JURNAL ILMIAH INFORMATIKA GLOBAL
VOLUME 14 Nomor 01 April 2023
e-ISSN 2477-3786 | p-ISSN 2302-500X
DOI : 10.36982/jiig.v13i3.2715

THE AVERAGE VALUE ALGORITHM FROM THE DISTANCE MATRIX FOR TRAVELING SALESMAN PROBLEM

Candra Setiawan[1], Bagus Aditya[2], Agustina Heryati[3]

[1] Computer System, Indo Global Mandiri University
Palembang, Indonesia
[2] Telecommunication Engineering, Telkom University
Bandung, Indonesia
[3] Information System, Indo Global Mandiri University
Palembang, Indonesia
Email : candra@uigm.ac.id[1], goesaditya@telkomuniversity.ac.id[2], agustina.heryati@uigm.ac.id[3]

ABSTRACT

The Traveling Salesman Problem (TSP) is a popular problem, but until now there is no algorithm that has the same search results as brute force with a fast search time. Many algorithms have been made previously related to solving this problem with the aim of finding the shortest route through a number of nodes to finally return to the initial node. The purpose of this research is to create an algorithm that can optimize the search for the shortest route with a fast search time. The approach taken is to find the average value of the distance matrix and look for routes with links that have values below the average value. Each route that has been passed will be marked and compared so that it can facilitate the search with a shorter processing time. In this paper the best and effective routes are limited to 12 nodes. The results obtained show that the Average Score Algorithm provides a relatively stable processing time from node 4 to node 12. The proposed algorithm has a tendency of decreasing processing capacity with increasing number of nodes.

Key word : *travelling salesman problem, distance matrix, computation time, complexity, shortest route*

ABSTRAK

Travelling Salesman Problem (TSP) merupakan masalah yang populer, namun sampai saat ini belum ada algoritma yang memiliki hasil pencarian yang sama dengan Brute Force dengan waktu pencarian yang cepat. Banyak algoritma yang telah dibuat sebelumnya terkait penyelesaian masalah ini dengan tujuan untuk mencari rute terpendek melalui sejumlah node hingga akhirnya kembali ke node awal. Tujuan dari penelitian ini adalah untuk membuat suatu algoritma yang dapat mengoptimalkan pencarian rute terpendek dengan waktu pencarian yang cepat. Pendekatan yang dilakukan adalah mencari nilai rata-rata matriks jarak dan mencari rute dengan link yang memiliki nilai di bawah nilai rata-rata. Setiap rute yang telah dilalui akan ditandai dan dibandingkan sehingga dapat memudahkan pencarian dengan waktu pengerjaan yang lebih singkat. Dalam makalah ini rute terbaik dan efektif dibatasi hingga 12 node. Hasil yang diperoleh menunjukkan bahwa Algoritma Nilai Rata-Rata memberikan waktu pemrosesan yang relatif stabil dari *node* 4 ke *node* 12. Algoritma yang diusulkan memiliki kecenderungan penurunan kapasitas pemrosesan dengan bertambahnya jumlah *node*.

Kata Kunci : *travelling salesman problem, distance matrix, computation time, complexity, shortest route*

Article History
Received : 29/07/2022
Revised : 01/04/2023
Accepted : 01/04/2023
Online : 30/04/2023

This is an open access article under the **CC BY-SA 4.0** License

VOLUME 14 Nomor 01 April 2023
e-ISSN 2477-3786 | p-ISSN 2302-500X

1. Introduction

Traveling Salesman Problem (TSP) is a classic problem in finding the best route but it is still difficult to be solved conventionally. Problems in the TSP arise if the route sought consists of many nodes, causing a high combination. The high combination causes the computational time needed to find the best route is also high (Abdulkarim, 2015).

The route taken by the TSP can only go through one node once and then go to another node until finally returning to the initial node. This route is also known as the Hamiltonian Cycle. The most common solution for TSP is through brute force by calculating all the shortest route probabilities with the lowest total cost. Calculations using brute force are still possible if the number of n nodes is still below 10 so that the required computing time is still not too high. The number of possible routes chosen using n! and each route will be compared between one another to find the best route. The number of combinations significantly influences the computational time required (Baidoo, 2016).

