A GRAPHICAL USER INTERFACE TO SIMULATE COLOUR ASSIGNMENT IN GRAPH COLOURING PROBLEMS

Mas Izuren Hishamuddin[1], **Syazwina Azwin Ahmad Sabri**[2] **and Nur Syazwani Ghazali**[3*]

Pusat Pengajian Sains Matematik, Kolej Pengajian Pengkomputeran, Informatik Dan Media, Universiti Teknologi MARA (UiTM), 40450 Shah Alam, Selangor, Malaysia

[1]2021125467@student.uitm.edu.my, [2]2021102575@student.uitm.edu.my,

[3*]syazwanighazali@uitm.edu.my

(indicates the corresponding author)*

ABSTRACT

Graph Theory is one of mathematics fields that is concerned with networks of points connected by lines. This research project aims to develop Graphical User Interface (GUI) and Graph colouring is an effective technique used in Graph Theory in solving a variety of practical and theoretical problems for simulating colour assignment in graph colouring problem. By developing a GUI for the graph colouring method, we wish to provide a base programme to visualise the colour assignment problems. The objectives of this study are to design and develop a graphical user interface that is able to simulate network graph topology and to apply and demonstrate colour assignment for graph in the developed graphical user interface. In order to achieve the objectives of this study, Java software is used as a tool to develop the GUI. As a result, a GUI is successfully developed, and it is hoped to serve as a user-friendly GUI for building a graph topology and architecture for graph colouring problems and its applications.

Keywords*: Graph Colouring, Graphical User Interface (GUI), Colour Assignment, Java software*

Received for review: 15-03-2023; Accepted: 09-04-2023; Published: 14-04-2023

1. Introduction

Graphical User Interface (GUI) is a computer program that allows interaction between users and computers or electronic devices through a visual indicator representation such as symbols and pointing devices. It is one of the most significant application components for business and safety-oriented software systems and makes use of human perceptual and motor abilities to perform basic tasks like command exploration and invocation, information seeking, and multitasking. In the classroom, GUIs are increasingly utilised to give simulation users a comfortable and visible way to specify all input parameters, making it simpler to explain what is required to execute a simulation. The user interfaces need to be modified to satisfy various design requirements and user interactions in connection with the services they provide. Research done by Lin et al. (2018) proposed a design of user interface (UI) operations and appropriate performance indices to evaluate the smoothness of smartphones. The main consideration for this GUI is to provide appropriate and understandable user interfaces to avoid errors that could have a direct impact on human life.

Graph colouring is a useful technique for solving a variety of practical and theoretical problems, such as timetabling, scheduling, job allocation, frequency allocation and team building. Shukla et al. (2021) investigated the Genetic Algorithm approach to graph colouring

as it relates to the timetable problem. There are a few graph colouring methods that can be applied in many different areas of science and technology, such as Group Special Mobile (GSM) where a vertex colouring algorithm can be used to assign four different frequencies. Additionally, there are schemes that have been developed to overcome the limited energy and resources of Wireless Sensor Networks (WSN). The use of graph colouring method in minimizing the time slots can improve the performance of a Wireless Sensor Network (WSN).

It is important to consider the efficiency of energy when designing a WSN due to the energy constraint posed by sensor nodes. In integrated circuit manufacturing, the graph colouring problem can be used to construct a graph G whose node set is the set of components, and two components are adjacent if they are separated by the lithography distance. Elhag & Özcan (2015) solved grouping problems using a novel generic selection hyper-heuristic framework and an unconventional move acceptance mechanism. De (2022) solved resource scheduling in cloud systems with multiple processes and limited resources using graph colouring techniques. Alfouzan et al. (2018) proposed a new protocol GC-MAC to produce a conflict-free transmission over the network by assigning time slots to each individual colour in the network. The goal is to reduce energy consumption by assigning each colour to a unique time slot.

This research paper will specifically design GUI to simulate colour assignment in graph colouring problem by using Java Programming Language. We wish to provide a base program to visualize the colour assignment problems and to study the graph colouring process and output. The main goal of developing GUI in this research is to provide a convenient method for creating input and evaluating the result of the process of colouring the graph. This GUI is developed as a base to solve other graph colouring problem applications. The contribution of this application is a user-friendly GUI for building a graph topology and architecture for graph colouring problems and their applications.

The objectives of this research study are to design and develop a graphical user interface that is able to simulate network graph topology and to apply a graph colouring method in the developed graphical user interface.

2. Methodology

In this section, the research methodology used in this research will be distinguished. The design of the proposed application is studied, and the procedures along with the process are used in forming the application and data collection together with the methods on how to analyze the collected data. The primary research method for this research is to simulate colour assignment in graph colouring problems. The methodology of designing a GUI for graph colouring problems by using Java software is discussed in this section.

2.1 Designing the layout for GUI

Figure 1 shows the first layout design for the GUI. The layout design for the first interface requires the user to input the number of nodes, the number of range and the number of channels. The number of nodes, range and channel are based on user's preference on what they wanted to see. The maximum number of nodes that a user can fill in is 800 nodes. There are empty fields for the user to input the number and a button enter for the user to proceed to the next interface which is a graph simulator.

