Software Engineering Department

Ort Braude College

Course 61999: Extended Project in Software Engineering

Capstone Project Phase B

Project Number: 23-1-R-19

Online minimization of switching cost in optical networks

Students:

Ahmad Aamar - 206713034

Email: ahmad.aamar1410@gmail.com

Mohamed abu assad - 316449495

Email: m7mad114@gmail.com

Supervisor:

Shmuel Zaks

Email: shmuel.Zaks@braude.ac.il

Git Hub link:

mohammad-abu-assad/finalProject (github.com)

Contents

1.Project Review

1.1 Introduction

In Phase A of our research project, we conducted a comprehensive analysis of the ONLINE-MINADM algorithm, which provides an online solution for minimizing the cost of optical networks by effectively coloring lightpaths. Our focus was on two network topologies: the path topology and the ring topology. By studying the algorithm and its performance bounds, we gained valuable insights into its efficiency and optimality.

Our analysis revealed that the ONLINE-MINADM algorithm achieves a competitive ratio of 3/2 for the path topology and 7/4 for the ring topology. This ratio indicates that, on average, the algorithm uses at most 3/2 (or 7/4) times the number of ADMs employed by the optimal offline algorithm in these respective network configurations. These results highlight the algorithm's effectiveness in minimizing the number of ADMs utilized, thus reducing the overall cost of the optical network.

Building upon this analysis, Phase B of our project aims to further investigate and evaluate the performance of the ONLINE-MINADM algorithm in practical scenarios. To achieve this, we will develop a simulation framework that allows us to visualize the algorithm's execution and assess its expected performance in various network topologies, with a specific focus on ring and path configurations.

Our objectives in Phase B are twofold. Firstly, we seek to provide a visual representation of the algorithm's operation in different network topologies. By visualizing the algorithm's execution, we can gain a better understanding of its behavior and identify any patterns or trends that may emerge. This visualization will enable us to observe how the algorithm efficiently assigns colors to lightpaths and manages the use of ADMs in real-time.

Secondly, we aim to evaluate the expected performance of the ONLINE-MINADM algorithm. While the worst-case competitive ratio is a standard metric for online algorithm analysis, we believe that assessing the algorithm's performance against the best possible outcome is more relevant in practical scenarios. By simulating the algorithm in various network topologies, we can gather empirical data on its expected cost compared to the optimal solution. This analysis will provide valuable insights into the algorithm's performance in different scenarios and allow us to identify areas where further improvements can be made.

Our ultimate goal in this project is to develop a practical, efficient, and cost-effective algorithm for minimizing switching costs in optical networks. By combining theoretical analysis with practical simulations, we aim to create an algorithm that not only performs well in theory but also demonstrates its effectiveness in real-world settings. Through our research

and development efforts, we strive to contribute to the optimization of optical networks and provide valuable solutions for reducing operational costs and improving network efficiency.

1.2 Background & related work:

1.2.1 Online Algorithms:

Online algorithms are a specific type of algorithm that process data incrementally as it becomes available, without prior knowledge of the entire input. These algorithms are designed to handle streaming data and make real-time decisions based on limited information. They have gained significant popularity and are widely used in various fields, including real-time systems, machine learning, and network management.

One of the key characteristics of online algorithms is their ability to make decisions without knowledge of future inputs, distinguishing them from offline algorithms that have access to the complete input dataset before making any decisions. This lack of complete information introduces uncertainty and requires online algorithms to be designed to handle such uncertainty and make decisions based on limited information.

In network management, online algorithms are used to dynamically allocate resources in response to changing network conditions. This includes tasks such as bandwidth allocation and routing decisions. Online algorithms in network management must be efficient, scalable, and robust to handle the large volume of requests that arrive in a network. They play a critical role in optimizing network performance and resource utilization.

To evaluate the performance of online algorithms, a methodology called competitive analysis is commonly employed. Competitive analysis provides a framework for measuring how well an online algorithm approximates the optimal solution. The competitive ratio is a key measure used in this analysis, which compares the cost incurred by the online algorithm to the cost of the optimal offline algorithm.

For example, let's consider the "rent-or-buy" problem as discussed in [3]. Imagine you are going on a ski vacation, but the number of ski days is unknown. Renting skis costs $10 per day, while buying skis incurs a fixed price of $100. The objective is to minimize the total cost of skiing. If the number of ski days were known in advance, the decision between renting and buying would be straightforward. If the number of days exceeds 10, it would be more cost-effective to purchase skis on the first day. However, if the number of days is less than or equal to 10, renting skis would be the better option.

In the context of competitive analysis, the worst-case scenario is considered. Suppose you decide to buy the skis and end up skiing only once during your vacation. In this case, you would pay $100, whereas the optimal solution would be to rent skis for $10. Hence, the competitive ratio for this scenario is 100/10 = 10. However, by employing a different strategy, such as renting skis until realizing the need to buy, it is possible to achieve a competitive ratio of ≤ 2.

Another illustrative example of online algorithms is the "list accessing problem" discussed in [2]. In this problem, you have a filing cabinet containing l labelled and unsorted files. Requests to access specific files arrive one by one, and the goal is to minimize the cost of accessing these files. Each access operation incurs a cost equal to the position of the accessed file in the list. Additionally, if a file is not found, it must be returned to the cabinet before the next request arrives. Reorganizing the list incurs a cost equal to the current number of files plus one.

The lower bound for this problem is established through a proof. It is shown that for any deterministic online algorithm, the competitive ratio is at least 2 - 2/(l+1). This implies that no deterministic online algorithm can achieve a competitive ratio better than this lower bound. Thus, competitive analysis provides insights into the inherent limitations of online algorithms for certain problems.

Online algorithms find applications in various domains, including data structures and graph theory. For instance, in the implementation of an Abstract Data Type (ADT) called a priority queue, an online algorithm may be used to handle incoming insertions and deletions efficiently.

Similarly, online algorithms are employed in graph colouring problems, where the goal is to assign colours to the vertices of a graph subject to certain constraints. The ability of online algorithms to make decisions based on partial information makes them well-suited for these types of problems.

In conclusion, online algorithms process input data incrementally as it becomes available, without prior knowledge of the entire input. They are commonly used in real-time systems, machine learning, and network management. Competitive analysis is a valuable tool for evaluating the performance of online algorithms by comparing their results to an optimal offline solution. While online algorithms may not always achieve the same level of optimality as offline algorithms, they play a crucial role in real-time decision-making scenarios and provide efficient solutions for a wide range of computational problems. This project

specifically focuses on online algorithms used to assign wavelengths to lightpaths in optical networks, aiming to minimize the network cost by reducing the number of ADMs used.

1.2.2 Optical Networks:

Optical networks are communication networks that use light waves to transmit data. These networks are built using optical fibers, which are thin strands of glass or plastic that can carry light over long distances. The light waves used in optical networks are at different frequencies, known as wavelengths, and this feature is used to multiplex multiple channels of data onto a single optical fiber, a process known as wavelength-division multiplexing (WDM).

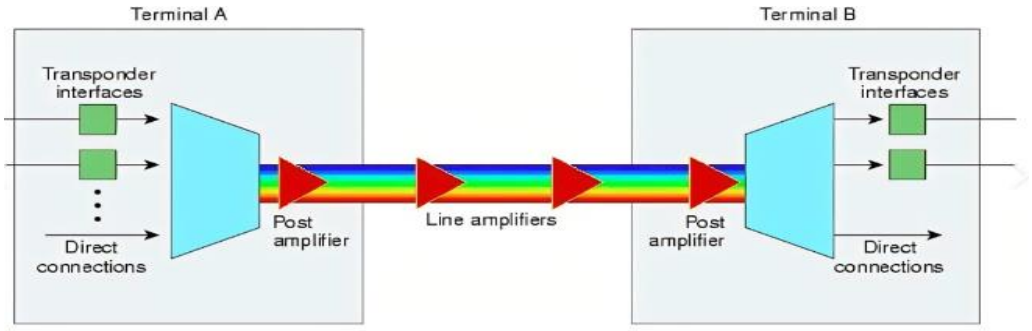


Fig.1: wavelength-division multiplexing (WDM). (See more in [4])

Optical networks have several advantages over traditional copper-based networks, such as increased bandwidth, longer transmission distances, and greater resistance to electromagnetic interference. These advantages have led to the widespread adoption of optical networks in various fields, including telecommunications, data centers, and long-distance data transmission.

In optical networks, data is transmitted in the form of optical signals, which are generated by a device called a transmitter and received by a device called a receiver. These signals are transmitted through optical fibers, which are connected to various network elements such as amplifiers, multiplexers, and switches. These network elements are used to control and manage the flow of data within the network.

One of the key elements of optical networks is the Add-Drop Multiplexer (ADM), which is used to add or drop specific wavelengths of light to or from a specific optical fiber. ADMs are used to create lightpaths, which are paths between two nodes in the network that are assigned a specific wavelength. The total number of ADMs used in a network is considered to be the total cost of the network, and minimizing this cost is an important goal in the design and management of optical networks.

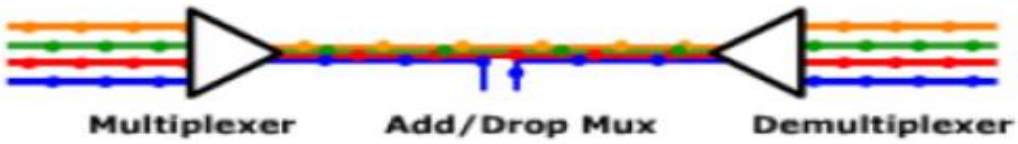


Fig.2: Add-Drop Multiplexer (ADM). (See more details in [5])

Multiplexer: A multiplexer (MUX) is a device that combines multiple signals into one signal for transmission over a shared medium. It allows multiple data streams to be transmitted over a single communication channel by switching between the inputs at a high rate

Demultiplexer: A demultiplexer (DEMUX) is a device that separates a single signal into multiple signals. It is the inverse of a multiplexer and is used to extract individual data streams from a multiplexed signal

1.3. The Algorithm:

The ONLINE-MINADM algorithm is a powerful online strategy for efficiently managing lightpaths in a network. It is specifically designed to dynamically assign wavelengths and minimize the number of Add-Drop Multiplexers (ADMs) used, resulting in cost reduction for the optical network .

With a competitive ratio of 7/4, the algorithm demonstrates its effectiveness by ensuring that the number of ADMs utilized is at most 7/4 times the number used by an optimal offline strategy. This competitive ratio provides a quantitative measure of the algorithm's performance, highlighting its efficiency and approximation capabilities in real-world scenarios.

The algorithm operates by employing a simple yet effective coloring procedure when adding lightpaths to the network incrementally. When a new lightpath arrives with endpoints u and v, the algorithm first checks for available free ADMs at these endpoints. An ADM is considered free if it is serving only one lightpath that ends at the same endpoint and does not share an edge with the new lightpath.

In cases where there are two ADMs of the same color, the algorithm attempts to connect them to form a cycle with existing lightpaths. If this is not possible, a path is formed instead. When free ADMs are available at one or both endpoints, the algorithm strives to connect to

them, optimizing resource utilization. However, if no free ADMs are available, the new lightpath is assigned a new color.

The crucial aspect of assigning colors is ensuring feasibility. A color λ is considered feasible for a lightpath pi if it does not overlap with any other lightpaths that already have the same color. This feasibility condition guarantees a proper coloring for the set of lightpaths Pi.

The main steps of the ONLINE-MINADM algorithm can be summarized as follows:

Upon the arrival of a new lightpath pi with endpoints ui and vi, the algorithm checks the following conditions:

1 .If there exists a chain of lightpaths with color λ that have endpoints ui and vi, and color λ is feasible for lightpath pi, then pi is assigned color λ (w(pi) = λ).

2 .If a chain of lightpaths with color λ exists that has one endpoint from {ui, vi}, and color λ is feasible for lightpath pi, then pi is assigned color λ.

3 .If no suitable color can be found based on the above conditions, a new unused color λ' is assigned to lightpath pi.

It is important to note that when the algorithm resorts to using an unused color in the last clause, it ensures that no two chains will have the same color. As a result, in the first clause, the algorithm will always form a cycle, optimizing the allocation of resources.

By employing the ONLINE-MINADM algorithm and carefully considering the order of lightpath arrivals, network operators can significantly minimize the number of required ADMs and achieve cost savings. This algorithm, with its competitive ratio and efficient coloring procedure, offers a practical and effective approach to manage lightpaths in dynamic optical networks.

Example:

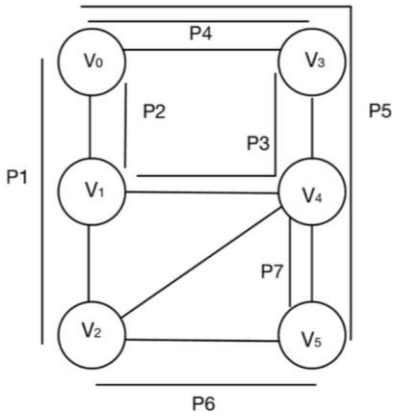


Fig.3. the input.

