Lecture Note - Traveling Salesman Problem and The Held-Karp Lower Bound

Lan Peng, Ph.D.

School of Management, Shanghai University, Shanghai, China

"O never go back."

1 The Traveling Salesman Problem

In this section, we are going to compare between different formulations of the Traveling Salesman Problem (TSP). Generally speaking, let $G = (V, A)$ be a graph where V is a set of n vertices, and A is a set of arcs (or edges). Let $C = c_{ij}$ be a cost (distance) matrix associated with A. The TSP consists of determining a minimum cost (distance) Hamiltonian circle (or cycle) that visits each vertex once and only once. If for all $i, j \in V, c_{ij} = c_{ji}$, then the TSP is symmetrical, otherwise is asymmetrical.

Define the decision variable x_{ij} as the following

$$x_{ij} = \begin{cases} 1, & \text{if goes from } i \text{ to } j \\ 0, & \text{otherwise} \end{cases}, \quad (i,j) \in A \tag{1}$$

The objective function will be

$$\min \quad \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{2}$$

1.1 Dantzig-Fulkerson-Johnson (DFJ) Formulation

The first famous formulations for TSP is the **Dantzig-Fulkerson-Johnson (DFJ) formulation**:

$$\sum_{j\in V,(i,j)\in A} x_{ij} = 1, \quad \forall i \in V \tag{3}$$

$$\sum_{i\in V,(i,j)\in A} x_{ij} = 1, \quad \forall j \in V \tag{4}$$

$$\sum_{j\notin S, i\in S,(i,j)\in A} x_{ij} \geq 1, \quad \forall S \subset V, 2 \leq |S| \leq n-1 \tag{5}$$

In the formulation, constraints (3) and constraints (4) are degree constraints, which specify that every vertex is entered exactly once. Constraints (5) is the sub-tour constraints, they prohibit the formation of sub-tours. S is a non-empty subset of V, and has at least 2 vertices. (5) can be replaced by

$$\sum_{i,j\in S,(i,j)\in A} x_{ij} \leq |S| - 1, \quad \forall S \subset V, 2 \leq |S| \leq n-1 \tag{6}$$

If we list all sub-tour constraints in DFJ, there will be $O(2^n)$ constraints and $O(n^2)$ binary variables. The exponential number of constraints makes it impractical to solve directly. Instead, lazy constraints are usually implemented for the sub-tour elimination constraints (5) or (6).

In the graph $G = (N, A)$, let $\bar{G} = (N, \bar{A})$ be the connected components of graph, where

$$\bar{G} = (G, \bar{A}), \bar{A} = \{(i, j) \in A | \bar{x_{ij}} = 1\}$$

denote

$$\bar{FS}(i) = \{(i, j) \in \bar{A}\}$$

The following algorithm describes an algorithm to find subtours:

Algorithm 1 Sub-tour Searching Algorithm

```
K = ∅
d_i = 0, ∀i ∈ N
for i ∈ N do
    C = ∅
    Q = ∅
    if d_i == 0 then
        d_i = 1
        C = C ∪ {i}
        Q.append(i)
        while Q ≠ ∅ do
            v = Q.pop()
            for u ∈ FS̄(v) do
                if d_u == 0 then
                    d_u = 1
                    C = C ∪ {u}
                    Q.append(u)
    K = K ∪ C
```

We can add those sub-tour constraints into the formulation on the fly in a Branch-and-cut manner.

1.2 Miller-Tucker-Zemlin (MTZ) Formulation

We can also formulate TSP using sequential formulations, namely, **Miller-Tucker-Zemlin (MTZ) formulation**. In the MTZ formulation, the degree constraints (3) and (4) are the same as in DFJ formulation.

Define a new set of integer decision variables u_i, u_i defined as the sequence in which node i is visited, $u_1 = 1$.

