidth signals from the stream of data and direct them to other network routes. **[2]**

A single fiber-optic cable offers a bandwidth that can potentially carry information at the rate of several terabits per second, much faster than any electronic device can handle. In order to utilize the potential of optical fiber, wavelength-division multiplexing (WDM) is used. The bandwidth is partitioned into a number of channels at different wavelengths. Several signals can be transmitted through a fiber link simultaneously on different channels. The number of channels (wavelengths) available in WDM systems is limited by the chosen technology. One of the important parameters affected by the technology is the network cost. Add/drop multiplexers (ADMs) are employed at the network nodes to insert light waves into the fiber and extract them. WDM ring networks are deployed by a growing number of telecom carriers. The problem of minimizing the number of wavelengths has been extensively studied. Variants of this problem such as to maximize the number of light paths given a limited number of wavelengths or to minimize the blocking probability of a light path were also studied. **[2]**

* 1. **Min ADM Algorithm** 
     1. **Preliminaries**

An instance of the problem is a pair where is an undirected graph and is a set of simple paths in . In an on-line instance, the graph is known in advance and the set of paths is given on-line. In this case we denote where is the -th path of the input and consists of the first paths of the input.

As for example we take a pair where ,set of vertices {v, u, x, w} and the set of paths P= {q, s, r} (as shown in fig 2).

r

s

q

Fig.2 A sample input .

Given such an instance we define the following:

**Definition 2.2.1.1:**The paths are conflicting or overlapping if they have an edge in common. This is indicated as **p ≍ p′**. The graph of the relation is called the conflict graph of .

*As we see in the example inf fig.2 we have q and s are overlapping or conflicting, s ≍ q.*

**Definition 2.2.1.2:**A proper coloring or wavelength assignment of **P** is a function , such that whenever .

*So, our input should look like Fig.3.*

Fig.3 Proper coloring.

r

s

q

**Definition 2.2.1.3:**A valid chain (cycle) of problem is a path (cycle) formed by the sequence of distinct paths that do not go over the same edge twice (not conflicted).

*In the sample input the valid chain is .*

**Definition 2.2.1.4:**A solution of an instance is a set of valid chains and valid cycles of such that each appears in exactly one of these sets.

*Our solution .*

**Definition 2.2.1.5:**The shareability graph of an instance , is the edge-labelled multi-graph such that there is an edge labelled in if and only if and are not overlapping , and is a common endpoint of and in .Note we understand from the definitions that the edges from the conflicted graph are in . Also, that for any node of , the edges next to its count is at most two.

u

w

Fig .4 Shareability graph

**Definition 2.2.1.6**:

A valid chain (cycle) of is a simple path of , such that any two consecutive edges in the path (cycle) have distinct labels and its node set is properly colorable with one color (in ), or in other words constitutes an independent set of the conflict graph.

Fig.5 Conflict graph

**Definition 2.2.1.7**:  
The sharing graph of a solution of an instance , is the following subgraph of . Two lightpaths are connected with an edge labelled in if and only if they are consecutive in a chain or cycle in the solution , and their common endpoint is . We will usually exclude the index and simply write . We indicate by the degree of node in .

Now we want to define the .

So, first for any solution , we partition the set of lightpaths into disjoint subsets depending on the degree of the corresponding node in .

We define:

And

Note that

*Let's calculate the degree in our example (as Fig.5):*

*So and it is true we have 3 lightpaths.*

Because the edge with label is a sequence of two paths with the same color at their common endpoint, these two endpoints can share an ADM at their node , thus saving one ADM. When no ADMs are shared, each path need two ADMs, a total of ADMs. Therefore, the cost of a solution is .

*In our example:.*

*Reminder our solution is S= {(q, r), (s)} => .*

The motive is to find a solution such that is minimum, in other words is maximum.

Given a solution , for every node . Therefore, the connected components of are either paths or cycles. Note that an isolated node is a special case of a path. Let be the set of the connected components of that are paths. Clearly,

Therefore, **.**

*In our example:.*

Let be a solution with minimum cost. For any solution we define

**In our example:** .

**Lemma 2.2.1.1**:  
For any solution

.

**Proof.**Clearly . On the other hand, is the sum of the degrees of the nodes in ,

We conclude:

*In our example:*

* + 1. **How the algorithm works**

As we explained before each lightpath has two end point and for each end point there is an ADM, lets denote the end points with u and v.

End point v

End point v

lightpath

ADM

ADM

The ligthpaths are given on-line, so we want to adopt a simple coloring procedure.

A free ADM in is an ADM serving one lightpath ending in , but not sharing an edge with the lightpath to be colored.

If we have two lightpaths assigned to same color, then if it is possible the first lightpath tries to make a cycle with the existing lightspaths else it makes a path.

If there are free ADMs at one endpoint (or at both endpoint but of a different colors), then it tries to connect to any of the lightpaths.

Otherwise, there is no free ADMs, the lightpath will be assigned to a new color

So, we want a color to be feasible for lightpath , and it shouldn't be no other lightpath with the same color overlapping with ,the we can assign and will be a proper coloring for .

**So, the coloring procedure goes like this:**

When a lightpath with endpoints and arrives:  
If there exists a chain of lightpaths of the same color whose endpoints are , and is feasible for then, assign .

Otherwise, if there exists a chain of lightpaths of the same color having one endpoint from and is feasible for then, assign .Otherwise , assign , where is an unused color.\

We can see easily that the Algorithm Online-MinADM is valid: *w* is a proper coloring for , because if is colored by one of the first two cases, then it is checked by algorithm for feasibility, otherwise *w ()* is assigned an unused color, therefore no other path, in particular no path conflicting with may have *w (.*

**Diagram

Description automatically generated**

The late studies used to deal with the problem of assigning colors to lightpaths, as for now large portion of the studies focus on the total hardware cost.

in our case the total cost considered to be the total number of Add-drop Multiplexers (ADMs), why?...

Each light paths uses two ADMs one at each end point, and if a two adjacent lightpaths share a common endpoint (with the same wavelength) that mean they share an ADM.

Note: because ADMs are designed to be used mainly in ring and path networks in which the degree of a node is at most two, an ADM may be shared by at most two lightpaths.

lightpath

Sharing

ADM

lightpath

lightpath

lightpath

Fig. 1 Two lightpaths with the same wavelength (WL)

Lightpaths sharing ADMs in a common endpoint can be thought as concatenated, so that they form longer paths or cycles. These paths/cycles do not use any edge *e* ∈ *E* twice, for otherwise they cannot use the same wavelength which is a necessary condition to share ADMs. The motivation for the on-line problem stems from the need to utilize the cost of use of the optical network. We assume that the switching equipment is installed in the network. Once a light path arrives, we need to assign it two ADMs, and our target is to determine which wavelength to assign to it so that we minimize the cost, measured by the total number of ADMs used. **[1]**

* + 1. **Results in line topology**

Online-MinADM is competitive in path topology.