In the given scenario, we have a graph represented as g = {V, E}, where V = {v0, v1, v2, v3, v4, v5} and E = {e0 = (v0, v1), e1 = (v1, v2), e2 = (v3, v4), e3 = (v4, v5), e4 = (v0, v3), e5 = (v1, v4), e6 = (v2, v4), e7 = (v2, v5)}. We also have a set of light paths P = {p1 = (e0, e1), p2 = (e0), p3 = (e2, e5), p4 = (e4), p5 = (e4, e2, e3), p6 = (e7), p7 = (e3)}, as shown in **Figure 3**.

The sequence of paths for analysis is p1, p5, p6, p2, p4, p3, p7. Let's examine how the ONLINE-MINADM algorithm operates in this scenario.

When p1 arrives, the algorithm assigns two ADMs to its endpoints (v0, v2) because it is the first path in the sequence. The algorithm assigns a new color, denoted as red, to p1, i.e., w(p1) = red. Subsequently, when p5 arrives, it finds that there is a free ADM in node v0, and the color red is feasible for it. Hence, the algorithm assigns the color red to p5 as well. Similarly, when p6 arrives, it discovers that there is one free ADM in v2 and one free ADM in v5, both with the same color red, making the color red feasible for p6. As a result, p6 closes the cycle and effectively saves two ADMs. Similarly, w(p1) = w(p4) = w(p3) = blue.

Finally, when p7 arrives, the algorithm determines that there are no free ADMs available at its endpoints. Thus, it introduces a new color, denoted as green, and assigns w(p7) = green, as depicted in **Figure 4.**

By meticulously managing the allocation of ADMs and utilizing the feasibility of colors, the ONLINE-MINADM algorithm effectively minimizes the number of ADMs required in the network. This approach optimizes resource utilization and contributes to cost reduction in the optical network.

It is important to note that the given sequence of paths p1, p5, p6, p2, p4, p3, p7 has been considered for analysis purposes. The order of path arrivals can significantly impact the overall cost and efficiency of the algorithm in different scenarios.

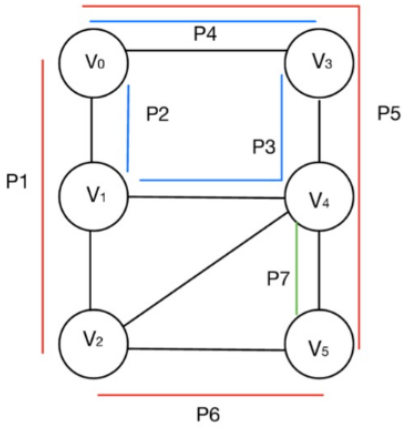


Fig.4. optimal solution

The solution S= {(p_1, p_5, p_6), (p_2, p_4, p_3), p_7} and this is the optimal.

the optimal solution saves 6 ADMs

8 ADMs.

Consider the given sequence of paths: p4, p1, p6, p7, p5, p2, p3. Let's analyze how the ONLINE-MINADM algorithm handles this scenario.

When p4 arrives, the algorithm recognizes that the network is initially empty, so it assigns a new color, denoted as blue, to p4. Consequently, we have w(p4) = blue. Next, when p1 arrives, the algorithm identifies a free ADM in node v0, allowing it to assign the color blue to p1 as well. Similarly, when p6 and p7 arrive, they encounter free ADMs at their respective endpoints and are assigned the color blue.

Moving on to p5, the algorithm introduces a new color, red, since there are no existing chains that can accommodate it. Hence, w(p5) = red. For p2 and p3, the algorithm opts to assign them the same color, denoted as green, because they can share an ADM at v1. Consequently, we have w(p2) = w(p3) = green. **Figure 5** provides an illustration of this scenario.

By applying the ONLINE-MINADM algorithm to this sequence of paths, we obtain the solution S = {(p4, p1, p6), (p7, p5, p2, p3)}. Notably, this solution represents the optimal allocation of ADMs for this particular sequence, resulting in a savings of 6 ADMs.

It's worth emphasizing that the order in which the paths arrive can significantly impact the resulting allocation and overall cost. The ONLINE-MINADM algorithm adapts to the dynamic nature of the network by intelligently assigning colors and leveraging existing chains of lightpaths, ultimately achieving an efficient utilization of resources.

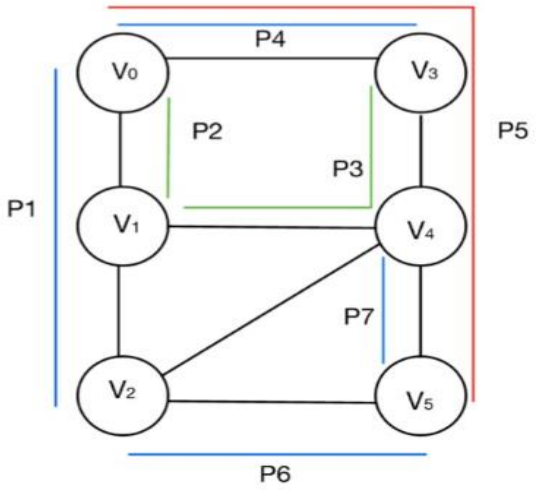


Fig.5. Online-MINADM solution

The solution S= {(p_4, p_1, p_6, p_7), (p_2, p_3), p_5}

The solution saves 6 ADMs.

The correctness of the Online-MINADM algorithm lies in its ability to assign a proper coloring for the set of lightpaths Pi. The algorithm considers two cases when assigning a color to a lightpath pi: either there exists a chain of lightpaths with the same color λ that have endpoints ui and vi, or there exists a chain of lightpaths with color λ that have one endpoint from {ui, vi}. In both cases, the algorithm checks the feasibility of the color λ for lightpath pi before making the assignment. If neither of these cases is satisfied, a new unused color is assigned to the lightpath, ensuring that no other conflicting lightpath will have the same color.

To prove the correctness of the algorithm, we need to show that the resulting coloring satisfies the constraints of a proper coloring. A proper coloring ensures that no two adjacent lightpaths share the same color. In the Online-MINADM algorithm, the assignment of colors follows a systematic process that guarantees the absence of conflicts.

In the first two cases, where a chain of lightpaths with the same color or one endpoint sharing the same color exists, the algorithm checks the feasibility of the color for the new lightpath pi. Feasibility means that assigning the color λ to pi will not result in a conflict with other lightpaths that already have the color λ. This step ensures that the resulting coloring is proper for those lightpaths.

In the third case, where a new unused color is assigned, the algorithm ensures that no two chains of lightpaths will have the same color. By assigning a new color to pi, the algorithm

prevents conflicts with existing chains and maintains a proper coloring for the set of lightpaths Pi.

Therefore, by applying these rules for assigning colors, the Online-MINADM algorithm guarantees a proper coloring for the set of lightpaths Pi, satisfying the constraints of the problem.

Furthermore, the Online-MINADM algorithm has been proven to be highly efficient for various network configurations. For general and ring network topologies, the algorithm achieves a competitive ratio of 7/4, meaning that the number of ADMs used by the algorithm is at most 7/4 times the number used by the optimal offline strategy. For path network topologies, the Online-MINADM algorithm is even more optimal, with a competitive ratio of 3/2.

In the subsequent sections, we will delve into the upper and lower limits of the algorithm's performance and provide further evidence to support the claim that the Online-MINADM algorithm is the best choice for general, ring, and path network configurations.

1.4.Research process:

In this project, we embark on tackling the challenging problem of minimizing the cost of an optical network by reducing the number of Add-Drop Multiplexers (ADMs) required. ADMs are crucial components in optical networks that facilitate wavelength switching. Traditionally, studies addressing this problem have focused on the offline scenario, where all the light paths are known in advance. However, in real-world scenarios, light path requests arrive dynamically, and it becomes essential to assign wavelengths efficiently while minimizing costs.

To address this dynamic and uncertain nature of optical networks, we turn our attention to online algorithms. Online algorithms are specifically designed to handle situations where inputs arrive incrementally, without prior knowledge of future inputs. They make decisions based on limited information, which aligns perfectly with the challenges presented by optical network management.

In our project, we explore the online algorithm proposed in [1] that effectively assigns wavelengths to light paths in optical networks while aiming to minimize the number of

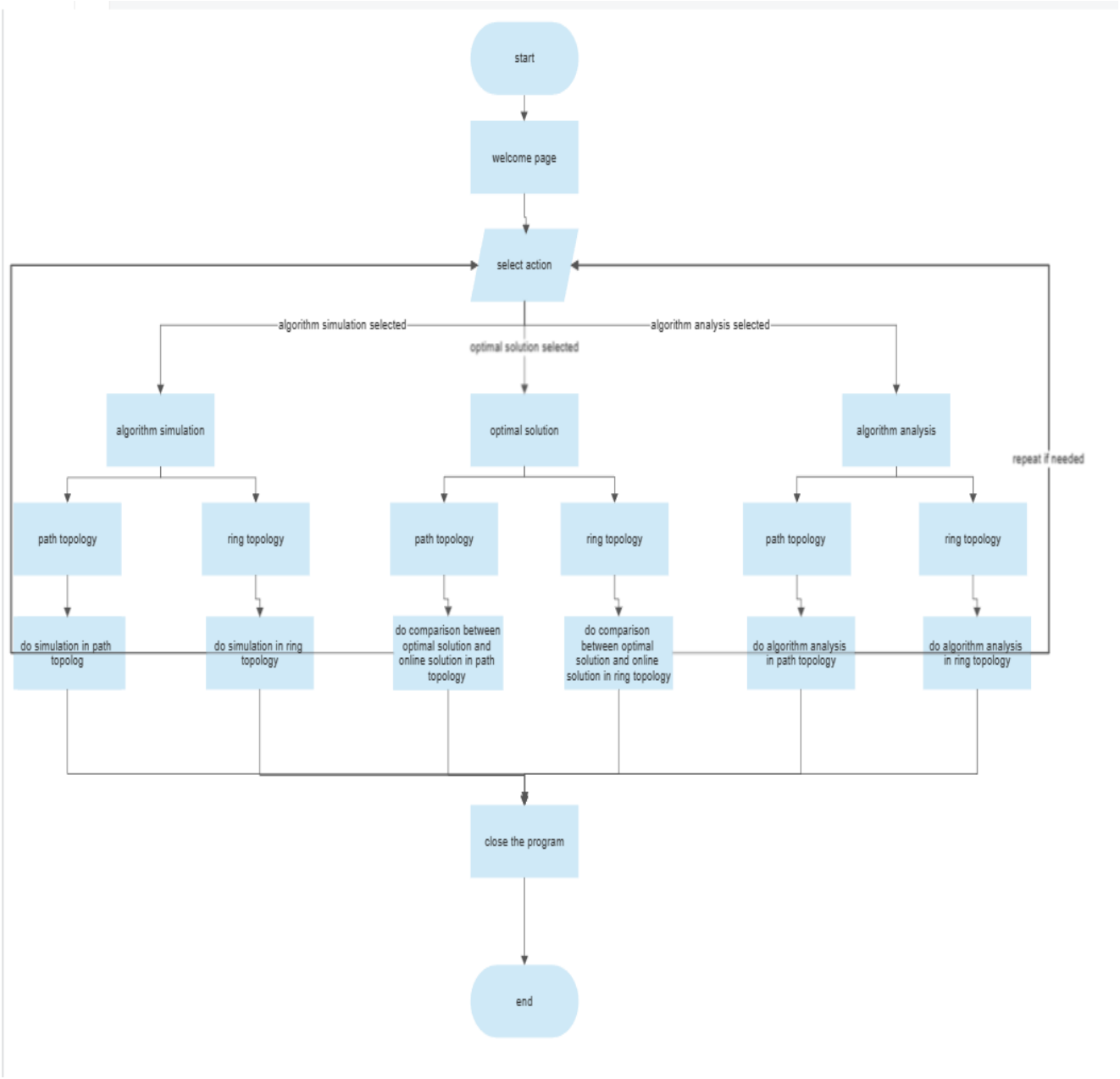
required ADMs. The algorithm takes into account the dynamic nature of incoming light path requests and makes optimal decisions in real-time.

In Part A of our project, we undertook thorough research into the domains of optical networks and online algorithms. We delved into the existing information, examining previous studies and the challenges they addressed. Building upon this foundation, we closely studied the online algorithm presented in [1]. To enhance the understanding and applicability of the algorithm, we provided insightful remarks, illustrative examples, and detailed proofs. By doing so, we aimed to expand upon the ideas and findings presented in the original paper, offering a comprehensive and enriched understanding of the algorithm's functionality and efficiency.

Moving forward to Part B of our project, we are focused on the practical implementation and analysis of the online algorithm. We plan to develop a software tool that will allow us to visualize the algorithm's execution and evaluate its average performance. While the analysis of the algorithm's worst-case performance, as measured by the competitive ratio, is crucial, we also aim to explore the average case performance. By examining the algorithm's behavior in different scenarios and analyzing its average performance, we can gain deeper insights into its practical efficiency and applicability in real-world optical networks.

Throughout the project, we strive to contribute to the understanding and advancement of online algorithms in the context of optical network management. By studying and evaluating the proposed algorithm's performance, both theoretically and empirically, we aim to shed light on its strengths, limitations, and potential areas for improvement. Ultimately, our project seeks to provide valuable insights into the dynamic assignment of wavelengths in optical networks and its implications for minimizing network costs.