The sub-tour constraints (5) or (6) are replaced by the following:

$$u_i - u_j + (n-1)x_{ij} \leq n - 2, \quad i, j = 2, \cdots, n \in V, (i, j) \in A \tag{7}$$

$$1 \leq u_i \leq n - 1, \quad i \in 2, \cdots, n \in V \tag{8}$$

In MTZ formulation, there are $O(n^2)$ constraints, $O(n^2)$ binary variables, and $O(n)$ continuous variables.

1.3 Flow Based Formulations

In this section, flow based formulations are discussed, which includes **Single Commodity Flow**, **Two Commodity Flow** and **Multi-Commodity Flow**. In these formulations, continuous variables are introduced to represent the flow on the arcs.

In Single Commodity Flow formulation, define y_{ij} as the flow in an arc $(i,j) \in A$. Degree constraints (3) and (4) are retained. The following constraints are introduced:

$$y_{ij} \leq (n-1)x_{ij}, \quad \forall i,j \in V, (i,j) \in A \tag{9}$$

$$\sum_{j \in V,(1,j) \in A} y_{1j} = n-1 \tag{10}$$

$$\sum_{i \in V,(i,j) \in A} y_{ij} - \sum_{k \in V,(j,k) \in A} y_{jk} = 1, \quad \forall j \in V \setminus \{1\} \tag{11}$$

Constraints (9) can be tighten by the following:

$$y_{ij} \leq (n-1)x_{ij}, \quad i = 1, j \in V \setminus \{1\}, (i,j) \in A \tag{12}$$

$$y_{ij} \leq (n-2)x_{ij}, \quad i,j \in V \setminus \{1\}, (i,j) \in A \tag{13}$$

In SCM formulation, there are $O(n^2)$ constraints, $O(n^2)$ binary variables and $O(n^2)$ continuous variables.

In Two Commodity Flow formulation, define y_{ij} as the flow in an arc $(i,j) \in A$, for commodity type 1, and define z_{ij} as the flow in an arc $(i,j) \in A$, for commodity type 2.

Besides degree constraints, the other constraints are as following

$$y_{ij} + z_{ij} = (n-1)x_{ij}, \quad \forall i,j \in V, (i,j) \in A \tag{14}$$

$$\sum_{j \in V \setminus \{1\}} (y_{1j} - y_{j1}) = n-1, \quad (1,j) \in A \tag{15}$$

$$\sum_{j \in V} (y_{ij} - y_{ji}) = 1, \quad \forall i \in V \setminus \{1\}, (i,j) \in A \tag{16}$$

$$\sum_{j \in V \setminus \{1\}} (z_{1j} - z_{j1}) = 1-n, \quad (1,j) \in A \tag{17}$$

$$\sum_{j \in V} (z_{ij} - z_{ji}) = -1, \quad \forall i \in V \setminus \{1\}, (i,j) \in A \tag{18}$$

$$\sum_{j \in V} (y_{ij} + z_{ij}) = n-1, \quad \forall i \in V \tag{19}$$

In TCM formulation, constraints (14) only allow flow in an arc if present. Constraints (15) and (16) forces $(n-1)$ units of commodity type 1 to flow in at node 1 and 1 unit to flow out at every other nodes. Constraints (17) and (18) are similar, those forces $(n-1)$ units of commodity type 2 to flow out at node 1 and 1 unit to flow in at every other nodes. Constraints (19) forces exactly $(n-1)$ units of combined commodity in each arc.

In TCM formulation, there are $O(n^2)$ constraints, $O(n^2)$ binary variables and $O(n^2)$ continuous variables.

The SCM and the TCM can be generalized into **Multi-Commodity Flow formulation**. As usual, degree constraints are retained. The following continuous variables are introduced. Define

y_{ij}^k as the flow of commodity type k in arc $(i,j) \in A$.