Let’s prove this proposition:

Let be the nodes of path from left to right, denote as the set of paths having vi as their right endpoint and denote as the set of paths having vi as their left endpoint. we know that the number of ADMs used by an optimal solution is . In an optimal solution, at each node , exactly pairs of paths are assigned one color per pair; therefore, these pairs constitute a maximum matching of the complete bipartite graph .

The solution saves ADMs at node , in other words . such that the matching in each node is obtained by augmenting the matching done by to a maximum matching, which is .We will now define a function.

An edge of Es\* - Es

An edge of Es

pi

pj

pi'

pi\*

e

e'

e"

vk

vk'

vk"

Let . Let for some node .   
Assume without loss of generality that , which is that path appears before in the input.  
As , none of are paired with any path at node . Therefore when appears in the input is feasible for , if it is not assigned a color , this can be only because it is assigned color , for some . Let the common node of and be .   
Then We define . Note that is defined uniquely because there cannot be a third path except and getting the same color and ending at node . Necessarily  
, because we know that is not paired at node .  
We claim that is one-to-one. Assume, by contradiction that there is some , such that  
. Then , therefore for some node .  
By the construction of , is the other endpoint of . Let . By the discussion in the previous paragraph, symmetrically it follows that ,a contradiction.  
Therefore is one-to-one, which is that , thus .

We conclude as follows.  **,** therefore:

.

**Lower bound in path topology**

For any ϵ>0, there is no ( - ϵ)-competitive deterministic algorithm for path (line) topology.

Let’s prove this proposition:

We want to build the adversary as the following, the it will work in two phases.

Let's take a graph G as a path with 2k nodes and denote them as , the value of k will be determined later on. Denote ALG as deterministic algorithm.

In phase one the input (lighpaths) of the algorithm ALG will be where the endpoints of the lightpaths is ( , as seen in the following figure.

In the second phase the input (lightpaths) depends on the algorithm ALG in the first phase:

as we see in the following example . Therefore, the input lighpaths is , and .

Let's denote x as the number of times the is satisfied be , therefore the number of is not satisfied will be times.

Now we want to calculate the competitive ratio of the algorithm ALG.

First possible solution:

In the first phase the algorithm uses 2k ADMs (one ADM for each node). As in the second phase for the lightpaths and , we take a color λ where λ=. Algorithm ALG can't assign color λ for nor because they are conflicted, therefore for each will assign different colors and the cost is 4 ADMs, so the total is 4x ADMs.

As for the lightpath (also in the second phase) let's assing the following coloring λ= and λ'=(≠λ) .So if it assign the lightpath to one of this colors the cost will be one ADM otherwise, the algorithm ALG will uses 2 ADMs. Therefore, the algorithm uses at least k-1-x in total.

After summing everything up we get that ALG uses at least ADMs .

As for the second possible solution which is the optimal:

For any consecutive lightpaths the algorithm will assign colors as the following the total ADMs of this solution is 2k+2x ADMs, one ADM at and one on that 2k, also x additional ADMs at and at that’s 2x.

Therefore the competitive ratio is at least .

For any ϵ>0 we can choose , so that the competitive ratio of ALG is bigger than .

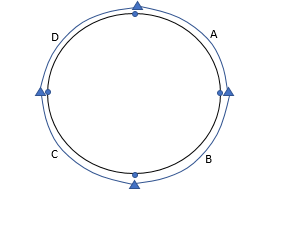
* + 1. **Results in ring topology**

**Lemma** **2.2.4.1**: no deterministic online algorithm has a competitive ratio better than 7/4, even for the ring topology.

Overview:

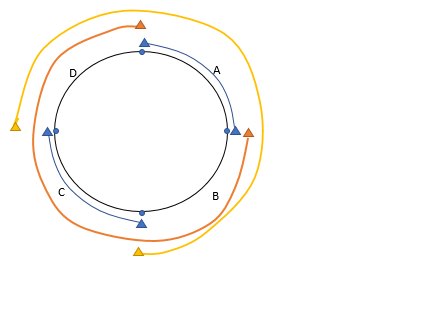
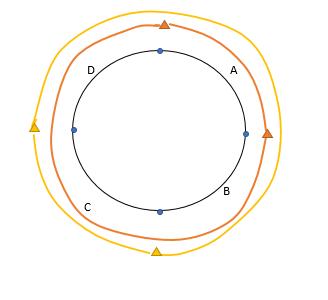
First of all, let’s suppose we want to divide the ring into 4 segments A, B, C, D.

Now first request is lightpaths A and B.



1. We want to check if the online algorithm assigns the same color to A and C, which mean the lightpaths in segments A and B are not conflicted as we explained earlier.

C

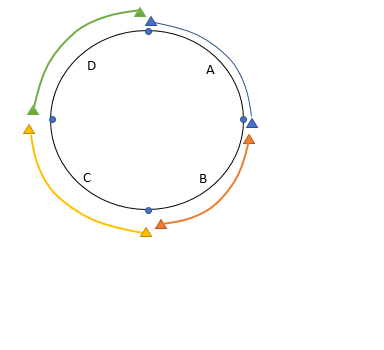
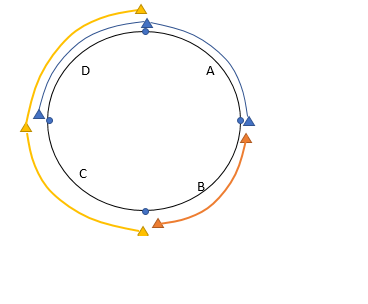


Then we request two lightpaths (B, C, D) and (D, A, B).

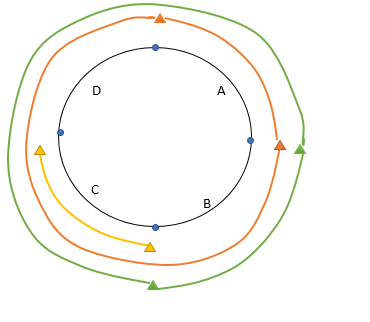
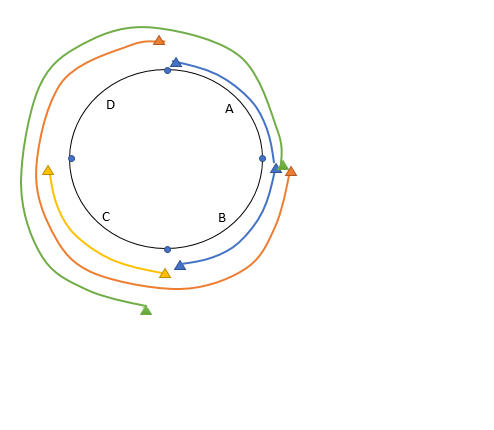
Obviously, the online algorithm has to use two new colors because the lightpaths are conflicted.

Therefore, we used 8 ADMs while the offline algorithm used 4 ADMs.