1.4. Flow charts

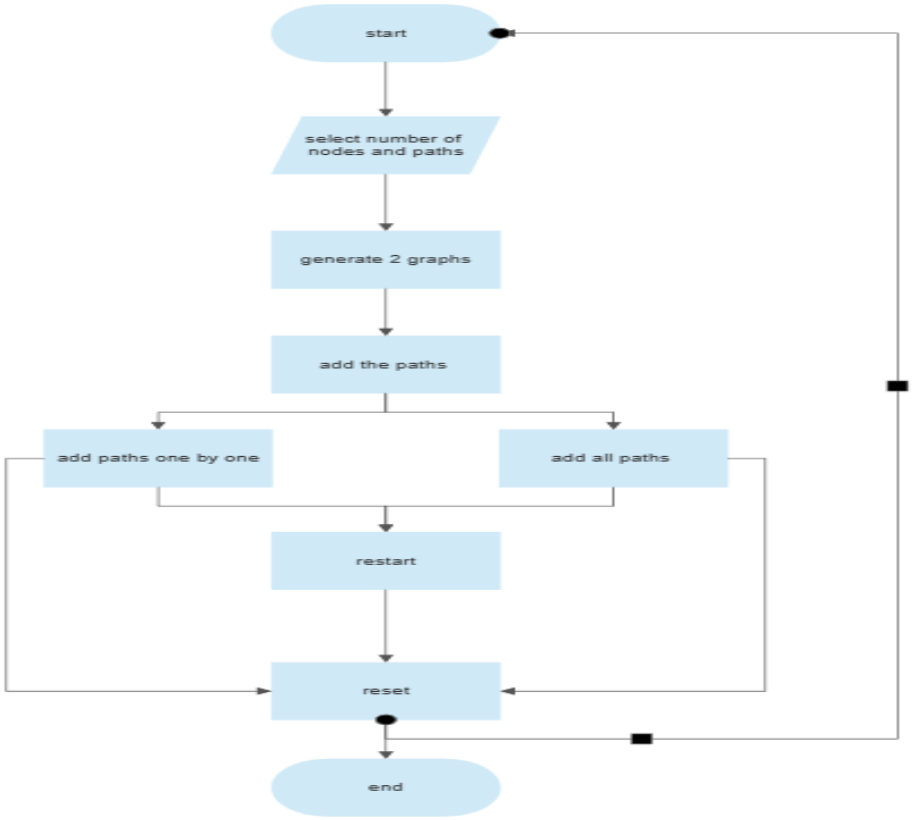


The above flow chart depicts the program that we developed, such that when the program starts the user will see a welcome page that displays 3 buttons, one for algorithm simulation, one for comparison the optimal solution with the online solution and the last one for algorithm analysis. Each one of these actions can be done in path or ring topologies.

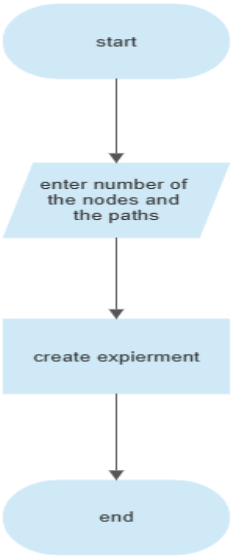
In the following flow charts, we will zoom in on each one of these actions.

The above flow chart describes how to do algorithm simulation in path topology.

The above flow chart describes how to do algorithm simulation in ring topology.



The above flow chat explains how to do comparison between optimal and online solution in ring and path topologies.



The above flow chat explains how to do algorithm analysis in ring and path topologies.

1.5. Results

Based on the aforementioned information, we conducted extensive testing of the algorithm on various network sizes and different orders of lightpaths within each network. The purpose was to calculate the average solution of the algorithm for each network and input, allowing us to assess its performance.

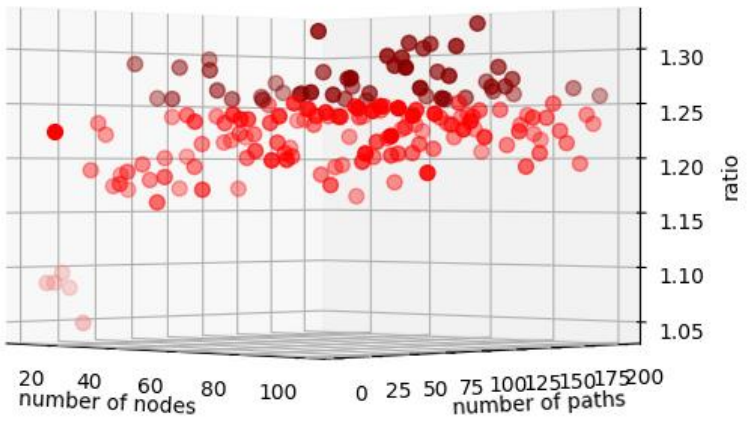
The algorithm was tested on networks ranging from 10 to 100 nodes. For greater precision, users can run the algorithm on additional network sizes. The average ratio, which represents the relationship between the algorithm's average solution and the optimal solution, was computed for both path and ring topologies.

In the case of the path topology, the average ratio obtained was 1.22. This indicates that, on average, the algorithm's solution was 22% worse than the optimal solution. It is noteworthy that the competitive ratio for this topology was initially determined to be 3/2 (1.5). However, our experimental results show a significantly better performance than anticipated.

For the ring topology, the average ratio obtained was 1.27. This indicates that, on average, the algorithm's solution was 27% worse than the optimal solution. Similarly, our experimental results outperformed the competitive ratio of 7/4 (1.75) initially estimated for this topology.

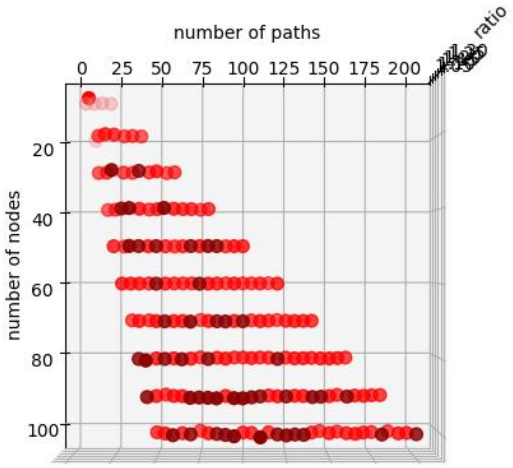
To visualize the results in the path topology, a 3D graph was created. The x-axis represents the number of nodes in the network, the y-axis represents the number of paths in the input, and the z-axis represents the average competitve ratio. In the graph, darker red points represent results with a ratio higher than 1.25, light red points represent ratios between 1 and 1.1, and the remaining points are red.

These findings indicate that the algorithm consistently produces solutions that are notably better than the worst-case scenario predicted by the competitive ratios. It demonstrates the algorithm's effectiveness and suggests that it can provide efficient solutions for various network sizes and input configurations in both path and ring topologies.



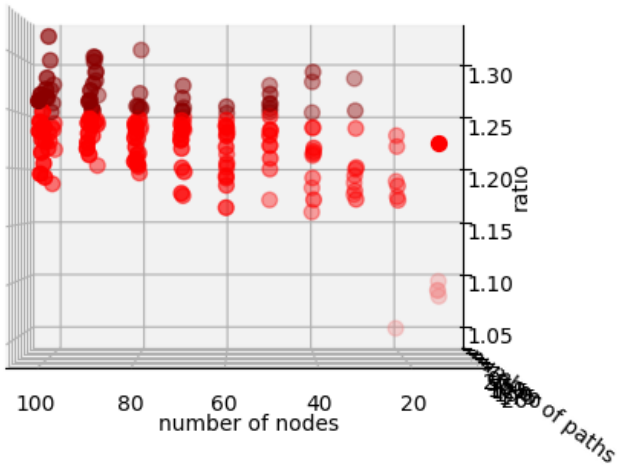
Path topology results

The software provides users with the ability to view the 3D graph from various angles, allowing them to examine it from different perspectives. By exploring the graph from these angles, users can gain valuable insights and draw meaningful conclusions. By focusing on three particularly interesting angles, additional insights can be gleaned, leading to further observations and conclusions.



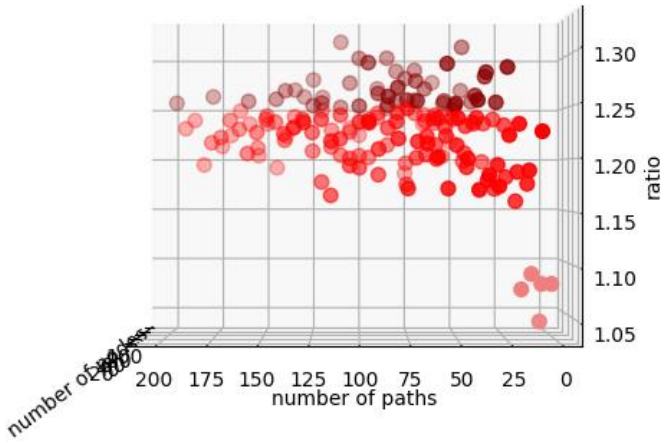
Path topology results, x-y angel

When considering the x-y angle, an analysis of the algorithm's performance was conducted on various networks. The algorithm was executed on networks with 'X' nodes, where the number of paths ranged from 'X/2' to '2X'. For instance, networks with 10 nodes were tested with 5, 10, 15, and 20 paths. For each network configuration, the algorithm was run on different orders of the input paths, and the average results were calculated.



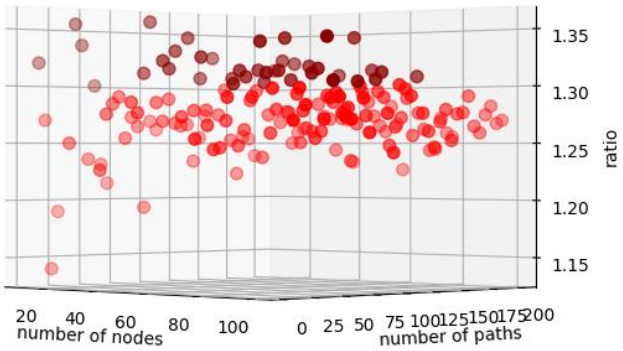
Path topology results, x-z angel

In the x-z angle analysis within the path topology, we examined the relationship between the number of nodes and the average ratio. Notably, the highest ratios were obtained in networks of size 80 and 100. However, it is worth mentioning that no points were found with ratios exceeding 1.5, which aligns with our expectations and indicates that the algorithm's performance remains within the expected bounds in this context.



Path topology result, y-z angel

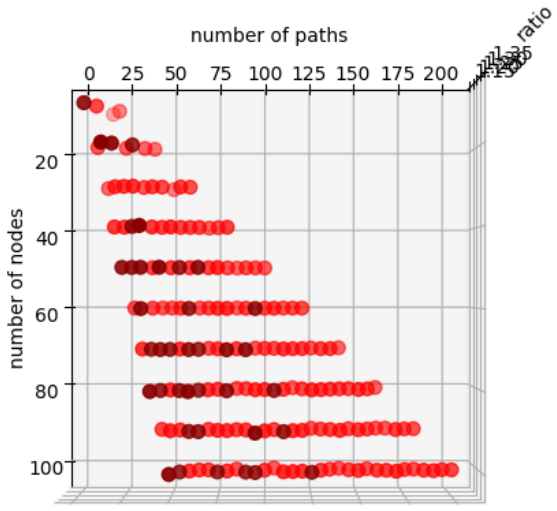
In the y-z analysis of the path topology, we examined the relationship between the number of paths and the resulting ratio. Notably, it was observed that the highest ratios consistently occurred on the right side of the graph, where the number of paths was smaller.



Ring topology results

This graph depicts the results from the ring topology. The subsequent graphs illustrate different angles of analysis within this topology. Similarly to the path topology, the points on these graphs are color-coded in varying shades of red based on the ratio of the result.

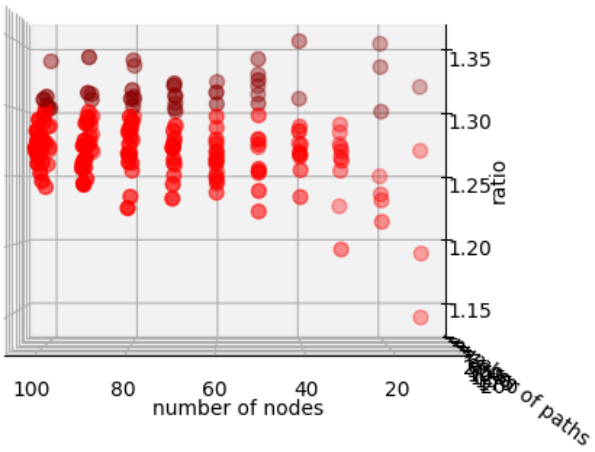
In particular, points with ratios ranging from 1 to 1.1 are represented by a light red color. Points with ratios between 1.1 and 1.3 are depicted as bright red. Lastly, all points with ratios exceeding 1.3 are shown as dark red on the graph. This color classification provides a visual distinction and helps identify the relative magnitudes of the result ratios within the ring topology.



Ring topology results, x-y angel

In the x-y analysis within the ring topology, we explored the relationship between the number of nodes, the number of paths, and the corresponding results. Interestingly, unlike the findings in the path topology, it appears that in almost every column of the graph, there is at least one data point marked as dark red.