The other constraints are

$$y_{ij}^k \leq x_{ij}, \quad \forall i,j,k \in N, k \neq 1 \tag{20}$$

$$\sum_{i \in V} y_{1i}^k = 1, \quad \forall k \in V \setminus \{1\} \tag{21}$$

$$\sum_{i \in V} y_{i1}^k = 0, \quad \forall k \in V \setminus \{1\} \tag{22}$$

$$\sum_{i \in V} y_{ik}^k = 1, \quad \forall k \in V \setminus \{1\} \tag{23}$$

$$\sum_{j \in V} y_{kj}^k = 0, \quad \forall k \in V \setminus \{1\} \tag{24}$$

$$\sum_{i \in V} y_{ij}^k - \sum_{i \in V} y_{ji}^k = 0, \quad \forall j,k \in V \setminus \{1\}, j \neq k \tag{25}$$

Constraints (20) only allow flow in an arc which is present. Constraints (21) forces exactly one unit of each type of commodity to flow in at node 1. Constraints (22) prevent any commodity flow out at node 1.Constraints (23), and Constraints (24), forces exactly one unit of type k commodity to flow out, and in, at every node except node 1. Constraints (25) forces balance of all types of commodities at every node except node 1.

This formulation has $O(n^3)$ constraints, $O(n^2)$ binary variables, and $O(n^3)$ continuous variables.

1.4 Shortest Path Formulation

In this section, we are going to introduce another form of formulation with different definition of decision variable and objective function.

Assuming for a completed graph $G = (V, A)$. Define x_{ij}^t as the following

$$x_{ij}^t = \begin{cases} 1, & \text{If path crosses arc } (i,t) \text{ and } (j,t+1) \\ 0, & \text{Otherwise} \end{cases}, \quad i \in V, j \in V \setminus \{i\}, t = 1, \cdots, n \tag{26}$$

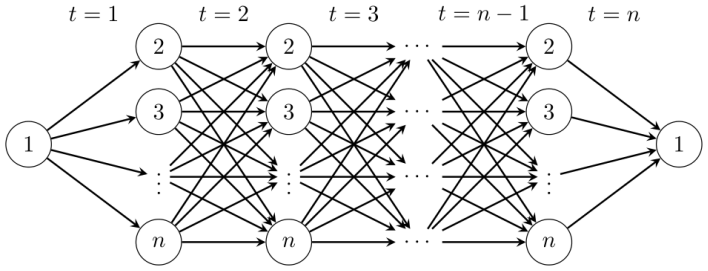


Figure 1: Time-staged graph

The objective function will be

$$\min \quad \sum_{i \in V} \sum_{j \in V \setminus \{i\}} c_{ij} \sum_{t=1}^{n} x_{ij}^t \tag{27}$$

The constraints are as following

$$\sum_{j\in V\setminus\{1\}} x_{1j}^1 = 1 \tag{28}$$

$$\sum_{j\in V\setminus\{1,i\}} x_{ij}^2 - x_{1i}^1 = 0, \quad \forall i \in V\setminus\{1\} \tag{29}$$

$$\sum_{j\in V\setminus\{1,i\}} x_{ij}^t - \sum_{j\in V\setminus\{1,i\}} x_{ji}^{t-1} = 0, \quad \forall i \in V\setminus\{1\}, t \in \{2,\dots,n-1\} \tag{30}$$

$$x_{i1}^n - \sum_{j\in V\setminus\{1,i\}} x_{ji}^{n-1} = 0, \quad \forall i \in V\setminus\{1\} \tag{31}$$

$$\sum_{i\in V\setminus\{1\}} x_{i1}^n = 1 \tag{32}$$

$$\sum_{t=2}^{n-1} \sum_{j\in V\setminus\{1,i\}} x_{ij}^t + x_{i1}^n \le 1, \quad \forall i \in V\setminus\{1\} \tag{33}$$

Notice that constraint (33) can be replaced by

$$x_{1i}^1 + \sum_{t=2}^{n-1} \sum_{j\in V\setminus\{1,i\}} x_{ji}^t \le 1, \quad \forall i \in V\setminus\{1\} \tag{34}$$

1.5 Quadratic Formulation (QAP)

In this section, we are going to go over a TSP formulation are super bad. However, it still has some value for further study.