1. the second case is lightpaths of and are conflicted. Therefore, the online algorithm assigns different colors. We then request :
2. If the online algorithm assigned third color to B, then D might be assigned either to same color as A or same color as C using 7 ADMs. Or the algorithm might assign a new color to D then we will have 8 ADMs. Therefore, the minimum is 7 ADMs, but the optimal solution in the offline algorithm is 4 ADMs only.



1. If the online algorithm assigns it to be one of the colors A or C, let's assume it choose the color of A, then we have the lightpaths (B, C, D) and (C, D, A). Obviously, they are conflicted, then they can’t care ADMs with the already existing paths. So, we have 3 ADMs for A and B, 2 ADMs for (B, C, D), 2 ADMs for (C, D, A) and 2 ADMs for C. that’s total of 7 ADMs plus 2 ADMs for C, which means 9 ADMs.



we see in this problematic stage is in 2(b).

In this case we might repeat the input in stages, the first stage taking A, B, (B, C, D), (C, D, A), for the second stage we take c and proceed in second stage by requesting A and repeating this process k times (k is a large number).

Therefore, in the online algorithm we use at least 7k+2 ADMs, as for the offline algorithm we use almost 4k+2 ADMs.

For very large, we can say that the competitive ratio at least 7/4 – ( ).

* 1. **Process description**

In this section we are going to explain our program and the decision we took throughout the making.

The website simulates the minADM algorithm, which can help us find the competitive ratio on different inputs easily.

It has been proven in the research that in general topology and ring topology the upper competitive ratio is 7/4, and for the line topology is 3/2. Therefor we must calculate the competitive ratio each time we run the algorithm, that is our main goal to prove that for general or ring topology the upper competitive ratio is 7/4, and for the line topology is 3/2.

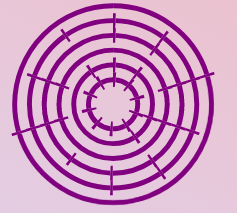
Since the general and ring topology has the same upper ratio, we can simulate one of them, we choose the ring topology because it’s easier to understand and more fun two watch.

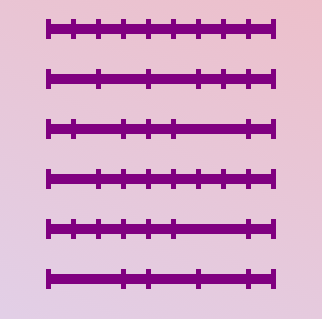
For simulating the minADM algorithm, we had to create a network with a specific number of nodes (the nodes are an input form the user). In the ring network number of edges equal to number of nodes, while in the line network it is number of nodes minus 1. For each edge we save the passing through lightpaths in array, so it will be easier later to know which lightpaths are conflicted.

For each online solution we have the offline solution which is the optimal solution when the lightpaths already known and calculating the offline solution is very important since we use it to calculate the competitive ratio.

**Offline solution:**

We created random number of random lightpaths, even thou we succeeded programing it focusing on the conflicting edges but running it on large input (number of nodes) will take an exponential time. That was going to restrict us with number of lightpaths and/or number of nodes. Therefore, we created full chains that contains random number of random lightpaths (smaller than number of nodes). Calculating the ADMS in the optimal solution now is much easier because we know each chain assigned its own color(wavelength), since we already have the lightpaths ordered in full chains, we don’t need to make an algorithm for assigning the color to the lightpaths like we do in the minADM algorithm. After all our goal is to calculate the competitive ratio by counting the ADMs.





**Online solution:**

Since we have the number of ADMs for the offline now, we need to count the ADMs in the online solution after running on it the minADM algorithm, for calculating the competitive ratio. The input for the minADM algorithm is the lightpaths we created earlier. In the process of the minADM algorithm we must now if lightpaths are conflicting and we can get this information from the passing-lightpaths-array in each edge of the current lightpath.

Because the lightpaths will come into the algorithm in random order the number of ADMs will be bigger, unless if coincidentally was the same order as the offline solution, then it will have same number of ADMs.

**Improvements:**

During the development, we faced some obstacles that will be mentioned in this section.

* Drawing a circle is never easy as human but for the computer with the right coordination it can happen easily with any size, we want but it wasn't easy to know that coordination, same goes to drawing a line. Since we had some trouble with drawing the circle and line, they would fit correctly in the screen, our thought was to change the simulation to be on the terminal. But to see only numbers is a very boring process and cannot fully deliver the main goal of the research. Therefore, we went back to drawing the circle and line by choosing how many optimal chains we would like to represent this way we have more control over how many circles we need to draw.
* The pervious obstacles also made us think to change from programing in HTML and JavaScript to python but later when we found the solution went back to programing in HTMAL and JavaScript
* Since the network is undirected graph, we had to make sure there is a direction (clockwise) for all the lightpaths, because lightpath P1 (V1, V2) differs from P2 (V2, V1).
* We had to consider that in line network for each lightpath P (V1, V2) endpoint V1 can’t be bigger than V2, and in the ring network it can so we had to check always if we are in the right range. Otherwise, all our calculation will be wrong, these little details we don’t think about because our human brain understand it but for a computer it’s not and we must constantly check if the random input is correct.
  1. **Testing Process Description**

For the minADM algorithm we already did all the tests we needed otherwise we cannot create and go thru the whole simulation, as we mention earlier in section 2.5, we also tested if it is below the upper competitive ratio.

To evaluate the system, we performed two kinds of evaluation tests. The first one is unit testing and the second is functional testing.

* + 1. **Unit testing**

To evaluate the system testing we will run the program on some significant input:

|  |  |  |
| --- | --- | --- |
| **Test No** | **Test subject** | **Testing plan** |
|  | Create a network with the given input. | Testing if network is created by the simulation itself, the drawing of the creating depends on that. So, if we can see on the screen the ring/line that means the input has been drawn correctly. |
|  | Creating a random lightpath | First of all, we printed it in the terminal, after that for bigger input we just test it by drawing it on screen (simulating the lightpath ). |
|  | Creating a lightpath | Our lightpaths is random so only for testing we gave it a specific endpoint and made it draw it on screen |
|  | Calculating the c-ratio | For testing the calculation, we gave the function a specific input then compared if it returns the c-ratio that should be for those specific inputs. |
|  | Counting ADMs | Checking if the test result for a specific input is equal to the actual result. |

