TRAVELLING SALESMAN PROBLEM MODELLING BY MIXED INTEGER LINEAR PROGRAMMING OF PYTHON (MIP)

Pham My Hanh

An Giang University, Vietnam National University, Ho Chi Minh City, Vietnam

Email: pmhanh@agu.edu.vn

Article history

Received: 14/3/2022; Received in revised from: 22/6/2022; Accepted: 14/7/2022

Abstract

A famous travelling salesman problem, appearing simple to state but complex to solve, has been widely investigated and various algorithms have been proposed. In this article, mixed integer linear programming of python (MIP) is used to model this problem with varying input data. The result shows that with small input data the modelling code of MIP executing quickly and converging to optimal value, while large scale input data require plenty of computation time; thereby algorithm improvement as well as parallel implementation are suggested.

Keywords: *Mixed integer linear programming, python, travelling salesman problem.*

MÔ PHỎNG BÀI TOÁN NGƯỜI BÁN HÀNG BẰNG QUY HOẠCH TUYẾN TÍNH HỖN HỢP NGUYÊN (MIP) CỦA PYTHON

Phạm Mỹ Hạnh

Trường Đại học An Giang, Đại học Quốc gia Thành phố Hồ Chí Minh, Việt Nam

Email: pmhanh@agu.edu.vn

Lịch sử bài báo

Ngày nhận: 14/3/2022; Ngày nhận chỉnh sửa: 22/6/2022; Ngày duyệt đăng: 14/7/2022

Tóm tắt

Bài toán người bán hàng là một bài toán nổi tiếng vì nó được trình bày đơn giản nhưng lời giải thì thật phức tạp. Bài toán này đã thu hút sự nghiên cứu của đông đảo nhà khoa học và nhiều thuật toán đã được đề xuất. Trong bài viết này, tác giả sử dụng phần mềm MIP (quy hoạch tuyến tính trên tập số nguyên) được viết bởi ngôn ngữ lập trình python để giải quyết bài toán với các kích cỡ khác nhau của dữ liệu đầu vào. Kết quả cho thấy đối với dữ liệu nhỏ thì thuật toán hội tụ khá nhanh về giá trị tối ưu, tuy nhiên với dữ liệu đầu vào lớn, khối lượng bước tính nhiều, cần sự cải tiến về mặt thuật toán và áp dụng tùy chọn tính toán song song.

Từ khóa: *Bài toán người bán hàng, quy hoạch tuyến tính hỗn hợp nguyên, python.*

DOI: https://doi.org/10.52714/dthu.11.5.2022.977

Cite: Pham, M. H. (2022). Travelling salesman problem modelling by mixed integer linear programming of python (MIP). *Dong Thap University Journal of Science*, *11*(5), 29-34. https://doi.org/10.52714/dthu.11.5.2022.977.

1. Introduction

The famous travelling salesman problem (TSP) can be stated simply that a salesman starting at his home city wants to visit to other *n*-1 different cities all at once and return back to his original position. He knows the distance between two arbitrary cities, so which path he should follow to achieve the shortest distance tour in which sub-tours are not allowed. This is a compounded problem that has many practical applications, attracting numerous researchers' interest so far. In addition, some following practical applications of this problem can be listed, for instance shipping company has to figure out an optimization route when delivering goods to customers at different locations in order to save time and fuel cost; a school bus driver has to consider the most appropriate way to pick up pupils; an airline has to set up commercial and sufficient flight route throughout *n* cities… TSP is also a particular case of travelling purchaser and vehicle routing.

In graph theory, this problem leads to finding the Hamiltonian cycle through *n* vertices. The graph presents here is a weight graph with *n* cities presented by *n* vertices and the edge connects two vertices having weight, which denotes the distance between two cities. Two versions of this problem are asymmetric TSP and symmetric TSP, depending on whether the graph is digraph or simple graph. This problem has been proved NP-complete as *n* grows to infinity, where the computation iterations needed might reach 2^n. This problem may not have exact optimal solution but feasible ones (see Dantzig, 1954; Hoffman *et al.*, 2013 for details). Thereby, many heuristics and exact algorithms have been proposed to find feasible solutions, especially linear programming is deeply concerned.

In linear programming model, simplex algorithm has long been used; however if the input values are integer, the model will be more complicated and subdivided into three following major types:

+ Integer model has its decision variables belong to integer set.