This observation aligns with expectations, considering that the competitive ratio of the algorithm in the ring topology is higher, specifically 1.75, compared to the 1.5 ratio in the path topology. Although the threshold for darker red points is higher in the ring topology, the presence of such points in most columns indicates that the algorithm achieves higher ratios in this particular network topology.

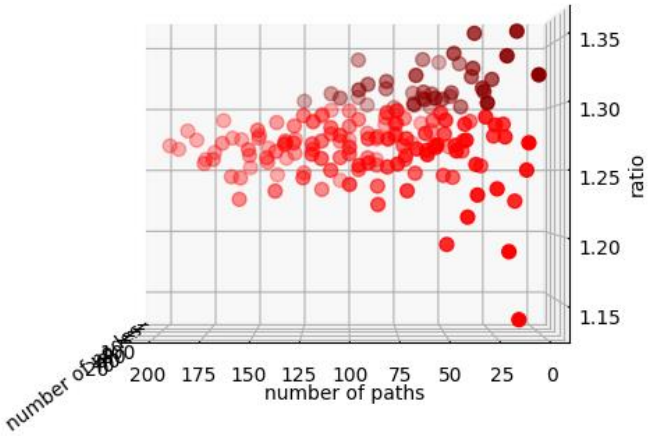


Ring topology results, x-z angel

In the x-z analysis within the ring topology, we examined the relationship between the number of nodes and the resulting ratio. Notably, for nodes ranging from 40 to 100, the minimum ratio obtained was higher than 1.2, indicating that the algorithm's performance remains above the specified threshold in this range.

The graph displays dense points, primarily concentrated within the range of 1.2 to 1.3 for the ratio. This clustering suggests that a majority of the results fall within this range, indicating consistent performance within that particular ratio interval.

As expected, no points with a ratio higher than 1.75 were observed, aligning with the anticipated behavior based on the competitive ratio of the algorithm in the ring topology.



Ring topology results, y-z angel

In the y-z analysis within the ring topology, we explored the relationship between the number of paths and the resulting ratio. Notably, it was observed that the lowest and highest ratios tend to occur on the right side of the graph, where the number of paths ranges from 10 to 50.

1.6. Conclusion

The "ONLINE-MINADM algorithm visualizer" software successfully achieves the primary objectives of this project phase. It effectively implements and visualizes the algorithm studied during phase A, allowing for a comprehensive understanding of its workings. The software also enables the algorithm to be tested on various network sizes and inputs, facilitating an assessment of its average performance.

2. User Documentation

2.1 User Guide

The purpose of the "ONLINE-MINADM algorithm visualizer" software is to provide a visual representation of the algorithm described in [1] and analyze its performance, serving as a valuable addition to the referenced paper.

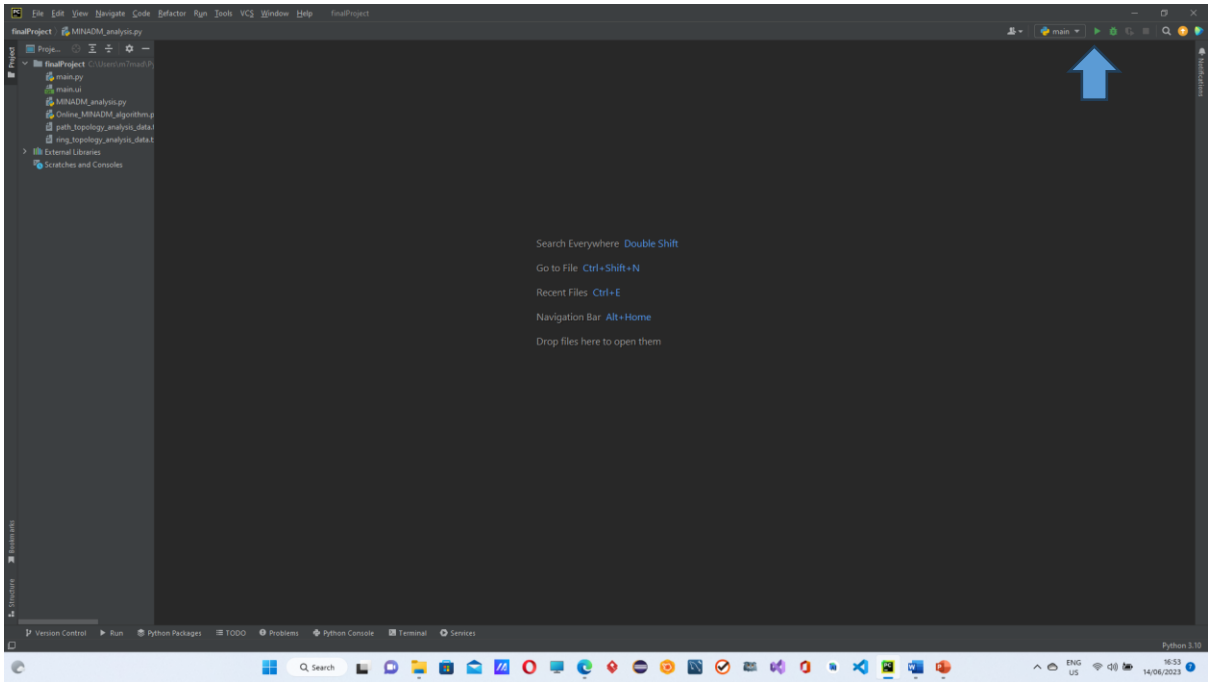
The algorithm is implemented in Python using PyCharm IDE version 2022.3.3. The software utilizes the "networkx" library for drawing networks and "matplotlib" for displaying these visualizations. To create the user interface, the "pyqt5" library is employed.

The software primarily focuses on path and ring topologies. It allows users to visualize the implemented algorithm on networks of their choice and provides a side-by-side comparison of the algorithm's coloring and the optimal solution for a specific network.

Furthermore, the software conducts an analysis of the algorithm's performance. It achieves this by executing the algorithm on various networks and inputs. Users can contribute to the statistics by adding their own data, and the software recalculates the average accordingly. The performance results for each network are displayed on a 3D graph, providing a comprehensive visualization of the algorithm's performance across different scenarios.

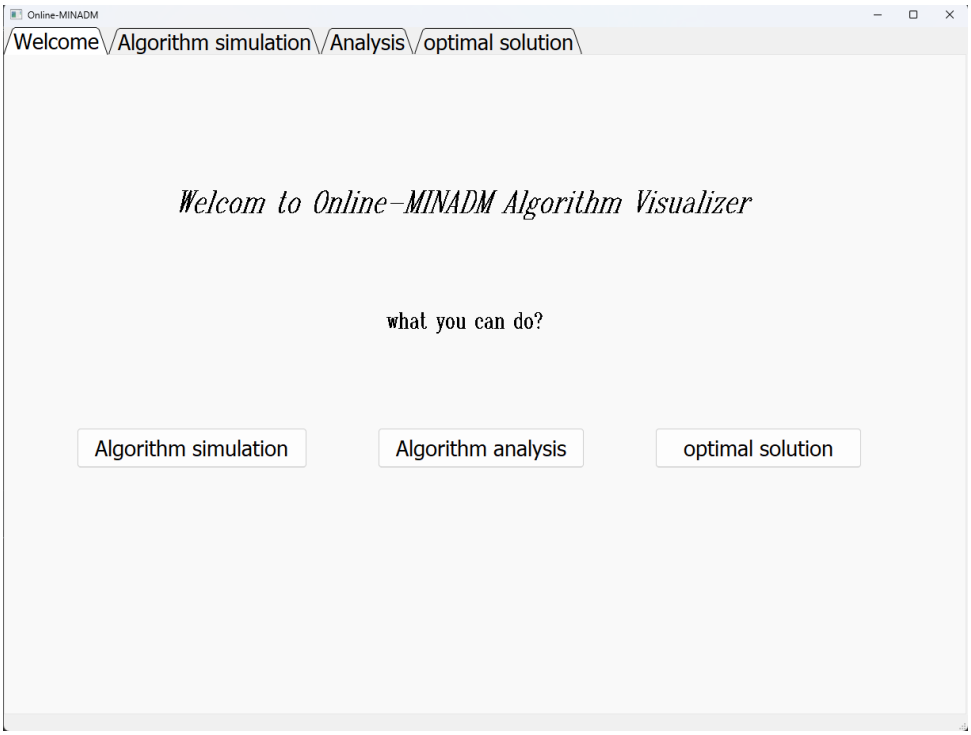
2.2 Operating Instructions:

Executing the software is a straightforward process requiring an IDE to open the code and a simple action of clicking on the "run" icon. Please refer to the screenshot below for visual reference.



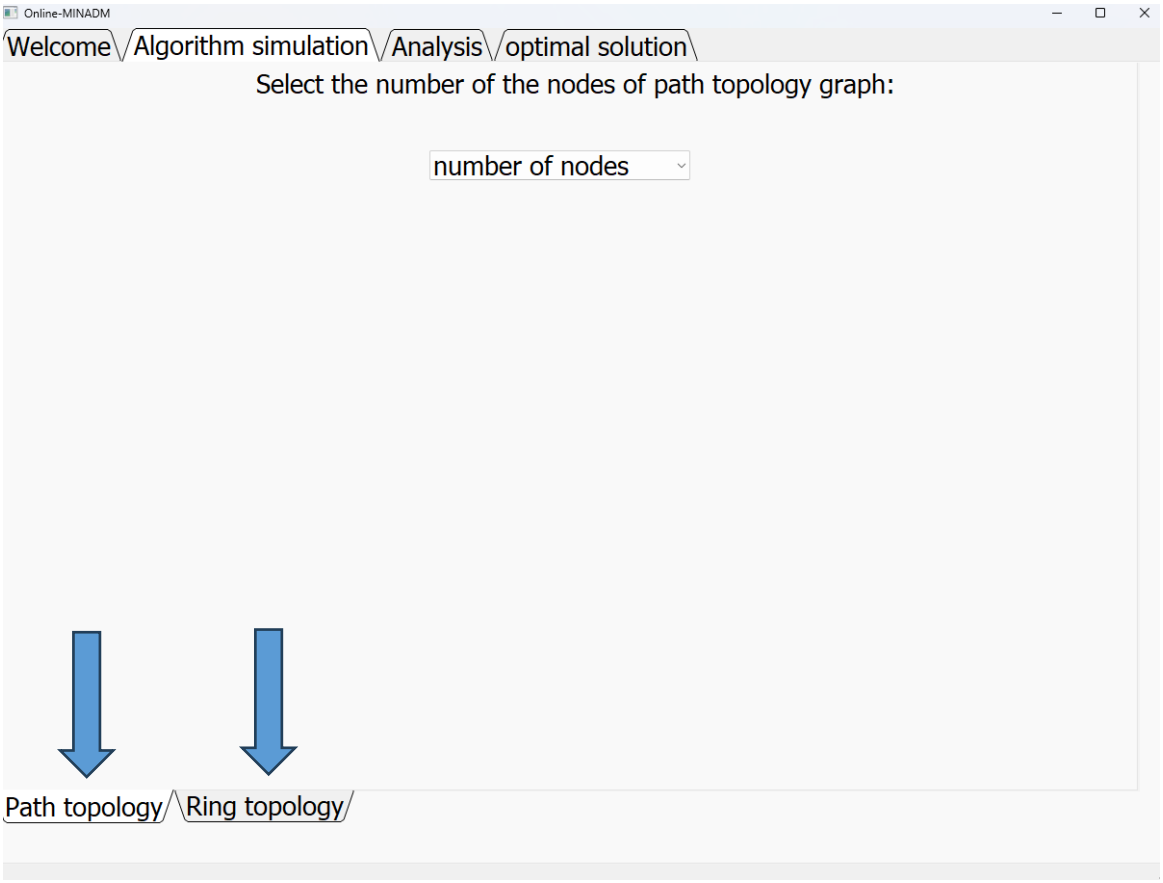
By following this procedure, the software will be launched and ready to use.

This will open the main welcome page of the software

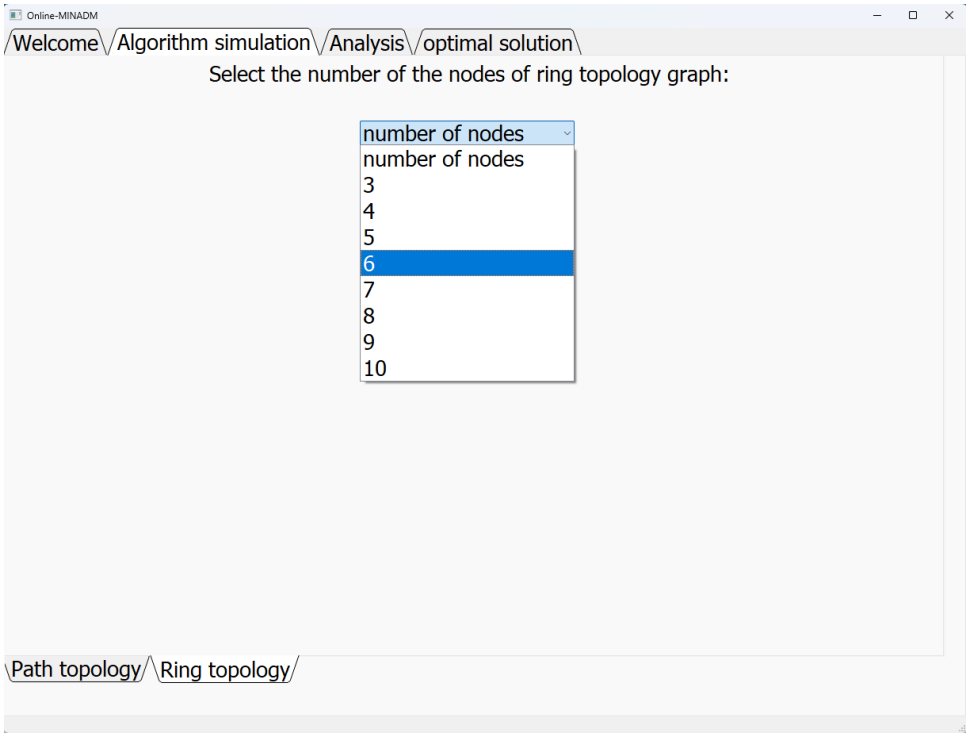


On the welcome page, the user can do choose to open algorithm simolution by pressing the button “Algorithm simulation” or the tab “Algorithm simulation” on the top to start visualizing the algorithm on certain topology. To look at the results or see comparison between the optimal and the algorithms solution, the user can click on the buttons ”Algorithm analysis” or “optimal solution” or by clicking on analysis and optimal solution tabs.