The idea is to transform TSP into an assignment problem. Assuming we have n boxes, which represents n steps in the path. Define x_{ij} as

$$x_{ij} = \begin{cases} 1, & \text{Vertex } i \text{ is assigned to box } j \\ 0, & \text{Otherwise} \end{cases} \tag{35}$$

The constraints are simple as an assignment problem as following

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \in V \tag{36}$$

$$\sum_{i\in V}^{n} x_{ij} = 1, \quad j = 1,\dots,n \tag{37}$$

However, the tricky part is in the objective function

$$\min \quad \sum_{i\in V} \sum_{j\in V\setminus\{i\}} \sum_{k=1}^{n-1} c_{ij} x_{ik} x_{j,k+1} + \sum_{i\in V} \sum_{j\in V\setminus\{i\}} c_{ij} x_{in} x_{j1} \tag{38}$$

Notice that the objective function is not linear function, with the multiplications of decision variables. Now we are going to linearize them. The linearized version is as following

$$\min \quad \sum_{i\in V}\sum_{j\in V\setminus\{i\}}\sum_{k=1}^{n-1} c_{ij}w_{ij}^k + \sum_{i\in V}\sum_{j\in V\setminus\{i\}} c_{ij}w_{ij}^n \tag{39}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \in V \tag{40}$$

$$\sum_{i\in V}^{n} x_{ij} = 1, \quad j = 1, \ldots, n \tag{41}$$

$$w_{ij}^k \geq x_{ik} + x_{j,k+1} - 1, \quad i \in V, j \in V \setminus \{i\}, k = 1, \cdots, n-1 \tag{42}$$

$$w_{ij}^k \geq x_{ik} + x_{j1} - 1, \quad i \in V, j \in V \setminus \{i\}, k = n \tag{43}$$

$$w_{ij}^k \in \{0,1\}, \quad i \in V, j \in V \setminus \{i\}, k = 1, \ldots, n \tag{44}$$

$$x_{ij} \in \{0,1\}, \quad i \in V, j \in V \setminus \{i\} \tag{45}$$

We can prove that this is very very bad. The optimal solution of the LP Relaxation for the QAP formulation is as follows

$$x_{ij} = \frac{1}{n}, \quad \forall i, j \in V \tag{46}$$

$$w_{ij}^k = \frac{2}{n} - 1, \quad \forall i \in V, j \in V \setminus \{i\}, k = 1, 2, \ldots, n \tag{47}$$

The solution indicates that all decision variables are symmetric in the LP Relaxation, thus, it did not provide any information for branching. In fact, such formulation will search all $O(2^n)$ branches and the lower bound will be difficult to converge.

2 The Held and Karp Lower Bound

In this section, we will solve the Dantzig-Fulkerson-Johnson formulation using Lagrangian Relaxation. Before finally converge, the LR finds an infeasible solution as lower bound. The bound found by this method is also known as Held & Karp Bound.

2.1 Minimum Spanning Tree

We first introduce the concept of minimum spanning tree by an example. A company wants to build a communication network for their offices. For a link between office v and office w, there is a cost c_{vw}. If an office is connected to another office, then they are connected to with all its neighbors. Company wants to minimize the cost of communication networks.

Definition 2.1 (Spanning Tree)**.** A subgraph T of G is a **spanning tree** if it is spanning ($V(T) = V(G)$) and it is a tree.

Definition 2.2 (Minimum Spanning Tree)**.** Given a connected graph graph G, and a cost $C_e, \forall e \in E$, find a minimum cost spanning tree of G

The follow are two algorithms that finds the minimum spanning tree on given graph G.

Algorithm 2 Kroskal's Algorithm, $O(m \log m)$

Require: A connected graph
Ensure: A minimum spanning tree
  Keep a spanning forest $H = (V, F)$ of G, with $F = \emptyset$
  while $|F| < |V| - 1$ **do**
    add to F a least-cost edge $e \notin F$ such that H remains a forest.