As a calculation, the number of combinations for 12 nodes can produce as many as 1,99584x107 possible routes. Because TSP is one-way, the combination of route A to node B is the same as B to node A or known as symmetric value, so that the same combination of routes can be eliminated and simplified using the formula (n-1)/2. The higher the number of nodes, the more routes will be compared to each other (Beardwood, 1959).

The large number of nodes makes brute force not the right solution to solve a TSP problem. Therefore we need an algorithm that can be an alternative solution with faster computing time. Although this TSP is a classic problem, but until now there has not been one algorithm that is proven to be able to find the lowest total cost with a short computing time. Research on TSP is not final and continues to the present (Droste, 2017).

2. Related Work

Brute Force algorithm in principle is the flow of problem solving by trying all possibilities that exist to find the best solution. The brute force solution only calculates the total distance for each possible route and then chooses the shortest one. This algorithm was chosen to solve a simple problem and does not require large calculations (Hannah Bast, 2015).

For a large number of nodes, a smarter algorithm is needed to find a shortcut in finding the best solution on the TSP. Intelligent algorithms can eliminate many routes so that computing time is shorter.

Many algorithms have been created as a solution to this problem including the Greedy, Ant Colony Optimization (ACO), Fast Bee Colony Optimization, and others. All of these algorithms try to find the best solution both in terms of total cost and computing time (Girsang, 2012) (Syambas, 2017).

3. Proposed Algorithm

A. Average Value of Distance Matrix

The proposed algorithm in principle looks for the lowest total cost with a short computational time. The first step taken is to calculate the average value of the distance matrix. Distance matrix used in this study is a matrix that has been used by previous studies (Hannah Bast, 2015) (Johnson, 2002).

Table 1. *Distance Matrix*

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	0	930	993	1138	218	458	1194	637	234	577	529	371
2	930	0	1748	857	510	901	1022	785	659	421	130	399
3	993	1748	0	564	795	901	1076	749	795	612	441	669
4	1138	857	564	0	768	1460	481	930	49	948	381	806
5	218	510	795	768	0	1380	583	534	434	516	1133	949
6	458	901	901	1460	1380	0	946	552	549	658	1030	929
7	1194	1022	1076	481	583	946	0	754	1372	928	816	930
8	637	785	749	930	534	552	754	0	1257	856	769	616
9	234	659	795	49	434	549	1372	1257	0	390	757	321
10	577	421	612	948	516	658	928	856	390	0	681	363
11	529	130	441	381	1133	1030	816	769	757	681	0	928
12	371	399	669	806	949	929	930	616	321	363	928	0

B. Pseudo Code

The simple idea for creating pseudo code is as follows:

1. Enter the distance matrix.
2. Determine the initial node, where the initial node will also be the last node.
3. Take the values in the distance matrix and do the calculation to find the average value. For the same value in the distance matrix, only one value is taken.
4. Use links with values below the average to find the best route.
5. Each route selected will be compared and marked to find the best value.
6. For the last route that reaches the final node, any value will be chosen so that the route forms a ring.

C. Flowchart

Flowchart for the process of the proposed algorithm can be seen in the figure below.

