Travelling Salesman Problem and Bellman-Held-Karp Algorithm

Quang Nhat Nguyen

May 10, 2020

1 Definitions

Definition 1.1 (Travelling Salesman Problem)**.** *Let $G = (V, E, \omega)$ be a* weighted *Hamiltonian graph, with $V = \{x_1, x_2, \ldots, x_n\}$, $i : E \to V \times V$ determines the edges, and $\omega : E \to \mathbb{R}_+$ determines the weighted edge lengths. The problem of finding its shortest Hamiltonian cycle is called the Travelling Salesman Problem (abbrv. TSP).*

Note: *If G is undirected, the TSP is called* symmetric TSP *(sTSP). If G is directed, the TSP is called* asymmetric TSP *(aTSP).*

This is one of the classical problems in Computer Science, and is an introductory problem to Dynamic Programming. The need of computer power arises when we consider a large graph (eg. one that contains hundreds or thousands of vertices) and want to find its shortest Hamiltonian cycle. Before discussing the approaches, let us introduce the *distance matrix.*

Definition 1.2 (Distance matrix)**.** *The $n \times n$ matrix D with elements*

$$D_{ij} := \min\{\omega(e) \,|\, e \in E,\, i(e) = (x_i, x_j)\} \quad \textit{if there exists such } e$$
$$D_{ij} := \infty \quad \textit{if there is no such } e$$

is called the distance matrix *of the weighted graph G.*

In simpler terms, the element D_{ij} denotes the shortest edge that originates at x_i and terminates at x_j. It can be seen that if the graph G is undirected, the matrix D is symmetric. If there is no edge from x_i to x_j, $D_{ij} = \infty$. If there are multiple edges from x_i to x_j, we only consider the shortest edge because our concern is the shortest Hamiltonian cycle.

A naïve approach to this problem would be for the computer to consider all permutations of the vertices, see if one permutation can make a cycle, and if it does then record the cycle's length for comparison. This method has a time complexity of $O(n!)$, and thus is not desirable when dealing with a large database.

Proposition 1.3. *One can reduce the time complexity of $O(n!)$ of the naïve method to $O(n^2 \times 2^n)$ by implementing Bellman-Held-Karp Algorithm.*

Let us discuss this algorithm in the next section.

2 Bellman-Held-Karp Algorithm

First, let us acknowledge the following.

Proposition 2.1. *The shortest Hamiltonian cycle does not depend on the choice of the starting vertex.*

This is obvious because the Hamiltonian cycle has cyclic symmetry, thus changing the starting vertex does not change the order of the other vertices in the cycle. Therefore, let us start constructing the desired path from x_1.

Definition 2.2. *Let $S \subset V\backslash\{x_1\} \equiv \{x_2, x_3, \ldots, x_n\}$ be a subset of size s $(1 \leq s \leq n-1)$. For each vertex $x_i \in S$, we define* $\text{cost}(x_i, S)$ *as the length of the shortest path from x_1 to x_i which travels to each of the remaining vertices in S once. More precisely, we implement this function by the following recursion formula:*

$$\text{cost}(x_i, S) = \min_{x_j}\{\text{cost}(x_j, S\backslash\{x_i\}) + D_{ji}\} \tag{2.1}$$

in which $x_j \in S\backslash\{x_i\}$. In case S has size 1, we define:

$$\text{cost}(x_i, S) = D_{1i} \tag{2.2}$$

Using the above recursive formula, we can gradually construct the cost function for subset S of size from 1 to $n-1$. Because of this recursive feature, which means that the current step is the base for the next step, the implementation of this algorithm in a programming language is called Dynamic Programming.

When we reach subset S of size $n-1$, which means $S = V\backslash\{x_1\}$, the only thing left is to find:

$$\begin{aligned}\text{Shortest Hamiltonian cycle} &= \min_{x_i}\{\text{cost}(x_i, S) + D_{i1}\} \\ &\equiv \min_{x_i}\{\text{cost}(x_i, V\backslash\{x_1\}) + D_{i1}\}\end{aligned}$$

with $x_i \in S \equiv V\backslash\{x_1\}$.