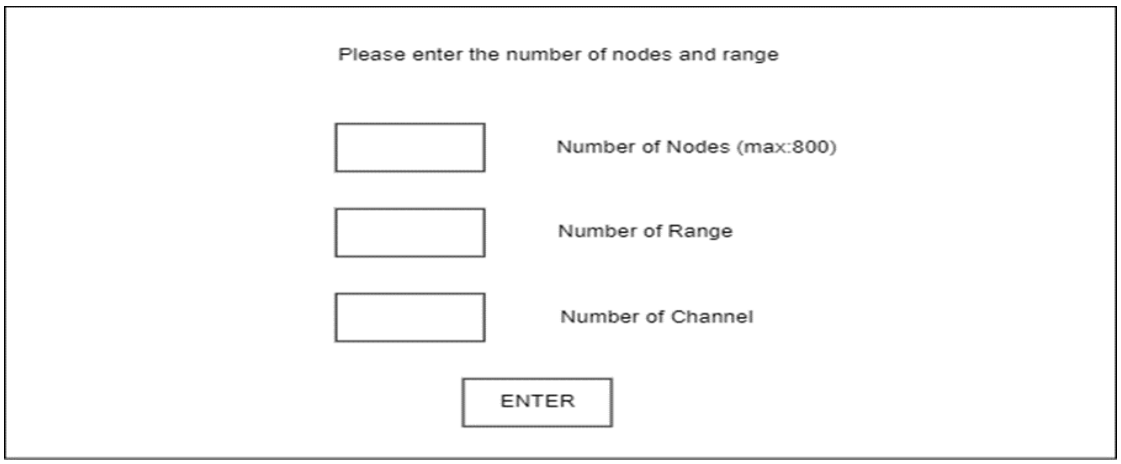

Figure 1. The first layout for GUI

Figure 2 shows the layout for the second interface. Then, for the next layout, there are 6 buttons at the right side of the window consisting of 'Generate Nodes' button, 'Draw Range' button, 'No. of Neighbors' button, 'Minimum Level' button, 'Assign Colour' button, and lastly 'Draw Coloured Graph' button. Each of these buttons have their own function in this interface.

Figure 2. The second layout for GUI

2.2 Developing the GUI

After designing the GUI that is shown in Figures under 2.1, GUI is programmed by using Java coding and ran using Eclipse IDE software. The interface of a GUI programme is made up of widgets that are shown in a window. As seen in Figure 1, a popup screen appeared asking users to enter the quantity of nodes, number of range and number of channels. Users must press enter to move on to the following section. After entering the data, a window with 6 buttons appeared, including the 'Generate Nodes', 'Draw Range', 'No. of Neighbors', 'Minimum Level', 'Assign Colour' and 'Draw Coloured Path'. Each button has different functions. Thus, the user must click each one of the buttons in order to see the output. The number of nodes and range entered in the first step is used to construct a random graph if the user clicks the Generate Node button. Then, the process is repeated for all buttons. The graph is redrawn in the same window with coloured vertices depending on the number of colours, and beneath each vertex is a count of its neighbors which are also using the same colour. Lastly, users can click the quit button in the menu to restart the same process with a different number of nodes and range to see a different output.

2.3 Algorithm for Creating and Colouring the Graph in the GUI

The details of the steps for colouring the graph is as shown in Figure 4, which is presented in the flowchart diagram. The purpose of this flowchart diagram is to develop a detailed understanding on the processes of how colouring the graph for each level is done.

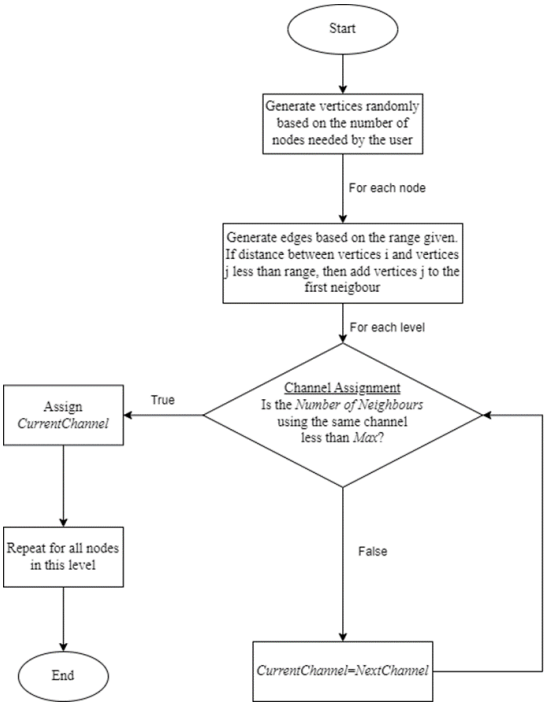

Figure 3. The flowchart of colouring the graph

Based on the flowchart involving channel assignment in colouring the graph shown in Figure 3 above, this section discussed in detail about the process of colouring the graph for each level. Firstly, the vertices shown in the graph are randomly generated based on the number of nodes inserted by the user. Next, for each node, it consists of edges connected to other nodes which are created based on the range inserted by the user. If the distance between vertices i and j is smaller than the range, vertices j is added to the first neighbor. Next, colour assignment for each level, if fewer neighbors are utilising the same channel than the maximum, the channel is designated as the *Current Channel*. The *Current Channel* must be equal to the *Next Channel* in order to continue with the Channel Assignment process until it is allocated as the *Current Channel*. For each node in each level of the graph, the same procedure is repeated.

