Faculty of Computers and Artificial Intelligence



Artificial Intelligence

Fall semester 2023-2024

**Artificial Intelligence project**

***Team Members***

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **ID** | Department |
| 1 | **احمد صابر محمود محمد** | **20210063** | **CS** |
| 2 | **عامر مدحت احمد** | **20210484** | **CS** |
| 3 | **مريم عزت حسن همام** | **20210916** | **CS** |
| 4 | **مريم محمود سيد محمود** | **20210926** | **CS** |
| 5 | **مريم مدحت الليثي** | **20210928** | **CS** |
| 6 | **مريم ياسر عمر شندي** | **20210931** | **CS** |
| 7 | **نورا على عبده محمود** | **20211010** | **CS** |

**AI Documentation**

**-Introduction and overview**

**Project Title:**

Intelligent Graph Coloring Solver with Backtracking and Genetic Algorithms.

**Introduction:**

The Graph Coloring Problem poses a significant combinatorial optimization challenge, requiring the assignment of colors to vertices in a graph such that no adjacent vertices share the same color. This project aims to develop an intelligent Graph Coloring Problem Solver, leveraging both the Backtracking Search Algorithm and a Genetic Algorithm. By combining these approaches, the solver seeks to efficiently explore the solution space and evolve optimal colorings for graph vertices.

**Problem Description:**

In the Graph Coloring Problem, a graph is defined by vertices and edges. The primary objective is to color each vertex in a manner that avoids conflicts between connected vertices. The chromatic number, representing the minimum number of colors needed for a valid coloring, is a key metric. The project's goal is to discover a valid coloring solution that minimizes the chromatic number, thus optimizing the graph's color assignment.

**- Applied algorithms**

**Backtracking Search Algorithm:**

The Backtracking Search Algorithm is a systematic exploration method that meticulously examines various color assignments for each vertex. Operating through recursive trials of different colors and backtracking upon conflict detection, this algorithm intelligently navigates the solution space. By pruning paths leading to infeasible colorings, the Backtracking Search Algorithm efficiently converges towards valid and optimized solutions.

**Genetic Algorithm:**

Inspired by natural selection, the Genetic Algorithm represents potential colorings as chromosomes. The algorithm initiates a population of colorings and evolves them over generations using genetic operations, including crossover and mutation. Fitter colorings, characterized by fewer conflicts, are more likely to be passed on to subsequent generations. Through this iterative process, the Genetic Algorithm explores diverse color combinations and converges towards solutions with minimized chromatic numbers.

**Expected Outcome:**

The project aims to deliver an intelligent Graph Coloring Solver that efficiently explores the solution space, leveraging the Backtracking Search Algorithm and Genetic Algorithm. The solver should produce optimized color assignments for graph vertices, minimizing the chromatic number and demonstrating the efficacy of the hybrid approach.

**-Main functionalities**

**Graph Representation:**

Define the graph structure with vertices and edges.

Establish methods for representing the graph and its properties.

**Backtracking Search Implementation:**

Develop a backtracking algorithm to explore different color assignments systematically.

Implement recursive functions for color trials and backtracking.

**Genetic Algorithm Integration:**

Design chromosome representation for potential colorings.

Develop genetic operators: crossover and mutation.

Initialize a population of colorings and evolve them over generations.

**Edge Connections:**

Responsible for extracting user-inputted edge connections and incorporating them into the graph. This step is crucial in defining the relationships between nodes in the graph, forming the basis for subsequent graph algorithms.

**Adding Nodes and Edges:**

-For each edge connection, the code adds the corresponding nodes (**v1** and **v2**) to the graph, ensuring that nodes are explicitly included even if they do not have edges connecting them.

-It then adds the edge between the source and destination nodes to establish the connection.

**User Interface:**

Develop a user-friendly interface for users to input graphs and visualize coloring solutions.

Provide informative feedback on the chromatic number and optimized color assignments.

**Measure Performance:**

Measuring performance by time involves assessing the execution time of a backtracking or genetic operation. It's a crucial aspect of performance evaluation, and it helps developers understand how efficient their code is in terms of runtime.

**-Similar applications in the market**

**Graph Coloring in Map Applications:**

Description: Develop an intelligent graph coloring algorithm to assist in map applications where different regions or areas need to be colored in such a way that adjacent regions have distinct colors. This could be applied to political districting, urban planning, or any scenario where color-coding regions is important.

Algorithm: Utilize a graph coloring solver that combines backtracking for systematic exploration and a genetic algorithm for optimization. The objective is to minimize the number of colors used in coloring different regions.

