Optimal travel planner

Nina Kaploukhaya
Optimization Class Project. MIPT

Introduction

The Traveling Salesman Problem (TSP) and its generalization the Vehicle Routing Problem (VRP) are two of the most popular problems in the field of combinatorial optimization. If we want to plan an optimal trip, first thing that we should do is to solve this problem. It can be done by several methods. By building an objective function and adding constraints corresponding to a problem, we can solve it like integer linear programming problem [3]. Using this and Google Maps API interface, we make a program that shows an optimal tourist route based on user's preferences, limit of time and location.

Methods

Many methods are used to solve this problem. In this work we implemented simulated annealing, genetics algorithm [7], branch and bound method ([4], [6]).

Integer linear programming approaches for this task

For the first version of Travel planner, that based on classic tsp problem we introduce this objective function and constraints:

$$\min_{x_{ij}\in\{0,1\}} \sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ij}$$
$$\sum_{j=0,j\neq i}^{n} x_{ij} = 1$$
$$\sum_{i=0,i\neq j}^{n} x_{ij} = 1$$
$$y_i - (n+1)\cdot x_{ij} \geq y_j - n$$

The second version solves a more difficult task. We have a dataset of tourist attractions and user wants to visit as much as possible from it in limited time and, of course, he prefers places with higher rating. Also, there is no point to visit places, that are already closed. To solve this, we add rating depended part in objective function. In order to make rating dependent part be competitive with distance dependent part, one point of rating upon 3 equals $2 * c.mean()$, and equities in constrains changes to equities (so we should also add additional constraints to make sure that we exit each entered place). And add additional constrained to visit only opened places.

$$\min_{x_{ij}\in\{0,1\}} \sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ij} - 2 * av(c)\sum_{i=0}^{n}(rating_i - 3) * \sum_{j=0}^{n} x_{ij} \quad (1)$$

$$\sum_{j=0}^{n} x_{ij} <= 1, \;\; \sum_{i=0}^{n}\sum_{j=0}^{n}(x_{ij} - x_{ji}) = 0 \quad (2)$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n}((c_{ij} + \text{speed} * 500)/(\text{speed} * 1000))) * x_{ij}) <= \text{Limited Time} \quad (3)$$

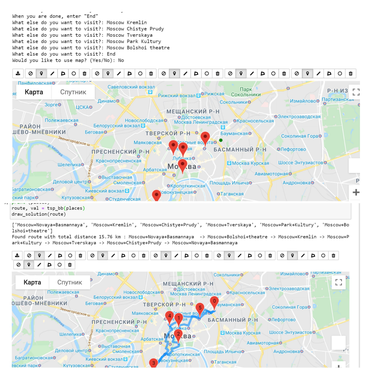
Results of method performance

On this numerical experiments, conducted on "starbucks uk" dataset of gmaps for different number of items ($5/15/30$) and different samples, average performances of method are (value in table corresponds to value of this method solution (distance) / min (solution) from all methods - 1):

	Methods		
Number of items	SA	GA	branch and bound
5 items	0.0008	0	0
15 items	0.2512	0.002	0.0005
30 items	0.4	0.001	0.024

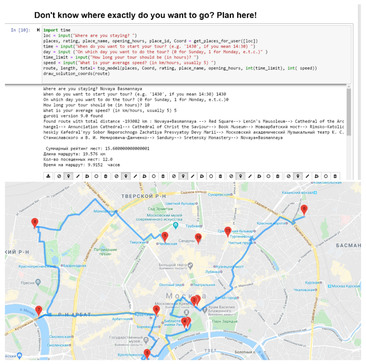
Results

Project code: [1].
Our first version of the travel planner asks user which places he wants to visit (he can type them or choose on map) and builds an optimal route between these places using different methods:



Work of travel planner.1

Our second version asks for user's start/end location, limitation of his time, his average speed, current time/day of week and suggests an optimal route that visits the most of opened tourist attractions with highest rating in the limited$_time$.



Work of travel planner.2

Conclusion

SA, GA, BB methods were analyzed on solving TSP problem for the dataset, GA and BB showed more stable results than SA on big amount of items. Solutions of this method were visualised with Google Maps API. Based on this ravel planner was successfully created.

Educational value of the project

- Learn how to work with Google Maps API
- Train to use simulated annealing, genetics algorithm, branch and bound method in practice and analyze their performances
- Learn how to create optimization models in python, using Gurobi and MIP solver
- Study different approaches to tsp problem and vehicle routing problem

Acknowledgements

This material is based upon works, stated in references and on advice of Merkulov Danya.

References

[1] My project
https://github.com/NinaOwl/OptProjects/blob/master/Travel_Planner.ipynb

[2] Google Maps Api documentation,
https://developers.google.com/maps/documentation/

[3] MPI documentation
https://mpi4py.readthedocs.io/en/stable/

[4] Branch and bound method
https://www.lix.polytechnique.fr/liberti/teaching/dix/inf431-11/bb-notes.pdf

[5] MPI examples
https://python-mip.readthedocs.io/en/latest/examples.html

[6] Branch bound method in MPI
https://iopscience.iop.org/article/10.1088/1742-6596/256/1/012018/pdf

[7] Genetics algorithm
https://www.sciencedirect.com/science/article/pii/S1877050918306100