3. Results and Discussion

In this section, all of the results of this research are presented. After completing and running the Java programming using Eclipse IDE software, the desired Graphical User Interface for graph colouring assignment is resulted as below. This GUI is named *'Polaris: Graph Colouring Simulator'*. It consists of two windows in total. The first window is for the user to input the data and the second window is to see the output or the outcome of the graph with several buttons with different functions of each button. Each of the outcomes of the result is explained in detail below.

3.1 Results

Figure 4 shows the first window which asks the user to enter the number of nodes, which is 250, into the number of nodes field and the number of range, which is 100. Five channels are allocated, starting with *Default Channel* and going all the way to Channel 4, into the number of channels field. The number of nodes, ranges, and channels that the user wants to create for their graph determines how many of each there will be. As long as it does not go over their limit number, the user is free to input as many nodes and ranges as they want. For this GUI, 800 nodes are the maximum number.

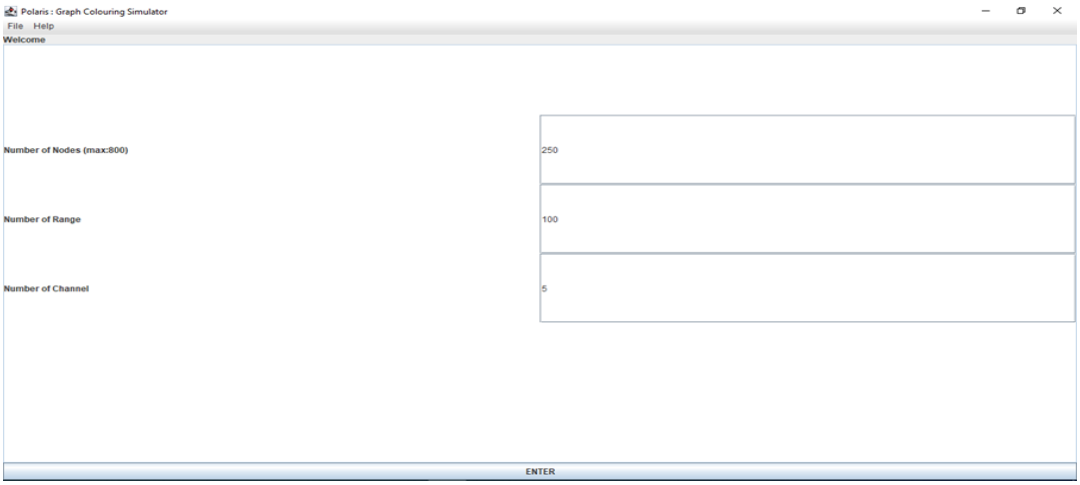

Figure 4. The first window

Figure 5 shows the second window of the developed GUI. Based on the first window shown before, the subsequent interface is shown when the user clicks enter. Six buttons are located on the right side of the window. The buttons include the following: 'Generate Nodes', 'Draw Range', 'Number of Neighbors', 'Minimum Level', 'Assign Color', and finally 'Draw Colored Path'. Each button has a distinct purpose.

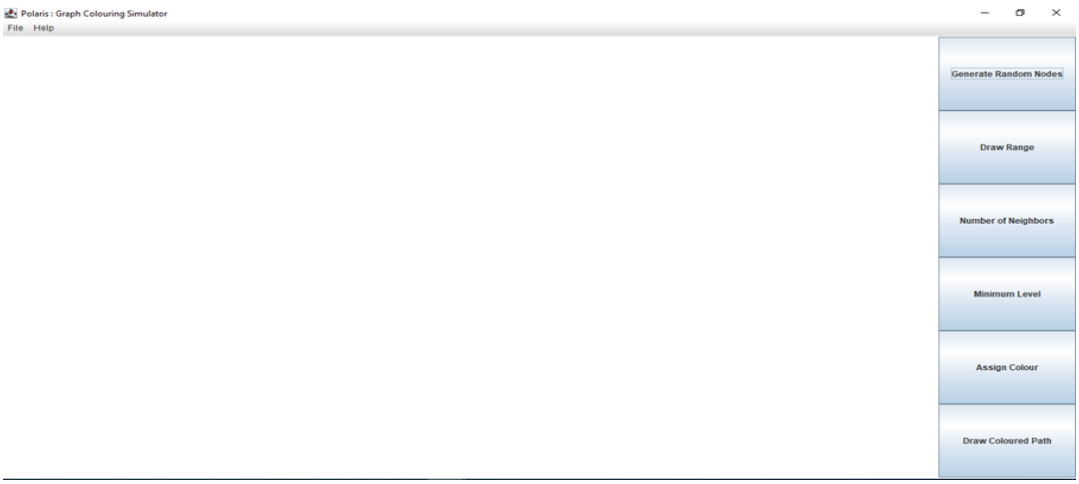

Figure 5. The second window

Figure 6 shows the result when the 'Generate Random Nodes' button is clicked. Based on the second window shown before, when the user clicks on the 'Generate Nodes' button, a window appears with a graph of 250 nodes. It is randomly generated based on the input of the number of nodes and the range. The colour assigned to each node is generated with the

default colour which is pink. The nodes generated are also represented with their own Id. For this graph containing 250 nodes, the Id for each node will start with Id=0 until Id=249.

