Robust Uncapcaitated Facility Location Solver

This project implements methods for solving Uncapcaitated Facility Location(UFLP) nominally and under data uncertainty.

Can be used for solving the UFLP approximately and to optimality and to produce robust solutions to the UFLP under spatial demand uncertainty.

Description

The project implements two methods for solving the UFLP a Local Search Algorithm and an Optimization model written in Pyomo. The optimization model is configured to run with CPLEX but can be reconfigured for use with any MIP solver with Pyomo Interface.

The Robust Solver implements Cardinality Constrained Robustness, where each demand node in the UFLP has an uncertain interval. It solves the problem with an input parameter gamma which corresponds to the number of demand nodes taking on their worst case realization within their demand interval. The solution is produced by solving nominal instances of the UFLP to optimality using the Pyomo model.

Getting Started

Dependencies

- Python 3.8.8 +
- Pyomo 6.1.2 +
- IBM ILOG CPLEX Interactive Optimizer 20.1.0.0 +
- Anaconda *(optional)*

Recommended Installation

- Install the latest Anaconda distribution https://www.anaconda.com/
- From the Anaconda CLI install Pyomo with `conda install -c conda-forge pyomo`
- If using CPLEX for the optimization model install from https://www.ibm.com/analytics/cplex-optimizer
- If using another Optimization solver with a Pyomo interface update the SolverFactory object parameters in
    - optimizationModel/solver.py
    - robustModel/robustModelVaryGamma.py
    - robustModel/worstCaseRealizations.py
    - robustModel/robustSingleSolve.py

Each folder contains a single dataset to run and produce output on. A more detailed exploration can be done by running on instances in the /Data directory All algorithms can accept datasets in the input format specified in https://resources.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/data-format.html

Running Local Search

The Fast Local Search implementation can be found in /localSearch/localSearchFast.py This is the fastest implementation and recommended for solving purposes

Iterated Local Search

Runs Iterated Local Search to solve UFLP instance. Outputs csv file to same directory containing Best value obtained over iterated descents to local optimum

- Navigate to /localSearch
- Execute following command in CLI

```
python iteratedLocalSearch.py
```

- When prompted by file dialog select the file ga250a-1 (or files from data folder) from same directory

Compare Local Searches

Compares running time between the Fast and Slow Local Search implementations. Outputs four csv files containing total running time and process time at each neighbourhood step in same directory

- Navigate to /localSearch/compareLocalSearches
- Execute following command in CLI

```
python compareLocalSearches.py
```

- When prompted by file dialog select the file ga250a-1 (or files from data folder) from the same directory
- CSV files will be saved in /localSearch/compareLocalSearches

Average Descent Time

Calculates the average descent time to local optimum using fast procedure

- Navigate to /localSearch
- Execute the following command in the CLI

```
python averageDescentTime.py
```

- When prompted by file dialog select ga250a-1 (or files from data folder) from same directory
- CSV files will be saved in /localSearch

Running Optimization Solver

Solves UFLP Data set using Pyomo model Outputs CPLEX Log to same directory as a txt file

- Navigate to /optimizationModel
- Execute the following command in the CLI

```
python solver.py
```

- When prompted by file dialog select either 3011EuclS.txt/ga250a-4 (or others from data folder) from same directory
- Txt files will be saved in /optimizationModel

Running Robust Model

Single Robust Solver

Produces a single robust solution with a specified number of demand nodes at their worst case realizations (75% by default) Output is a single text file containing Robust Solution Objective Value

- Navigate to /robustModel
- Execute the following command in the CLI

```
python robustSolverSingle.py
```

- When prompted by file dialog select 111EuclS.txt from the same directory
- txt file will be saved in /robustModel

solves m instances of the UFLP using Pyomo model

Robust Solver Multiple Gamma Values

Solves M robust solutions varying the number of worst case realizations from 0 to M where M is the number of customer nodes. Output is a csv file containing entries in the form wc,robust solution value and a text file containing the solution differential for all these values

- Navigate to /robustModel
- Execute the following command in the CLI

```
python robustModelVaryGamma.py
```

- when prompted by file dialog select 111EuclS.txt from the same directory
- csv files outputted to /robustModel

  *Note this file takes ~6 hours to run to completion since it needs to solve m*m instances of the UFLP*

Worst Case Realizations

Solves instances of UFLP problem with worst case realizations with varying percentage deviations. Solves for both interval and box uncertainty.Outputs are two csv files interval.csv and box.csv. Each file contains entries with the percentage deviation and worst case realization value.

- Navigate to /robustModel
- Execute the following command in the CLI

```
python robustWorstCaseRealizations.py
```

- when prompted by file dialog select 111EuclS.txt from the same directory
- csv files outputted to /robustModel

Data

Contains extracts from the two data sets from the Max Planck Institute For Informatics. Data is Separated into folders according to which section of the report they are used in. Access raw data at https://resources.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/packages.html

- Euclidian Benchmarks
- Koerkel-Ghosh Asymmetric Benchmarks

Data folder contains data sets and their optimal/best known solutions https://resources.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/data-format.html

- Data sets are in .txt or FILE formats
- Optimal & Best Known solutions are in .opt & .bub formats

Results

Contains results used for data analysis in report. Separated into folders according to which section of report results used in. Can be reproduced by running algorithms described specified in report on the corresponding data

Authors

Harsha Ramachandran

Version History

- 1.0
    - Initial Release