Nonogram Solver Implementations

This repository contains a set of Python implementations for solving and visualizing Nonograms using techniques such as backtracking and line-solving. It includes visualization of the solving process and support for configurable Nonogram puzzles through constraints.

Features

- **Backtracking-based Nonogram Solver**:
  - Implements a constraint satisfaction problem (CSP) approach with
  - heuristics like Minimum Remaining Values (MRV) and forward checking.
  - Outputs the solving process as a visual animation in MP4 format.
- **Line Solver**:
  - Solves Nonograms using domain consistency and line-by-line processing.
  - Provides visualization for solving steps.
- **Interactive Visualizations**:
  - Displays solving steps using Matplotlib animations.
  - Supports custom clue input for both rows and columns.

Prerequisites

To run the scripts, ensure you have Python installed (version 3.7 or later). The following

Python libraries are required:

- numpy
- matplotlib
- tqdm
- opencv-python

Install the dependencies using pip: *pip install numpy matplotlib tqdm opencv-python*

File Descriptions

1. Backtracking Algorithm Implementation.py

This script implements a CSP-based Nonogram solver with the following steps:

- Initializes the grid and domains based on row and column constraints.
- Applies heuristics like MRV and forward checking to reduce the search space.
- Saves the solving process as a video.

Usage

Run the script directly:

```
bash
Copy code
python Backtracking Algorithm Implementation.py
```

It includes sample constraints, but you can modify `row_constraints` and `col_constraints` for custom puzzles.

2. Line Solver Visualization.py

This script focuses on line-solving techniques without backtracking. It uses arc consistency to deduce the puzzle's solution iteratively.

Usage

Run the script:

```
bash
Copy code
python Line Solver Visualization.py
```

3. Line Solver Implementation.ipynb

A Jupyter Notebook providing interactive exploration of Nonogram solving techniques. Useful for debugging and step-by-step analysis.

Usage

Open the notebook:

```
bash

Copy code

jupyter notebook Line Solver Implementation.ipynb
```

Example Configuration

For each script, you can customize the puzzle by modifying the row_clues and col_clues. Example clues for a 5x5 grid:

```
row_constraints = [
    [1, 1], [1, 2], [2], [5], [1]
]
col_constraints= [
    [1, 1], [1, 2], [3], [2, 1], [1, 1]
]
```

Output

- **Backtracking Solver**:
  - Final solution printed in the console.
  - Video saved as nonogram_solution.mp4.
- **Line Solver**:
  - Solution grid displayed in a Matplotlib animation.