+ Binary integer model whose decision variables are binary, having value either 0 or 1.

+ Mixed integer linear is a linear programming where its decision variables belong to both integer and real number sets.

To solve TSP, plenty of heuristic methods have been investigated so far. Ant colony optimization methods and its improvement were presented by Munkres, 1957; Dorigo & Gambardella, 1997; Chawda & Sureja, 2012. Meanwhile, nearest neighbour algorithm was introduced by Dhakal and Chiong, 2008. Especially, branch and cut algorithm was proposed by Padberg & Rinaldi (1991) to solve symmetric TSP by using FORTRAN to find the incidence vectors of a Hamiltonian cycle. The principle of this method is firstly solving linear program without the integer constraint by using the simplex algorithm to obtain an optimal solution then applying a cutting plane algorithm to reach all feasible integer points with have optimal value.

Besides some powerful computing libraries written in FORTRAN and C++, Python has plenty of efficient packages applied for modelling and optimization study. PuLP library was presented in Mitchell et al. (2011). Pyomo package was thoroughly described in Hart et al. (2017). CVXOPT package for convex optimization was proposed by Diamond et al. (2016). In addition, Linderoth & Lodi (2010) presented some major components of mixed integer linear programming solver of Python. More precisely, python computational tools have been used by Asani et al. (2020) to solve TSP by applying the convex-hull and nearest neighbor heuristic algorithm to construct a tour technique. As a consequence of the rapid development of algorithm and software, MIP (mixed integer linear programming of python) is one of the recent efficient packages of python used in optimal computation.

In this article, TSP investigated hereafter is symmetric, modeled by MIP. The aim of this article is to present a modelling result of TSP by different size of the vertex set *n*, then proposing some recommendations to use MIP for TSP. More precisely, in the next section, the general mathematical formulation of this problem is stated and an overview of the branch and cut algorithms as well as MIP basic program and finally modelling results of TSP are presented.

2. Mathematical formulation of TSP, mixed integer linear programming, branch-and cut algorithm and modelling results of TSP by MIP

2.1. Mathematical formulation of TSP

The mathematical formulation of symmetric TSP can be generally presented as follows.

Let n and c_{ij} denote the number of cities and the distance between city i to city j respectively.

Let A be a set of edge from vertex i to vertex j.

$x_{ij} = 1$ if the salesman travels from city i to city j and $x_{ij} = 0$ otherwise.

y_{ij} denotes the flow from vertex i to j.

TSP can be formulated as:

- Objective function: Minimize $\sum_{(i,j)\in A} c_{ij} x_{ij}$.
- Constraints:

$\sum_{\substack{i=0 \\ i\neq j}}^{n} x_{ij} = 1 \ (0 \le j \le n)$ since the salesman has to travel to each city i.

$\sum_{\substack{j=0 \\ j\neq i}}^{n} x_{ij} = 1 \ (0 \le i \le n)$ since the salesman must leave for another city after visiting j.

$\sum_{\substack{j=0 \\ i\neq j}}^{n} N_{ij} y_{ij} = 1, \ i = 0,...,n$ to prevent sub-tours, where each column of N_{ij} denotes the flow variable y_{ij} in arc (i, j).

$$y_{ij} \le n x_{ij} \quad \forall (i, j) \in A,$$
$$y_{ij} \ge 0 \quad \forall (i, j) \in A,$$
$$x_{ij} \in \{0,1\} \quad \forall (i, j) \in A.$$

2.2. Mixed integer linear programming (MIP) solver and its basic components of python

Mixed integer linear programming (MIP) is an efficient collection of Python tools to model, especially for mixed integer optimal problems. MIP, originally written in modern and typed Python, works with the Python compiler Pypy. MIP can solve large-scale problems with more complicate and computationally intensive methods like simplex method or the branch-and-cut-method and their variants.

In general, MIP consists of these following basic components. Firstly, in the presolving phase, it detects some necessary changes of the input to improve the solution process in next phase. Secondly, cutting plane process strengthens approximation iterations especially in a convex hull. Then, branching strategies are established at either node selection or variable selection. Its two final stages are primal heuristics and parallel implementation.

Additionally, a python MIP modelling code consists of these main parts as follows.

