Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 19 No. 21 (2025) | OPEN ACCESS

https://doi.org/10.3991/ijim.v19i21.55895

REPORT

Mobile Application for Analyzing and Graph Coloring

Toma Katsarski(✉)

South-West University, Blagoevgrad, Bulgaria

t.katsarski@swu.bg

ABSTRACT

The present topic aims to present a mobile application for graph coloring. A demonstration of creating and vertex coloring of the most commonly used undirected graphs for this purpose is made, and the other functions in the application are also presented. An analysis of the technologies used is made and graphs with three heuristic algorithms for vertex coloring are studied. This paper provides presented and analyzed results from the execution of the algorithms on randomly generated graphs with vertices between 5000 and 10000 (5000, 6000, 7000, 8000, 9000, 10000) with an increasing number in 1000. The obtained results are considered for each graph and the minimum number of colors required for the coloring of the graph is calculated and the execution time is taken into account. In conclusion, it can be seen that the application is user-friendly and optimized for trouble-free operation on a large number of mobile devices. The technologies used allow for viewing results and performing analyses on devices with different screen resolutions and orientations.

KEYWORDS

mobile app, graph coloring, algorithms, Greedy, Welsh Powell, DSatur, Vertex coloring, ReactJS, DataBase

1 INTRODUCTION

The vertex coloring of graphs is a branch of graph theory where no two adjacent vertices can have the same color. The second thing is the chromatic index (X) for coloring the edges of the graph, or the chromatic number for coloring the vertices. In this paper, we consider (calculate) the chromatic number, or the minimum number of colors needed to color the vertices of a graph so that the rule for two adjacent vertices to be marked with different colors is observed. A graph can be represented as G = (V, E), where V is the vertex index and E are the edges. The graph coloring problem (GCP) is one of the well-studied NP-hard problems in graph theory [1]. The idea of graph coloring originated with Francis Guthrie in 1852 when he discovered that the counties of England could be colored with exactly four colors. It was initially used for planar graphs but was later applied to other areas. The reasons why the graph coloring problem is the subject of numerous studies are its

Katsarski, T. (2025). Mobile Application for Analyzing and Graph Coloring. *International Journal of Interactive Mobile Technologies (iJIM)*, 19(21), pp. 199–215. https://doi.org/10.3991/ijim.v19i21.55895

Article submitted 2025-04-05. Revision uploaded 2025-07-14. Final acceptance 2025-09-09.

application in various fields such as scheduling and planning [2, 3], radio frequency assignment [4], computer register allocation [5, 6], channel routing [7], printed circuit board testing [8], etc.

Now digital technologies, software, and smartphones are developing rapidly and play a huge role in modern lifestyle [9]. This creates the need to create mobile graph coloring software to be accessible from anywhere and convenient to use with just a few clicks on the screen of a mobile device. For the purpose of this study and for the development of the mobile application, technologies for graph visualization and calculations were carefully selected. The resolution of the different devices was also taken into account so that it looks as good as possible regardless of the model, brand, resolution, and orientation of the phone. This topic includes several stages of the development of the application. The first section provides an introduction to the terminology of graph coloring. The second section examines similar applications, mainly web-based, that are adaptable to mobile devices. The third section continues by showing the most commonly used graphs for demonstration and describes three heuristic algorithms for graph coloring. The technologies used to create the mobile application are examined in the fourth section, and the fifth section analyzes the results of the algorithms. The study ends with a conclusion in section 6.

In the process of studying existing graph applications, mainly WEB-based ones were considered. This application differs from the considered graph coloring applications with several key innovations:

- **Mobility and adaptability:** The application is designed for mobile devices, which provides accessibility and convenience in its use, expanding the potential audience.
- **Two types of graphs:** The application allows the creation of both manually and randomly generated graphs with a dynamic number of vertices. This provides a field for experiments and analysis with the integrated algorithms.
- **Graph exploration and analysis:** The inclusion of three different algorithms and their comparison in terms of chromatic number and execution time gives a new dimension to the evaluation of these algorithms, which is not common in existing solutions.

The goal of this topic is the development of a mobile application for graph analysis and coloring available for different mobile devices and operating systems, such as Android, iOS and Windows.

2 DEVELOPMENT OF GRAPH COLORING APPLICATIONS

Graphs are an important and rapidly developing field in graph theory and computer science. In recent years, there has been significant progress in methods for computing and visualizing graphs on different platforms. These developments are widely used in various fields, such as social networks, transportation networks, biological networks, and even in artificial intelligence.

Visualizing graphs on mobile devices is a particular challenge due to hardware and user interface limitations. Developing a mobile application for analyzing and coloring graphs in real time is a key focus of many studies.

2.1 Popular applications and tools

In recent years, various visualization applications and tools have become widely available and used in many industries:

- **Gephi:** Gephi is a powerful open source tool for graph visualization and analysis. Although it is primarily a desktop application, it is the basis for creating many mobile applications that offer similar functionality.
- **D3.js and mobile libraries:** D3.js, a basic JavaScript library for data visualization, has also been adapted for mobile web applications. Although primarily for web-based visualizations, there are mobile solutions that integrate D3.js through React Native and similar technologies.
- **Sigma.JS:** Sigma.js is an open source JavaScript library aimed at visualizing graphs of thousands of nodes and edges using WebGL. It was developed primarily by Alexis Jacomy and Guillaume Plique and is built on graphology [22].

2.2 Recent developments in mobile graph computing

There have been several significant developments in mobile graph computing in recent years:

- **Use of machine learning:** Modern mobile applications increasingly integrate machine learning algorithms to optimize the processes of graph analysis, cluster detection, and relationship prediction. This is especially useful in areas such as social network analysis, where algorithms can recognize hidden patterns in large networks.
- **Parallel and distributed algorithms:** New algorithms for distributed computing are also finding application in mobile graph processing, using cloud services to accelerate the computation and storage of large graph structures. This allows mobile applications to handle large and dynamic graphs without overburdening the hardware of mobile devices.
- **Optimization for small screens:** Visualizing graphs on small screens requires new approaches in user interface design. Much research focuses on creating interactive visualizations that allow the user to explore the graph through gestures such as zooming, swiping, and clicking on nodes for more information.