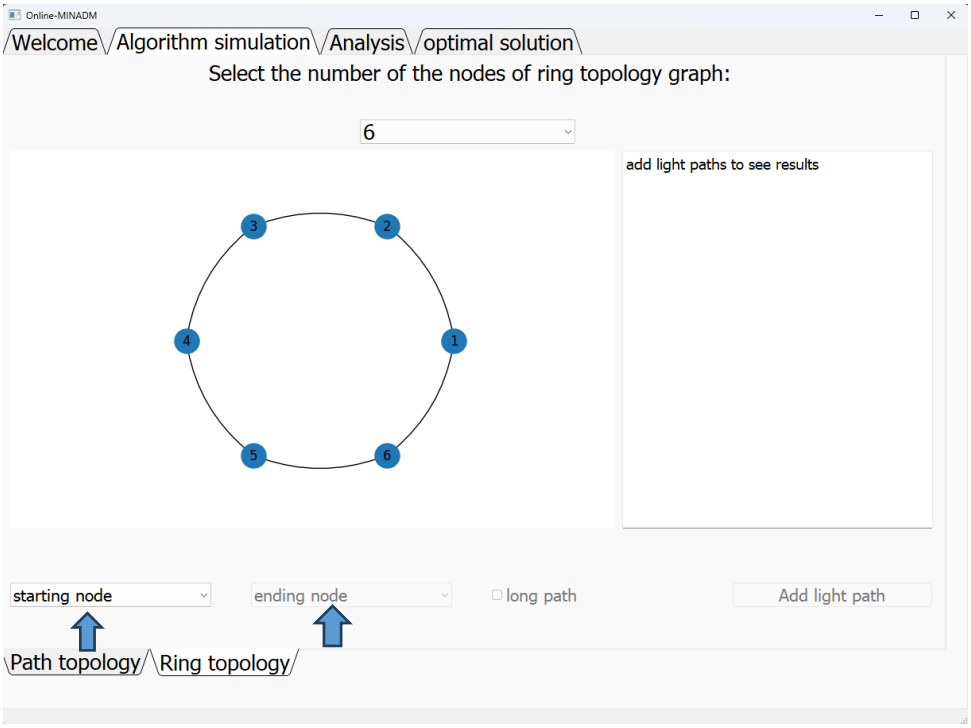
Pressing on “Algorithm simulation” button will open a tab that enables the user to start visualizing the algorithm in ring topology or path topology graph, the user can choose the topology by selecting the appropriate tab.



the user can choose the number of nodes by clicking on the combo box and select a number 3-10

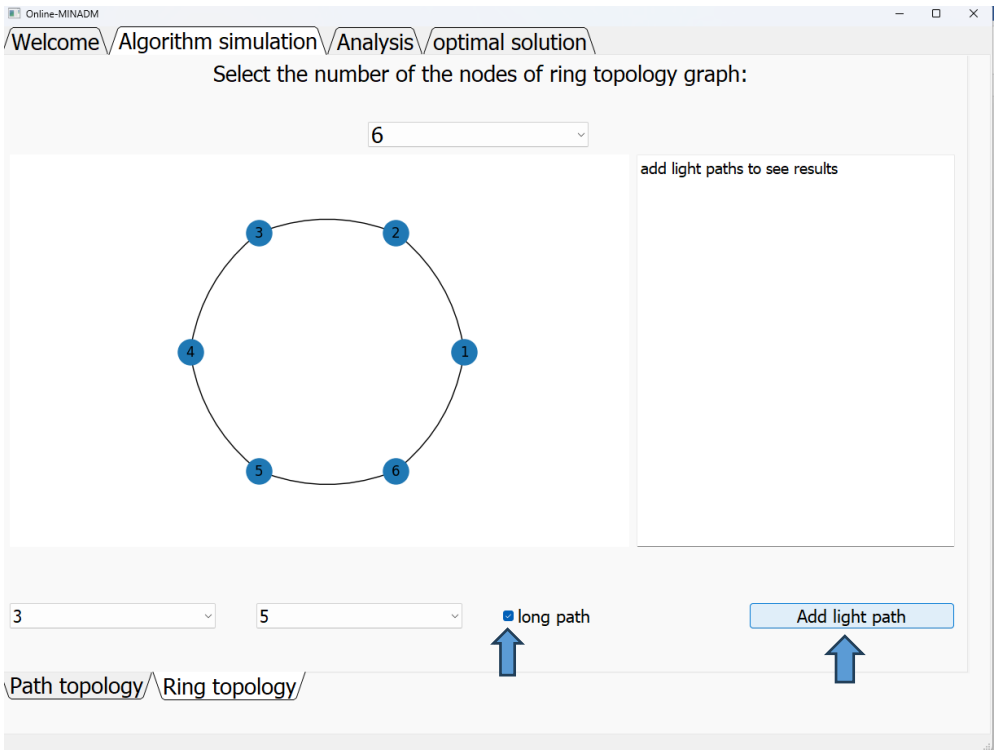


For example, if the user chooses 6, the system will generate new cycle graph with 6 nodes and then the user can start adding light paths

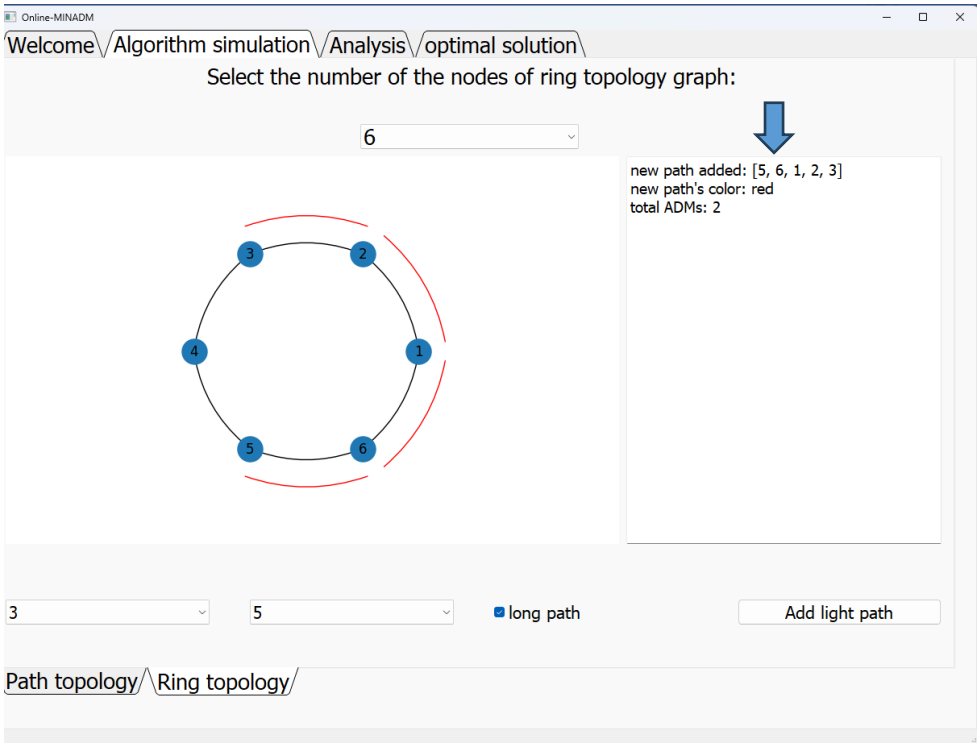


To add new light light path, the user need select a starting node first and then selecting ending node, this fields is mandatory. in addition, the user can choose which path he wants, if he wants the long path he needs to check the check box as shown in the picture bellow, or leave it unchecked if he wants the short path, at the end he needs to cilck on “Add light path” button. For example if the user chooses the starting node and the ending node to be 3 and 5

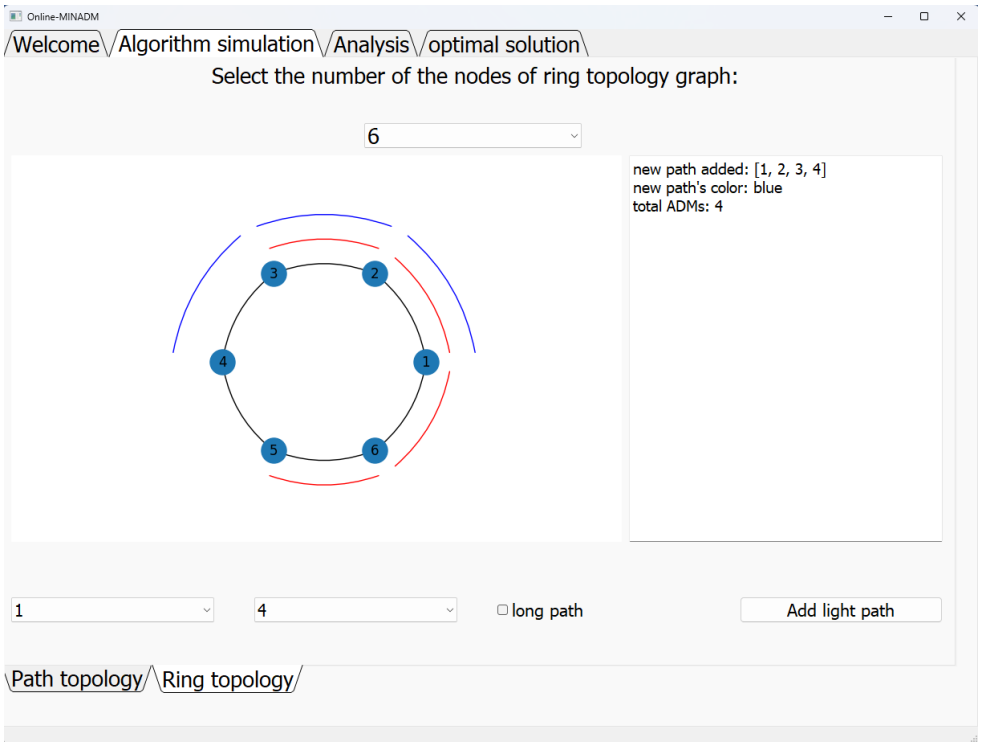
like the picture bellow and checked the check box and clicked on the button, the program will add new ligth path to the graph that starts from 3, ends in 5 and goes through 2,1,6.



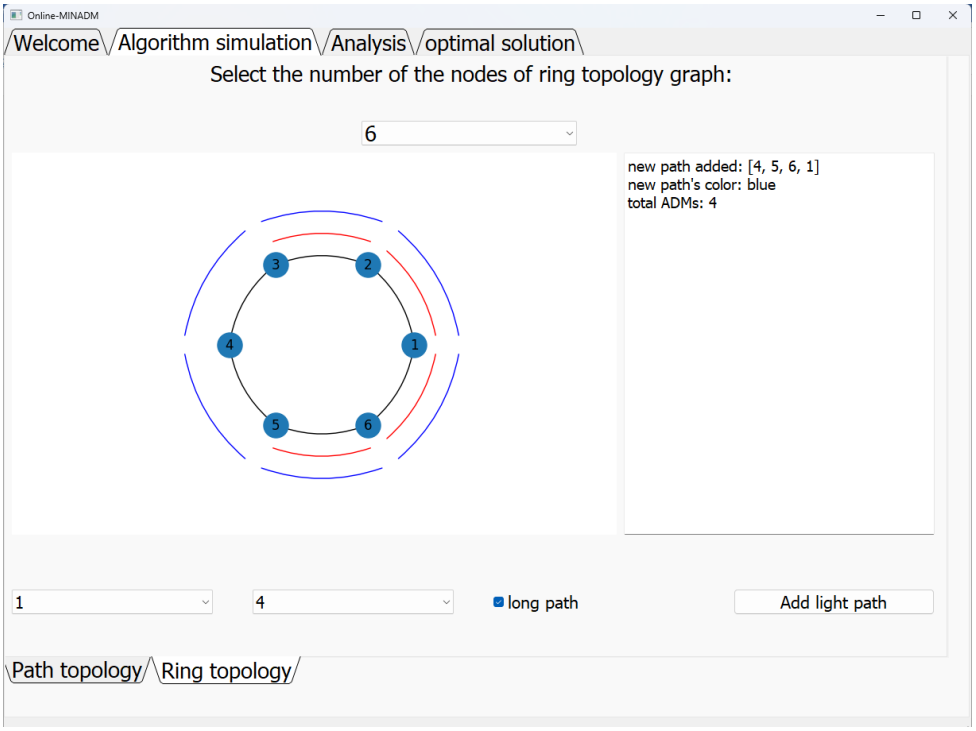
After clicking on “Add light path” button, the program will take the data from the input fields and generate a path based on the data and run the “Online MINADM” algorithm on it, and draw it on the with the proper color. Moreover, on the right side the user can see the results of the algorithm.



