Vehicle routing problem with simulated annealing using python programming

R. Aurachman¹, D B Baskara², J Habibie¹

¹Telkom University, Bandung, West Java, Indonesia
²Institut Teknologi Telkom Surabaya, East Java, Indonesia.

Abstract. Vehicle Routing Problems (VRP) are a type of linear programming problem. This paper tries to explain the completion of VRP using Python Programming with the Simulated Annealing algorithm. Python Programming is used as a tool by utilizing the wealth of packages in python. Python is a popular programming language. Python is easy to learn for beginners. However, there is still no example and guide for solving VRP using simulated annealing in python programming with a process paradigm. This design can be a reference and guide for beginners in dealing with similar problems or the development of existing problems.

rio_aurachman@yahoo.com

1. Introduction

The transportation system is one of the important systems in human life. The transportation system connects one location to another. Between locations there are entities that move using the Transportation system. Transportation is a waste. As much as possible the movement is minimized. However, transportation is often very difficult to avoid. So humans try to find ways so that transportation activities can be minimized.

One form of transportation is the delivery of goods in a network. One vehicle must take many items to many places in turns. The simple model of moving goods is the Traveling Salesman Problem (TSP). Several studies have examined the mathematical modelling of the TSP. Other research studies the method of its solution using certain algorithms [1] [2]

The form of development of the Traveling Salesman Problem is the VRP. In VRP, there are more than one vehicle involved in the transportation process. Each vehicle has a capacity. VRP has developed rapidly and one of the studies has mapped various variations of the VRP model. Some of our research has also discussed Transportation problem, especially VRP [3] [4] [5] [6].

On the other hand, programming is growing and advancing rapidly. Academics have begun to explore several alternative programming languages that can be used. Among students in Indonesia and also many researcher around the world, many solve transportation problems using MATLAB [7] [8] [9] [10] [11]. But not enough literature has tried to solve the problem of the use of open source programming software such as Python, Julia, and others. Therefore, this research tries to contribute the programming design using Python Programming to solve VRP Heterogenous Fleet. It is hoped that the results of this research can be a reference for academics and students to solve the transportation problems they face.
2. Method
The research process goes through several stages. The framework used is mathematical modelling. The process of mathematical modelling begins with understanding the problem. After that, an interpretation of the variables representing the problem elements is carried out. After that the relationship between variables is examined. The relationship between these variables is modelled in the form of mathematical equations. After the mathematical model is formed, the next step is to validate and verify the mathematical model.

In designing source code and algorithms, mathematical equations are converted into functions in programming. The paradigm we use in this study is a functional approach. These functions are interrelated with one another to form systems and sub-systems.

The last step is to test the mathematical model and test the software that has been designed. Then evaluate the efficiency and computational time.

3. Result and Discussion
Here is the designed programming code

```python
import functools, operator, os, pandas, math, random, numpy, copy, matplotlib.pyplot as plt
xls = pandas.ExcelFile('data node VRP.xls')
sheet1 = pandas.read_excel(xls,'Sheet1')
sheet2 = pandas.read_excel(xls, 'Sheet2')
nodes = sheet1.as_matrix()
depot = nodes[len(nodes) - 1]
cities = nodes[:len(nodes) - 1]
vehicle = sheet2.as_matrix()
tour = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
n = len(cities)

def distance(x1, y1, x2, y2):
    return math.sqrt((x1 - x2)**2 + (y1 - y2)**2)

def totaldistancetour(tour):
    d = 0
    for i in range(1, len(tour)):
        x1 = cities[tour[i - 1]][0]
        y1 = cities[tour[i - 1]][1]
        x2 = cities[tour[i]][0]
        y2 = cities[tour[i]][1]
        d = d + distance(x1, y1, x2, y2)
    x1 = cities[tour[len(tour) - 1]][0]
    y1 = cities[tour[len(tour) - 1]][1]
    x2 = depot[0]
    y2 = depot[1]
    d = d + distance(x1, y1, x2, y2)
    x1 = cities[tour[0]][0]
    y1 = cities[tour[0]][1]
    x2 = depot[0]
    y2 = depot[1]
    d = d + distance(x1, y1, x2, y2)
    return d
```
Programming begins with import packages or libraries that will help the programming process become more efficient and faster. After that, in the second row import the spreadsheet file which is used as reference data. The data used is dummy data. Line 3 and line 4 are the specific import process sheets containing data to be processed. The fifth to seventh lines are instructions for defining the point that is the depot and the point that is the destination. This programming imports the coordinate data and not the distance matrix.