3 GRAPHS AND ALGORITHMS

Vertex coloring of graphs in graph theory is applicable to all types of graphs, i.e., there is no limit on the number of vertices and edges. Since the idea was initially applied to planar graphs, most often demonstrations are observed for planar graphs, binary, and cyclic. During the research, graphs that seem to be the same at first glance with different construction sequences were tested. In graph coloring, the sequence of connecting the vertices matters, as will be demonstrated later in the section. In the process of selecting the algorithms, the speed of execution of the algorithm and the best (optimal, close to optimal) solution were taken into account.

It can be noted here that for graphs with a larger number of vertices and edges, it is possible to find a solution, but it may not be optimal, or in other words, heuristics are applied. For the best solution, the minimum number (chromatic number) of vertices required for coloring the graph will be taken into account.

3.1 Types of graphs to color

Vertex coloring is not an obstacle to the type of graph; there are such graphs that are more often used for demonstrations and acquiring knowledge on the topic. With graphs with many vertices, visualization will be difficult and even impossible, especially on mobile devices, so in order to make a demonstration, as well as to consider a specific problem, existing ones can be used. Such can be cyclic (circular) graphs, wheel graphs, binary graphs, and others.

Essentially, cyclic graphs are graphs in which there are at least three vertices, and they are connected in such a way that they form a cycle. Also, each vertex has two adjacent ones. The structure of circular graphs resembles a circular figure. The number of colors required for coloring this type of graph depends on the number of vertices—whether it is even or odd. For even, the colors required are 2, and for odd, 3.

The other type that has been considered is wheel graphs. In this type, we have one main vertex located in the center surrounded and connected by the other vertices. The surrounding vertices are connected to the two neighboring vertices, i.e., each vertex has 3 neighboring vertices, one of which is located in the center. The neighboring vertices of the central vertex are equal to n – 1.

The study also considers binary graphs. The vertices are divided into groups—most often in two columns (A and B) or in two rows. Bipartite graphs are known to contain no cycles of odd length, i.e., each vertex of the first group (A) is connected by an edge to each vertex of the second group (B) [10, 11]. The optimal solution for this type of graph is two colors, but there are cases in which this is not the case and the number of colors is equal to the number of vertices.

In this topic, the focus will be on randomly generated graphs. Six graphs with a number of vertices between 5000 and 10000, with a density of 3%, 5% and 10% per vertex of the total number, will be studied. The density of the graphs is defined as the ratio of the number of edges $|E|$ to the ratio of the maximum possible edges.

For undirected simple graphs, the density of the graph is:

$$D = \frac{|E|}{\binom{|V|}{2}} = \frac{2|E|}{|V|(|V|-1)}$$

where E is the number of edges and V is the number of vertices in the graph.

The maximum number of edges for an undirected graph is

$$\binom{|V|}{2} = \frac{|V|(|V|-1)}{2}$$

so the maximum density is 1 (for complete graphs), and the minimum density is 0 [20].

3.2 Algorithms

Within the scope of this study, three algorithms with a consistent degree of complexity have been selected. The first algorithm chosen for integration into the mobile application for analyzing vertex coloring in graphs is Greedy.

The execution of this type of graph starts from the first vertex, marking it with the first color. As we mentioned at the beginning, the colors can be indicated by indices from 1 to n (C = 1 ... N). Then we move on to the next vertex and color it with the first unused color relative to its neighbors. That is, the first vertex is adjacent to vertex 2 and colored with the first color, then the color with the lowest index unused color is taken, i.e., the second. Then the steps are repeated until the entire graph is colored. The Greedy algorithm takes a short time to execute and gives accurate results, but they are not optimal. The optimal result in graph coloring is finding the chromatic number, or the minimum number of necessary markers (colors). The algorithm would be useful for use in situations where it is necessary to find a solution to a problem in a short time or for urgent research that requires quick results [23].

Welsh-Powell (WP) is another algorithm with fast result finding but also gives more accurate results. It works on the principle of sorting the vertices of the graph according to the degree of saturation, i.e., according to the number of successors of a given vertex. Sorting is performed in descending order—from the vertex with the most successors to the vertex with the fewest. After the sorting is done, the first vertex is marked with the first color, and the coloring continues with the next non-adjacent, uncolored graph and is marked with the same color. The action is repeated for all uncolored and non-adjacent vertices. Coloring continues with the first uncolored vertex.

DSatur (Degree of Saturation) is a graph coloring algorithm proposed by Daniel Brelaz in 1979 [12]. It sorts vertices in descending order according to the saturation of the vertices, i.e. the vertex with the highest number of colored neighboring vertices. If there are an equal number of vertices with equal saturation, we choose a second criterion, for vertex density or vertex index. The second thing in the “degree of saturation” algorithm is to color the first uncolored vertex with the first available color. Similar to the Greedy algorithm, it moves to the next uncolored vertex and labels it with the first lowest index color that is not used by its neighbors. The labeling cycle ends when all vertices are colored. DSatur is a heuristic graph coloring algorithm, but it gives accurate results for bipartite [12], cyclic, and cyclic graphs [13]. DSatur is also called the saturation algorithm in the literature [14]. This method is suitable [23] in cases where more accurate results are sought.

3.3 Adding new graphs and algorithms

Due to the specificity of the topic, the continuous research in the field and the discovery of new algorithms, it is of utmost importance that the application is flexible and adaptable. Therefore, the easy addition or generation of more new graphs has been taken into account. The integration of new algorithms is also affected.

4 METHODOLOGY

This section will look at the technologies and components required to develop a mobile application for vertex coloring of graphs. They include:

4.1 Graph visualization

Graph visualization is a key component for a mobile application due to the screen size, which makes visualization with a large number of vertices difficult for the device. Tests have been made with graphs up to 2000 vertices, taking into account the device screen, and it is possible. With a large number of vertices and edges, it is possible to perform the calculation in the background for optimization purposes. Another important thing is adding new vertices and the ability to easily create adjacencies between them, i.e., adding edges between vertices. It is also necessary to consider the types of mobile architectures and operating systems.