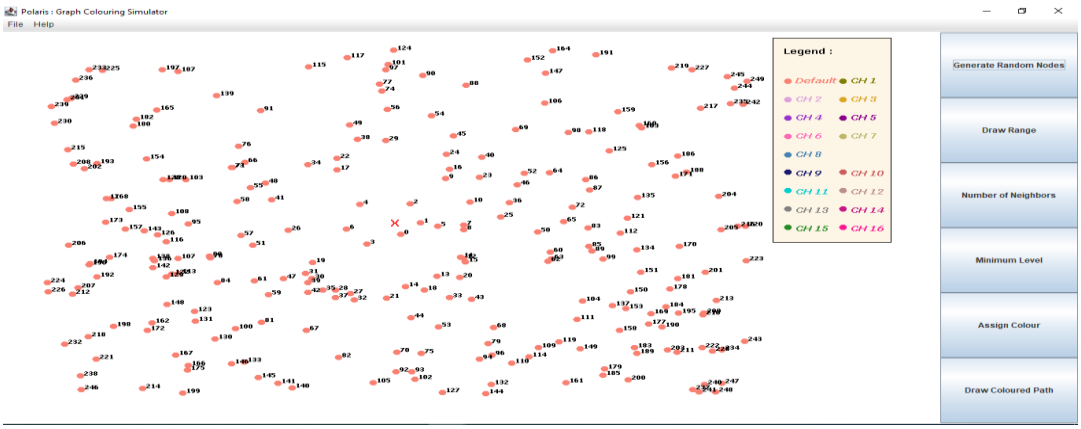

Figure 6. The result when the 'Generate Random Nodes' button is clicked

Figure 7 shows the result when the 'Draw Range' button is clicked. the window opens with an updated graph containing the edges linking each node when the user clicks the 'Draw Range' button. Depending on how many ranges the user enters, the edges are connected. Each node will link to its closest neighbor starting with X. For instance, X is linked to the following IDs: 0, 1, 2, 3, 4, 5, 6, 7, and 8.

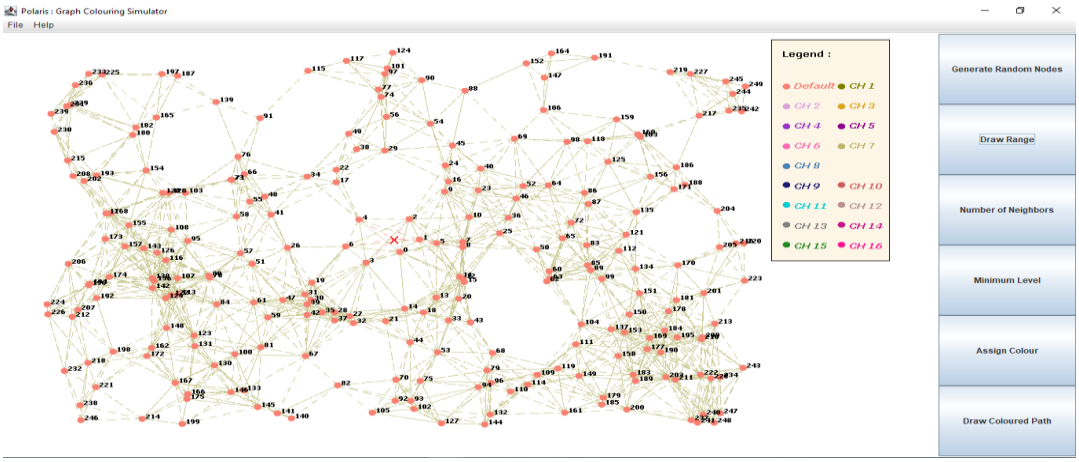

Figure 7. The result when the 'Draw Range' button is clicked

Next, Figure 8 shows the result when the 'Number of Neighbors' button is clicked. The graph is then updated with the number of neighbors for each node that is situated close to its node with the default colour if the user clicks the 'Number of Neighbors' button. Using the result above, the number of neighbors reflects the relationship between each node.