Algorithm 3 Prim's Algorithm, $O(nm)$

Require: A connected graph
Ensure: A minimum spanning tree
  Keep $H = (V(H), T)$ with $V(H) = \{v\}$, where $r \in V(G)$ and $T = \emptyset$
  while $|V(T)| < |V|$ **do**
    Add to T a least-cost edge $e \notin T$ such that H remains a tree.

- Kroskal start with a forest that contains all vertices, Prim start with a tree that only contain one vertex.
- Kroskal cannot guarantee every step it is a tree but can guarantee it is spanning, Prim can guarantee every step it is a tree but cannot guarantee spanning.

2.2 1-Tree

We first introduce an intuitive lower bound for TSP, which is the Minimum Spanning Tree Lower Bound. As we known, an optimal solution for the TSP is a Hamilton Cycle which enumerated all vertices on the graph. The length of such Hamilton cycle is denoted as TSP^*. If we randomly remove one of the edges, e.g., $\{vw\}$, then, the cycle becomes a path, denoted by $TSP - \{vw\}$. In this case, the degree of vertices v and w reduce to 1 while all the other vertices remains the same as 2. By definition, such path $TSP - \{vw\}$ is a spanning tree on graph G. Therefore, the minimum spanning tree of graph G defines a lower bound.

$$MST \leq spanningtree = TSP - \{vw\} < TSP^*$$

We can further improve the lower bound by introducing 1-tree. A spanning tree is a tree with no cycle, if the graph has 1 cycle, it is called 1-tree. Notice that a Hamilton Cycle has only 1 cycle, thus, the Hamilton Cycle is an 1-tree as well. Naturally, the length of all edges of any 1-tree is larger than the length of all edges of the minimum spanning tree. The lower bound of TSP is further improved as follows

$$MST < M1T \leq 1 - tree \leq TSP^*$$

The following algorithm defines an 1-Tree on graph G corresponding to vertex v.

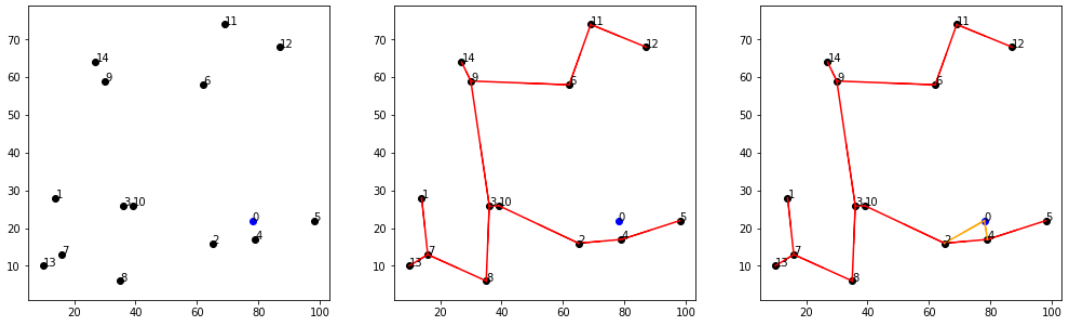
A M1T is a good lower bound, however, we can still further improve the lower bound by finding 1-trees.

2.3 Held and Karp Lower Bound and Lagrangian Relaxation

Notice that in the minimum 1-tree example, vertices have different degrees, some are of degree 1, 2, 3, or more. However, the "optimal" 1-tree that we look for, is an 1-tree such that all the vertices

Algorithm 4 1-Tree

Require: A connected graph
Ensure: A minimum 1-Tree
Remove v from the graph
Use Kroskal's algorithm or Prim's Algorithm to find the minimum spanning tree of $G \setminus \{v\}$
Find the shortest edge induced by v to the rest of graph $G \setminus \{v\}$, denoted by vu and vw, add them to the MST
return 1-Tree



(a) Remove v (b) Find MST (c) Add back v

Figure 2: Steps in finding minimum 1-Tree

are of degree 2. By intuition, if an 1-tree has fewer "branches", it's closer to be "optimal". That is actually the principle of the Held and Karp Lower Bound.