4.2 Mobile application design

The first important thing in the design of such an application is the ability to easily create a graph with just a few clicks. Considering the information in the previous subsection (4.1), it could use functionality with resizing the vertices of the graph and thus save space and allow for visualization of more vertices. It is necessary that the workspace on the mobile device screen be as simple and clean as possible for more detailed visualization. All unnecessary components should be hidden and shown when necessary. Such components would be the menu. It is ideal to use a SIDEBAR, with the ability to hide. Other options, such as editing, should also be hidden when necessary.

For optimal operation of the application, a server part will be considered on which the calculations will be performed, i.e., traversing the graph with the three types of algorithms. For graphs with a large number of vertices, it takes several hours. For this, it is rather mandatory to have a server part.

The mobile application (see Figure 3) should have the following menus:

- Menu for creating a new graph – New Graph
- Menu for viewing and visualizing existing graphs – here it is good for the application to use user sessions so that each user can see their own graphs
- Ability to edit the viewed graph
- Results of the algorithm execution – Results
- Device information – can be on a separate page or on the Home page
- Testimonials page

Figure 1 shows printscreens of the application pages. The first is the Home Page, where you can see a welcome message and information about the device (model, manufacturer, device name, Idion, platform, screen width and height). On the second page (Graph page) is the simulation environment [24] where we select from the graphs already created by the user, which are visualized in a drop-down menu. After selecting a graph, you can choose an algorithm for coloring the graph. It is also possible to edit the graph on this page. The parameters that can be edited are – graph name, vertex coordinates, whether it is visible to everyone (i.e., to be marked as public), and it is also possible to delete it. The last page shown in Figure 1 is creating a graph where vertices are placed by clicking the mouse on the graph visualization field, and edges are placed manually after entering the number of the vertices from/to which they should be connected.

Result pages, recommendations and settings have also been created, and the results page is discussed in the analysis section of the study.

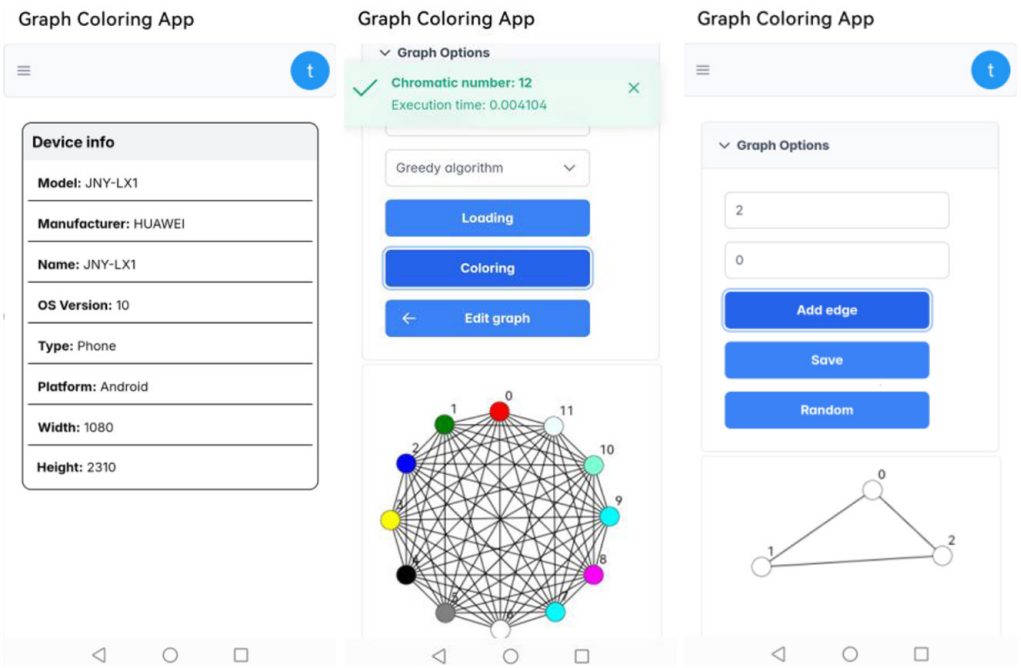

Fig. 1. Home page, Graph page, add new page preview

Figure 2 illustrates the system hardware and software identification screen extracted from an Acer Aspire series mobile computer running Windows 11 operating system.

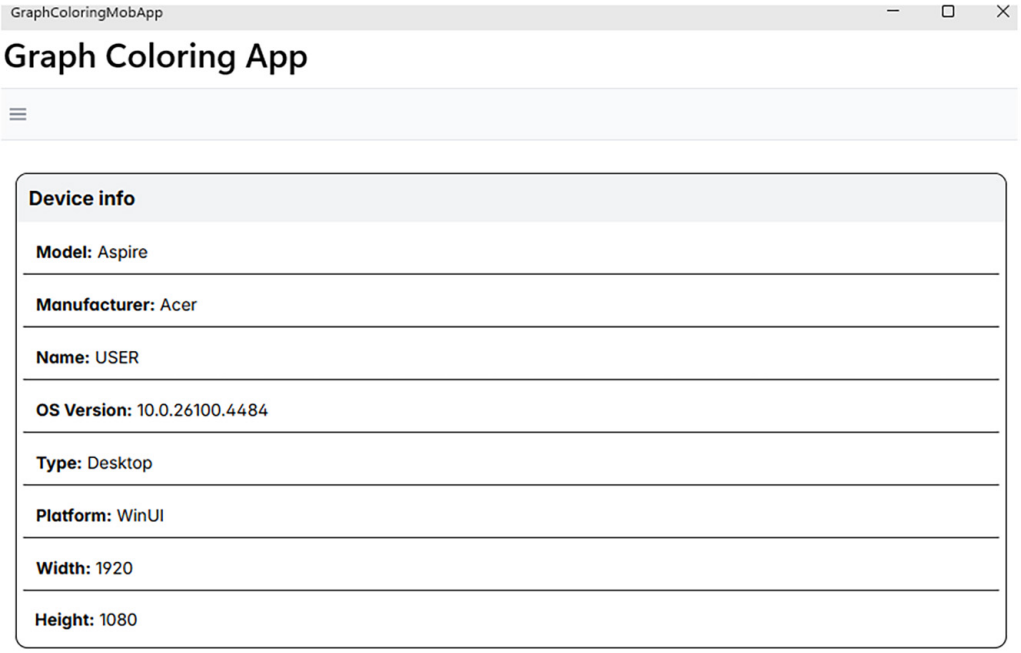

Fig. 2. Visualization of the system information page on a Microsoft Windows device