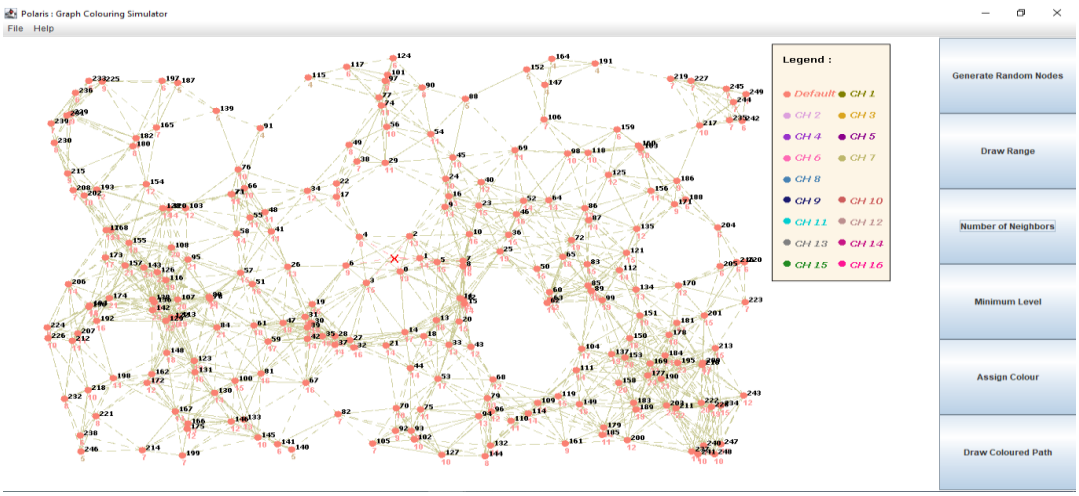

Figure 8. The result when the 'Number of Neighbors' button is clicked

As can be seen in Figure 9, the close up view of Number of Neigbours below, Id=0 has thirteen connections between its neighbors. Hence, the number of neighbors for Id=0 is thirteen.

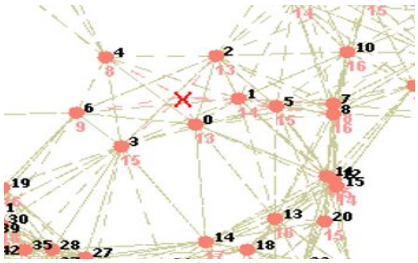

Figure 9. The close up view of Number of Neighbors

Table 1 below displays result from Id=0 until Id=8 as an example.

Table 1. Result for Number of Neighbors

No. of Id	**Number of Neighbors**
Id=0	13
Id=1	14
Id=2	13
Id=3	15
Id=4	8
Id=5	15
Id=6	9

Id=7	18
Id=8	16

Figure 10 shows the result when the 'Minimum Level' button is clicked. The interface below demonstrates how the Minimum Level button functions. The graph is updated when the user clicks the 'Minimum Level' button, adding a new number at the graph's node that is coloured green which represents the smallest level corresponds to each node's number of hops.

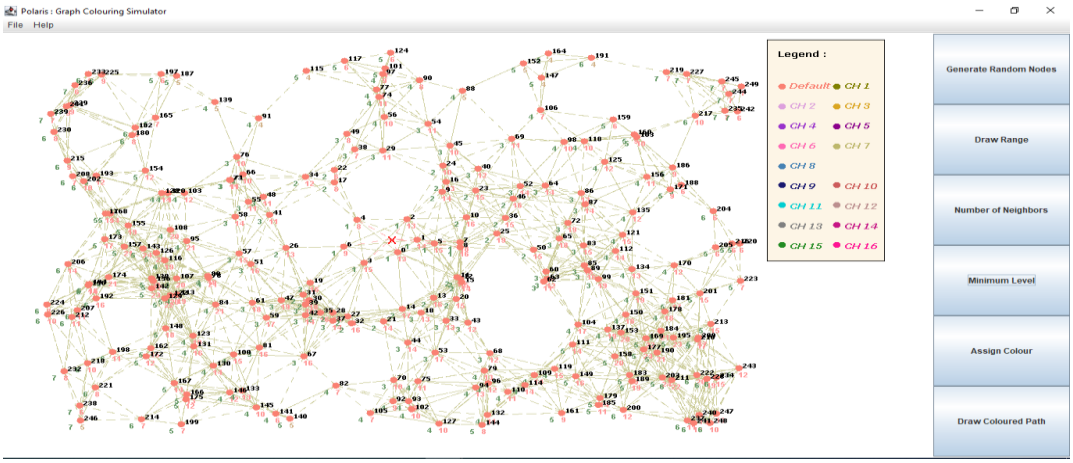

Figure 10. The result when the 'Minimum Level' button is clicked

As an illustration for close up view of Minimum Level shown in Figure 11, the minimal level for Id=0, Id=1, Id=2, Id=3, Id=4, Id=5, Id=7, and Id=8 is one, as they only need one level to get from X, the starting point, to themselves.

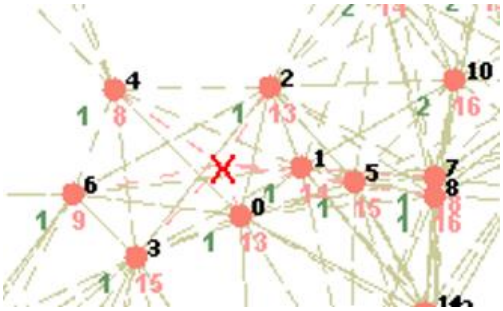

Figure 11. The close up view of Minimum Level

Table 2 displays the Ids with the minimum level 1.