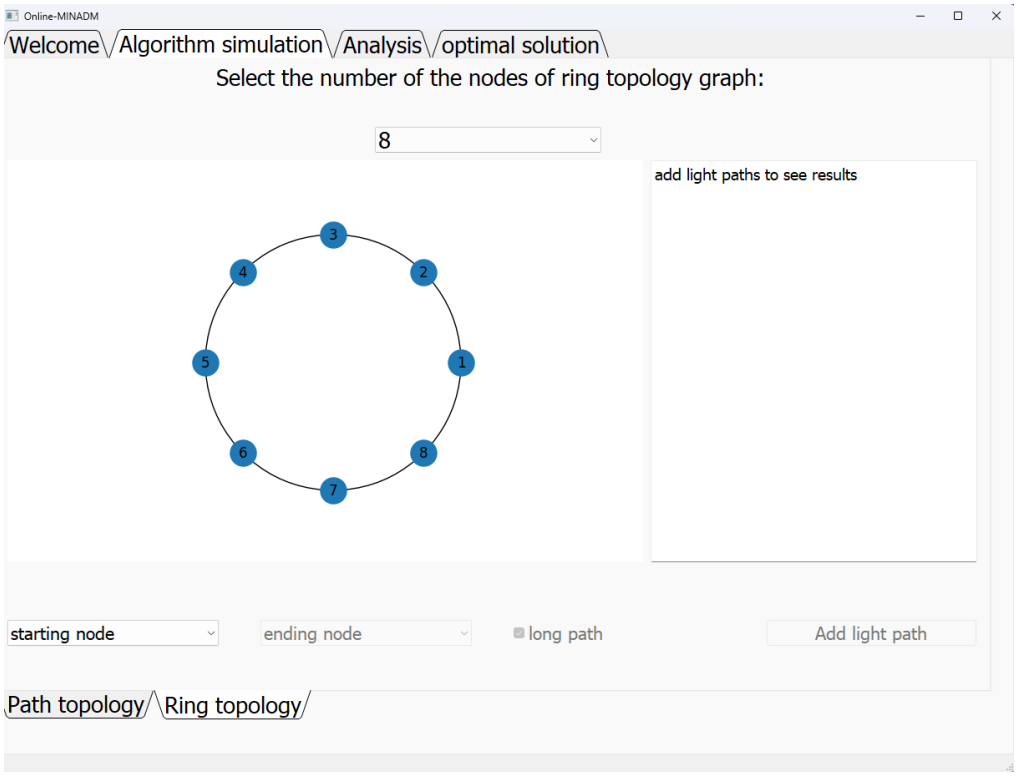
By rpeating the same process, the user can add as many light paths as he wants.



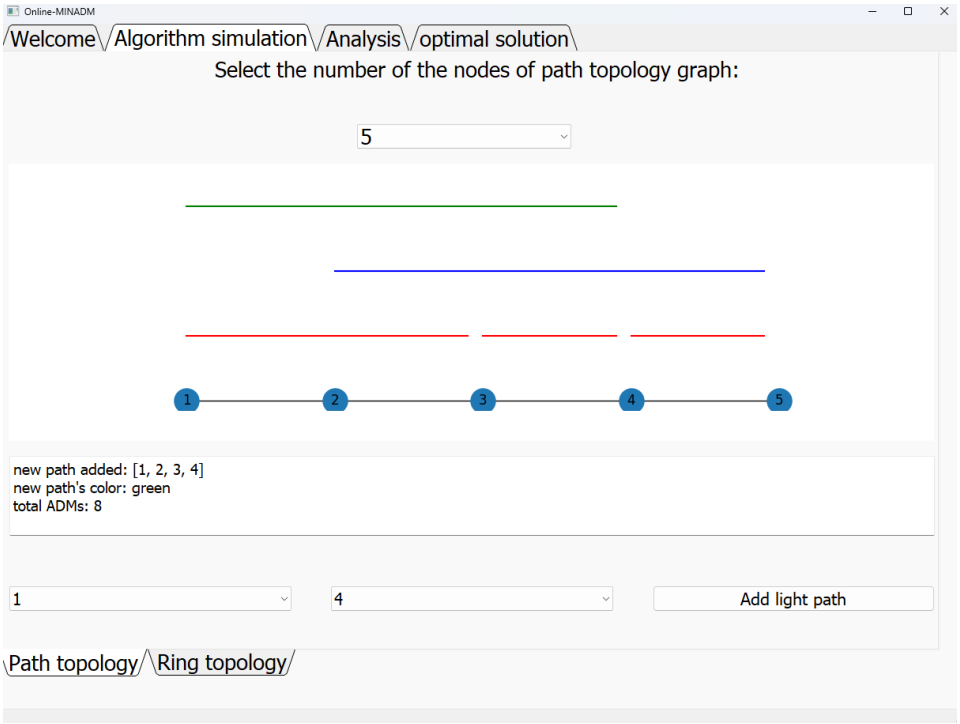
If the user adds new light path that there is no proper color for it, the program will add it with new color as we can see in the picture above



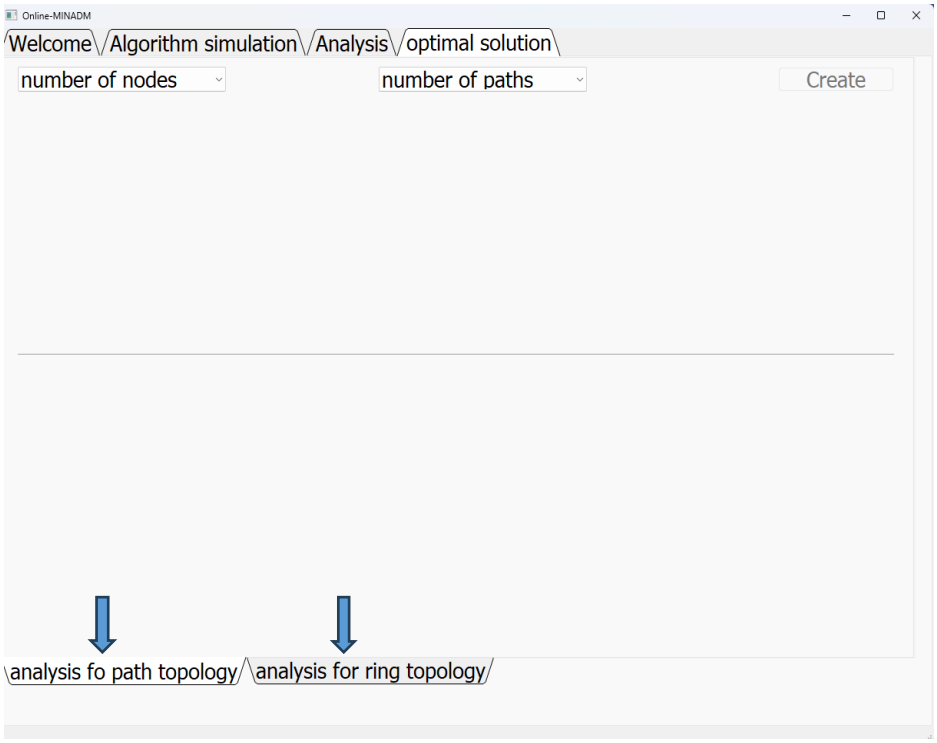
The user can start new experiment by rechoose the number of node from the combo box and the program will automatically clear the graph and draw new clear one.



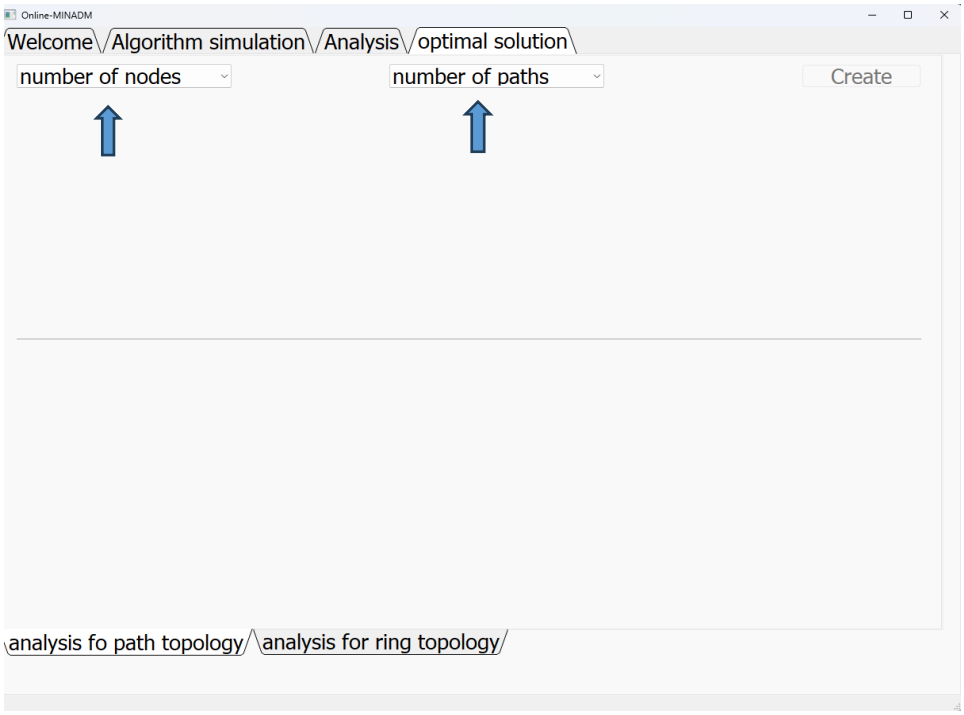
The user can do the same thing on path topology graphs by entering the “path topology” tab.



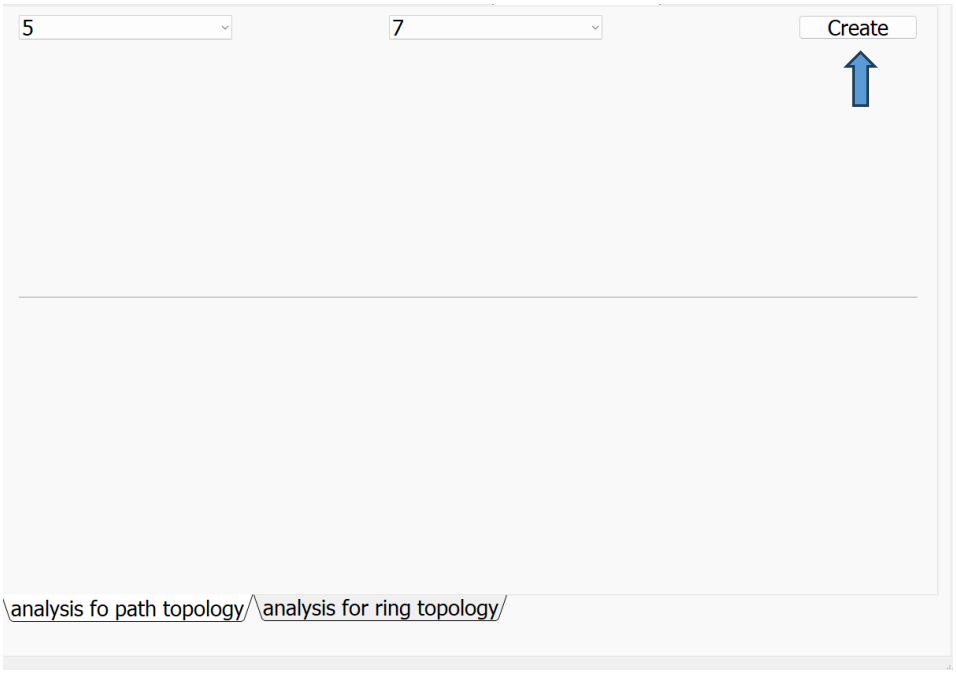
The user can enter to the “optimal solution” tab to do comparison between optimal solution and online solution. Inside this there is to tabs one for ring topology and the another for path topology.