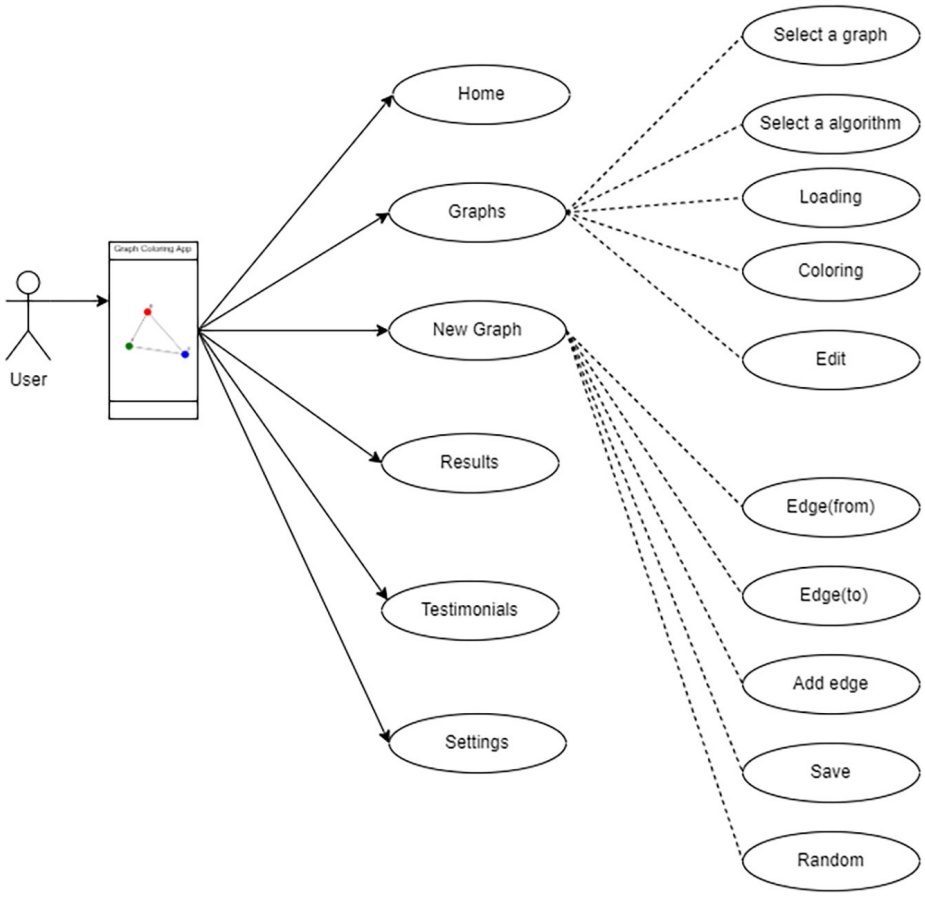

Fig. 3. UseCase diagram of the mobile graph coloring application

4.3 Mobile application development

According to the design phase, this research aims to develop an application that works with both iOS and Android operating systems [1], taking into account the resolution of the mobile device. In this paper, we will develop a multi-layered application using the following technologies:

- The ReactJS—also known as React.js or ReactJS—is a free open-source library [15] [16] that aims to make building component-based user interfaces easier and more seamless [15]. It also has the ability to integrate and use multiple libraries such as Axios for communicating with web services, Redux for storing state, React-bootstrap for responsive design, and others. Canvas is a great tool for drawing and visualizing shapes, which makes it suitable for graph visualization. The library is used to build the user interface components including Pages, Menu, Fields for creating and visualizing graphs, a results table, and a registration and login form.
- .NET Multi-Platform App UI (.NET MAUI) is a cross-platform framework for creating mobile and desktop applications with C# and XAML that can run on Android, iOS, macOS, and Windows. It contains useful components for launching applications developed with other technologies, such as the HybridWebView implemented in version 9 for displaying local HTML files [17]. WebView works in a similar way by displaying remote web pages [18]. This allows developers of other technologies to launch their applications in the .NET Maui framework. For the current study, WebView will be used.

- Backend part where all calculations (traversal) on the graph will be performed. Communication between fronted and backend will be carried out via WebServices and JSON containing data for visualization. The data will include information about the graph such as coordinates, size and color (after calculation) of the vertex, label, neighboring vertices, and others. Information from the calculations will also be processed via JSON. All this will be executed by using Spring Boot.
  - Spring Boot is an open-source Java framework used for programming stand-alone Spring-based production-grade applications with a package of libraries that facilitate the launch and management of the project [19]. The backend houses the repository for communication with the database; the services will execute the business logic. The selected algorithms will be integrated into the services, and the controllers will submit the JSON data to the frontend.
  - MariaDb will be used to store the information about the graphs. The data that will be saved in the database is the name of the graph, vertices, and neighbors of the vertices. For the results, there will be columns created for the graph, the number of colors found for coloring the vertices, and the execution time of the algorithm for the specific graph. Thus, with this architecture, saving the results in a database allows for more in-depth analyses.

4.4 Demonstration of vertex coloring of graphs with the mobile application

- **Cyclic graphs** – As mentioned above in this topic, the vertices in cyclic graphs have at least three vertices forming a cycle and each has two adjacent vertices. Figure 4 shows two cyclic graphs, the first with an odd number (5) of vertices and the second with an even number (6) of vertices. From the demonstration, it can be concluded that a cyclic graph with an odd number of vertices can be colored with three colors, and with an odd number of vertices with two. This is an optimal solution and the chromatic index has been found.