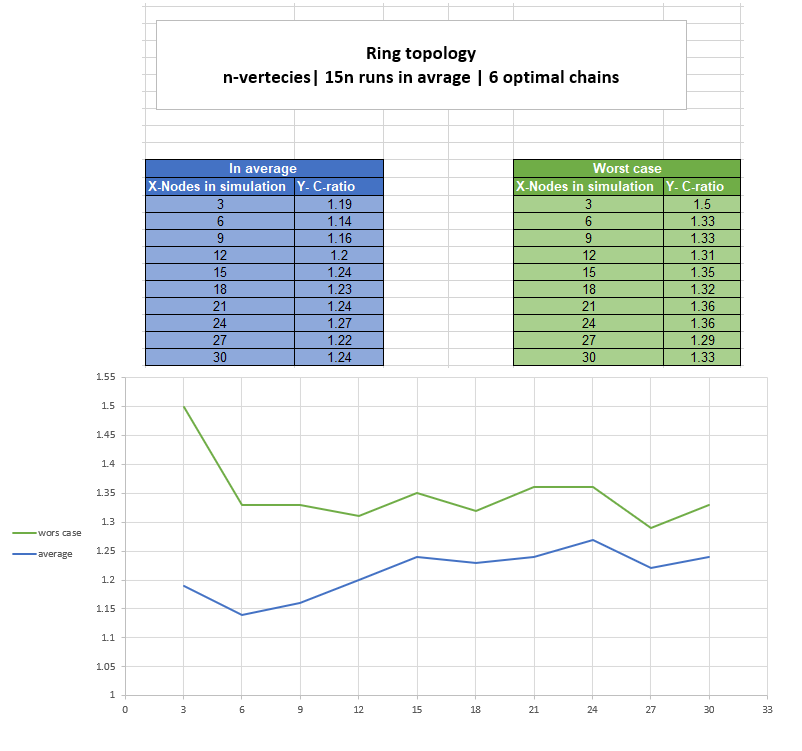
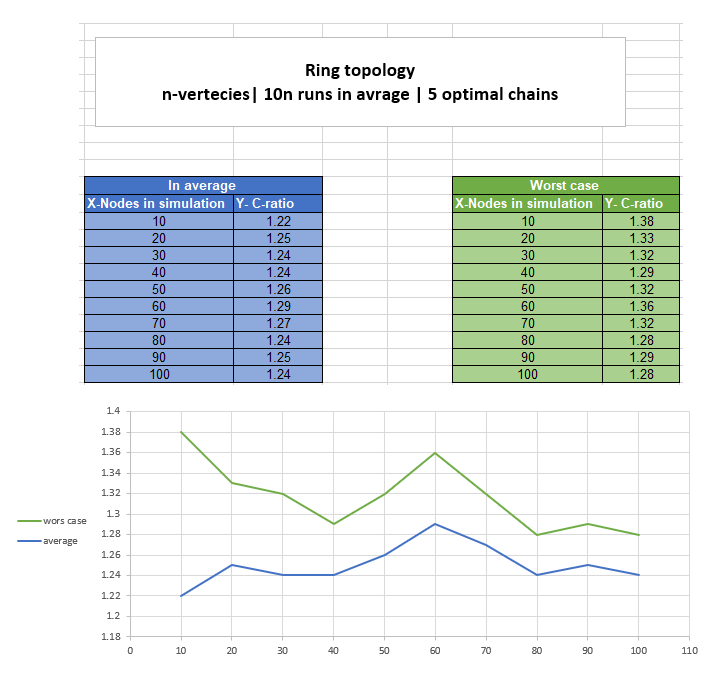
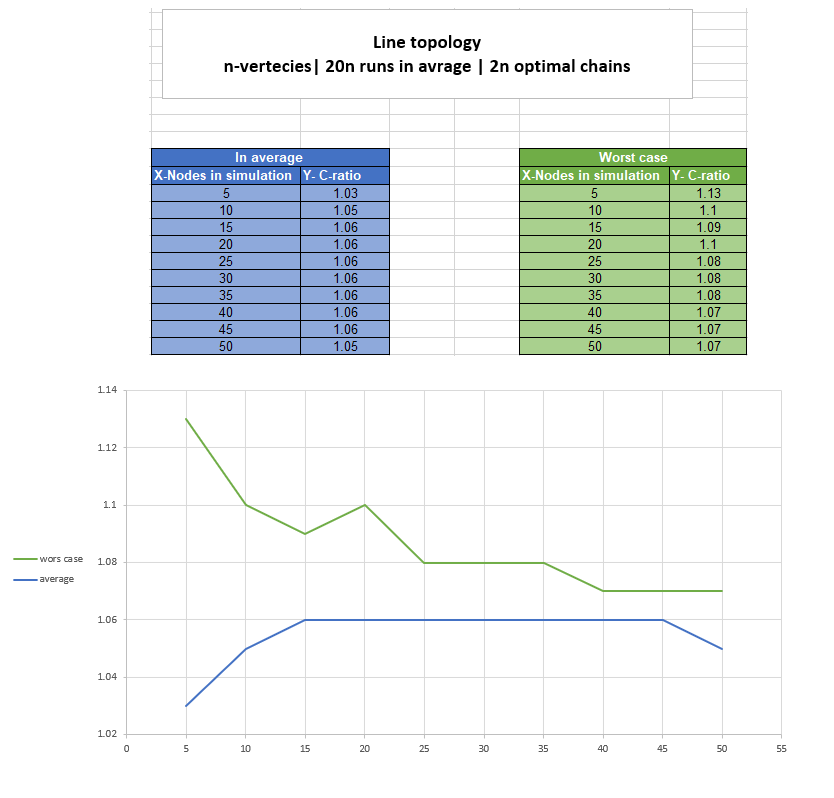
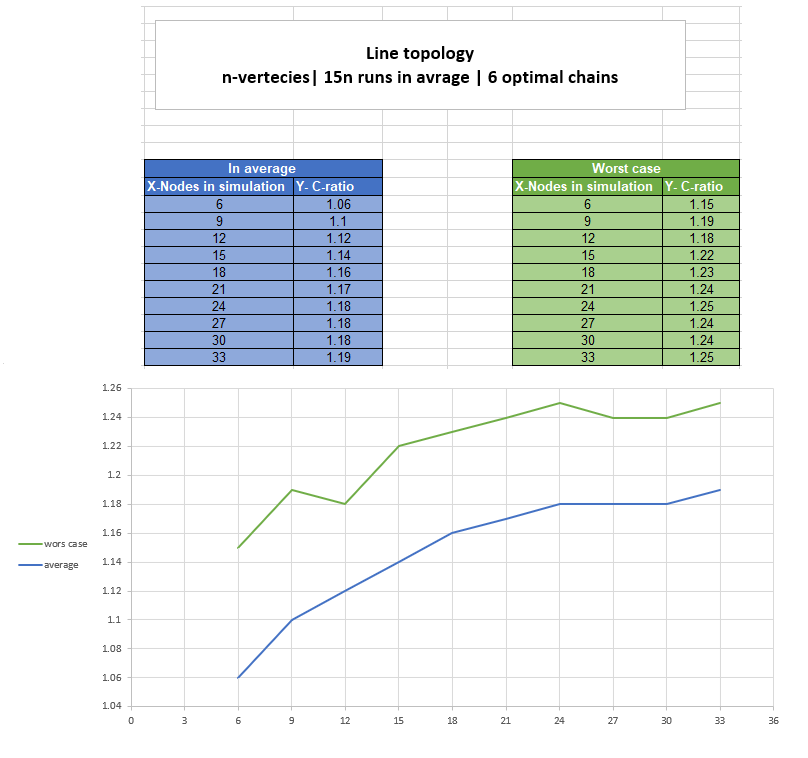
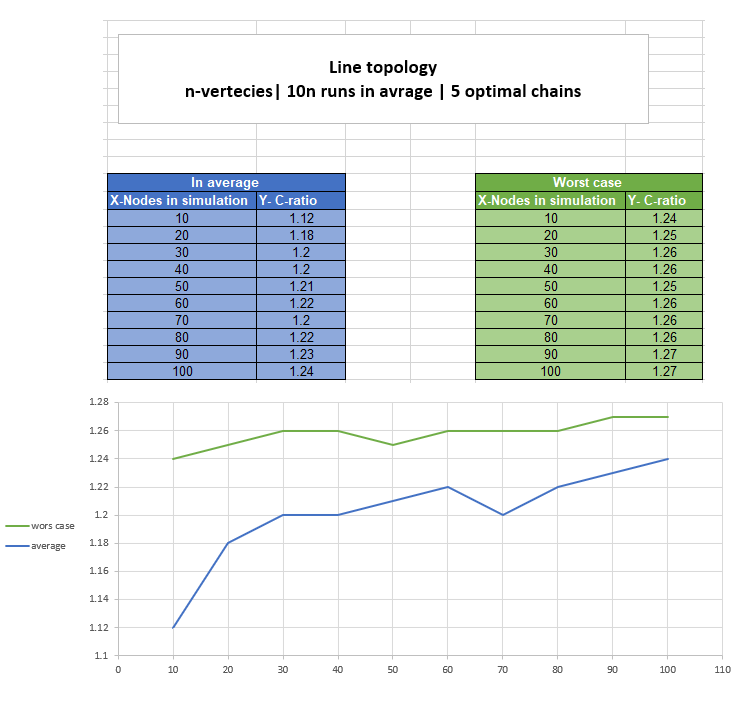
* + 1. **Functionality testing**

