Multi-Agent Path Finding

Using the simulated annealing technique

Problem Solving and Search in Artificial Intelligence

Summer Semester 2024: Assignment 2

Daria Stefan & Olanrewaju Ajibua

Algorithm Description

- Overview:
  - Simulated Annealing is used for finding an approximate solution to the optimization problem.
  - It involves exploring the solution space by probabilistically accepting changes that improve the objective function and, occasionally, changes that do not, to escape local optima.
- Key Components:
  - Agent: Represents entities within the algorithm.
  - SimulatedAnnealing: Executes the optimization process.
  - Visualization: Plots the results.

- Algorithm Workflow

- Initialization: Set initial temperature and parameters.
- Iterative Optimization: Perform optimization over a set number of iterations.
- Temperature Adjustment: Gradually decrease the temperature to refine the search.
- Result Storage: Save and prepare results for visualization.

Experiments and Selection of Parameters

Key Parameters:

- Initial Temperature
- Cooling Rate
- Number of Iterations

Tuning Methodology:

- Empirical testing with various parameter sets.
- Selection based on performance metrics such as convergence speed and solution quality.

Example Parameter Sets:

- Initial Temperature: 1000, 500, 100
- Cooling Rate: 0.95, 0.90, 0.85
- Iterations: 10000, 5000, 1000

Automated Algorithm Configurator

Utilizes a grid search approach to optimize parameters

Parameter Ranges:

- Initial Temperature: From initialTempMin to initialTempMax
- Cooling Rate: From coolingRateMin to coolingRateMax
- Maximum Iterations: From maxIterationsMin to maxIterationsMax

Grid Search:

- Systematically explores combinations of parameters in defined steps.

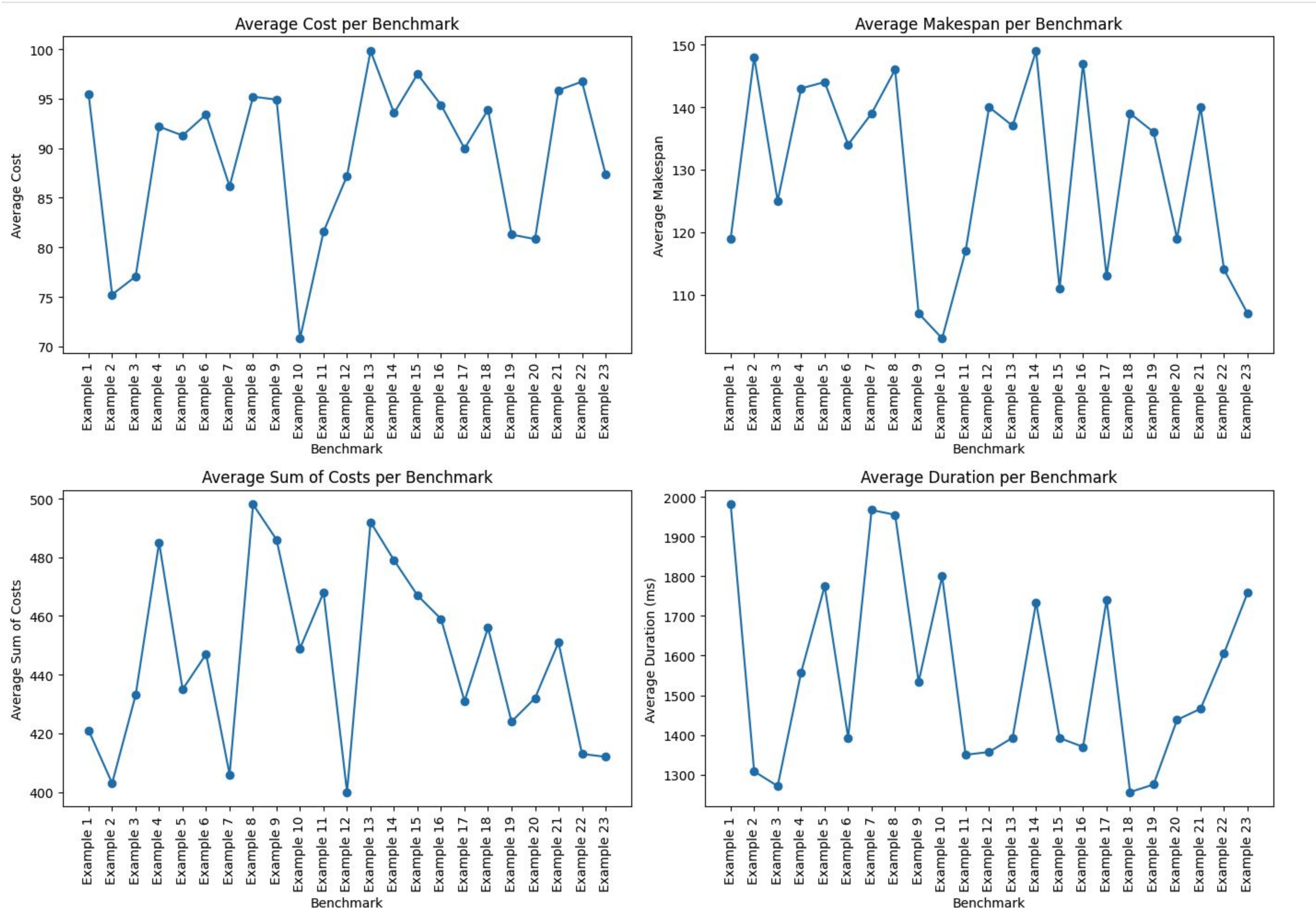
Performance Metrics:

- Evaluates configurations based on cost, makespan, sum of costs, and execution duration.

Best Configuration Selection:

- Selects the best parameter set based on performance metrics.

Results for Benchmark Examples



Conclusions and Lessons Learned

- Summary of Findings:
  - The algorithm performs well on a variety of benchmarks, consistently finding near-optimal solutions.
- Performance Insights:
  - Higher initial temperatures help explore the solution space more thoroughly.
  - A slower cooling rate allows for better refinement but requires more iterations.
- Strengths and Weaknesses:
  - Strengths: Effective at escaping local optima, adaptable to various problems.
  - Weaknesses: Performance highly dependent on parameter settings.

Instructions on Running the Program

- Prerequisites:
  - C++ compiler (e.g., g++)
  - Simple and Fast Multimedia Library (SFML) for visualization

- Compilation Steps:
  - make clean
  - make
  - ./mapf_visualizer <filename> <configfile>

DEMO

Thanks for your Attention!