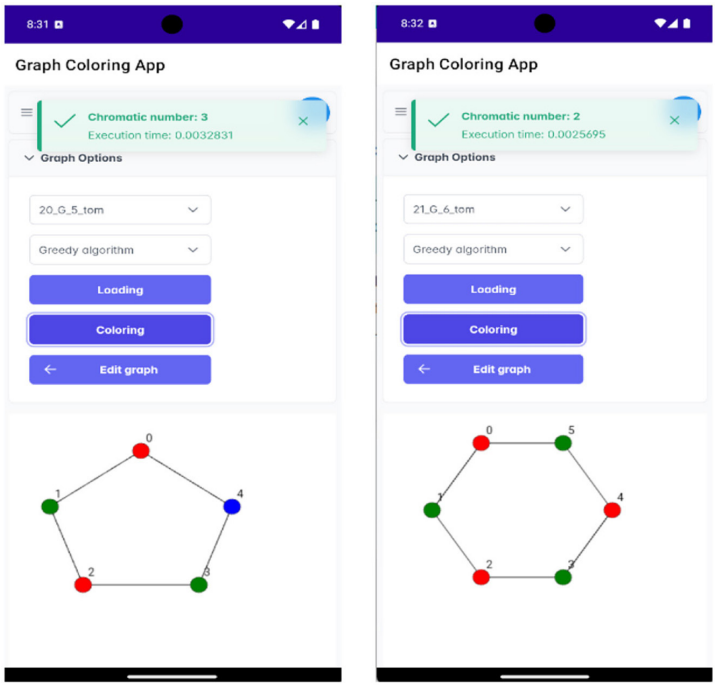

Fig. 4. Example of coloring cyclic graphs with odd and even number of vertices

- **Wheel graphs** – In wheel graphs (see Figure 5) the chromatic index also depends on whether the number of vertices is divisible by 2 without remainder or with remainder. They are similar to the cyclic graph in that all vertices are adjacent to a vertex located in the center.

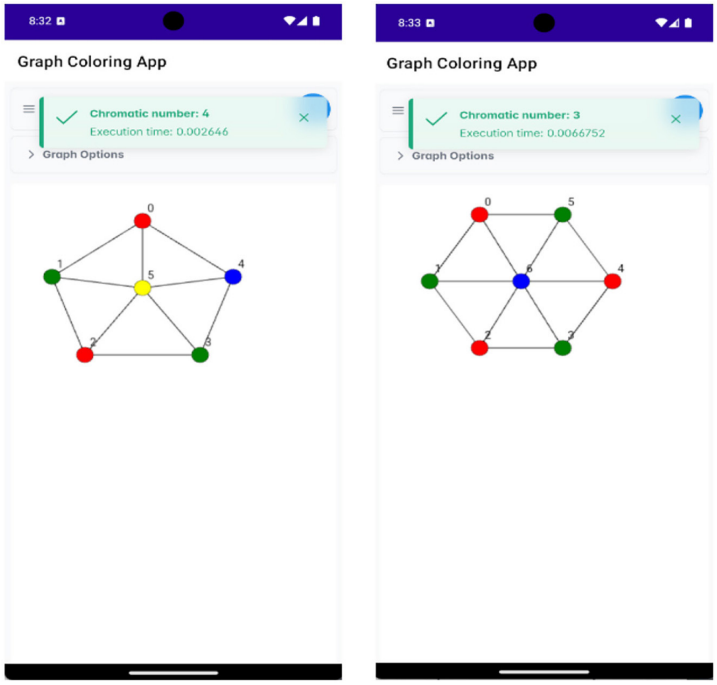

Fig. 5. Example of visualization and coloring of wheel graphs with even and odd number of vertices

- **Bipartite graph** – The vertices of a bipartite graph (see Figure 6) are divided into two groups, either horizontally or vertically. It is known that bipartite graphs does not contain any odd-length cycles [10, 11].

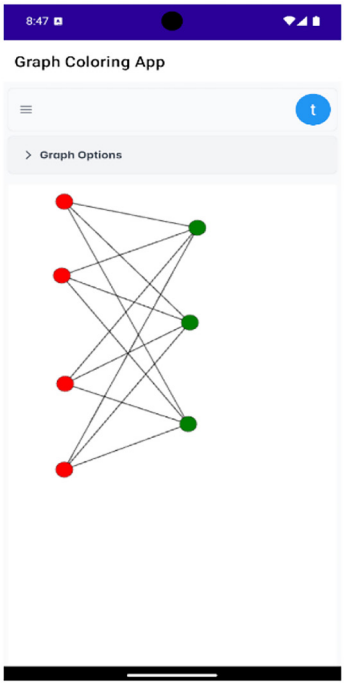

Fig. 6. Shows coloring of Bipartite graph

4.5 Testimonials and evaluation of the mobile application

The application has a specially designed page for recommendations and ratings. Figure 7 shows a page with recommendations in horizontal screen orientation. 50% of the opinions are positive, 30% said they would use the application out of curiosity and the rest said they did not find it useful in their work.

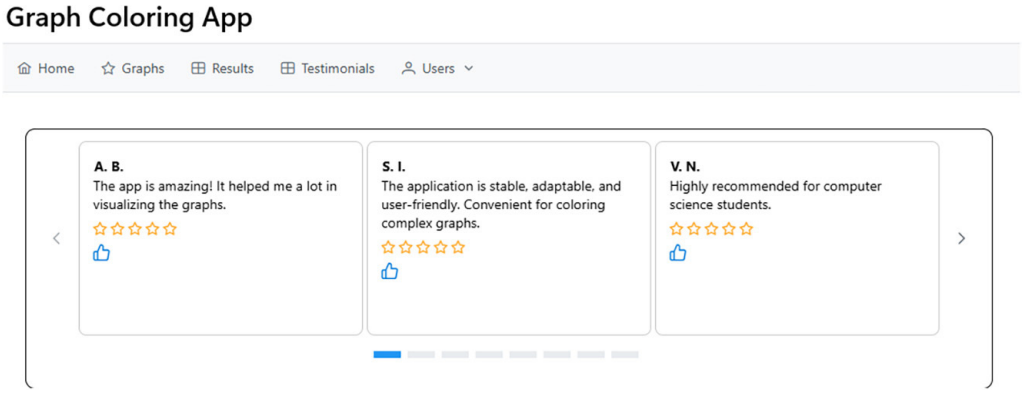

Fig. 7. Visualization with user opinions displayed in Carousel

Each registered user can give their opinion on the functionality, and the submitted recommendations will be visible to everyone. In the future development of the application, it is planned to implement other heuristic algorithms, as well as optimization algorithms.

