Project Overview

This project provides a framework for performing random search, simulated annealing and improved simulated annealing on configurable systems and visualising the results. The tool allows users to evaluate configurations across different datasets, identify optimal configurations, statistically analyse results and generate visualisations to aid in performance analysis.

Features

- Random search, standard simulated annealing, improved simulated annealing for exploring configurations in datasets.
- Automatic handling of maximisation and minimisation problems.
- Visualisation of search results with performance trends and optimal points.
- Statistical analysis of result data.
- Clear organisation of datasets, search results, and visualisation outputs.

Installation

1. Clone the Repository

```
git clone https://github.com/Co11eague/SAForConfigurationTuning.git
```

2. Install Dependencies

Ensure `pip` is up-to-date and install required packages:

```
pip install --upgrade pip
pip install -r requirements.txt
```

Dataset Description

The `datasets` folder contains CSV files representing different configurable systems. Each CSV file has the following structure:

- **Columns (1 to n-1):** Configuration parameters (discrete or continuous values).
- **Column n:** Performance objective (a numeric value).

Included Systems

System	Optimization Type
7z	Minimisation
Apache	Minimisation
Brotli	Minimisation
LLVM	Minimisation
PostgreSQL	Minimisation
Spear	Minimisation
Storm	Minimisation
x264	Minimisation

- **Minimisation Problems:** Lower performance values are better.
- **Maximisation Problems:** Higher performance values are better.

Usage

Find out what settings are needed

Type in the budget, temperature and other settings in the code before running the program. Each program file has `runs` value in the main function.

- If `runs` is equal to one, only results for one run are displayed in the console.
- If `runs` is anything greater than one, it runs the algorithm for the specified amount of times on each system, finally displaying results as `best_performances.csv` in `search_results` folder. In this case the `visualize_search_results.py` will only display the results of the last run.

1. Perform Search

Random Search

Run the RS.py script to perform random search on all datasets:

```
python RS.py
```

Simulated Annealing Search

Run the SA.py script to perform simulated annealing search on all datasets:

```
python SA.py
```

Improved Simulated Annealing Search

Run the ISA.py script to perform improved simulated annealing search on all datasets:

```
python ISA.py
```

The search results for each dataset will be stored in the `search_results` folder as CSV files.

2. Visualize Search Results

Run the visualisation script to generate performance plots for all datasets:

```
python visualize_search_results.py
```

The visualisations are automatically generated and stored in the `visualisation_results` folder after running script. Each dataset's visualisation shows:

- Performance trends across search iterations.
- The optimal performance point highlighted in red *.

3. Statistically analyse result data

Run the statistical analysis script to analyse the data from two different algorithms in specific systems:

```
python statistically_process_results.py
```

Make sure to change the variables in the script to reflect what you are analysing.

- Each algorithm has its own separate CSV file, where each column is a system (see examples in data folder) and each row is a different run.
- To analyse different systems you need to change variable `system` to the name of the system you are comparing.
- You can also switch analysis for minimisation and minimisation with `MAXIMISATION` flag.

4. Execute via IDE (Optional)

If you prefer using an IDE like PyCharm, you can:

- Run `RS.py` to execute the random search process.
- Run `SA.py` to execute the simulated annealing search process.
- Run `ISA.py` to execute the improved simulated annealing search process.
- Run `visualize_search_results.py` to generate visualisations of the results.
- Run `statistically_process_results.py` to statistically analyse result data.

Project Structure

```
project-folder/
├── datasets/               # Contains input datasets (CSV files).
├── data/                   # Used for storing result data
├── search_results/         # Stores search results after running the script.
├── visualisation_results/  # Stores visualisations of search results.
├── ISA.py                  # Script to run improved simulated annealing search and generate results.
├── SA.py                   # Script to run simulated annealing search and generate results.
├── RS.py                   # Script to run random search and generate results.
├── requirements.pdf        # Project dependencies requirements.
├── manual.pdf              # Project manual.
├── replication.pdf         # Project manual for replicating results.
├── visualize_search_results.py  # Script for generating visualisations.
├── statistically_process_results.py  # Script for statistically processing result data.
└── requirements.txt        # Python dependencies.
```

Notes

- For non-existing configurations during the search, the tool assigns:
    - **Minimisation Problems:** Twice the maximum performance value in the dataset.
    - **Maximisation Problems:** Half the minimum performance value in the dataset.
- Ensure datasets are formatted correctly with valid configuration and performance columns.