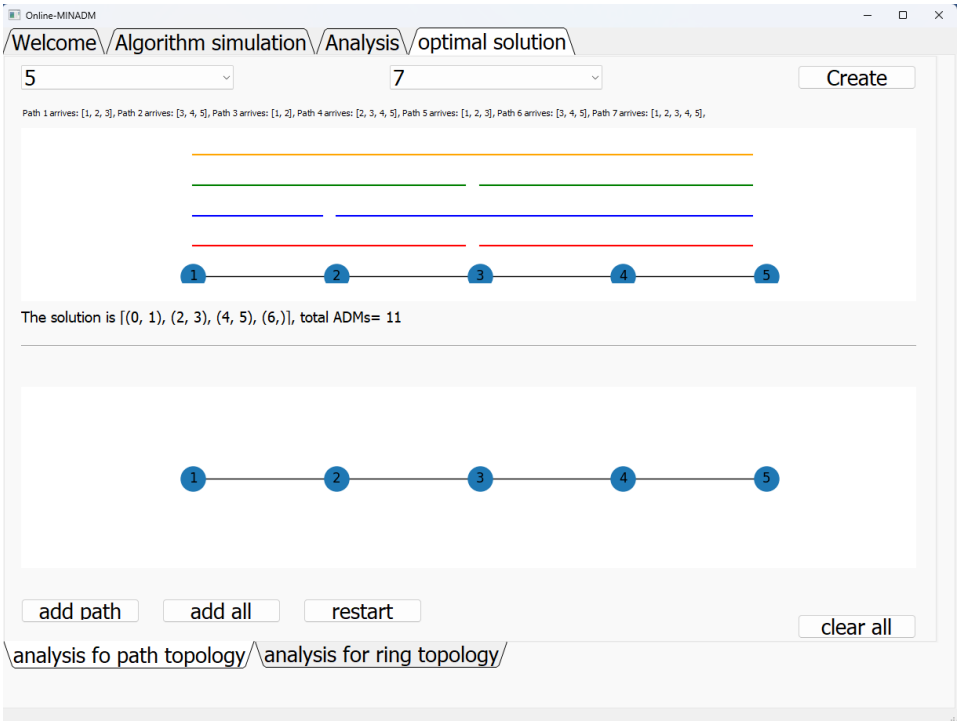
To begin, the user needs to choose the number of nodes and the number of paths



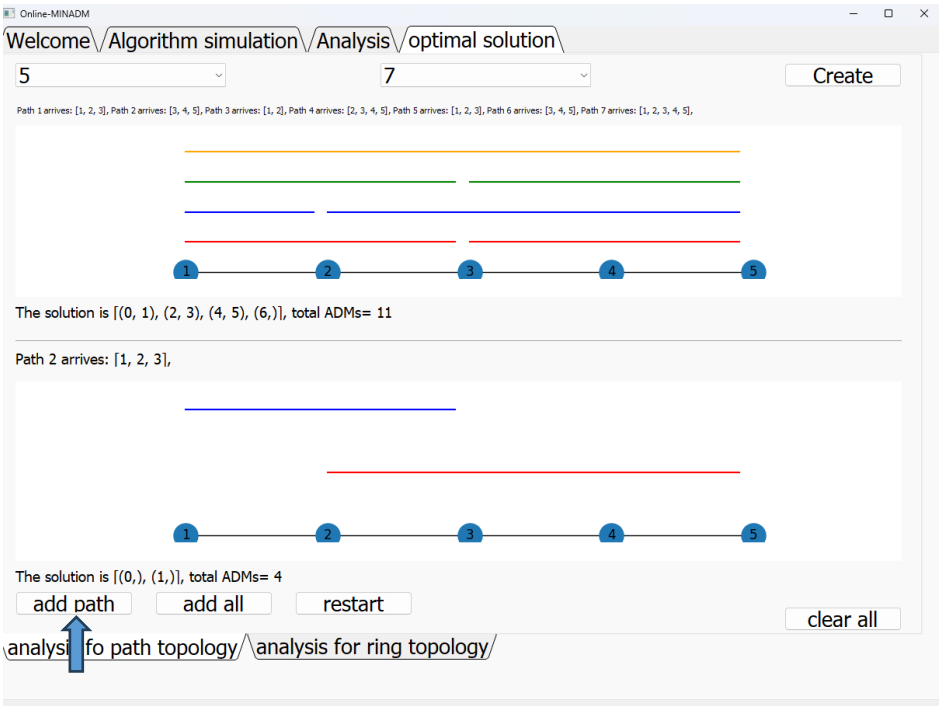
After that the program will enables the “Create” button, so the user can press it to generate new graphs.



In this case the program will generate and draw 2 path graphs with five nodes, the first one for the optimal solution and the second one for online solution. The optimal graph will contain 7 light paths that colored in the optimal way, as shown in the picture bellow.

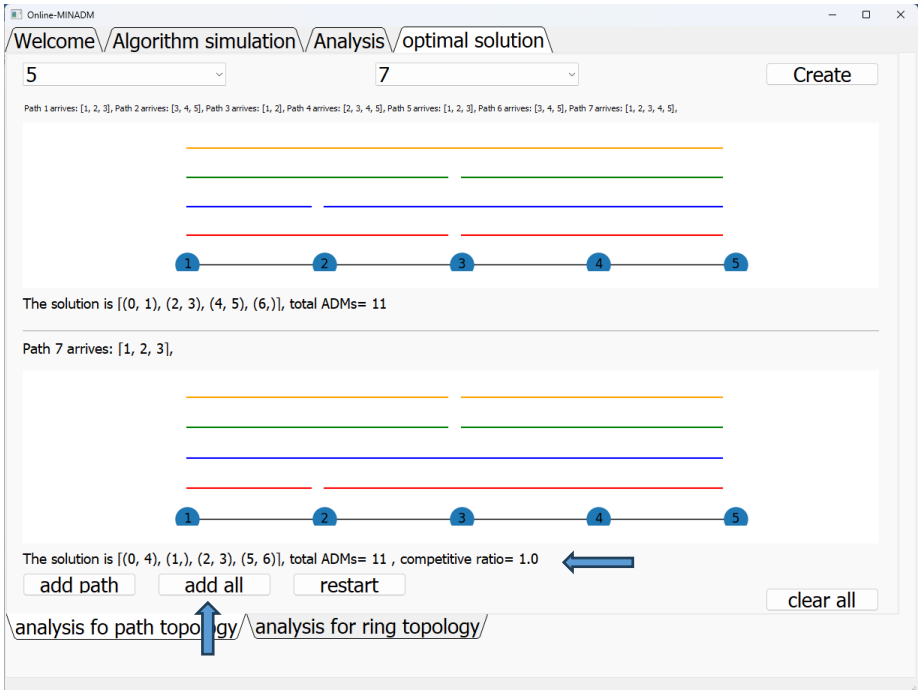


The user can add paths one by pressing on “add path” button, so the program will take a random path and add it to the graph.

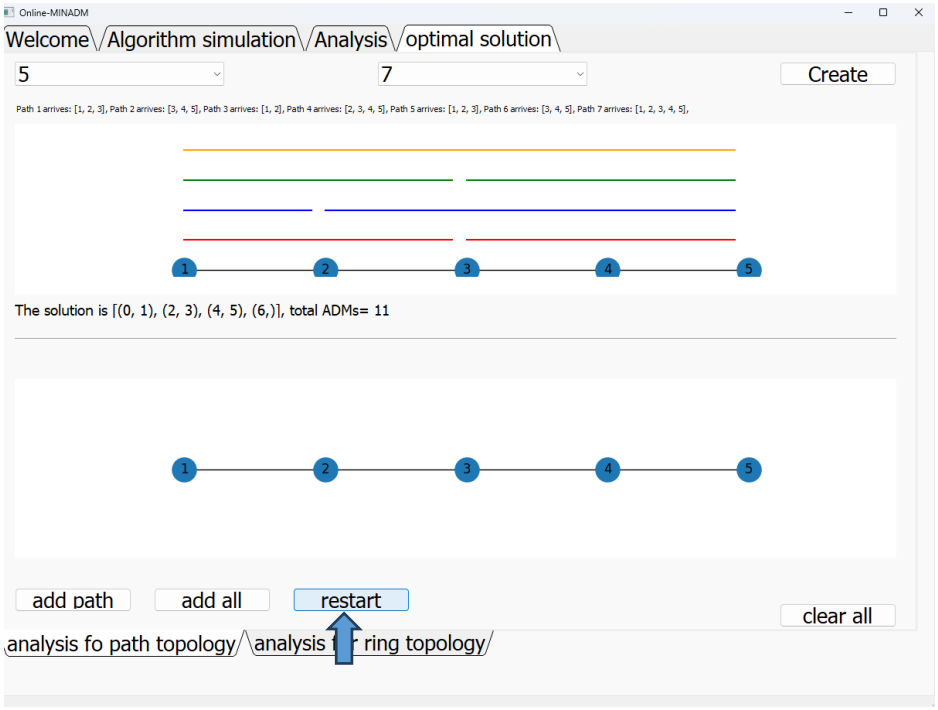


Also, the user can add all the paths in one click, by pressing on “add all” button.

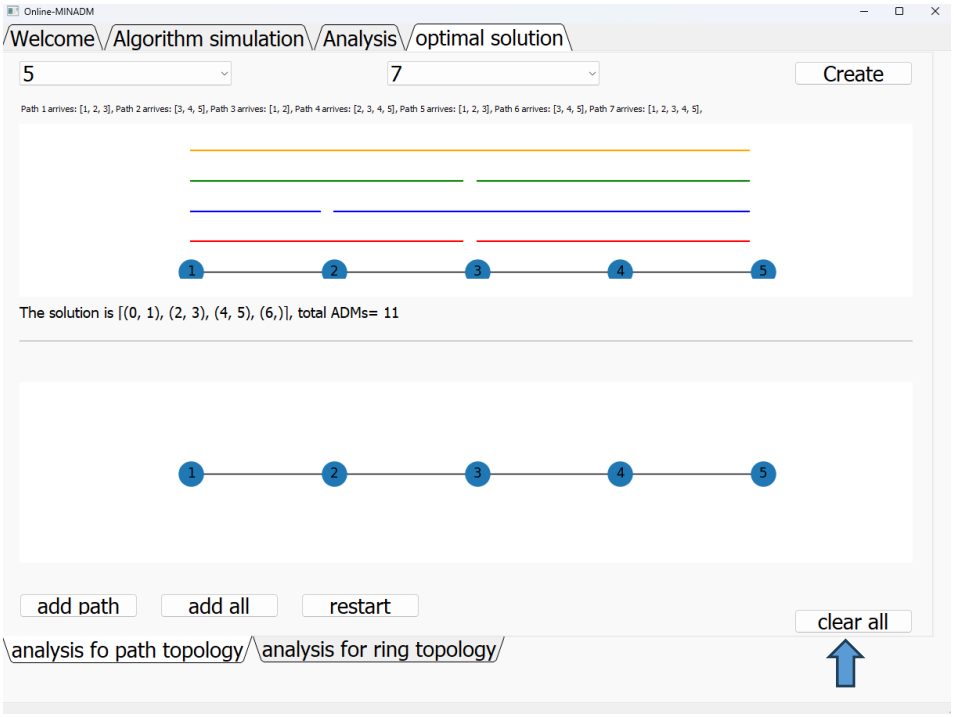
At the end, the program will calculate the competitive ratio and display it.



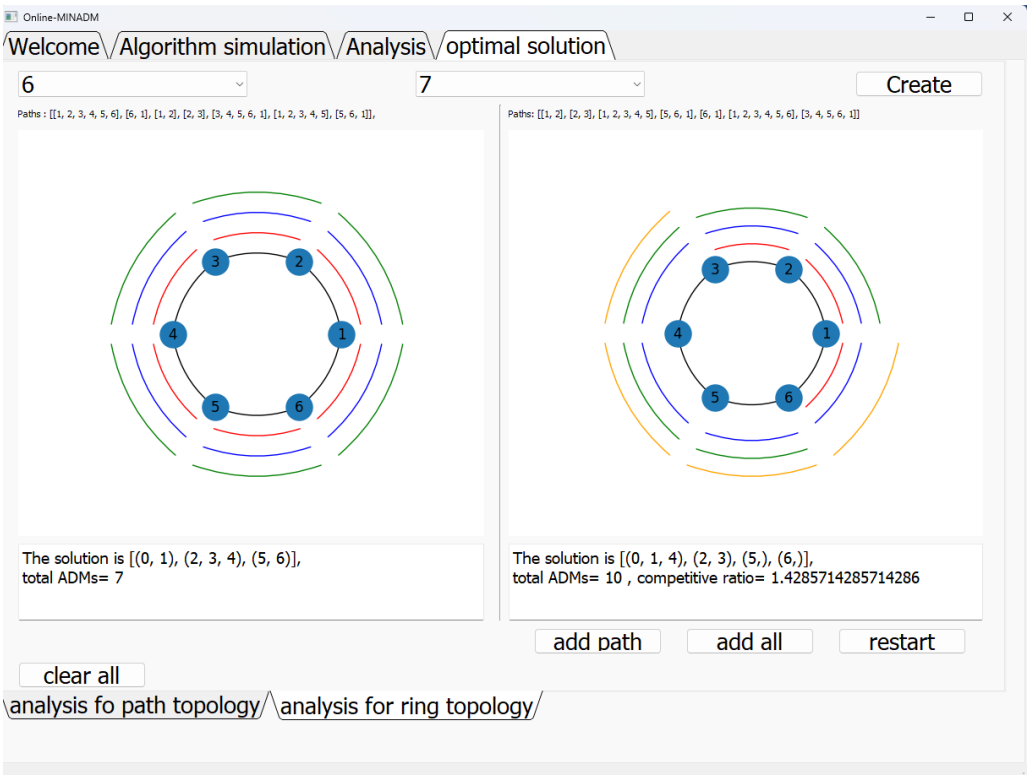
The user can reset the online graph by pressing on “restart button” and start again.