It is also possible to integrate artificial intelligence for calculations. The application and the system as a whole maintain a data structure understandable for existing language models to perform calculations and predictions on the data.

The mobile application for coloring graphs can also be adapted for other areas of graph theory.

5 RESULTS

The experimental setup will be performed on a Huawei p40 Lite mobile device, with 6 GB Ram, Processor Kerin 810 and resolution 2310×1080. On the backend server side, it will be with a 13th Gen IntelCore(TM) i5 processor and 16.0 GB Ram.

The focus of this study will be directed towards graphs divided into three categories [23] with 5000, 6000, 7000, 8000, 9000 and 10000 vertices. The density will be as follows – first group 3, 5 and 10%. The simulation results show clear differences between the algorithms [24] and will be shown with a screenshot of the tables from the application in the following subsections.

5.1 Results of the study of the first group at a density of 3%

Figure 8 shows the results generated through the application and displayed in a table on the “Results” page of the application.

Fig. 8. Visualization of the table with results from the execution of the Greedy, WP, and DSatur algorithms at 3% density

The "Results" page consists of five columns containing the name of the graph (consisting of the number of vertices, density and the user who created the corresponding graph), in the second column are the number of neighbors between the vertices, and the third, fourth and fifth columns are the results (execution time and number of colors found) from the execution of the algorithms as follows: Greedy algorithm, WelshPowell (WP) Algorithm and DSatur. After reviewing the results for graphs with a density of 3%, it is seen that DSatur finds the closest results to the optimal solution, and the Greedy algorithm is executed in the shortest time. WP also gives accurate results and is more accurate than the Greedy algorithm. It can be seen that the Greedy algorithm executes in under one second from 5000 to 8000 vertices, WP in under five seconds for the same graphs and DSatur in about one minute. From 9000 to 10000 vertices, Greedy finds a solution in under two minutes, WP in under one minute, and DSatur for over two minutes.

Figure 9 graphically shows the execution time of the algorithms with a linear graph and the number of colors found in the form of a vertical bar.

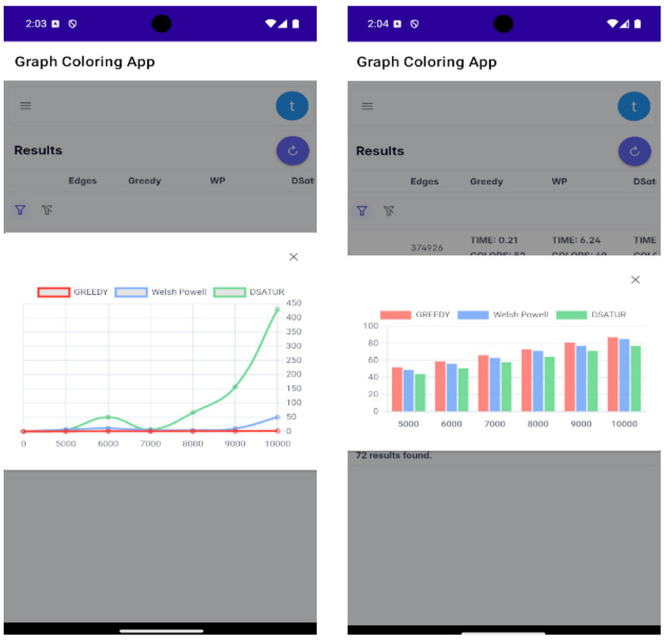

Fig. 9. Graphical representation of execution time and number of found colors at 3% graph density

5.2 Results of the study of the second group at a density of 5%

To visualize the data in all tables as in Table 1, Datatables is shown again. DataTables is a Javascript HTML library for enhancing tables. It is a very flexible tool, built on the foundations of progressive enhancement, which adds advanced features to any table [21].

Table 1. Visualization of the table with results from the execution of the Greedy, WP, and DSatur algorithms at 5% density

Graph Name	Vertices	Greedy	WP	DSatur
G5000_density5	624876	Time: 0.33 Colors: 70	Time: 4.8 Colors: 67	Time: 9.69 Colors: 62
G6000_density5	899850	Time: 0.79 Colors: 81	Time: 8.64 Colors: 78	Time: 59.91 Colors: 71
G7000_density5	1224826	Time: 0.86 Colors: 91	Time: 14.26 Colors: 87	Time: 79.36 Colors: 81
G8000_density5	1599800	Time: 1.19 Colors: 100	Time: 19.73 Colors: 97	Time: 276.61 Colors: 90
G9000_density5	2024776	Time: 2.45 Colors: 112	Time: 23.99 Colors: 108	Time: 701.17 Colors: 100
G10000_density5	2499750	Time: 2.25 Colors: 120	Time: 18.55 Colors: 117	Time: 141.01 Colors: 109

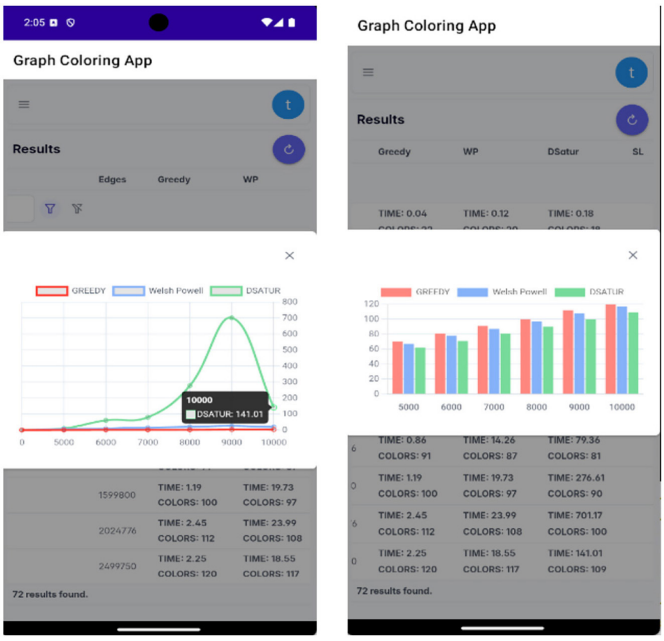

Fig. 10. Graphical representation of execution time and number of found colors at 5% graph density

Figure 10 graphically shows the execution time of the algorithms with a linear graph and the number of colors found in the form of a vertical bar.

