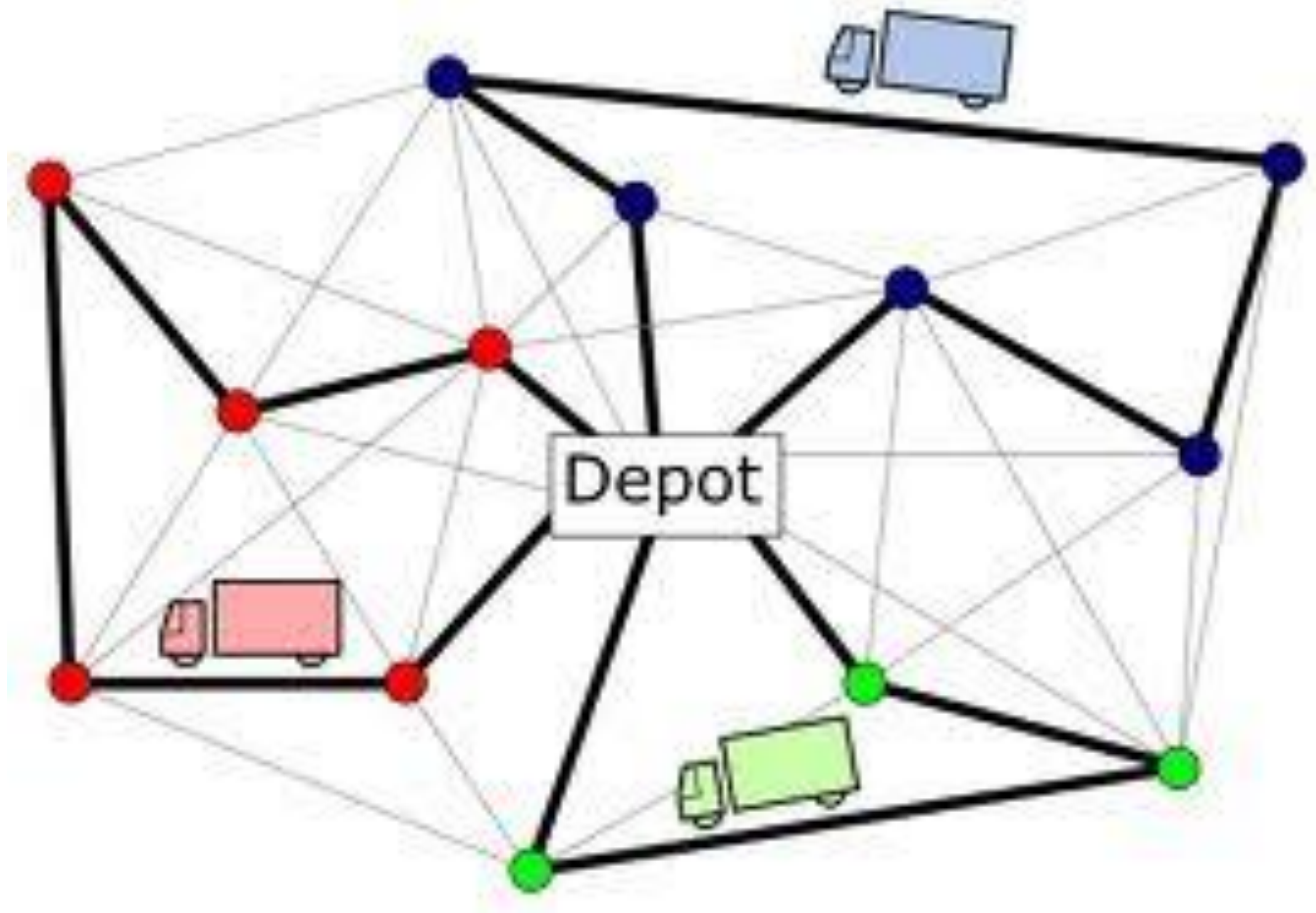
SISTEM TRANSPORTASI DAN DISTRIBUSI BARANG

Perutean dalam Aktivitas Transportasi



Muhammad Nashir Ardiansyah, S.T., M.T., Ph.D.

Program Studi S1 Teknik Industri – Telkom University

Image Source: https://www.r-bloggers.com/any-r-packages-to-solve-vehicle-routing-problem/

Algoritma Penyelesaian TSP

Metode Pendekatan Penyelesaian TSP

- Beberapa algoritma pendekatan sederhana untuk menyelesaikan permasalahan TSP:
  1. Algoritma Nearest Neighbor
  2. Algoritma Branch and Bound
  3. Algoritma Lin-Kernighan
  4. Algoritma Farthest Insertion
  5. Algoritma V-opt

Algoritma Farthest Insertion

- Algoritma Farthest Insertion dimulai dari pemilihan tur yang terdiri dari dua kota dengan jarak antar kota maksimum, berulang kali memilih kota dengan jarak maksimum ke tetangga terdekat di antara titik yang belum terpilih, dan memasukkannya seperti dalam Nearest Neighbor.

