Dynamic programming approaches for the traveling salesman problem with drone

1 Problem Description

该研究[1]针对的问题是 Traveling Salesman Problem with Drone (TSP-D)，目标函数是求卡车和无人机返回仓库的最短时间，单次无人机飞行只能够服务一个顾客，允许无人机的配送路径是个环（即卡车在某个顾客节点等待无人机服务后返回），且卡车和无人机会合的节点只能够是顾客节点或者仓库。该研究用到的符号及其含义如表 table 1所示。

表 1: TSP-D 用到的符号及含义

符号	含义
V	位置节点集合，包含仓库 v_0 和 $n-1$ 个顾客位置，即 $V=\{v_0, v_1, \ldots, v_{n-1}\}$
n	位置节点总数（包含仓库 v_0）
$c(v,w)$	卡车从位置 v 到位置 w 的行驶时间（或距离），$v,w \in V$
$c^d(v,w)$	无人机从位置 v 到位置 w 的飞行时间（或距离），$v,w \in V$
α	无人机与卡车的速度关联常数，满足 $\alpha c^d(v,w) \geq c(v,w)$ 对所有 $v,w \in V$，表示无人机最大速度为卡车的 α 倍
o	单个操作（operation），由起始节点、终止节点、最多一个无人机节点（由无人机单独服务的节点）和若干卡车节点（卡车单独服务的节点）组成，当没有无人机节点时，卡车搭载无人机进行服务，当起始节点和终止节点一致且没有卡车节点但有无人机节点时，卡车在起始节点等待无人机服务完后会合
$t(o)$	单个操作 o 的持续时间，取卡车行驶总时间与无人机飞行总时间的最大值
S	V 的子集，用于表示动态规划中已覆盖的节点集合
$D_{\text{TSP}}(S,w)$	标准 TSP 中，从起点 $v \in S$ 出发覆盖子集 S 中所有节点并结束于 $w \in S \setminus \{v\}$ 的最短路径成本

2 Solution Approaches

该研究的解决方法主要基于 Bellman-Held-Karp dynamic programming algorithm for the TSP[1]进行改良[2-3]，该方法分为三个阶段：

1. 对于任意一个起始节点、终止节点以及中间卡车服务的节点集合，枚举所有这样组合的卡车行驶的最短路
2. 在阶段一的卡车行驶最短路径的基础上引入无人机节点，然后计算最小时间花费的覆盖所有

[1]关于 Bellman-Held-Karp 算法的进一步理解可以参考本文的末尾的附录部分，该文档来源于Nagoya University 的 Serge Richard 教授。

服务节点集合的可行操作（efficient operation）

3. 计算满足服务所有顾客节点且起始节点和终止节点都在仓库的最优的可行操作排序

2.1 Dynamic Programming Approach for The Standard TSP

通常的 Traveling Salesman Problem (TSP) 是找到一条从任意一个起点 v 出发，访问所有 V 中的节点并且最终回到起点 v 的最短路。路径的权重可以是距离、时间、排放量等可以被分解到每一步的量，表示为 $D_{\text{TSP}}(V, v)$。

接下来简单介绍一下 Bellman-Held-Karp (BHK) 算法，对于一个包含了 v 和 $w \in S \setminus \{w\}$ 的集合 $S \subset V$，用 $D_{\text{TSP}}(S, w)$ 表示从起点 v 出发，访问 S 中所有节点并到达终点 w 的最短路所需要的花费（时间或者距离等）。而 $D_{\text{TSP}}(S, w)$ 又可以分解为一个更小的子问题（subproblem），即该问题可以由从某个节点 $u \in S$ 到达终点 w 的弧和从 v 出发访问集合 $S \setminus \{w\}$ 中所有节点并到达终点 u 的最短路组合而成。因此解决问题 $D_{\text{TSP}}(S, w)$ 可以变成解决考虑所有弧 $(u, w) : u \in S \setminus \{w\}$ 和子问题 $D_{\text{TSP}}(S \setminus \{w\}, u)$ 的所有组合。可以通过递归公式来表达上述过程：

$$D_{\text{TSP}}(S, w) = \begin{cases} \infty & \text{if } w \notin S \\ c(v, w) & \text{if } S = \{w\} \\ \min_{u \in S} \{D_{\text{TSP}}(S \setminus \{w\}, u) + c(u, w)\} & \text{otherwise} \end{cases} \tag{1}$$