Proposition 2.3 (Time complexity of this algorithm)**.** *The time complexity of this algorithm is: $O(n^2 \times 2^n)$.*

Proof. This algorithm considers 2^{n-1} subsets of $V\backslash\{x_1\}$. For each subset, one computes the cost function for all of its elements, which there are no more than n of them. For each execution of the cost function, one again computes no more than n values based on the values obtained from the previous step. In total, the time complexity of this algorithm is: $O(n^2 \times 2^n)$. □

Remark 2.4. *Compared to the naïve method which has time complexity $O(n!)$, the time complexity of this algorithm, $O(n^2 \times 2^n)$, is much better. For example, if $n = 100$:*

$$\begin{aligned}O(n!) = O(100!) &= O(9.33 \times 10^{157}) \\ O(n^2 \times 2^n) = O(100^2 \times 2^{100}) &= O(1.27 \times 10^{34})\end{aligned}$$

which is about 7.35×10^{123}, or 0.7 septillion googols, times faster.

3 Illustration

Let us illustrate the algorithm through the following sample problem.

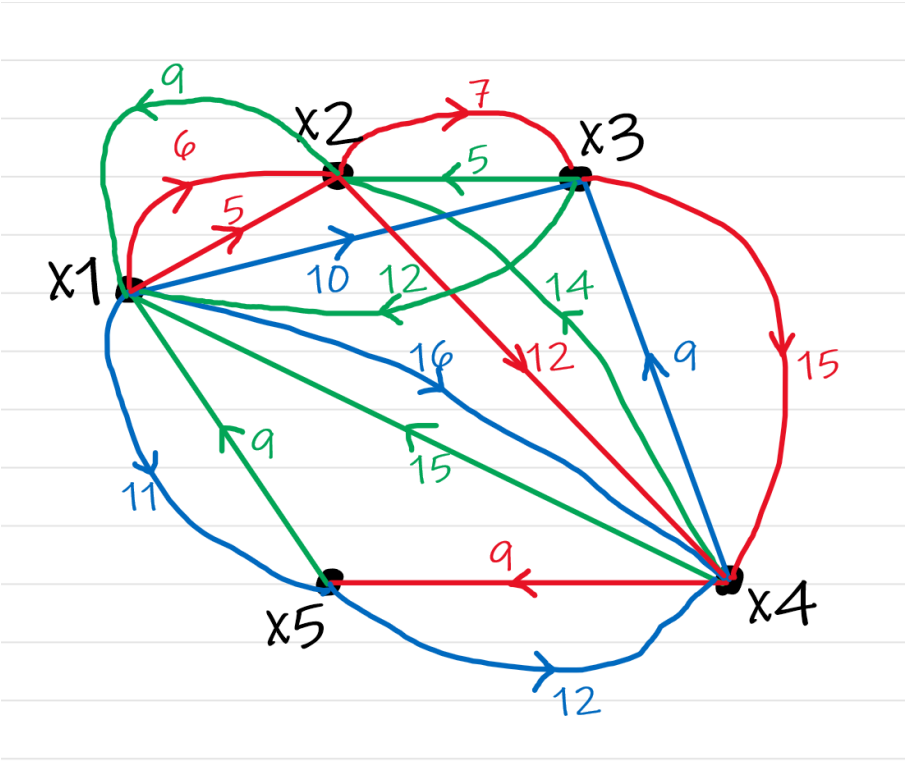

Figure 1: Graph for a sample asymmetric TSP

The distance matrix can be generated as follows:

$$\begin{bmatrix} 0 & 5 & 10 & 16 & 11 \\ 9 & 0 & 7 & 12 & \infty \\ 12 & 5 & 0 & 15 & \infty \\ 15 & 14 & 9 & 0 & 9 \\ 9 & \infty & \infty & 12 & 0 \end{bmatrix}$$

Now that we have the distance matrix, we can proceed with the recursive formula. First, let us consider subsets S of size 1:

Subset S	cost function	Result
$S = \{x_2\}$	$\text{cost}(x_2, \{x_2\}) = D_{12}$	5
$S = \{x_3\}$	$\text{cost}(x_3, \{x_3\}) = D_{13}$	10
$S = \{x_4\}$	$\text{cost}(x_4, \{x_4\}) = D_{14}$	16
$S = \{x_5\}$	$\text{cost}(x_5, \{x_5\}) = D_{15}$	11