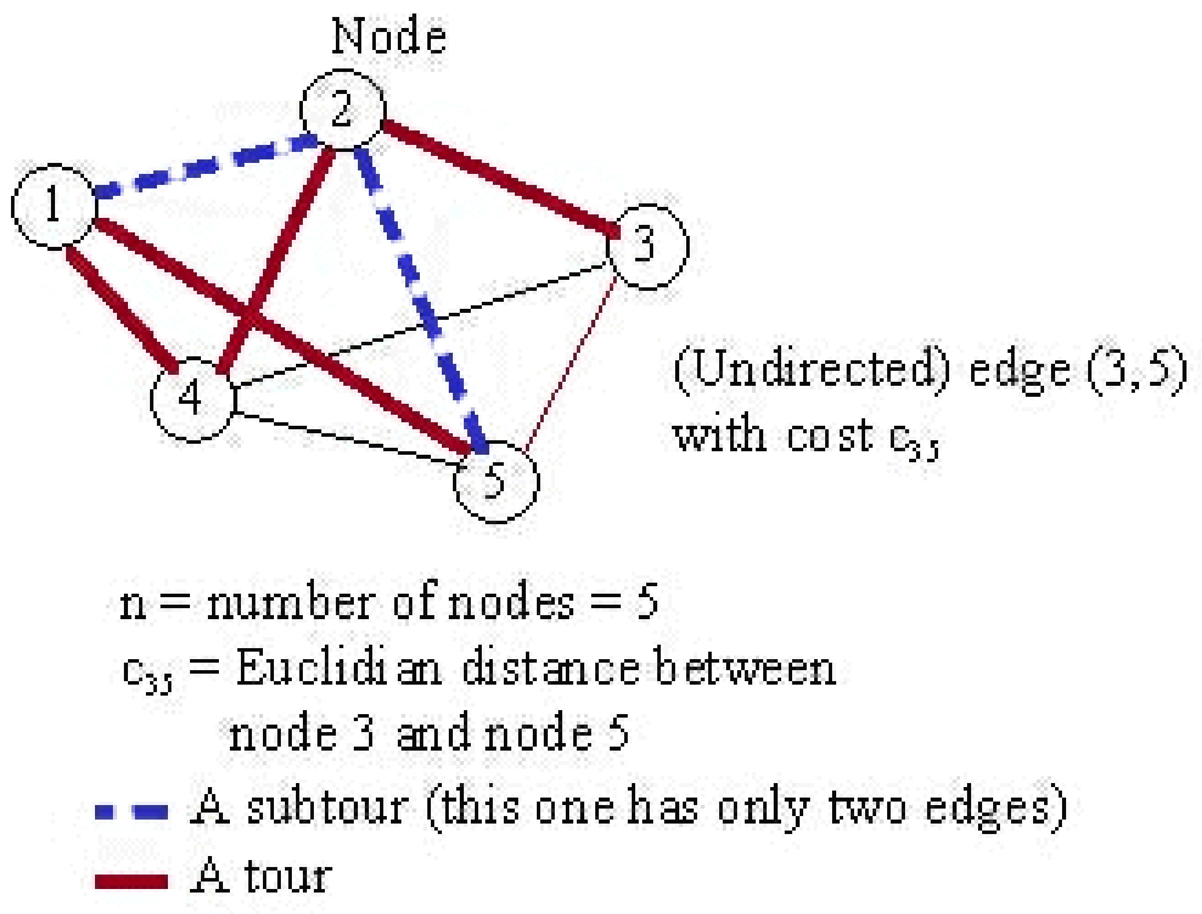


Image Source: https://www2.isye.gatech.edu/~mgoetsch/cali/VEHICLE/TSP/TSP003__.HTM

Tahapan Algoritma Farthest Insertion

1. For every node v not in the cycle, $dist(v)$ is the distance to v from that node in the current cycle from which v is closest
2. Each time a new node f is added to the cycle, the $dist$ array is updates such that its entries are the minimum of the current entries in the $dist$ array and the f th row in W
3. Having settled on the Selection step, let us now look at the Insertion step. Assume that there are k nodes in the current cycle, and the next (farthest) node to be inserted is f.
4. We examine every edge (i,j) in the current tour to determine the insertion cost of/between node i and j, which is

$$c_{ij} = w_{if} + w_{fj} - w_{ij}$$

Tahapan Algoritma Farthest Insertion

6. Among all k edges in the cycle we select $edge(t, h)$ — with tail t and head h — for which c_{th} has the smallest value (c_{ij} could be negative). Then insert node f between t and h. The weight of the cycle is updated. We also update the $dist$ array
7. To keep track of V_T, the nodes in the current cycle, as well as E_T, the edges in the current cycle, we will maintain an array, $cycle$, of length n, defined — as follows; $cycle(i) = 0$ if and only if node i is not in the current cycle; and $cycle(i) = j$ if and only if (i, j) is an edge in the current cycle

