

River Crossing Problem Solver: User Documentation

1. Introduction

The **River Crossing Problem Solver** is a Python-based application designed to find and visualize solutions to the classic **Missionaries and Cannibals Problem**. This problem is a well-known state-space search puzzle often used to demonstrate various Artificial Intelligence search algorithms.

The application provides two primary modes of operation:

- 1 **Command-Line Interface (CLI):** Used to run and compare the performance metrics of five different search algorithms.
- 2 **Graphical User Interface (GUI):** Used to visually simulate the step-by-step solution for a chosen algorithm, complete with a visual representation of the river, boat, and characters.

2. System Requirements and Setup

2.1. Requirements

To run the application, you will need:

- **Python 3.x** (version 3.6 or higher is recommended).
- The **pygame** library, which is used by the GUI for audio playback.

2.2. Setup Instructions

- 3 **Download the files from github**

```
git clone [https://github.com/EyadEhab/River-Crossing-Problem.git](https://github.com/EyadEhab/River-Crossing-Problem.git)
```

This will create a directory named River-Crossing-Problem.

- 4 **Navigate to the Directory:** Open your terminal or command prompt and change the directory to the extracted folder:

```
cd River-Crossing-Problem
```

- 5 **Install Dependencies:** Install the required pygame library. It is highly recommended to use a Python virtual environment to manage dependencies.

```
# (Optional but Recommended) Create and activate a virtual environment
```

```
python3 -m venv venv  
source venv/bin/activate # On Linux/macOS  
# venv\Scripts\activate # On Windows
```

```
# Install pygame
```

```
pip install pygame
```

3. Command-Line Interface (CLI) Usage

The CLI mode is ideal for quickly comparing the efficiency of the different search algorithms.

3.1. Running the CLI

Execute the main.py file using your Python interpreter:

```
python3 main.py
```

3.2. Interpreting the Output

The program will automatically run all five implemented algorithms (Breadth-First Search, Depth-First Search, A*, Greedy, and Constraint Satisfaction Problem solver) and display a summary table of their performance.

Column	Description
Algorithm	The name of the search algorithm used.
Path Length	The number of moves (steps) required to solve the puzzle. A lower number indicates a more optimal solution.
Nodes Explored	The total number of states the algorithm visited during the search. A lower number indicates better search efficiency.
Time (ms)	The time taken (in milliseconds) for the algorithm to find the solution.

Note: The Missionaries and Cannibals problem has a known optimal solution length. Algorithms like A* and BFS are typically guaranteed to find the shortest path, while DFS and Greedy Search may find longer paths or fail to find a solution quickly depending on the implementation.

4. Graphical User Interface (GUI) Usage

The GUI provides a visual, step-by-step animation of the solution path found by the chosen algorithm.

4.1. Running the GUI

Execute the `gui.py` file using your Python interpreter:

```
python3 gui.py
```

4.2. Start Menu

Upon launching, the application will present a start menu with the following options:

- *Individual Algorithm Buttons (e.g., BFS, A):** Click any of these to run the selected algorithm and immediately begin the visual simulation of its solution.
- **Compare All Algorithms:** Click this button to display the same performance comparison table as the CLI, but within the GUI window.

4.3. Simulation Controls

Once a simulation begins, the following controls are available:

Control	Location	Function
Pause/Resume	Bottom Center	Toggles the animation state, allowing you to stop and inspect the current state of the puzzle.
End Game	Bottom Center	Stops the current simulation and returns to the main menu.
Metrics	Top Left	Toggles an overlay displaying the performance metrics (Path Length, Nodes Explored, Time) for the currently running algorithm.
Volume Slider	Top Right	Controls the volume of the background audio.

4.4. Visual Elements

The simulation screen displays the following elements:

- **River:** The central blue area.
- **Banks:** The green areas on the left (Start) and right (Goal).
- **Characters:** Yellow circles represent **Missionaries** and orange circles represent **Cannibals**.
- **Boat:** The brown rectangle used to transport characters across the river.

The animation will proceed automatically, showing the boat moving back and forth until all characters have successfully crossed the river to the goal state. If the algorithm finds a solution, the animation will stop when the goal is reached.