Table 2. Result for Minimum Level

Starting point (from)	**The minimum level**	**No. of Id**
X	1	*Id=0*
		Id=1
		Id=2
		Id=3
		Id=4
		Id=5
		Id=7
		Id=8

Figure 12 shows the result when 'Assign Colour' button is clicked. for the 'Assign Colour' button, the graph is updated with the nodes coloured. The number of channels that the user has added determines the colour that is assigned to each node.

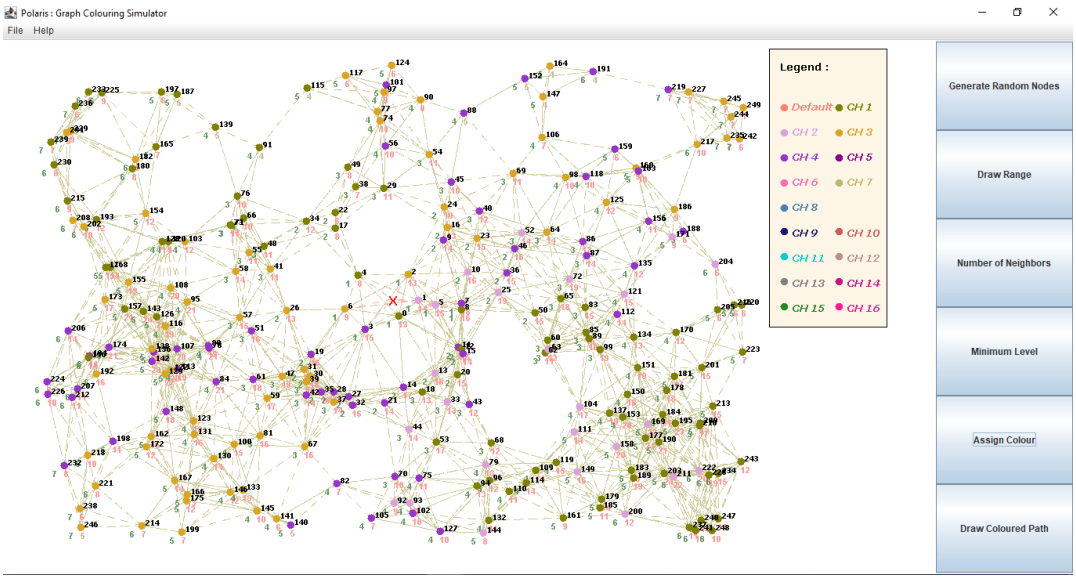

Figure 12. The result when 'Assign Colour' button is clicked

Figure 13 shows the close up view of Assign Colour. The channel allocated to this graph, according to the interface, is five. The algorithm will begin at Id=0 from starting point X. Node Id=0 is assigned to Channel 1 (green), Id=1 is assigned to Channel 2 (lilac), Id=2 is assigned to Channel 3 (yellow), Id=3 is assigned to Channel 4 (purple), Id=4 is assigned to Channel 1 (green), Id=5 is assigned to Channel 2 (lilac). For the remaining nodes, the procedure is repeated using the same algorithm.

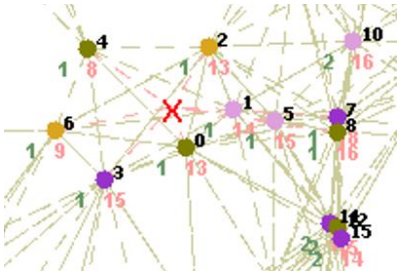

Figure 13. The close up view of Assign Colour

Table 3 displays the channel assigned for *Id=0* until *Id=4*.

Table 3. Result for Assign Colour

Starting point (from)	**No. of Id**	**Channel Assigned**
X	*Id=0*	Channel 1 (green)
	Id=1	Channel 2 (lilac)
	Id=2	Channel 3 (yellow)
	Id=3	Channel 4 (purple)
	Id=4	Channel 1 (green)

Figure 14 shows the result when 'Draw Coloured Path' is clicked which is the final result of the coloured graph. According to the number of channels assigned to each node, the edges separating each node from the others will change colour.