Contoh Permasalahan TSP

- Terdapat 6 kota yang harus dikunjungi oleh seorang pedagang.
- Buat rute kunjungan pedagang ke 6 kota!

$$
\begin{array}{c|cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & \infty & 3 & 93 & 13 & 33 & 9 \\
2 & 4 & \infty & 77 & 42 & 21 & 16 \\
3 & 45 & 17 & \infty & 36 & 16 & 28 \\
4 & 39 & 90 & 80 & \infty & 56 & 7 \\
5 & 28 & 46 & 88 & 33 & \infty & 25 \\
6 & 3 & 88 & 18 & 46 & 92 & \infty
\end{array}
$$

Algoritma Farthest-Insertion

1. Let us arbitrarily pick node 1 as the starting node s. The $dist$ array at this juncture will be

$$dist = (-, 3, 93, 13, 33, 9)$$

which is row 1 of weight matrix W, except $dist\ (1)$, which is immaterial. The other array is $cycle = (1,0,0,0,0,0)$; the sub-tour is $(1,1)$.

2. The largest entry in $dist$ array is 93, corresponding to node 3. Therefore, the sub-tour is enlarged to $(1,3,1)$ and the total distance traveled $(tweight)$ is

$$w_{13} + w_{31} = 93 + 45 = 138.$$

The $dist$ array is now modified to have entries that are the smaller of $dist$ and row 3 of W.

$$\begin{array}{c|c} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{bmatrix} \infty & 3 & 93 & 13 & 33 & 9 \\ 4 & \infty & 77 & 42 & 21 & 16 \\ 45 & 17 & \infty & 36 & 16 & 28 \\ 39 & 90 & 80 & \infty & 56 & 7 \\ 28 & 46 & 88 & 33 & \infty & 25 \\ 3 & 88 & 18 & 46 & 92 & \infty \end{bmatrix} \end{array}$$

That is $dist$ = (-, 3,-, 13, 16, 9)

and $cycle = (3,0,1,0,0,0)$;

the sub-tour is (1, 3, 1).

	1	2	3	4	5	6
$w_{1.}$	—	3	—	13	33	9
$w_{3.}$	—	17	—	36	16	28
$\min\{w_{1.}, w_{3.}\}$	—	3	—	13	16	9

Algoritma Farthest-Insertion

- Now in the second iteration the farthest node from the current sub-tour is 5, corresponding to the largest value, 16, in the $dist$ array. Node 5 can be inserted in two different ways. The insertion costs are

$$c_{13} = w_{15} + w_{53} - w_{13} = 33 + 88 - 93 = 28$$
$$c_{31} = w_{35} + w_{51} - w_{31} = 16 + 28 - 45 = -1\ (*)$$

Performing the insertion with lower cost, we obtain $tweight = 138 - 1 = 137$, and the two arrays are

$cycle = (3, 0, 5, 0, 1, 0)$; the sub-tour is $(1, 3, 5, 1)$.

$$dist = (-, 3, -, \mathbf{13}, -, 9)$$

Algoritma Farthest-Insertion

- In the third iteration, node 4 is the farthest. The three insertion costs of node 4 are

$$\begin{aligned} c_{13} &= w_{14} + w_{43} - w_{13} = 0\ (*) \\ c_{35} &= w_{34} + w_{45} - w_{35} = 76 \\ c_{51} &= w_{54} + w_{41} - w_{51} = 44 \end{aligned}$$

We, therefore, perform the lowest-cost insertion (at zero cost). The new sub-tour is $(1,4,3,5,1)$, with a value $tweight$ of 137.

The updated arrays are

$$\begin{aligned} cycle &= (4,0,5,3,1,0) \\ dist &= (-,3,-,-,-,7) \end{aligned}$$

Algoritma Farthest-Insertion

- In the fifth and the last iteration, we must insert node 2. Its five insertion costs are

  $c_{14} = 32$, $c_{46} = 99$, $c_{63} = 147$, $c_{35} = 22\ (*)$, and $c_{51} = 22\ (*)$.

  There are two minimum values; we could pick either. Let us choose C_{35}. Then we obtain the final solution as $(1,4,6,3,2,5,1)$, with the total distance traveled

$$tweight = 82 + 22 = 104$$