5.3 Results from the study of the third group of graphs at a density of 10%

Since the execution results may vary depending on the server load, the execution time and number of colors data in Table 2 have been analyzed in summary.

Table 2. Visualization of the table with results from the execution of the Greedy, WP, and DSatur algorithms at 10% density

Graph Name	Vertices	Greedy	WP	DSatur
G5000_density10	1249750	Time: 0.81 Colors: 115	Time: 4.79 Colors: 111	Time: 83.22 Colors: 103
G6000_density10	1799700	Time: 2.57 Colors: 132	Time: 12.9 Colors: 128	Time: 214.33 Colors: 120
G7000_density10	2449650	Time: 2.23 Colors: 151	Time: 37.67 Colors: 146	Time: 3881.22 Colors: 136
G8000_density10	3199600	Time: 2.29 Colors: 166	Time: 31.06 Colors: 162	Time: 1923.49 Colors: 153
G9000_density10	4049550	Time: 7.86 Colors: 184	Time: 98.77 Colors: 179	Time: 5181.98 Colors: 168
G10000_density10	4999500	Time: 10.29 Colors: 200	Time: 96.04 Colors: 196	Time: 2098.33 Colors: 185

The Greedy algorithm performed a total of 948 color calculations for graphs from 5000 to 10000 vertices in 26.05 seconds. WP took 281.23 seconds or 4.69 minutes and found 922 colors. DSatur completed the execution for all six graphs (5000, 6000, 7000, 8000, 9000, 10000 vertices) at a density of 10% in 13382.57 seconds or 4 hours and 11 minutes and found 865 colors.

It can be concluded that for graphs with a density of 10%, the best results, i.e. the smallest number of colors for coloring the graphs by DSatur were found, considering that it took many times more time than the other two algorithms. Greedy spent a total of about 26 seconds, and WP about five minutes.

Fig. 11. Graphical representation of execution time and number of found colors at 10% graph density

Figure 11 graphically shows the execution time of the algorithms with a linear graph and the number of colors found in the form of a vertical bar.

The application uses a database where the same graph can be searched multiple times, and the number of colors needed to color the graph will be recorded, but the time may vary depending on the server load. Load may occur when the system is used by many users, multiple requests are made. Then the time will increase and when few requests are sent to the server the execution time for a particular graph will be low.

6 CONCLUSION

With the help of the Javascript and Java programming languages and frameworks based on them, a mobile application for vertex coloring of graphs in graph theory was developed. Demonstrations were made on manually creating graphs and graphs created randomly. As mentioned at the beginning, graph coloring is a subject of research and applicable in many areas, and with the help of the developed application, vertices can easily be replaced by circles with pictures of wifi networks, pictures of mobile cells and others. For visualization of the results, a table with the DataTables and Charts components was used for the graphical visualization of the execution time of the algorithms and the number of markers found.

After analyzing the results of the research in the three categories with randomly generated graphs with a density of 3%, 5%, and 10% with 5000, 6000, 7000, 8000, 9000, and 10000 number of vertices, it is seen that the most accurate or close to the optimal solution results are given by the DSatur algorithm. It is also observed that the execution of DSatur requires more time and more hardware resources. Greedy also finds the correct solution and completes the execution many times faster than DSatur. Welsh Pawell finds more accurate results than the Greedy algorithm and takes less time to execute than DSatur. In conclusion, if a more accurate solution is not necessarily sought and time is limited, Greedy is the best solution, and WP can also be used. Despite these advantages, it should be considered that the results from such a sample may not reflect accurate results [23]. On the other hand, if resources (time and hardware) are not limited and better results are sought, it is desirable to use DSatur. It is very important to note that the execution time of a selected algorithm depends on the server load and the number of users using the system, as well as the complexity of the generated graph. Therefore, the same combination (algorithm and graph) may return different results. It should also be noted that the number of found markers (colors) necessary for coloring a graph does not depend on the execution time.

Table 3 compares the three algorithms based on the following criteria:

- Accuracy of results
- Algorithm execution time (complexity)
- Advantages and disadvantages

Table 3. Comparison table of algorithms by criteria

Algorithm	Accuracy	Execution Time	Advantages	Disadvantages
Greedy	Poor quality solution, could use many colors	O(n + m)	Very fast, easy to implement	Often does not give good results, does not guarantee optimality
WP	Good quality solution, closer to optimal	O(n log n)	Good for graphs with relatively even degrees	Does not guarantee an optimal solution
DSatur	Better solution quality, especially for complex graphs	$O(n^2)$	Reduces the number of colors used	Slower than the greedy algorithm

The developed mobile application is suitable for calculations of the chromatic number (minimum number of colors needed to color the vertices of a graph), and can be used to do:

- lists of tasks and their distribution
- visualization of the Sudoku game and solving
- drawing computer and mobile networks where the vertices will be the mobile cells, and the edges the connection between and can be calculated the minimum number of frequencies to use, etc.

It is planned to integrate more algorithms for optimization purposes to find more accurate results and collect sufficient data for integration into optimization algorithms.

The data structure for the graphs and results is well-formed for understanding by artificial intelligence, as integration with the application can perform calculations and provide predictions on graphs with a large number of vertices and edges.

7 ACKNOWLEDGEMENT

I thank South-West University "Neofit Rilski," Blagoevgrad, Bulgaria, for the opportunity to prepare and publish the article on the development of "**Mobile application for analyzing and graph coloring**."