We first look at the DFJ formulation for the TSP.

$$\min \quad \sum_{e \in E} \tau_e x_e \tag{48}$$

$$\text{s.t.} \quad \sum_{e \in \delta(i)} x_e = 2, \quad \forall i \in 1, 2, \ldots, n \tag{49}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1, \quad \forall S \subset V, 2 \leq |S| \leq n - 1 \tag{50}$$

$$x_e \in \{0, 1\}, \quad \forall e \in E \tag{51}$$

In the DFJ model, the objective is to sum up all the cost of the edges that are chosen on the TSP path. Constraints (49) is a flow balancing constraint, the index $e \in \delta(i)$ represents all the edges that are induced by vertex i. Constraints (50) is the sub-tour constraint.

Replace the Constraints (49) by the following

$$\sum_{e \in \delta(i)} x_e = 2, \quad \forall i \in 1, 2, \ldots, n-1 \tag{52}$$

$$\sum_{e \in E} x_e = n \tag{53}$$

We can reformulate the DFJ formulation as follows:

$$\min \quad \sum_{e\in E} \tau_e x_e \tag{54}$$

$$\text{s.t.} \quad \sum_{e\in\delta(i)} x_e = 2, \quad \forall i \in 1,2,\ldots,n-1 \tag{55}$$

$$\sum_{e\in E} x_e = n \tag{56}$$

$$\sum_{e\in E(S)} x_e \leq |S| - 1, \quad \forall S \subset V, 2 \leq |S| \leq n-1 \tag{57}$$

$$x_e \in \{0,1\}, \quad \forall e \in E \tag{58}$$

Look closely, Constraints (56) means, the number of edges in the subgraph should be the same as the number of vertices, so the subgraph has 1 cycle. The Constraints (57) guarantee that the graph is connected. Which means, Constraints (56) and (57) finds an 1-tree.

Until now, we can move the Constraints (55) to the objective function, and use the Lagrangian Relaxation to solve the problem

$$z(\mathbf{u}) = \min \quad \sum_{e\in E} \tau_e x_e + \sum_{i=1}^{n-1} u_i (2 - \sum_{e\in\delta(i)} x_e) \tag{59}$$

$$\text{s.t.} \quad \mathbf{x} \text{ defines an 1-tree with vertice } n \tag{60}$$

For the vertex i, define u_i, the formulation can be further rewritten as

$$z(\mathbf{u}) = \min \quad 2\sum_{i=1}^{n-1} u_i + \sum_{e\in\delta(i)} (\tau_e - u_{e^+} - u_{e^-}) x_e \tag{61}$$

$$\text{s.t.} \quad \mathbf{x} \text{ defines an 1-tree with vertice } n \tag{62}$$

2.4 Subgradient Descendant Method

Notice that in the previous section, we derived a function $z(\mathbf{u})$ of Lagrangian scalar u_i as the lower bound of the TSP. The solution space of $\mathbf{u}$ is a convex space, in this section, we will search the $\max_{\mathbf{u}} z(\mathbf{u})$ by subgradient descendant method.

In the algorithm, different descendant function may be applied, for example, an easier way for implementation is to update

$$u_i \leftarrow u_i + (d_i - 2)\rho^k$$

Algorithm 5 Subgradient descendant method for Held-Karp Lower Bound

Require: A connected graph

Ensure: A lower bound of TSP

Initialize, for each vertex i, let $u_i = 0$, $d_i = \emptyset$. Define lower bound $L \leftarrow 0$, $L' \leftarrow 0$

while do$|L - L'| \geq \epsilon$

Update weights of edges $\tau_{ij} = \tau_{ij} - u_i - u_j$

Find the M1T on the graph G with edges updated. Let D be the summation of the length of all edges, let d_i be the degree of vertex i on the M1T.

Update $u_i \leftarrow u_i + \lambda_i \dfrac{UB - D}{\sum_i (d_i - 2)^2}$

Update $L' \leftarrow L$

Update $L \leftarrow D + 2\sum_i u_i$

return L