To start new experiment, the user can press on “clear all” button and repeat the same process or just recreate another graphs by pressing on “create” button and the program will clear all first automatically.



The user can do the same thing in ring topology by enter to “analysis for ring topology” tab.



In the "Analysis" tab, the software presents a 3D graph that showcases the relationship between the number of nodes (x-axis), the number of paths (y-axis), and the ratio between

the average performance and the optimal solution (z-axis). The points in the graph are color-coded into three different shades of red based on the ratio values.

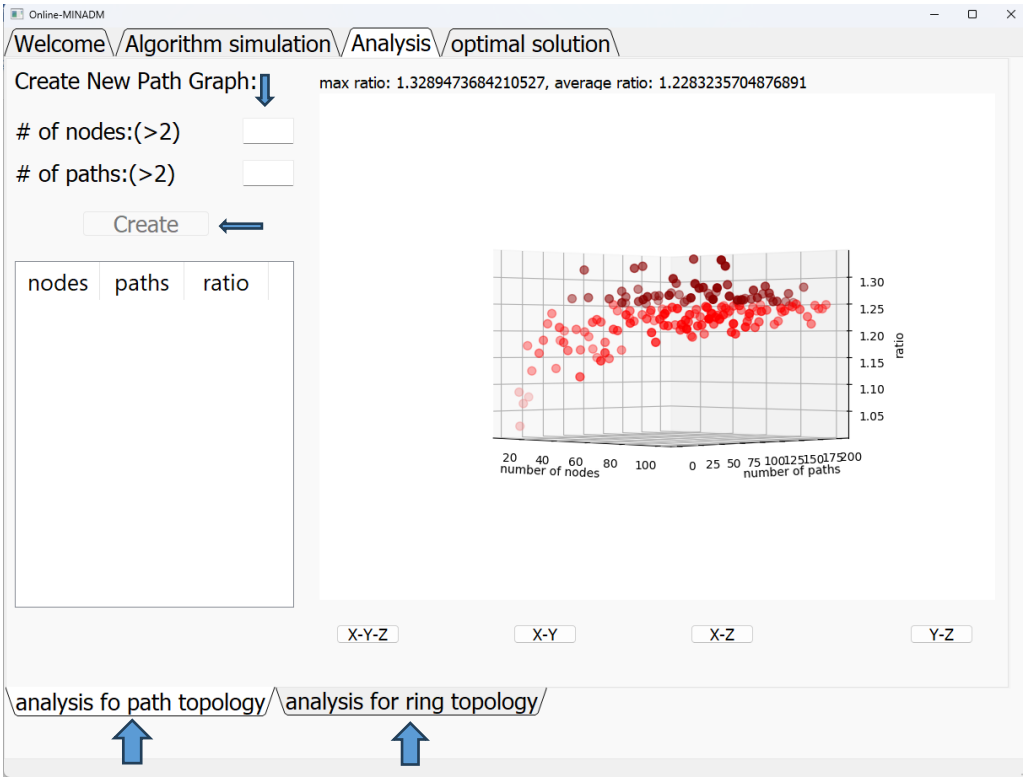
The color distinction provides visual differentiation for understanding the ratio variations. Typically, lighter shades of red may represent lower ratio values, while darker shades of red may indicate higher ratio values. This color-based representation allows users to identify and interpret the performance characteristics of the algorithm across different combinations of node and path numbers within the given network topology.

In the "Path Topology" analysis, points on the 3D graph are color-coded based on the ratio values. Points with a ratio (z) between 1 and 1.1 are represented by a light red color. Points with a ratio between 1.1 and 1.25 are depicted as bright red, while points with a ratio greater than 1.25 are shown as dark red. It is important to note that these color shades and ranges were chosen arbitrarily to emphasize the varying levels of ratios in the results.

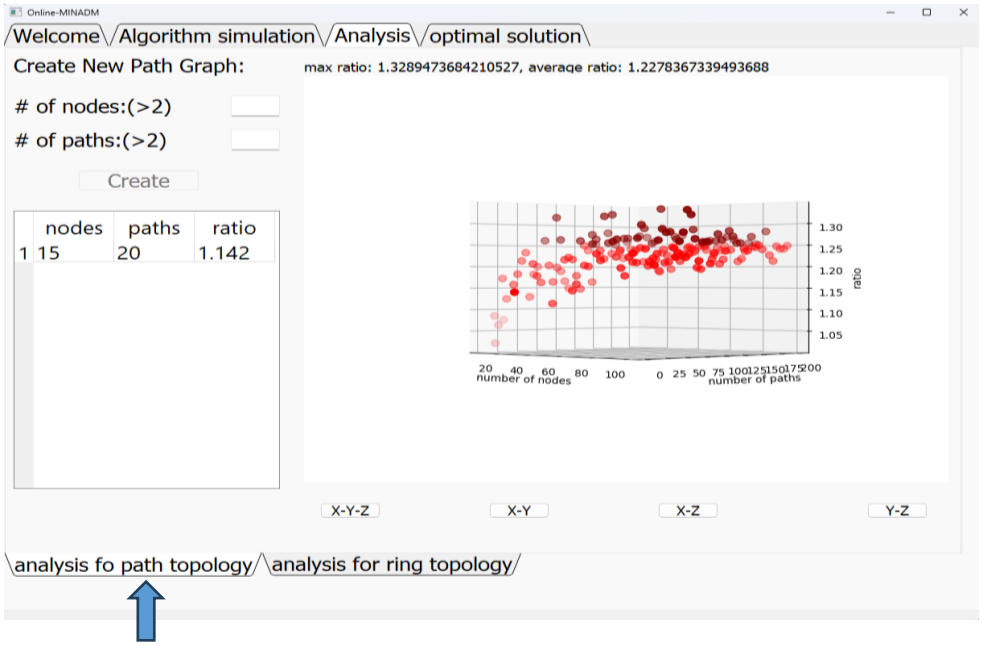
Similarly, in the "Ring Topology" analysis, the points are color-coded based on the ratio values. Points with a ratio between 1 and 1.1 are represented by a light red color. Points with a ratio between 1.1 and 1.3 are depicted as bright red, and points with a ratio greater than 1.3 are shown as dark red.

These results were obtained by running the algorithm on numerous networks of various sizes and calculating the average result. Each point (x, y, z) on the graph represents a network with x nodes, y paths arriving sequentially, and an average ratio of z. The average is calculated by running the algorithm on networks with x nodes while changing the order in which the paths arrive.

The 3D graph can be interactively explored by moving it around using the mouse. Additionally, specific angles of the graph can be viewed conveniently by using the provided buttons.

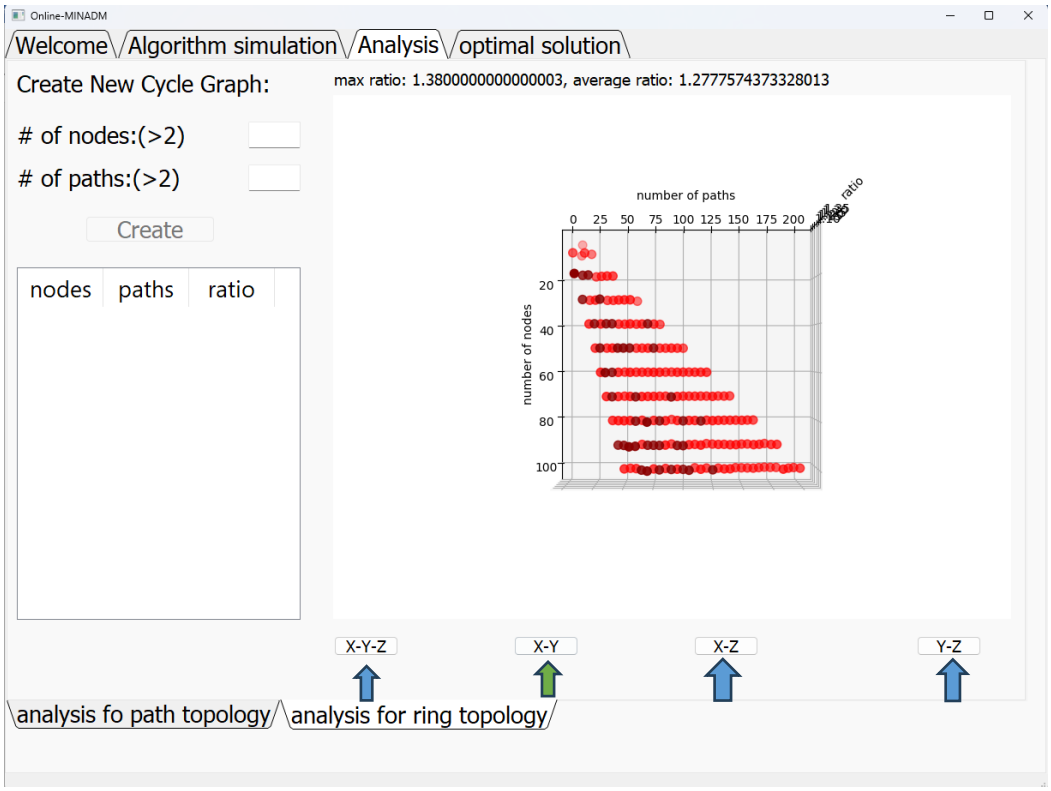


The user can choose between path topology and ring topology by clicking on the desired tab and see the results or add new point to the graph, to add new point to the graph the user needs fill input field (# of nodes, # of paths) and press on “create” button.



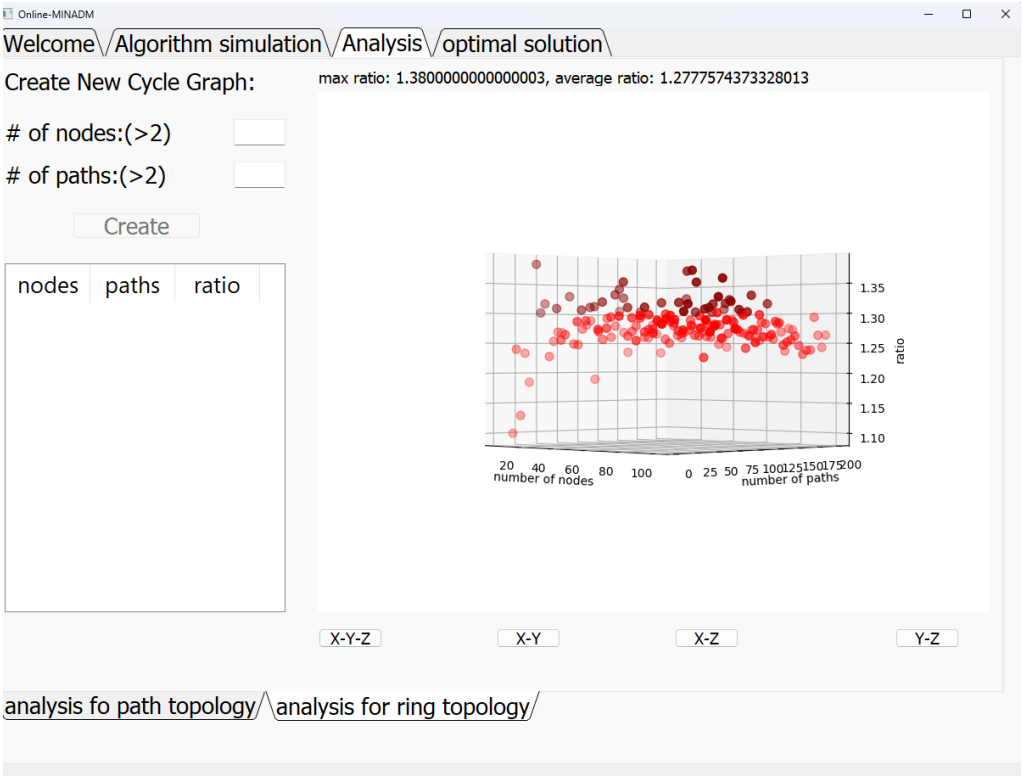
After pressing on the “create” button, the program will read the input fields and generate new graph with the specified number of nodes, create list of paths in length of the specified number of paths, the paths are arranged in the optimal so the algorithm can calculate the optimal solution.

After calculating the optimal solution the program will do 10 random runs then calculate the average and add the data to table and to the graph.



By clicking on the buttons (“X-Y-Z”, “X-Y, “X-Z”, “Y-Z”), the user can see different angels of the graph

The user can do the same for ring topology by click on the tab “analysis for ring topolog”



3. References

[1] *Shalom, M., Wong, P. W, & Zaks,* S. *(2007, September). Optimal on-line colorings for minimizing the number of ADMs in optical networks. In International Symposium on Distributed Computing (pp. 435-449). Springer, Berlin, Heidelberg.*

[2] Borodin, A., & El-Yaniv, R. (2005). *Online computation and competitive analysis*. Cambridge university press.

[3] Gollapudi, S., & Panigrahi, D. (2019, May). Online algorithms for rent-or-buy with expert advice. In *International Conference on Machine Learning* (pp. 2319-2327). PMLR.

[4] Technical Papers and Presentations: Dense Wavelength-Division Multiplexing (DWDM) (topicstash.blogspot.com)

[5] Part 8: WDM SYSTEM. - ppt video online download (slideplayer.com)