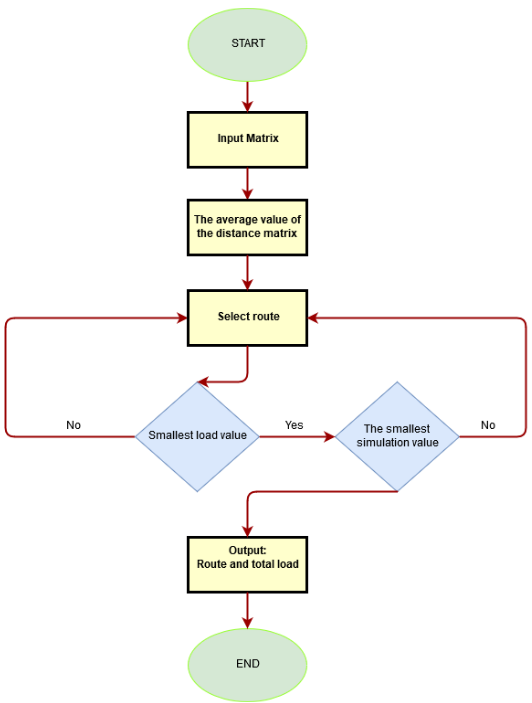

Figure 1. *Flowchart of proposed algorithm*

D. Coding Logic for Proposed Algortyhm

Based on the pseudo code above, the following encoding logic will be used by the TSP simulation.

```
1.   Inisialisasi data membaca distance matrix
2.   ARRAY beban[point1,point2]
3.   ARRAY pointTerpakai
4.   ARRAY bebanList
5.   jumlahColumn: integer
6.   line_count: integer
7.   point1: integer
8.   point2: integer
9.   costPebanding: integer
10.  pointPilihan: integer
11.  route:varchar
12.  namaFileCsv:varchar
13.      point1 <- 1
     poin2 <- 0
     route <- "1"
14.  nextMinCost(poin1,numcol,nodeTerpakai):
       costPebanding <- 0
       FOR y in range(1, numcol):
         IF poin1!=y and costPebanding == 0 and y not in nodeTerpakai:
           costPebanding <- beban[poin1,y]
         END IF
         ELIF poin1!=y and y not in nodeTerpakai and costPebanding > beban[poin1,y]:
           costPebanding <- beban[poin1,y]
         END ELIF
       NEXT y

       RETURN costPebanding
     END FUNCTION
15.  FUNCTION simulasi(p1<-0, p2<-0, jumlah_p<-0, bebanTerpakai<-[], rute<-'1', jp<-0, bebanRouting<-[]):
     bebanTotal <- 0
     arrP1 <- []
     arrP1.append(p1);
     bebanTerpakai <- diff(bebanTerpakai, arrP1)
     poin1 <- p1
     poinPilihan <- p1
     FOR x in range(0, jumlah_p-(1+len(bebanTerpakai))):
       costPebanding <- 0
       FOR y in range(1, jumlah_p):
         poin2 <- y
         IF poin1!=poin2 and costPebanding == 0 and poin2 not in bebanTerpakai:
           costPebanding <- bebanRouting[poin1,poin2]
           poinPilihan <- poin2
         ELIF poin1!=poin2 and poin2 not in bebanTerpakai and costPebanding > bebanRouting[poin1,poin2]:
           IF int(nextMinCost(poinPilihan,numcol,pointTerpakai)) > int(nextMinCost(poin2,numcol,pointTerpakai)) :
             costPebanding <- bebanRouting[poin1,poin2]
             poinPilihan <- poin2

       bebanTotal<- int(bebanTotal)+int(costPebanding)
       bebanTerpakai.append(poinPilihan)
       rute <- rute+" > "+str(poinPilihan)
       poin1 <- poinPilihan

     bebanTotal<- int(bebanTotal)+int(beban[poin1,1])

     rs <- rute.split(' > ')
     totbbn <- 0
     b1<-0
     b2<-0

     FOR x in range(0, len(rs)-1):
       IF x>0:
         b1 <- int(rs[x-1])
         b2 <- int(rs[x])
         totbbn <- int(beban[b1,b2])+totbbn

     totbbn <- int(bebanRouting[b2,1])+totbbn
     RETURN totbbn

16.    FOR x IN range(0, numcol-2):
     costPebanding <- 0
     pointTerpakai.append(poin1)
     PRINT(pointTerpakai)
     FOR y in range(1, numcol):
       poin2 <- y
       IF poin1!=poin2 and costPebanding == 0 and poin2 not in pointTerpakai:
         costPebanding <- beban[poin1,poin2]
         poinPilihan <- poin2
       END IF
```

```
        ELIF   poin1!=poin2 and poin2 not in pointTerpakai and costPebanding >= beban[poin1,poin2]:
            IF int(simulasi(poinPilihan, poin2, numcol, pointTerpakai, route, numcol, beban)) >= int(simulasi(poin2, poin2, int(numcol), pointTerpakai, route, numcol, beban)) :
                costPebanding <- beban[poin1,poin2]
                poinPilihan <- poin2
            END IF
        END ELIF


    costTotal=int(costTotal)+int(costPebanding)
    bebanList.append(costTotal)
    route <- route+" > "+str(poinPilihan)
    poin1 <- poinPilihan

   costTotal <- int(costTotal)+int(beban[poin1,1])
   route <- route+" > 1"
17.        PRINT(costTotal)

   rs = route.split(' > ')
   totbbn = 0

   for x in range(0, len(rs)-1):
       if x>0:
           b1 = int(rs[x-1])
           b2 = int(rs[x])
           PRINT("jarak dari"+rs[x-1]+" ke "+rs[x]+" = "+beban[b1,b2])
           totbbn = int(beban[b1,b2])+totbbn

   PRINT("jarak dari"+str(b2)+" ke 1 = "+beban[b2,1])
   totbbn = int(beban[b2,1])+totbbn
   PRINT("Total Beban = "+str(totbbn))

#selesai
END
```