|  |  |  |
| --- | --- | --- |
| **Test No.** | **Test subject** | **Results** |
|  | Try to simulate without inserting a number of nodes or/and number of circles | The button simulate will not appear if there isn't an input |
|  | Trying to simulate different input without resting | The text fields will be locked until pressing on reset |
|  | Pressing on the button "Next Step", when went through all the lightpaths | This message will appear "End of simulation !!" |
|  | Calculating the average without giving number of runs. | This message will appear " Please enter number runs !!" |
|  | Trying to calculate another average without resting. | The text fields will be locked until pressing on reset |

* 1. **Result and conclusion**

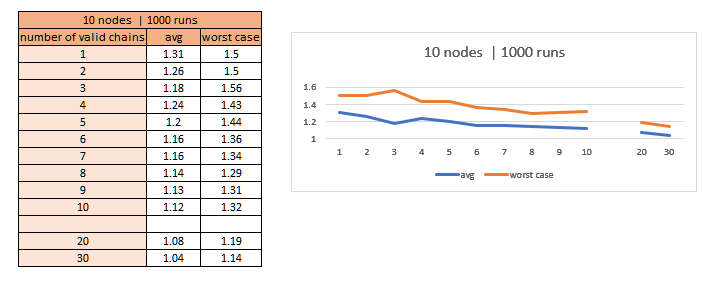
The final and the most important goal is that the competitive ratio must be under the upper ratio for the given topology. Even though it has been proven in the research, seeing it and trying all kind of examples can convince more. That's exactly what happen to us we were very satisfied to see the explanations and the proof come alive, it’s much easier and will help others to understand. Our part was to simulate a ring and line topology with random lightpaths to demonstrate how the lightpaths comes online, what the difference when the lightpaths already known (not taken as input), the cost of ADMs to each case and calculate the competitive ratio. Whenever we had an obstacle, we tried to think each one of us alone to overcome the issue faster and better and share our ideas. The main obstacle was the drawing the network on screen which made us almost rewrite the program 3 time.

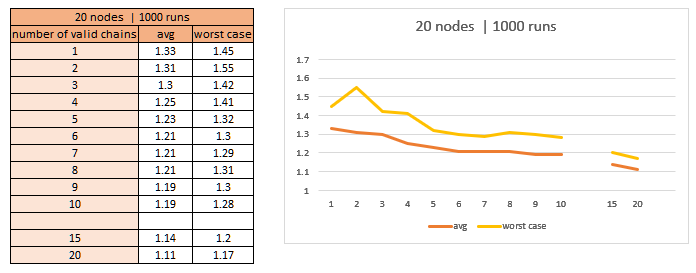
Eventually the result as we see in the graph for the average competitive ratio was is better than the worst case, which means that the algorithm meets the standards that we have discussed in the theoretical part of the project.

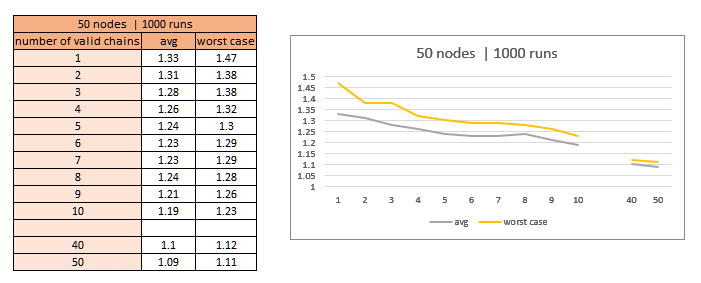


We have also taken another approach to show analysis which takes a similar number of nodes **n** and with a 1000 runs in average over the variable of number of valid chains in each average of runs the changing. Taken the results of the c-ratio in average and in worst case.