- After inputting data and implementing MIP packages, an initial step is creating a model, in which it can be an empty model and minimize or maximize mode can be selected. More precisely, TSP problem is used as an example.

```
model = Model()
```

- Including variables,

```
x = [[model.add_var(var_type=BINARY) for j in
V] for i in V]
y = [model.add_var() for i in V]
```

- Adding objective function,

```
model.objective = minimize(xsum(c[i][j]*x[i][j] for
i in V for j in V))
```

- Adding all constraints of the model,

```
# constraint: leave each city only once
for i in V:
   model += xsum(x[i][j] for j in V - {i}) == 1
# constraint: enter each city only once
for i in V:
   model += xsum(x[j][i] for j in V - {i}) == 1
# subtour elimination
for (i, j) in product(V - {0}, V - {0}):
   if i != j:
      model += y[i] - (n + 1)*x[i][j] >= y[j] - n
```

- Executing the model,

```
model.optimize()
```

- Checking if feasible solutions are found and writing them out the screen,

```
# checking if a solution was found
if model.num_solutions:
   out.write('route with total distance %g
```

```
found: %s' % (model.objective_value, place[0]))
nc = 0
while True:
    nc = [i for i in V if x[nc][i].x >= 0.99][0]
    out.write(' -> %s' % place[nc])
    if nc == 0:
        break
out.write('\n')
```

After executing MIP, the optimal results can be some following status.

- Optimal: if the optimal solution is found;

- Feasible: if a feasible solution is acquired but the program cannot check whether or not this is an optimal one;

- No solution found: if no solution is achieved;

- Infeasible: If there is no feasible solution after execution;

- Unbounded: If it lacks constraints;

- Error: If there are some errors occurring while executing;

Besides, if a truncated execution is performed, it will stop due to the time limit and no feasible solution is found.

(Source of MIP)

2.3. The branch-and-cut-method used in MIP

In general, branch and cut algorithm as a combinatorial optimization method is used to solve linear programs whose input values or constraints are restricted to integer set. This algorithm combines two components which are branch and bound algorithm and cutting plane technique.

Initially, simplex algorithm is used to solve the linear program without integer constraints. When the optimal value is obtained, which might not be integer value, a cutting plane algorithm is executed to search for all feasible integer points. The non-integer solutions play a role as upper bounds and integer solutions as lower bounds. If the upper bounds are less than the lower, then one node can be pruned. When solving the linear programming relaxations, additional cutting planes may be generated. These cutting planes can be either global cuts or local cuts, then the branch and bound part of the algorithm is selected. During this process, the algorithm searches for all candidate solutions of the feasible space. The set of candidate solution forms a rooted tree, and the algorithm explores the branches of the tree. Obviously, the tree is a subset of the solution set. During the search of the branches of tree, the candidate solution can be discarded if it does not provide a better solution than the previous obtained one. The algorithm for maximization objective function can be briefly stated as follows.

1) Set up list of active problems denoted by L. Initially assuming that the solution $x^* = null$ and objective value $v^* = -\infty$.

2) While L is not empty, select or remove the queued problems.

i) Solve the linear programming relaxation of the problem.

ii) If the solution is infeasible, go back to 2) (while loop), otherwise to denote the solution by x and objective value by v.

iii) If $v \le v^*$ then go back to 2).

iv) If x is an integer then $v^* = v, x^* = x$. Move to 2).

v) If desired solution is found, then searching for the cutting plane violated by x. If they are found, then add them to the linear programming relaxation and return to 2.i.

vi) Branching and partitioning the problem into new problem with restricted feasible regions. Adding these to L and return 2).

3) Returning the solution x^* and the objective value v^*.

Source Wikipedia.org

2.4. Modelling results of TSP by MIP and some remarks

In this article, the source data of TSP (retrieved from https://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.html) is selected and MIP is applied to model symmetric TSP with various size of vertex set. The smallest input data is 5 vertices and the largest is 26 vertices. The other maximal size TSP data of this link is 42 vertices and 48 vertices or more are not chosen because of the limit of computational equipment.

Let denote M_i the city i of the travelling route. It can be also considered as a vertex i in a simple graph with n vertices when modelling. In addition,

the vertex M_1 is the starting point of the salesman's journey. The modelling purpose is to find a Hamiltonian cycle through *n* vertices. In this context, TSP is modeled by python 3.9, with personal computer Intel(R) Core(TM) i3-7100U CPU, ram 4.00GB. The obtained results are as follows.