**Resource Allocation in Cloud Computing:**

Description: Create a system for optimizing resource allocation in a cloud computing environment where different tasks or processes (vertices) share resources (colors). The goal is to allocate resources efficiently, avoiding conflicts among tasks that share the same resources.

Algorithm: Implement a hybrid approach combining backtracking for exploring resource allocation possibilities and a genetic algorithm for evolving optimal resource allocation strategies over time.

**Timetable Scheduling for Educational Institutions:**

Description: Design an intelligent system for scheduling classes in educational institutions where courses (vertices) need to be assigned time slots (colors) such that no two overlapping courses occur simultaneously. The objective is to create an optimized timetable with minimal conflicts.

Algorithm: Develop a graph coloring solution using backtracking for exploring different scheduling options and a genetic algorithm to evolve and refine timetables over multiple iterations.

**Wireless Network Channel Assignment:**

Description: Build a tool for optimizing channel assignment in wireless networks, where different network nodes (vertices) need to be assigned communication channels (colors) to avoid interference. The goal is to minimize channel conflicts and optimize network performance.

Algorithm: Implement a graph coloring solution that combines backtracking for systematic channel assignment exploration and a genetic algorithm to evolve channel assignment strategies over time.

**Circuit Board Layout Optimization:**

Description: Develop an intelligent system for optimizing the layout of components on a circuit board, where components (vertices) need to be assigned positions (colors) to minimize interference and optimize the efficiency of the circuit. The objective is to find an optimal layout with minimal conflicts.

Algorithm: Create a graph coloring solution using backtracking for exploring different component layouts and a genetic algorithm to evolve and refine optimal circuit board layouts.

**-Research Papers:**

[**https://www.sciencedirect.com/science/article/pii/S1877050918313024**](https://www.sciencedirect.com/science/article/pii/S1877050918313024)

[**https://link.springer.com/chapter/10.1007/978-3-540-69384-0\_92**](https://link.springer.com/chapter/10.1007/978-3-540-69384-0_92)

[**https://link.springer.com/article/10.1007/s13369-017-2686-9**](https://link.springer.com/article/10.1007/s13369-017-2686-9)

**https://www.sciencedirect.com/science/article/abs/pii/0196677485900446**

[**https://dergipark.org.tr/en/pub/ijisae/issue/25999/273053**](https://dergipark.org.tr/en/pub/ijisae/issue/25999/273053)

**https://www.sciencedirect.com/science/article/abs/pii/S0167819112000592**

**-Development platform:**

* **Integrated Development Environment (IDE):**

The code was written and tested in Visual Studio, an integrated development environment (IDE) that provides a comprehensive set of tools for Python development.

* **Programming Language:**

The application is developed using Python, a high-level, interpreted programming language.

* **GUI Framework:**

The graphical user interface (GUI) is implemented using Tkinter, which is the standard GUI toolkit that comes with Python. Tkinter provides a set of tools for creating graphical interfaces, making it widely used for desktop applications.

* **Graph Representation and Manipulation:**

The networkx library is employed for graph representation and manipulation. networkx is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

* **Graph Visualization:**

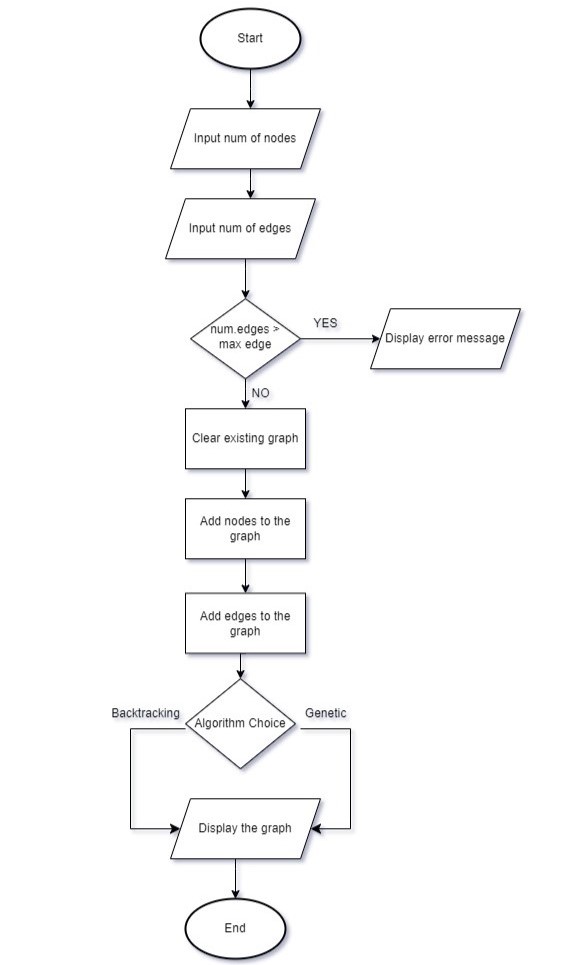
Graph visualization is achieved using matplotlib, a 2D plotting library for Python. The integration with Tkinter is facilitated by the FigureCanvasTkAgg class from matplotlib.backends.backend\_tkagg library.