Table 1: Process for subsets S of size 1

Next we consider subsets S of size 2 as follows:

Subset S	cost function	Result
$S = \{x_2, x_3\}$	$\text{cost}(x_2, \{x_2, x_3\}) = \min\{\text{cost}(x_3, \{x_3\}) + D_{32}\} = 10 + 5$	15
	$\text{cost}(x_3, \{x_2, x_3\}) = \min\{\text{cost}(x_2, \{x_2\}) + D_{23}\} = 5 + 7$	12
$S = \{x_2, x_4\}$	$\text{cost}(x_2, \{x_2, x_4\}) = \min\{\text{cost}(x_4, \{x_4\}) + D_{42}\} = 16 + 14$	30
	$\text{cost}(x_4, \{x_2, x_4\}) = \min\{\text{cost}(x_2, \{x_2\}) + D_{24}\} = 5 + 12$	17
$S = \{x_2, x_5\}$	$\text{cost}(x_2, \{x_2, x_5\}) = \min\{\text{cost}(x_5, \{x_5\}) + D_{52}\} = 11 + \infty$	∞
	$\text{cost}(x_5, \{x_2, x_5\}) = \min\{\text{cost}(x_2, \{x_2\}) + D_{25}\} = 5 + \infty$	∞
$S = \{x_3, x_4\}$	$\text{cost}(x_3, \{x_3, x_4\}) = \min\{\text{cost}(x_4, \{x_4\}) + D_{43}\} = 16 + 9$	25
	$\text{cost}(x_4, \{x_3, x_4\}) = \min\{\text{cost}(x_3, \{x_3\}) + D_{34}\} = 10 + 15$	25
$S = \{x_3, x_5\}$	$\text{cost}(x_3, \{x_3, x_5\}) = \min\{\text{cost}(x_5, \{x_5\}) + D_{53}\} = 11 + \infty$	∞
	$\text{cost}(x_5, \{x_3, x_5\}) = \min\{\text{cost}(x_3, \{x_3\}) + D_{35}\} = 10 + \infty$	∞
$S = \{x_4, x_5\}$	$\text{cost}(x_4, \{x_4, x_5\}) = \min\{\text{cost}(x_5, \{x_5\}) + D_{54}\} = 11 + 12$	23
	$\text{cost}(x_5, \{x_4, x_5\}) = \min\{\text{cost}(x_4, \{x_4\}) + D_{45}\} = 16 + 9$	25

Table 2: Process for subsets S of size 2

For the subsets S of size 3, we have the following table:

Subset S	cost function	Result
$S = \{x_2, x_3, x_4\}$	$\text{cost}(x_2, \{x_2, x_3, x_4\}) = \min\{25 + 5, 25 + 14\}$	30
	$\text{cost}(x_3, \{x_2, x_3, x_4\}) = \min\{30 + 7, 17 + 9\}$	26
	$\text{cost}(x_4, \{x_2, x_3, x_4\}) = \min\{15 + 12, 12 + 15\}$	27
$S = \{x_2, x_3, x_5\}$	$\text{cost}(x_2, \{x_2, x_3, x_5\}) = \min\{\infty, \infty\}$	∞
	$\text{cost}(x_3, \{x_2, x_3, x_5\}) = \min\{\infty, \infty\}$	∞
	$\text{cost}(x_5, \{x_2, x_3, x_5\}) = \min\{\infty, \infty\}$	∞
$S = \{x_2, x_4, x_5\}$	$\text{cost}(x_2, \{x_2, x_4, x_5\}) = \min\{23 + 14, \infty\}$	37
	$\text{cost}(x_4, \{x_2, x_4, x_5\}) = \min\{\infty, \infty\}$	∞
	$\text{cost}(x_5, \{x_2, x_4, x_5\}) = \min\{\infty, 17 + 9\}$	26
$S = \{x_3, x_4, x_5\}$	$\text{cost}(x_3, \{x_3, x_4, x_5\}) = \min\{23 + 9, 25 + \infty\}$	32
	$\text{cost}(x_4, \{x_3, x_4, x_5\}) = \min\{\infty, \infty\}$	∞
	$\text{cost}(x_5, \{x_3, x_4, x_5\}) = \min\{25 + \infty, 25 + 9\}$	34

