5.2. EXACT METHODS FOR THE TSP

5.2.1. BRANCH AND BOUND METHODS

Branch & Bound methods are normally based on some relaxation of the ILP model of TSP. The decision variables of the model are

$$x_{ij} = \begin{cases} 1 & \text{if the cycle goes along arc } i \rightarrow j \\ 0 & \text{otherwise} \end{cases}$$

In the following optimization models, the variables x_{ii} are either excluded from the model or prevented by setting $c_{ii} = \infty$.

ILP-model:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad (5.1)$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for every } i=1,...,n \qquad (5.2)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{for every } j=1,...,n \qquad (5.3)$$

$$x_{ij} = 0 \text{ or } 1 \quad i,j=1,...,n \qquad (5.4)$$

In addition, *subtour elimination constraints* (= constraints preventing separate cycles) must be enclosed:

• Tucker formulation:

$$u_i - u_j + n x_{ij} \le n-1 \quad \text{for every } i,j = 2,...,n,\ i \neq j \qquad (5.5)$$
$$u_i \in \mathbb{R}, \quad i=2,...,n$$

• Dantzig-Fulkerson-Johnson formulation:

$$\sum_{i \in S} \sum_{j \in \overline{S}} x_{ij} \ge 1 \quad \text{for subsets } S \subset \{1,...,n\},\ 2 \le |S| \le n-2 \qquad (5.6)$$

($\overline{S}$ = V\S, complement of S)

An alternative formulation:

$$\sum_{i,j \in R} x_{ij} \le |R| - 1 \quad \text{for all nonempty subsets } R \subset \{2,...,n\}. \qquad (5.7)$$

Constraints (5.5) prevent any subtours not containing vertex 1. Consider a subtour with vertices $i_1,...,i_k$ not containing vertex 1. Write down the constraints (5.5) for the subtour edges i-j having variable values $x_{ij}=1$. Summing these constraints yields $nk \le (n-1)k$, which is false.
Constraints (5.6) force a connection (an edge to be used) between every pair of separate subsets.
Constraints (5.7) prevent cycles of |R| vertices that do not include node 1, because the number of nodes equals the number of edges in a cycle.

The DFJ formulations are impractical for large scale problems, because the number of subtour elimination constraints is of order 2^n.

In Branch&Bound methods the lower bound for the length of the cycle is determined by an appropriate relaxation, e.g. by discarding the subtour elimination constraints. This results in an assignment problem, and it is called an assignment relaxation.

BRANCH AND BOUND ALGORITHM FOR THE TSP, WITH ASSIGNMENT RELAXATION

0. Initialization: Construct an initial feasible tour by some heuristic or use $1 \to 2 \to \ldots \to n \to 1$ and set $f_{min} := c_{12} + c_{23} + \ldots + c_{n1}$.

1. Solve the assignment relaxation. If the solution is a Hamiltonian cycle, stop: it is the optimum solution.

2. Choose a subtour with the smallest number of nodes, in order i(1), ..., i(k), k<n. In the previous solution
$x_{i(1),i(2)} = x_{i(2),i(3)} = \ldots = x_{i(k-1),i(k)} = x_{i(k),i(1)} = 1$.

3. Branching: Define k subproblems by deleting each edge of the subtour in turn, by setting

$x_{i(1),i(2)} = 0$
or $x_{i(2),i(3)} = 0$
⋮ ⋮
or $x_{i(k),i(1)} = 0$.

4. Choose a subproblem to be solved. Solve the associated assignment problem by setting $c_{ij} = \infty$ for the deleted edge. Denote the solution by $\mathbf{x}$, f.

5. (i) If $f < f_{min}$, but subtours exist, go to 2.
(ii) Bounding: if $f < f_{min}$ and the solution has no subtours, update the solution $\mathbf{x}_{min} := \mathbf{x}$, $f_{min} := f$. Go to 6
(iii) If $f \geq f_{min}$, go to 6.

6. If unsolved subproblems exist, go to 4.
Else, stop: the optimum solution is $\mathbf{x}_{min}$, f_{min}.

Example 5.4. TSP with an asymmetric distance matrix C:

i\j	1	2	3	4	5
1	0	2	0	6	1
2	1	0	4	4	2
3	5	3	0	1	5
4	4	7	2	0	1
5	2	6	3	6	0

Initialization: Tour 1-2-3-4-5-1 gives $f_{min} = 10$.