The distance function is to measure the distance between two Cartesian coordinates. Whereas the totaldistance tour function is a function whose task is to measure the total distance at which the tour becomes an input. Example tour is located in line 9. Tour is an array that contains the destination city numbers ranging from 1 to 20.

```
def subtour(slice,tour):
    capacityused=numpy.zeros(len(vehicle))
    k=0
    slice=[]
    mass=[]
    for i in range(len(vehicle)):
        while capacityused[i] <= vehicle[i][1] and k<=(len(tour)-1):
            capacityused[i]=capacityused[i]+cities[tour[k]][2]
            if capacityused[i] > vehicle[i][1]:
                capacityused[i]=capacityused[i]-cities[tour[k]][2]
            k=k-1
            slice.append(k)
            k=k+1
            break
        k=k+1
    sub=[]
    sub.append(tour[:slice[0]+1])
    for i in range(0,len(slice)-1):
        sub.append(tour[(slice[i]+1):(slice[i+1]+1)])
    return sub
```

In code above there are two functions, namely subtourslice and subtour. The function of the subtourslice is to cut a long tour into a small tour. Each small tour will be assigned to a different vehicle. Then each deduction is based on the capacity of each vehicle. If for example between tours 1,2,3,4,5,6 have a total weight of goods that must be transported beyond the capacity of vehicle 1, then this algorithm will cut off so that vehicle one only goes to subtour 1,2,3. Whereas the function of subotour is to combine all the subtours that have been cut by the subtourslice function.

In code below, it starts with the allvehicle distance function. This function is used to measure the total distance of all vehicles that have each subtour. While the tour distance function is a function whose task is to summarize the function of the previous line, so that when inputting the initial tour, and the vehicle capacity, this function will automatically generate the total distance travelled by all vehicles.

```
lines 80 to 90 contain the iteration process that operates functions from the previous line. Iteration is performed using the Simulated Annealing Algorithm [12].
```

Figure 1 shows the final appearance of the programming that has been designed. The system can generate routes from many vehicles. Each circle shows one vehicle route. It tries to find solutions that produce optimum performance, which is the minimum mileage. Each point shows the coordinates of the Destination city while the point in the centre of the picture is the depot where the vehicle will start departing and will return after delivering to several points.
def allvehicledistance(sub):
    alldistance = functools.reduce(operator.add, (totaldistancetour(x) for x in sub), 0)
    return alldistance

def tourtodistance(tour, vehicle):
    u = subtourslice(tour, vehicle)
    v = subtour(u[0], tour)
    total = allvehicledistance(v)
    return total

for temp in numpy.logspace(0, 5, num=100000)[::-1]:
    existingDistances = tourtodistance(tour, vehicle)
    [i, j] = sorted(random.sample(range(n), 2))
    updateTour = tour[i] + tour[j:j+1] + tour[i+1:j] + tour[i:i+1] + tour[j+1:]
    updateDistances = tourtodistance(updateTour, vehicle)
    if math.exp((existingDistances - updateDistances) / temp) > random.random():
        tour = copy.copy(updateTour)
        betterDistances = copy.copy(updateDistances)
        betterslice = subtourslice(tour, vehicle)
        bettersubtour = subtour(betterslice[0], tour)
        combinesubtour = numpy.concatenate([x+[21] for x in bettersubtour])
        combinesubtour = numpy.append([21], combinesubtour)
        plt.plot([nodes[combinesubtour[i]][0] for i in range(len(combinesubtour))],
                 [nodes[combinesubtour[i]][1] for i in range(len(combinesubtour))], 'xb-')
        plt.annotate(combinesubtour[i], xy=(nodes[combinesubtour[i]][0], nodes[combinesubtour[i]][1]))
        plt.show()