由于 V 的所有可能的子集数量为 2^n，并且 v 的选择有 n 种（即有 n 个地点作为可能的起点），因此最多可能有 $n \cdot 2^n$ 个子问题。对于解决每个递归的子问题时，最多考虑 n 次递归（因为每次递归都减少一个节点，最多有 n 个节点），所以该算法的时间复杂度为 $O(n^2 \cdot 2^n)$。

参考文献

[1] BOUMAN P, AGATZ N, SCHMIDT M. Dynamic programming approaches for the traveling salesman problem with drone[J/OL]. Networks, 2018, 72(4): 528-542. DOI: 10.1002/net.21864.

[2] BELLMAN R. Dynamic Programming Treatment of the Travelling Salesman Problem[J/OL]. J. ACM, 1962, 9(1): 61-63. DOI: 10.1145/321105.321111.

[3] HELD M, KARP R M. A dynamic programming approach to sequencing problems[C/OL]//Proceedings of the 1961 16th ACM National Meeting On -. Not Known: ACM Press, 1961: 71.201-71.204. DOI: 10.1145/800029.808532.

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.

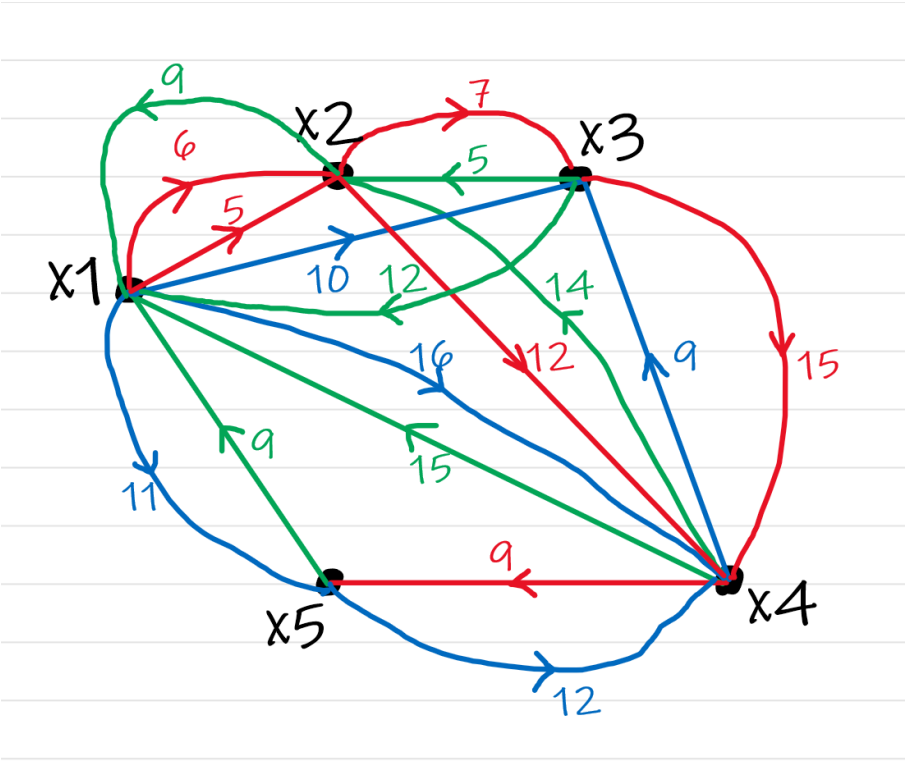

Figure 1: Graph for a sample asymmetric TSP

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