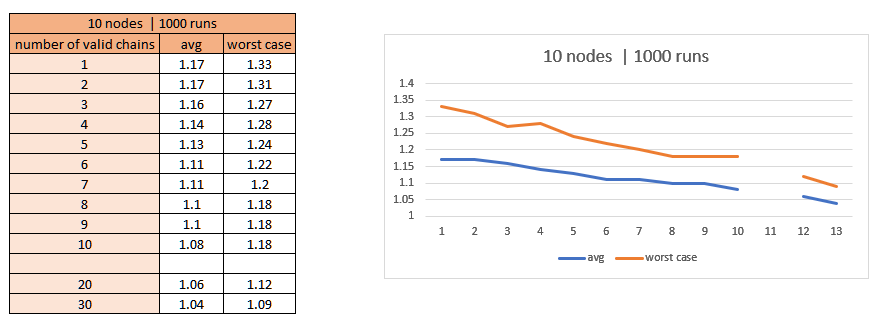
***Ring Topology:***

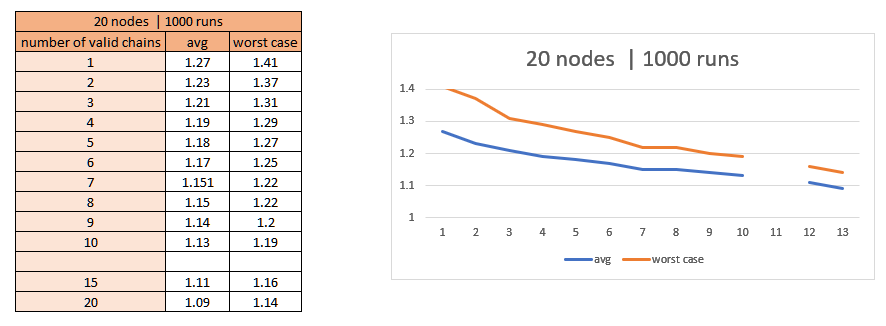


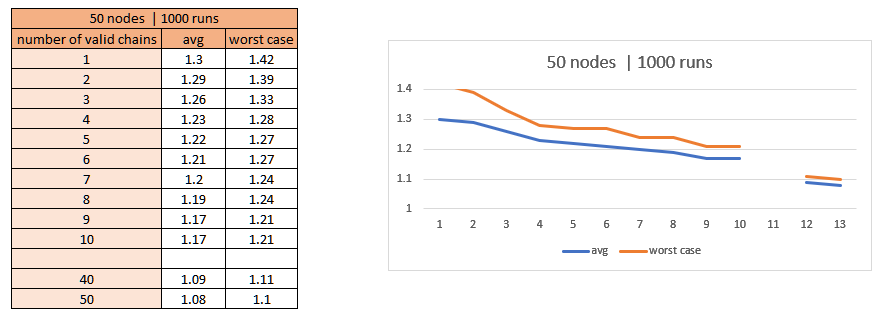




***Line Topology:***







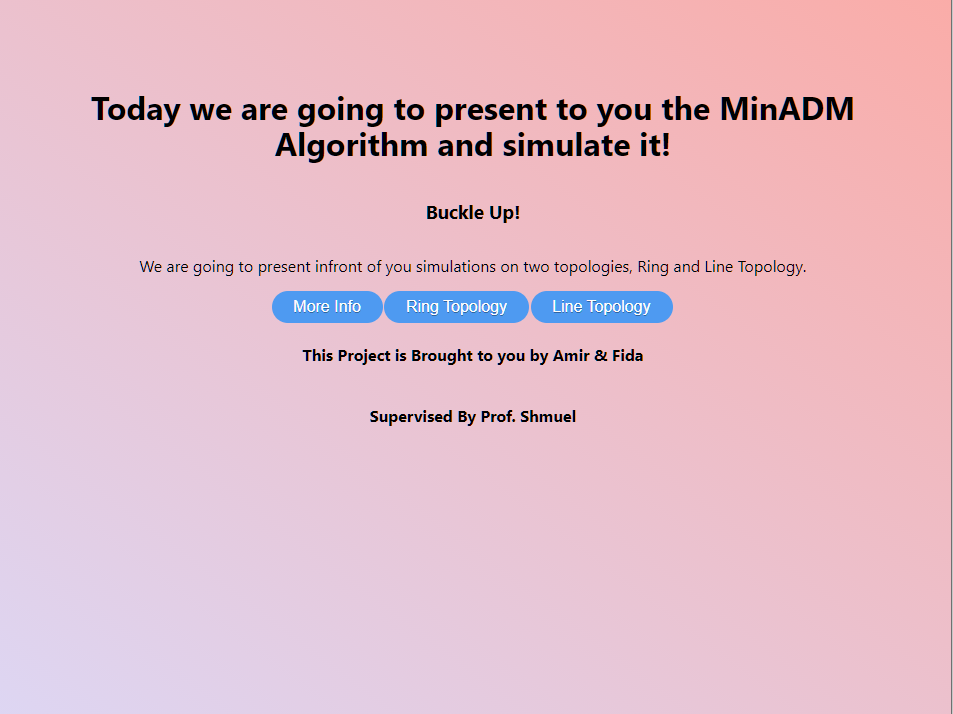
We can see that our analysis shows that with a large number of valid chains the algorithm starts getting closer to the optimum results which needs more research to show what is happening and maybe tune the algorithm itself but for the results shown over here the algorithm meets its requirements and behaves in such a way that does not pass the c-ratio bounds that we set in theory.

1. **User Documentation**
   1. **User's Guide**

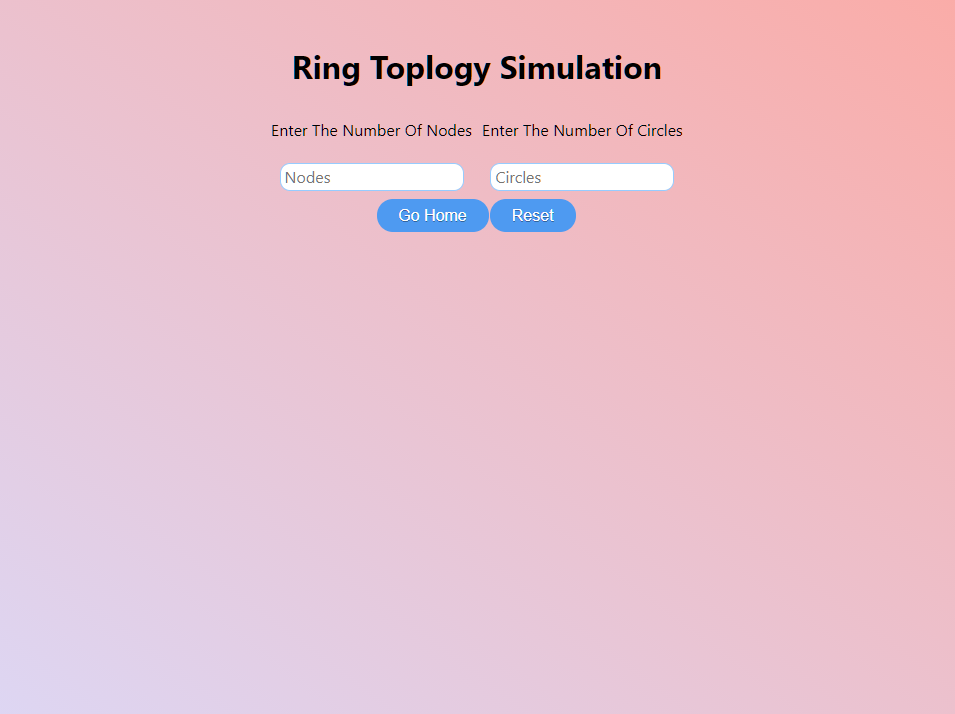
Our project’s main goal is to simulate the algorithm that was presented which is the minADM algorithm, our app was designed in a way that the user can simulate the algorithm itself and get a visual sense of how it works.