* **Additional Modules:**

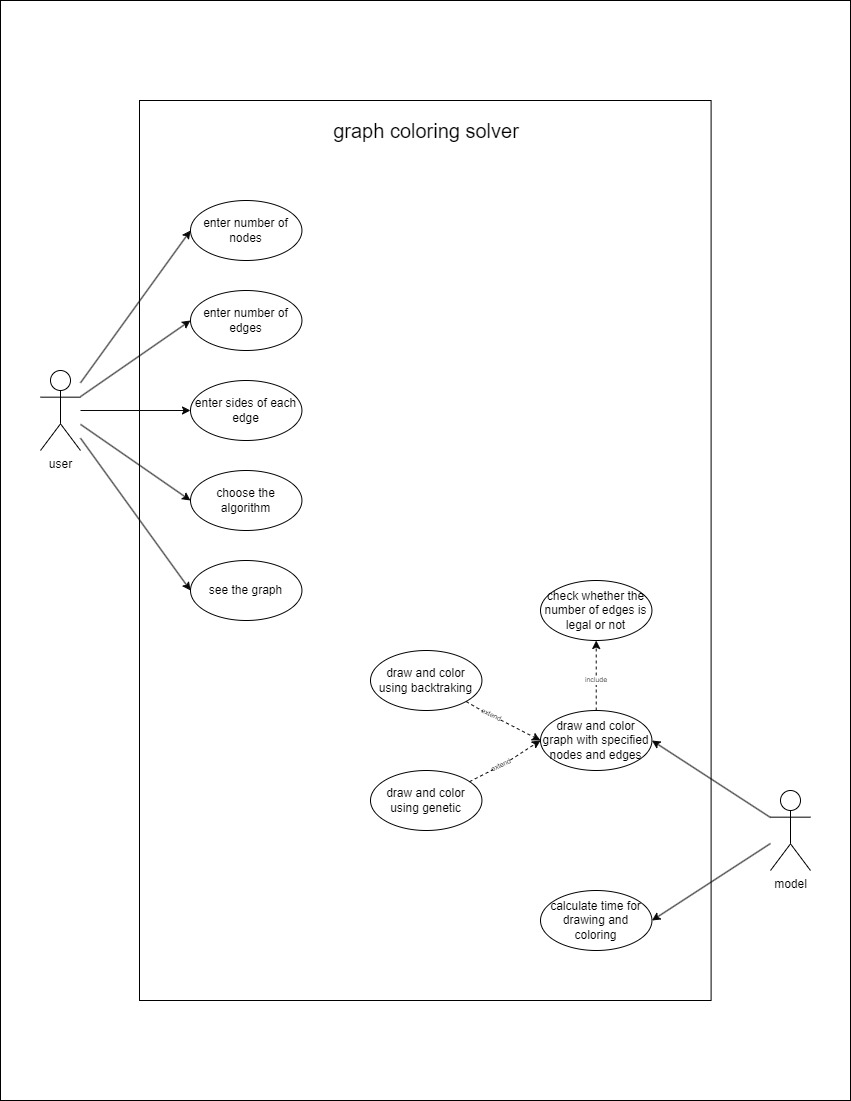
**Time Module:**

* The time module provides various time-related functions. In this application, it is used for measuring the time taken by different algorithms during graph processing.

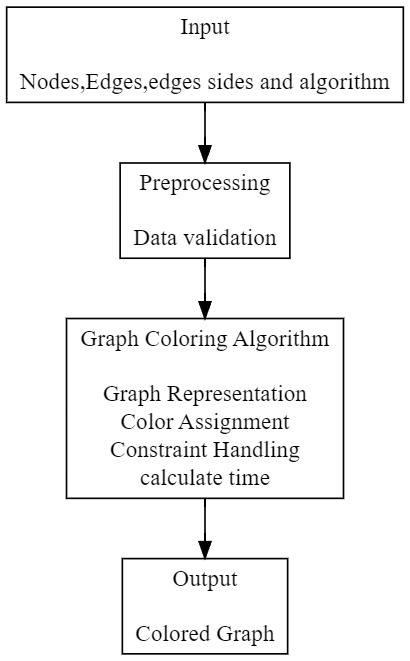
**-Workflow Diagram:**

****

**-USE CASE Diagram:**

****

**-Block Diagram:**

****