4. Conclusion
Based on the research, Python programming code has been obtained to solve VRP using the Simulated Annealing algorithm. The function uses 6 functions. Programming is also assisted by 9 libraries; functools, operators, os, pandas, math, random, numpy, copy, matplotlib. It is hoped that this programming will become a reference for academics to solve VRP problems using open source software.
Python is a preferrable programming language for Meta heuristic Optimization, both for learning and for professional practice.

The next research can continue the programming design using the Genetic Algorithm algorithm or other algorithms. You can also use alternative open source programming code like Julia.

References

[1] S. Kaewman and R. Akararunguangkul, "Heuristics Algorithms for a Heterogeneous Fleets VRP with Excessive Demand for the Vehicle at the Pickup Points, and the Longest Traveling Time Constraint: A Case Study in Prasitsuksa Songkloe, Ubonratchathani Thailand," Logistics, vol. 2, p. 15, 2018.

[2] L. Zheng, Z. He and W. Liang, "VRP Problem Solving Based on Adaptive Dynamic Search Ant Colony Algorithm".

[3] R. Aurachman, "Optimization model of the number of one-way servers to minimize cost and constraint," in Understanding Digital Industry: Proceedings of the Conference on Managing Digital Industry, Technology and Entrepreneurship (CoMDITE 2019), July 10-11, 2019, Bandung, Indonesia, 2020.

[4] R. Aurachman and A. Y. Ridwan, "Perancangan Model Optimasi Alokasi Jumlah Server untuk Meminimalkan Total Antrean pada Sistem Antrean Dua Arah pada Gerbang Tol," JRSI (Jurnal Rekayasa Sistem dan Industri), vol. 3, no. 2, pp. 25-30, 2016.

[5] A. Desiana, A. Y. Ridwan and R. Aurachman, "Penyelesaian Vehicle Routing Problem (vrp) Untuk Minimasi Total Biaya Transportasi Pada Pt Xyz Dengan Metode Algoritma Genetika," in eProceedings of Engineering, Bandung, 2016.

[6] B. M. Muttalqin, S. Martini and R. Aurachman, "Perancangan Dan Penjadwalan Aktivitas Distribusi Household Product Menggunakan Metode Distribusi Requirement Planning (DRP) Di PT. XYZ Untuk Menyelaraskan Pengiriman Produk Ke Ritel," JRSI (Jurnal Rekayasa Sistem dan Industri), pp. 56-61, 2017.

[7] E. Harahap, A. Harahap, A. Suryadi, D. Darmawan and R. Ceha, "LINTAS: Sistem simulasi lalu lintas menggunakan SimEvents MATLAB," 2018.

[8] M. K. A. Al Mufti, W. Wahyudi and B. Setiyono, "Simulasi Lalu Lintas Simpang Empat Menggunakan Cellular Automata Dengan Sistem Kontrol Berbasis Artificial Neural Network pada Matlab," TRANSIENT, vol. 7, pp. 508-514, 2018.

[9] G. O. N. G. Jin-li and Z.-m. CHENG, "The Realization of Genetic Algorithm of VRP Based on the Matlab [J]," Logistics Sci-Tech, vol. 7, 2006.

[10] N. Rathore, P. K. Jain and M. Parida, "A MATLAB-Based Application to Solve Vehicle Routing Problem Using GA," in Advances in Simulation, Product Design and Development, Springer, 2020, pp. 285-298.

[11] Z. Li, "Improved ant colony optimization for emergency rescue VRP based on matlab," in 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), 2010.

[12] F. Mar’i,i, W. F. Mahmudy and P. B. Santoso, "An Improved Simulated Annealing For The Capacitated Vehicle Routing Problem (Cvrp)," Jurnal Ilmiah Kursor, vol. 9, no. 3, pp. 119-128, 2018.