We are going to go over the functions that each user can use in our app:

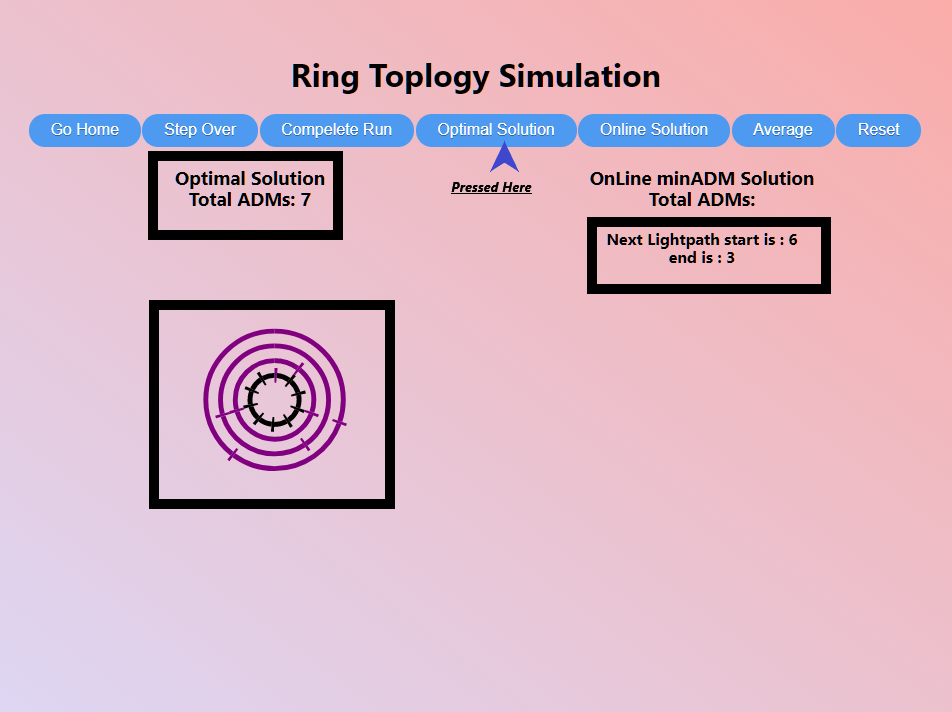
1. At the first screen the user is asked to choose which topology he would like simulate the algorithm on, the user chooses the topology desired.



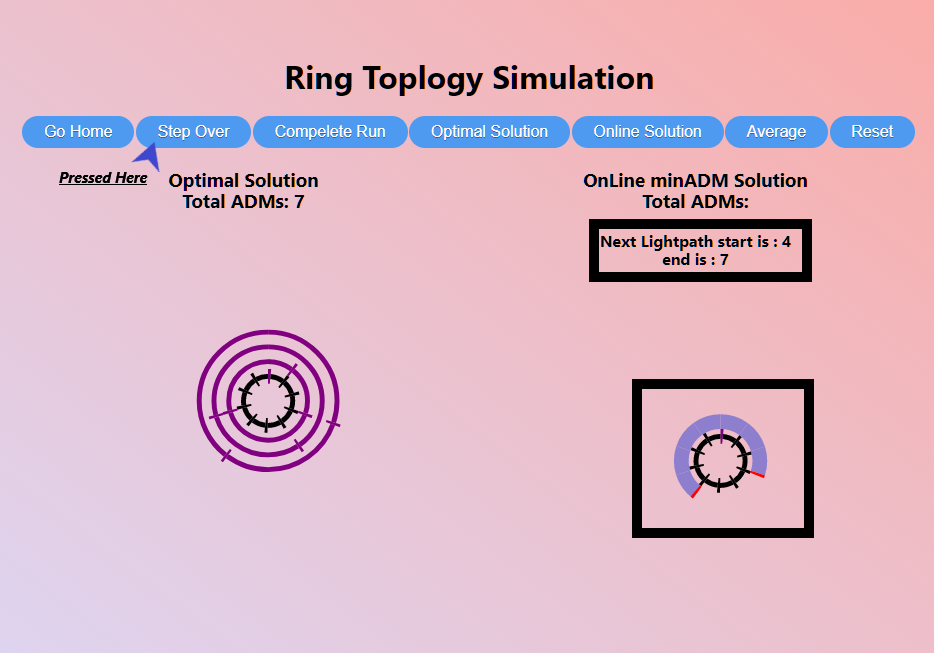
1. After choosing the topology the functions are similar the only change is the way of the calculations in the back and the visual representation of the simulation, for the sake of the example the user selects **RING TOPOLOGY,** the user is then asked to enter the number of the nodes desired to simulate and the number of the optimal chains that the algorithm will produce.



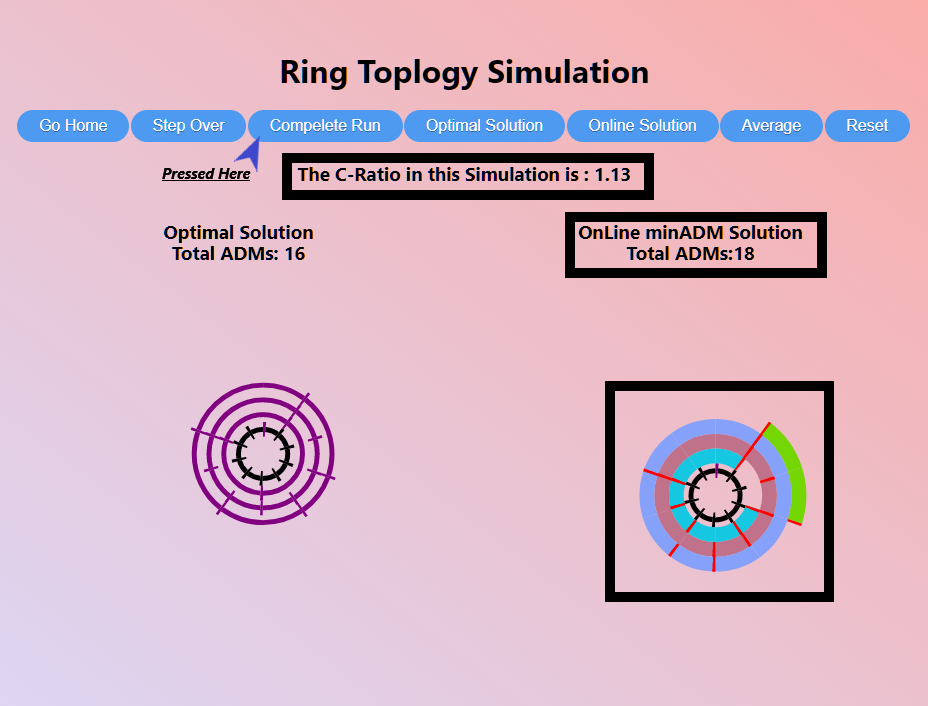
1. The user then enters the desired parameters, *Nodes = 10 , Circles = 3*,and hits ***Simulate***, then the user is moved to another page that in it the user has different buttons that he can use, and is presented with the total number of ADMs that were used in making the optimal ring topology network and which is the next lightpath that the algorithm will use to start calculating the total number of ADMs in the online calculations,  
   the user can press ***Optimal Solution*** to view the representation of the ring network alongside with the optimal chains of lightpaths.