E. Simulation of Traveling Salesman Problem

Applications for TSP simulations are made using Python 3.7.2. This simulation is run on Microsoft Windows 10 with a computer that has an Intel i5 processor and 8 Giga bytes Random Access Memory. Data is entered in files using CSV format, by using the distance matrix created by previous research. The simulation created will show the route used, total cost and time needed to complete the computation. The solution with brute force uses source code that comes from Github (Westphahl, 2010).

4. Simulation Results

The simulation starts with 4 nodes and finally with 12 nodes. This is because of the limitations of the hardware used in the simulation. Brute force will be a reference point in comparing the total cost and computation time obtained by this algorithm.

Table 2. *Comparison of Total Costs*

Node	Average Value of Distance Matrix	Brute Force	Proposed Algorithm
4	1038.33	3344	4736
5	852.1	3142	4756
6	908.06	3508	6376
7	901.09	4154	8506
8	852.28	4554	7753
9	811.47	4750	10472
10	780.42	4649	9590
11	759.74	4472	9586
12	743.43	4766	11139

Whereas for computing time can be seen in table 3.

Table 3. *Comparison of Computation Time*

Node	Computation Time	
	Brute Force	Proposed Algorithm
4	0,0134	0,0480
5	0,0142	0,0415
6	0,0151	0,0445
7	0,0248	0,0365
8	0,1288	0,0354
9	0,5745	0,0376
10	19,0063	0,0394
11	210,102	0,0391
12	2383,984	0,0371

A. Comparison of Computational Time

From the table above we can make a computational time comparison chart, where a significant difference starts at node 10. At the 12th node the time needed by brute force reaches 2383,984 seconds while the proposed algorithm takes 0.0371 seconds.

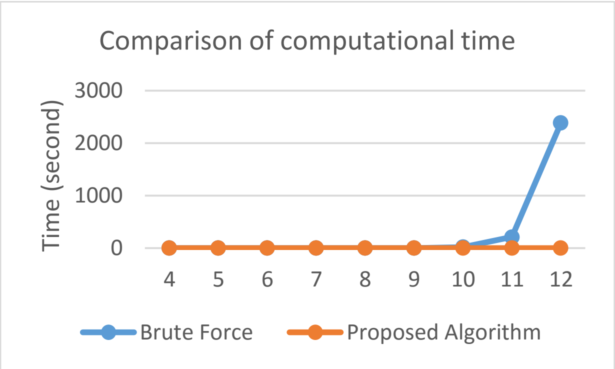

Figure 2. *Comparison of computational time between proposed algorithm and brute force*

B. Comparison of Total Cost

For the total cost, the results obtained through the proposed algorithm have a very big difference when compared to the results obtained through brute force. This is caused by the still not fitting the coding logic used in implementing the pseudo code that was made. So that the simulation made needs to be improved in terms of programming.