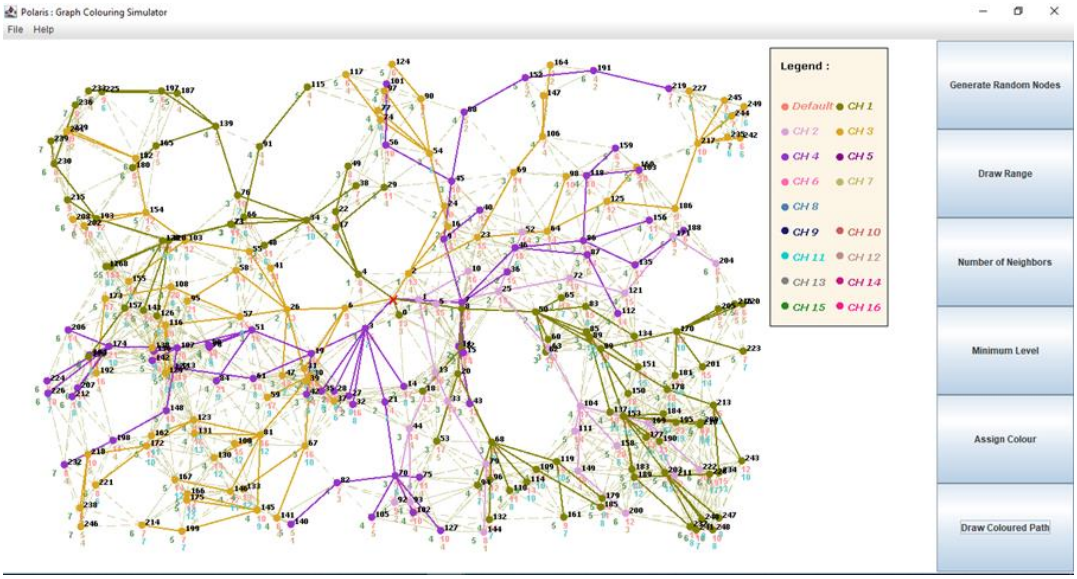

Figure 14. The final result when 'Draw Coloured Path' is clicked

Figure 15 shows the close up view of Draw Coloured Path as an illustration, from X to node Id=2, it is assigned to Channel 3 with the yellow colour. As a result, the path's colour changed to match that of the corresponding nodes. This displays a yellow path between nodes with Id=16, Id=23 and Id=24. The connection from X to node Id=3 then, connects with Id=19, Id=35, Id=28, Id=27, Id=32, Id=21 and Id=14 which are assigned to Channel 4. As a result, the range's colour shifted to purple.

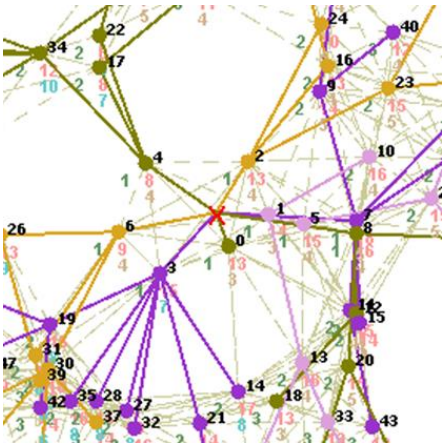

Figure 15. The close up view of Draw Coloured Path

Table 4 displays the result for coloured path from starting point *X* to *Id=2* and *Id=3*.

Table 4. Result for Coloured Path

Starting point	No. of Id (from)	No. of Id (to)	Channel	Coloured Path
X	*Id=2*	*Id=16*	Channel 3	Yellow
		Id=23		
		Id=24		
	Id=3	*Id=19*	Channel 4	Purple
		Id=35		
		Id=28		
		Id=27		
		Id=32		
		Id=21		
		Id=14		

The coloured numbers in Figure 15 have their own meaning which is explained below in Table 6.

Table 5. The meaning of coloured numbers

Colours	Meaning
Black	The number of Id assigned for each node starting from Id=0
Pink (default)	The number of neighbors for each node

Green	The minimum level represents the number of hops for each node
Blue	The number of neighbors that has been reduced

3.2 Discussion

From the results shown in subsection 3.1, step-by-step interface is shown using the developed GUI as well as the process of graph colouring. In the first interface, the process of generating vertices is shown specifically based on the values that users inserted. Next, the edges connecting to each other in the graph were also shown. Then, the number of neighbors for each of the vertices is generated and located near each vertex. Next, the minimum level process, the process of calculating the number of hops for each vertices occur and located near each vertex is shown. Then, the vertices are ready to be coloured. The process of assigning colour to the graph is updated with the vertices coloured. The number of channels that the user has added determines the colour that is assigned to each vertex. The channel allocated to the graph, according to the result above, is five. On the last interface, the graph consists of 250 vertices and the line connecting to each vertex are coloured with their respective channel assigned to them. The coloured number in blue represents the number of neighbors that has been reduced. In graph colouring, the chromatic number of a graph is the minimum number of colours for which graph colouring is possible. Thus, in this research the reduction of chromatic numbers has been achieved. By drawing the coloured graph, the chromatic number can be reduced since minimum number of colour is used and the number of neighbors also had been reduced.

There are various types of graphs that can be learned in graph theory, but this research is specifically about graph colouring. The aim to develop a GUI for graph colour assignment process in graph theory is completed. With this, all of the objectives of this study which are to design and develop a graphical user interface that is able to simulate network graph topology and to apply a graph colouring method in the developed graphical user interface has been successfully achieved.

4. Conclusion and Recommendations

This research focuses on the colour assignment process in graph colouring problems presented in the form of a Graphical User Interface (GUI). The GUI provides users with accurate and related information that they need quickly regarding graph colouring in graph theory. It also meets user's requirements and assists the users in the process of learning graph colouring through the 'Polaris: Graph Colouring Simulator'. The user interfaces developed had already been modified to satisfy varied design criteria and user interactions. The main purpose of this research study is for educational learning purposes that can help and attract people to learn more about graph theory, which is one of the fields of mathematics that is beneficial to solve some real-life problems.