Table 3: Process for subsets S of size 3

Note: Exlanation for the first entry:

$$\begin{aligned}\text{cost}(x_2, \{x_2, x_3, x_4\}) &= \min\{\text{cost}(x_3, \{x_3, x_4\}) + D_{32}, \text{cost}(x_4, \{x_3, x_4\}) + D_{42}\} \\ &= \min\{25 + 5, 25 + 14\} = 30\end{aligned}$$

For the subsets S of size 4, which means $S = \{x_2, x_3, x_4, x_5\}$, we have the following:

cost function	Evaluation	Result
$\text{cost}\{x_2, \{x_2, x_3, x_4, x_5\}\}$	$\min\{\text{cost}(x_3, \{x_3, x_4, x_5\}) + D_{32},$ $\text{cost}(x_4, \{x_3, x_4, x_5\}) + D_{42}, \quad = \min\{32+5, \infty, \infty\}$ $\text{cost}(x_5, \{x_3, x_4, x_5\}) + D_{52}, \}$	37
$\text{cost}\{x_3, \{x_2, x_3, x_4, x_5\}\}$	$\min\{\text{cost}(x_2, \{x_2, x_4, x_5\}) + D_{23},$ $\text{cost}(x_4, \{x_2, x_4, x_5\}) + D_{43}, \quad = \min\{37+7, \infty, \infty\}$ $\text{cost}(x_5, \{x_2, x_4, x_5\}) + D_{53}, \}$	44
$\text{cost}\{x_4, \{x_2, x_3, x_4, x_5\}\}$	$\min\{\text{cost}(x_2, \{x_2, x_3, x_5\}) + D_{24},$ $\text{cost}(x_3, \{x_2, x_3, x_5\}) + D_{34}, \quad = \min\{\infty, \infty, \infty\}$ $\text{cost}(x_5, \{x_2, x_3, x_5\}) + D_{54}, \}$	∞
$\text{cost}\{x_5, \{x_2, x_3, x_4, x_5\}\}$	$\min\{\text{cost}(x_2, \{x_2, x_3, x_4\}) + D_{25},$ $\text{cost}(x_3, \{x_2, x_3, x_4\}) + D_{35}, \quad = \min\{\infty, \infty, 27+9\}$ $\text{cost}(x_4, \{x_2, x_3, x_4\}) + D_{45}, \}$	36

Table 4: Process for subsets S of size 4

For the final step, we have:

$$\begin{aligned}\text{Shortest Hamiltonian cycle} = \min\{&\text{cost}\{x_2, \{x_2, x_3, x_4, x_5\}\} + D_{21},\\ &\text{cost}\{x_3, \{x_2, x_3, x_4, x_5\}\} + D_{31},\\ &\text{cost}\{x_4, \{x_2, x_3, x_4, x_5\}\} + D_{41},\\ &\text{cost}\{x_5, \{x_2, x_3, x_4, x_5\}\} + D_{51}\}\\ = \min\{&37 + 9, 44 + 12, \infty, 36 + 9\}\\ = 45\end{aligned}$$

This result means that this graph has *at least* three Hamiltonian cycles, and the shortest one has length 45. To trace back the order of the vertices in the shortest one, one simply trace back the vertices x_i in the minimal cost functions $\text{cost}(x_i, S)$. In this sample problem, the back-tracing order will be:

$$x_1 \leftarrow x_5 \leftarrow x_4 \leftarrow x_2 \leftarrow x_3 \leftarrow x_1$$

However, this back-tracing order is also correct:

$$x_1 \leftarrow x_5 \leftarrow x_4 \leftarrow x_3 \leftarrow x_2 \leftarrow x_1$$

So, the shortest Hamiltonian cycle has the following forward-going order:

$$x_1 \rightarrow x_3 \rightarrow x_2 \rightarrow x_4 \rightarrow x_5 \rightarrow x_1$$

or:

$$x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \rightarrow x_5 \rightarrow x_1$$

One can easily look at Figure 1 and check that the above Hamiltonian cycles both have length 45. This concludes the illustration of Bellman-Held-Karp Algorithm, which is based on Dynamic Programming