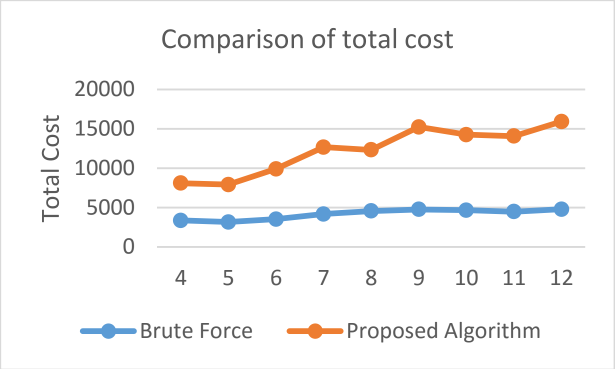

Figure 3. *Comparison of Total Cost*

C. Computation Time for Proposed Algorithm

In the proposed algorithm, computational time tends to decrease with increasing number of nodes. This decrease graph can be seen in Figure 4.

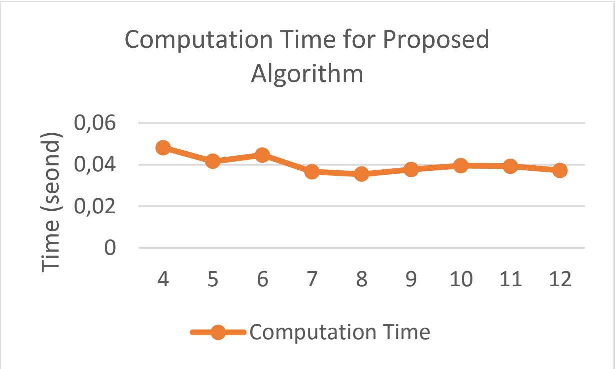

Figure 4. *Computation Time for Proposed Algorithm*

5. Conclusion

Based on the simulation it appears that the results obtained are still large differences compared to the brute force. There are still many weaknesses and need to be improved in terms of coding logic. On the computational time side, the proposed algorithm has a relatively stable time from node 4 to node 12. Proposed algorithms have a trend that tends to decrease with increasing number of nodes.

References

Abdulkarim, H. A. &. A. I. F., 2015. Comparison of Algorithms for Solving Traveling. *International Journal of Engineering and Advanced Technology (IJEAT)*, 4(6), pp. 76-79.

Baidoo, E. &. O. S. O., 2016. Solving the TSP using Traditional Computing Approach. *International Journal of Computer Applications*, 152(8), pp. 13-19.

Beardwood, J. H. J. H. &. H. J. M., 1959. The shortest path through many points. *Mathematical Proceedings of the Cambridge Philosophical Society*, Volume 55, pp. 299-327.

Droste, I., 2017. *Algorithms for the travelling salesman problem*. Utrecht: Universiteit Utrecht.

Girsang, A. S. T. C.-W. &. Y. C.-S., 2012. A Fast Bee Colony Optimization for Traveling Salesman Problem. *Third International Conference on Innovations in Bio-Inspired Computing and Applications*.

Hannah Bast, D. D. A. G. M. M.-H. T. P. P. S. D. W. R. F. W., 2015. *Route Planning in Transportation Networks*, s.l.: Microsoft Research.

Johnson, D. S. &. M. L. A., 2002. Experimental Analysis of Heuristics forthe STSP. *The Traveling Salesman Problem and its Variations*, pp. 369-443.

Saiyed, A. R., 2012. *The Traveling Salesman Problem*. Indiana: Indiana State University.

Syambas, N. R. S. S. &. S. G. M., 2017. *Fast heuristic algorithm for travelling salesman problem*. Lombok, IEEE.

Thomas, J. &. C. N., 2011. *Two Approaches for Hamiltonian Circuit Problem using Satisfiability*. Tumkur, Indian International Conference on Artificial Intelligence (IICAI-11).

Vukmirović, S. &. P. D., 2013. *The Travelling Salesman Problem In The Function Of Transport Network Optimalization*. [Online] Available at: https://www.semanticscholar.org/paper/The-Travelling-Salesman-Problem-In-The-Function-Of-Vukmirovic-Pupavac/2d4cfb6155224604c93bd362ad64239cf8e344ed [Accessed 2020].

Wang, Z. D. H. &. Z. X., 2009. *An Improved Greedy Genetic Algorithm for Solving Travelling Salesman Problem*. Tianjin, 2009 Fifth International Conference on Natural Computation.

Westphahl, S., 2010. *TSP Brute-Force Solution*. [Online] Available at: https://gist.github.com/westphahl/432876 [Accessed 2020].