In order to make this GUI more efficient, there are a few recommendations that have been recognized and can be improved in the future. Firstly, a toolbar button for zooming in the graph can be added in the GUI. Users can click the button whenever they want to zoom in to look at the graph that has been generated. The zoom button provides an additional and more direct way for the users to access the ‘Polaris: Graph Colouring Simulator’ commands. Thus, with the zoom button added, it helps the users understand and the graph can be clearly visible for the users’ convenience. Besides, the thickness of each edge generated in the graph can be improved for a much better visualization for the users. The thickness of the edges in the graph

generated can be increased. This improvement can help the users distinguish each edge in the graph since the overlapping edges might be confusing to some people. Lastly, each coloured number generated in the graph can be changed to bold colours. This is because the current colours used are too light and not that visible along with other edges and vertices generated. Hence, with the recommendations provided above, the visualization of the graph generated inside the GUI will be efficient, clear and consistent for the users.

Acknowledgement

The authors would like to thank the Department of Mathematics, Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA for supporting this project.

References

Ait-Ferhat, D., Juliard, V., Stauffer, G., & Torres, J. A. (2020). The k-path coloring problem in graphs of bounded treewidth: An application in integrated circuit manufacturing. *Operations Research Letters*, *48*(5), 652–657. https://doi.org/10.1016/j.orl.2020.07.017

Akça, E., & Tanriöver, Ö. Ö. (2021). A comprehensive appraisal of perceptual visual complexity analysis methods in GUI design. *Displays*, *69*. https://doi.org/10.1016/j.displa.2021.102031

Alfouzan, F., Shahrabi, A., Ghoreyshi, S. M., & Boutaleb, T. (2018). Graph colouring MAC protocol for underwater sensor networks. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, *2018-May*, 120–127. https://doi.org/10.1109/AINA.2018.00030

Assi, M., Halawi, B., & Haraty, R. A. (2018). Genetic Algorithm Analysis using the Graph Coloring Method for Solving the University Timetable Problem. *Procedia Computer Science*, *126*, 899–906. https://doi.org/10.1016/j.procS.2018.08.024

De, S. (2022). An efficient technique of resource scheduling in cloud using graph coloring algorithm. *Global Transitions Proceedings*, *3*(1), 169–176. https://doi.org/10.1016/j.gltp.2022.03.005

Elhag, A., & Özcan, E. (2015). A grouping hyper-heuristic framework: Application on graph colouring. *Expert Systems with Applications*, *42*(13), 5491–5507. https://doi.org/10.1016/j.eswa.2015.01.038

Hao, X., Yao, N., Wang, L., & Wang, J. (2020). Joint resource allocation algorithm based on multi-objective optimization for wireless sensor networks. *Applied Soft Computing Journal*, *94*. https://doi.org/10.1016/j.asoc.2020.106470

Kumari, G. M. P. S., Thiagarajan, K., & Bapu, T. B. B. R. (2019). N-hop network determination for irregular connected topology. *AIP Conference Proceedings*, *2112*. https://doi.org/10.1063/1.5112244

Le-Ngoc, K. K., Tho, Q. T., Bui, T. H., Rahmani, A. M., & Hosseinzadeh, M. (2022). Optimized fuzzy clustering in wireless sensor networks using improved squirrel search algorithm. *Fuzzy Sets and Systems*, *438*, 121–147. https://doi.org/10.1016/j.fss.2021.07.018

Lin, Y. D., Chu, E. T. H., Wen, C. L., Lai, Y. C., & Chen, I. C. (2018). Benchmarking handheld graphical user interface: Smoothness quality of experience. *Computers and Electrical Engineering*, *68*, 76–91. https://doi.org/10.1016/j.compeleceng.2018.03.044

Majeed, A., & Rauf, I. (2020). Graph theory: A comprehensive survey about graph theory applications in computer science and social networks. In *Inventions* (Vol. 5, Issue 1). MDPI Multidisciplinary Digital Publishing Institute. https://doi.org/10.3390/inventions5010010

Martin, B., Paulusma, D., & Smith, S. (2022). Colouring graphs of bounded diameter in the absence of small cycles. *Discrete Applied Mathematics*, *314*, 150–161. https://doi.org/10.1016/j.dam.2022.02.026

Narayan Shukl, A., & Garg, M. (2018). A List based Approach to Solve Graph Coloring Problem; A List based Approach to Solve Graph Coloring Problem. In *2018 International Conference on System Modeling & Advancement in Research Trends (SMART)*.

Oulasvirta, A., Dayama, N. R., Shiripour, M., John, M., & Karrenbauer, A. (2020). Combinatorial Optimization of Graphical User Interface Designs. *Proceedings of the IEEE*, *108*(3), 434–464. https://doi.org/10.1109/JPROC.2020.2969687

Shukla, A. N., Bharti, V., & Garg, M. L. (2021). A Greedy Technique Based Improved Approach to Solve Graph Colouring Problem. *EAI Endorsed Transactions on Scalable Information Systems*, *8*(31), 1–9. https://doi.org/10.4108/eai.16-2-2021.168716

Thadani, S., Bagora, S., & Sharma, A. (2022). Applications of graph coloring in various fields. *Materials Today: Proceedings*. https://doi.org/10.1016/j.matpr.2022.06.392