Table 1. Modelling results of Travelling salesman problem by MIP

Vertices set (*n*)	Optimal value	Routes	CPU time (seconds)
5	19	M_1 -> M_3 -> M_2 -> M_5 -> M_4 -> M_1	0.01
15	291	M_1 -> M_{11} -> M_4 -> M_6 -> M_8 -> M_{10} -> M_{14} -> M_{12} -> M_3 -> M_7 -> M_5 -> M_9 -> M_{15} -> M_2 -> M_{13} -> M_1	0.68
17	2085	M_1 -> M_{16} -> M_{12} -> M_9 -> M_5 -> M_2 -> M_{10} -> M_{11} -> M_3 -> M_{15} -> M_{14} -> M_{17} -> M_6 -> M_8 -> M_7 -> M_{13} -> M_4 -> M_1	4.59
26	937	M_1 -> M_{25} -> M_{24} -> M_{23} -> M_{26} -> M_{22} -> M_{21} -> M_{17} -> M_{18} -> M_{20} -> M_{19} -> M_{16} -> M_{11} -> M_{13} -> M_{12} -> M_{15} -> M_{14} -> M_{10} -> M_9 -> M_8 -> M_7 -> M_5 -> M_6 -> M_4 -> M_3 -> M_2 -> M_1	6.75

From Table 1 for symmetric TSP problem, where the graph has a small number of nodes, the model executes perfectly and optimal solution is found. The optimal routes and the shortest travelling distance are obtained. However, whenever the size of vertex set increases, the computation iterations needed rise rapidly, since TSP is the NP-complete problem. Thereby, for problem with large input data, it is suggested to use parallel implementation and to improve the branch and cut algorithm.

3. Conclusion

In short, this article has presented some modelling results of symmetric TSP problems. The results show the efficiency of the code and algorithm especially with small scale input data. However, TSP is NP-complete problem, the modelling where input value consists of more than 30 cities executes with huge computational cost, which requires a sufficient improvement of branch and cut algorithm as well as using parallel implementation. For further study, some improvement of this MIP code can be applied for asymmetric TSP problem. Besides, the branch and cut algorithm should be developed when input data is more than 30 cities.

References

Asani, E. O., Okeyinka, A. E., & Adebiyi, A. A. (2020). A construction tour technique for solving the travelling salesman problem based on convex hull and nearest neighbour heuristics. *In 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS)* IEEE, 1-4.

Chawda, B. V., & Sureja, N. M. (2012). An ACO approach to solve a variant of TSP. *International Journal of Advance Research in Computer Engineering and Technology*, *1*(5), 222-226.

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, *2*(4), 393-410.

Dhakal, S., & Chiong, R. (2008, August). A hybrid nearest neighbour and progressive improvement approach for Travelling Salesman Problem. In *2008 International Symposium on Information Technology*. IEEE, *1*, 1-4.

Diamond, S., & Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, *17*(1), 2909-2913.

Dorigo, M., & Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem. *Biosystems*, *43*(2), 73-81.

Hart, W. E., Laird, C. D., Watson, J. P., Woodruff, D. L., Hackebeil, G. A., Nicholson, B. L., & Siirola, J. D. (2017). *Pyomo-optimization modeling in python* (Vol. 67). Berlin: Springer.

Hoffman, K. L., Padberg, M., & Rinaldi, G. (2013). Traveling salesman problem. *Encyclopedia of operations research and management science*, *1*, 1573-1578.

Liberti, L., & Maculan, N. (Eds.). (2006). *Global optimization: from theory to implementation* (Vol. 84). Springer Science & Business Media.

Linderoth, J. T., & Lodi, A. (2010). MILP software. *Wiley encyclopedia of operations research and management science*, *5*, 3239-3248.

Mitchell, S., OSullivan, M., & Dunning, I. (2011). PuLP: a linear programming toolkit for python. *The University of Auckland, Auckland, New Zealand*, 65. Retrieved from https://www.dit.uoi.gr/e-class/modules/document/file.php/216/PAPERS/2011.%20PuLP%20-%20A%20Linear%20Programming%20Toolkit%20for%20Python.pdf

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, *5*(1), 32-38.

Padberg, M., & Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, *33*(1), 60-100.