Travelling Salesman Problem using Integer Programming

This repository contains code to find solution of Single Vehicle Pickup and Delivery travelling salesman problem (1-PDTSP) using integer programming for any list of cities. 1-PDTSP problem is, given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city also fulfilling demands of each node.?

1-PDTSP MILP Formulation

Let variable $x_{i,j}$ is a binary variable denoting if edge connecting i, j is part of optimal solution (1) or not (0). Number of these variables is equal to total number of edges.

$$x = \{1 \quad \text{if edge } (i,j) \text{ is part of optimal solution } 0 \quad \text{if edge } (i,j) \text{ is not part of optimal solution}$$

Another variable u will track edge number, it starts from 1 and goes till number of nodes. Number of these variables is equal to total number of nodes.

Let $|N|$ denotes total number of nodes and $|E|$ denotes total number of edges.

$$u \in 1, 2, 3, \ldots, |N|$$

Let $d_{i,j}$ denote distance between cities i and j

Model Formulation

$$min \sum_{i=1}^{|E|} \sum_{j=1}^{|E|} d_{i,j} x_{i,j}$$

$$s.t. \quad \sum_{i=1}^{n} x_{i,k} = 1 \ \ \forall k \in 1, 2, 3...., |E|, i \neq j \tag{1}$$

$$\sum_{j=1}^{n} x_{k,j} = 1 \ \ \forall k \in 1, 2, 3...., |E|, i \neq j \tag{2}$$

$$u_1 = 1 \tag{3}$$

$$u_i - u_j + 1 \leq (n-1)(1 - x_{i,j}) \ \ \forall (i,j) \in 1, 2, 3...., |E|^2, i \neq j \tag{4}$$

$$u_i \geq 2 \ \ \forall i \in 2, 3...., |E| \tag{5}$$

$$u_i \leq |N| \ \ \forall i \in 2, 3...., |E| \tag{6}$$

$$\sum_{i=1}^{n} f_{i,k} - \sum_{j=1}^{n} f_{k,j} = q_i \ \ \forall k \in 1, 2, 3...., |E|, i \neq j \tag{7}$$

$$0 \leq f_{i,j} \leq Q x_{i,j}, \ \ \forall i \in E \ \ \forall (i,j) \in 1, 2, 3...., |E|^2 \tag{8}$$

- $constraint(1)$ denotes from each node there is one incoming edge.

- $constraint(2)$ denotes from each node there is one outgoing edge.
- $constraint(3)$ denotes when $x_{i,j} = 1$ then $u_j = u_i + 1$ ie. j^{th} edge will be labelled 1 more than previous label (i^{th}) label if edge $x_{i,j}$ is selected. This constraint denotes ordering of each node in optimal solution. This constraint is not considered for last edge of path which connect last node with first node to create a cycle.
- $constraint(4)$ denotes first node ordering is always 1
- $constraint(5)$ and $constraint(6)$ are bound constraints on u.
- $constraint(7)$ and $constraint(8)$ are demand constraints and bound constraints on f respectively.

Features

- Provide method to get city details, latitude and longitude information from openstreetmap API using which distance between these cities is calculated using Haversine formula.
- Provide a formulation of TSP using pyomo that can be solved by any MILP solver supported by pyomo.

Installing dependencies

The command below will install all the required dependencies python from `requirements.txt` file.

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

We also need to install a MILP solver. This project uses `cbc` solver which is an open-source mixed integer linear programming solver written in C++.

Linux

- using apt package manager

```
apt install -y -q coinor-cbc
```

- using pacman package manager

```
sudo pacman -S  coin-or-cbc
```

Windows

Download binary from github release and install.

MacOS

```
brew install coin-or-tools/coinor/cbc
```

Documentation

- Input Parameters for a script can be set by using config file.
- A test config file is provided named `config.txt` it uses json formatting.

Create Data Files

- Data file to run 1-PDTSP on cities can be provided or can be created using `get_cities_data.py`.
- A `.txt` file will be needed by script that contains name of the cities. A default `cities.txt` is provided.

Create data for cities

```
python get_cities_data.py --names=cities.txt --save_folder=/data/
```

Arguments

- `--name` or `-n`: provide path to name of cities `.txt` file. (required)
- `--save_folder` or `-s` provide path to folder where data will be saved in csv. It will save two files in that folder named `cities_data.csv` or `cities_distances.csv`. (not required)

Run 1-PDTSP on cities

- 1-PDTSP can be run on files that we get from `get_cities_data.py` script or data in similar format generated by this script. Test data is provided inside `\data\` folder.
- A `config` file will be needed by script that contains settings related to run script. A test config file is provided as `config.txt`

Run 1-PDTSP on cities

```
python pdtsp_1.py --config=config.txt
```

Arguments

- `--config` or `-c`: provide path to config file. (required)

Some Links

- **Results on 34 cities of Maharastra Demand 1**
- **Results on 34 cities of Maharastra Demand 2**
- **Results on 34 cities of Maharastra Demand 3**
- **Results on 34 cities of Maharastra Demand 4**

References

- Hernández-Pérez and Salazar-González, 2003 Hernández-Pérez, H. and Salazar-González, J.-J. (2003). The one-commodity pickup-and-delivery travelling salesman problem. In Combinatorial Optimization

—Eureka, You Shrink! Papers Dedicated to Jack Edmonds 5th International Workshop Aussois, France, March 5–9, 2001 Revised Papers, pages 89–104. Springer.

- Mosheiov, 1994 Mosheiov, G. (1994). The travelling salesman problem with pick-up and delivery. European Journal of Operational Research, 79(2):299–310.
- Stein, 1978 Stein, D. M. (1978). Scheduling dial-a-ride transportation systems. Transportation Science, 12(3):232–249.