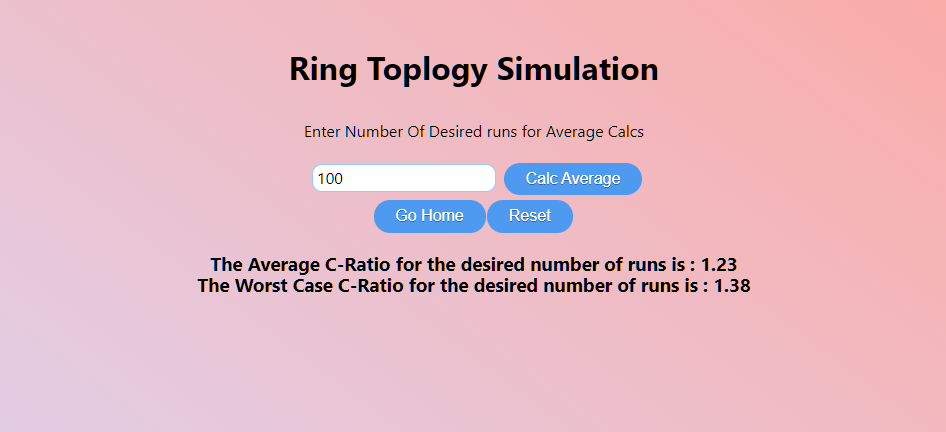
1. The user can hit the ***Step Over*** button to go over the steps that the algorithm will go over while getting as an input the lightpaths, and to get a better sense of the algorithm’s way of working in real-time.



1. The user can press the ***Complete Run*** button to complete the run for the current number of lightpaths that are coming as an input to the algorithm, which will result in showing the user the total number of ADMs that were used in the online version of constructing the lightpaths with the help of our algorithm, alongside with the Competitive ratio.



1. The user can press the ***Average*** button and then he can simulate the algorithm on the same set of lightpaths that were randomly produced, multiple times and calculate the average of these runs and also the worst case of these runs. The user types the desired number of runs that he wishes to calculate the average for.



1. The user can press ***Reset*** and go over the process again or ***Go Home.***  
   1. **Maintenance Guide**

**Running the project in CMD.**

*Requirements for the development environment:*

*-Node.js installed*

*-NPM package manager*

1- after extracting the .zip file we enter ***cd*** into the directory via CMD.  
2- run ***npm install***to install the dependencies of the project.

3- run ***npm start*** to run the start script included which opens the project on localhost with a given port.

**Preparing the project for Development.**

*Requirements for the development environment:*

*-Node.js installed*

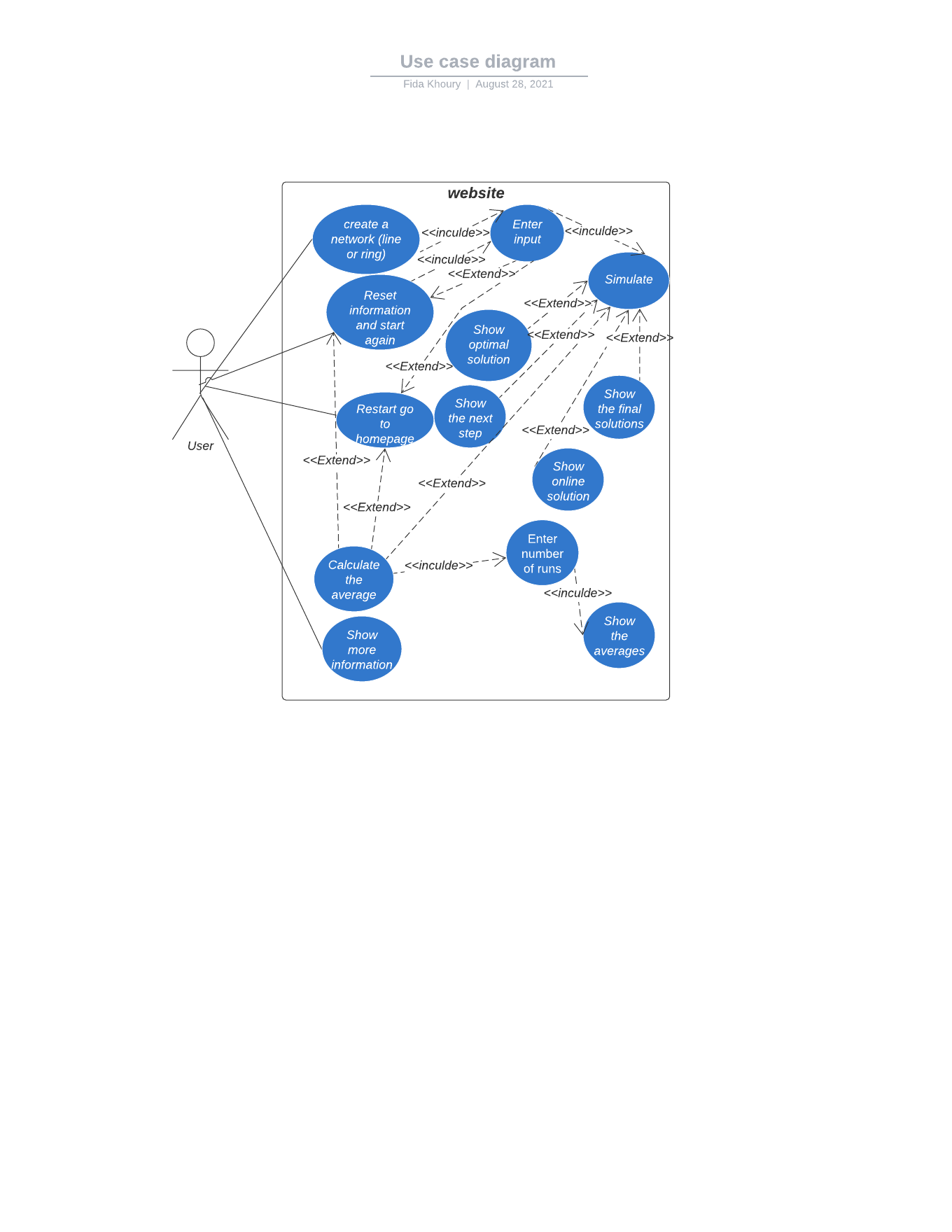
*-NPM package manager*

*-Preferable IDE*

1- after extracting the .zip file we enter ***cd*** into the directory via CMD.  
2- run ***npm install***to install the dependencies of the project.

3-Open the main directory of the project inside your desired IDE and start hacking.

1. **UML Diagrams** 
   1. **Use-Case Diagram**

****

1. **Reference**
2. Liwu Liu, Xiangyang Li, Peng-Jun, Wan Ophir Frieder, Wavelength Assignment in WDM Rings to Minimize SONET ADMs, pp 1-6
3. <https://www.techopedia.com/definition/24120/add-drop-multiplexer-adm>
4. <https://en.wikipedia.org/wiki/Online_algorithm>
5. <https://www.slideshare.net/vikasiitp/online-algorithms-and-their-applications>