I also express my gratitude to Assoc. Prof. Dr. Velin Kralev and Assoc. Prof. Dr. Radoslava Kraleva from South-West University "Neofit Rilski" in Blagoevgrad, Bulgaria, for their valuable guidance in selecting and integrating appropriate algorithms and technologies. I thank them for their helpful comments and professional support during the development of the application and the preparation of this article, which were essential for the completion of the mobile application and the research. Thanks to Dimitar Chakalov, a doctoral student and assistant professor at South-West University "Neofit Rilski" in Blagoevgrad, for the conversations and exchange of ideas on the topic of modern technologies, whose opinion was a valuable source of reflection and inspiration during the work on the article. Special thanks to Petya Dimitrova for her reading and editing of the article, which significantly improved the quality of the material to make it more successful.

8 REFERENCES

[1] M. R. Garey and D. S. Johnson, "Computers and intractability," *A Guide to the Theory of NP-Completeness*. New York, NY, USA: W.H. Freeman & Co., 1979. ISBN: 0716710447.

[2] D. de Werra, "An introduction to timetabling," *Eur. J. Oper. Res.*, vol. 19, no. 2, pp. 151–162, 1985. https://doi.org/10.1016/0377-2217(85)90167-5

[3] K. Dowsland and J. Thompson, "Ant colony optimization for the examination scheduling problem," *J. Oper. Res. Soc.*, vol. 56, no. 4, pp. 426–438, 2005. https://doi.org/10.1057/palgrave.jors.2601830

[4] A. Gamst, "Some lower bounds for a class of frequency assignment problems," *IEEE Trans. Veh. Technol.*, vol. 35, no. 1, pp. 8–14, 1986. https://doi.org/10.1109/T-VT.1986.24063

[5] G. J. Chaitin, M. A. Auslander, A. K. Chandra, J. Cocke, M. E. Hopkins, and P. W. Markstein, "Register allocation via coloring," *Computer Languages*, vol. 6, no. 1, pp. 47–57, 1981. https://doi.org/10.1016/0096-0551(81)90048-5

[6] F. C. Chow and J. L. Hennessy, "The priority-based coloring approach to register allocation," *ACM Trans. Program. Lang. Syst. (TOPLAS)*, vol. 12, no. 4, pp. 501–536, 1990. https://doi.org/10.1145/88616.88621

[7] M. R. Garey, D. Johnson, and H. So, "An application of graph coloring to printed circuit testing," *IEEE Circuits Syst.*, vol. 23, no. 10, pp. 591–599, 1976. https://doi.org/10.1109/TCS.1976.1084138

[8] S. Sen Sarma, R. Mandal, and A. Seth, "Some sequential graph colouring algorithms for restricted channel routeing," *Int. J. Electron.*, vol. 77, no. 1, pp. 81–93, 1994. https://doi.org/10.1080/00207219408926037

[9] S. Nuanmeesri, "Mobile application development of managing elderly household accounts using speech recognition," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 14, no. 2, pp. 84–100, 2020. https://doi.org/10.3991/ijim.v14i02.11651

[10] R. Diestel, *Graph Theory, Graduate Texts in Mathematics*. Springer, 2005.

[11] S. A. Asratian, M. J. T. Denley, and R. Häggkvist, *Bipartite Graphs and their Applications*. Cambridge, UK: Cambridge University Press, 1998. https://doi.org/10.1017/CBO9780511984068

[12] D. Brélaz, "New methods to color the vertices of a graph," *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, 1979. https://doi.org/10.1145/359094.359101

[13] R. M. R. Lewis, *A Guide to Graph Colouring: Algorithms and Applications. Texts in Computer Science,* 2nd ed. Berlin, Germany: Springer, 2021. https://doi.org/10.1007/978-3-030-81054-2

[14] M. Kubale, Eds., *Graph Colorings*, vol. 352, Contemporary Mathematics, 2004. https://doi.org/10.1090/conm/352

[15] React (software). https://en.wikipedia.org/wiki/React_(software)

[16] "Chapter 1. What is React? – What React is and Why it Matters [Book]," https://www.oreilly.com/library/view/what-react-is/9781491996744/ch01.html

[17] Microsoft Ignite, "HybridWebView," 2025. https://learn.microsoft.com/en-us/dotnet/maui/user-interface/controls/hybridwebview?view=net-maui-9.0

[18] Microsoft Ignite, "WebView," 2025. https://learn.microsoft.com/en-us/dotnet/maui/user-interface/controls/webview?view=net-maui-9.0&pivots=devices-android

[19] GeeksforGeeks, "Spring Boot Tutorial – Learn Spring Boot," 2023.

[20] F. T. Coleman and J. J. Moré, "Estimation of sparse Jacobian matrices and graph coloring problems," *SIAM Journal on Numerical Analysis*, vol. 20, no. 1, pp. 187–209, 1983. https://doi.org/10.1137/0720013

[21] DataTables, "Advanced interactive data grid for your finance data," 2025. https://datatables.net/

[22] Sciences-Po médialab and OuestWare, "Sigma.js," 2025. https://www.sigmajs.org/docs/

[23] M. Hakiki, R. Fadli, A. Sabir, A. Prihatmojo, Y. Hidayah, and Irwandi, "The impact of blockchain technology effectiveness in Indonesia's learning system," *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 20, no. 7, pp. 4–17, 2024. https://doi.org/10.3991/ijoe.v20i07.47675

[24] K. Tsachrelias, C.-A. Katsigiannis, V. Kokkinos, A. Gkamas, C. Bouras, and P. Pouyioutas, "Game theory algorithms for resource allocation in 5G MIMO," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 19, no. 13, pp. 183–203, 2025. https://doi.org/10.3991/ijim.v19i13.56051

9 AUTHOR

Toma Katsarski holds a Master's degree in Informatics from South-West University "Neofit Rilski," Blagoevgrad, Bulgaria. His areas of work and research are in the field of "Graph Theory," Software Technologies, and Computer Networks (E-mail: t.katsarski@swu.bg).