For the Hungarian method, we first set $c_{ii} = \infty$ to force $x_{ii}=0$.

Solution of the associated assignment problem with the Hungarian method:
$x_{12} = x_{21} = x_{34} = x_{45} = x_{53} = 1$, with subtours 1-2-1 and 3-4-5-3, $f = 8 < f_{min}$.

Branching on the first subtour generates subproblems
(1) $x_{12} = 0 \Rightarrow c_{12} = \infty$
(2) $x_{21} = 0 \Rightarrow c_{21} = \infty$.

Solution of subproblem (1): tour 1-3-4-5-2-1, $f = 9 < f_{min}$.
Set $f_{min} := 9$ and save the solution $\mathbf{x_{min}}$.

Solution of subproblem (2): subtours 1-2-5-1 and 3-4-3, $f = 9$.
This lower bound is no better than the incumbent solution.

All subproblems are fathomed.
Optimum solution is the tour 1-3-4-5-2-1, of length $f_{min} = 9$.

The Branch&Bound with Assignment relaxation is simple but inefficient because of its bounding technique. The tree of subproblems may become impracticable. A stricter lower bound can be calculated using other relaxations, e.g. the Lagrangean function for a modified problem and a minimal spanning tree relaxation, resulting in relatively limited branching. These and other Branch&Bound variants are reported in G. Laporte: The Traveling Salesman Problem: An overview of exact and approximate algorithm. European Journal of Operations Research 59 (1992), pp. 231-247.

5.2.2. DYNAMIC PROGRAMMING

Dynamic programming is a divide-and-conquer-strategy with the following features:
The problem is handled is smaller parts in a sequential way so that small subproblems are solved first and their solutions are stored for future reference. Larger subproblems are solved by a recursion formula from the smaller ones. In this way, the next stage is calculated from the previous stage in a bottom-up fashion. Finally, we back-track through the table to obtain the complete solution.

Consider the instance of Example 5.4.

i\j	1	2	3	4	5
1	0	2	0	6	1
2	1	0	4	4	2
3	5	3	0	1	5
4	4	7	2	0	1
5	2	6	3	6	0

Notation:
V = {1,...,n} vertex set
S = subset of V
g(i,S) = length of a shortest path from vertex i to vertex 1 passing through every vertex of S exactly once

In the example, if
$S = \{3\}$, $g(2,S) = c_{23}+c_{31} = 4+5 = 9$
$S=\{3,4\}$, $g(2,S) = \min \{ c_{23}+c_{34}+c_{41}, c_{24}+c_{43}+c_{31}\} = \min \{4+1+4, 4+2+5\} = \min \{9, 11\} = 9$

Recursion formula:

$$g(i,S) = \min_{j \in S} \{c_{ij} + g(j, S-\{j\})$$

The minimum value is

$$f_{min} = \min_{j=2,\ldots,n} \{c_{1j} + g(j, V-\{1,j\})$$

DYNAMIC PROGRAMMING ALGORITHM FOR THE TSP

Input: Number of vertices n, n×n matrix $C = (c_{ij})$

1. For i = 2 to n, set $g(i,\varnothing) = c_{i1}$

2. For k = 1 to n-2
For all subset $S \subseteq V-\{1\}$ containing k vertices
For all $i \notin S$, $i \neq 1$

$$g(i,S) := \min_{j \in S} \{c_{ij} + g(j, S-\{j\})$$

p(i,S) := value of j that gave the minimum

3. Length of optimum TSP-tour:

$$f := g(1,V-\{1\}) = \min_{j=2,\ldots,n} \{c_{1j} + g(j, V-\{1,j\})$$

p(1,V-{1}) := value of j that gave the minimum.

The array p has n rows for all the vertices and columns for all subsets of V-{1},

p(i,S) = the first vertex after i on a shortest path from i to 1 that passes through all the vertices of S exactly once.

The optimal tour is retrieved from the array p:

The first vertex after 1: p(1, V-{1}) = k
The second after 1: p(k, V -{1,k})
etc.

In can be shown that the number of operations is $T(n) = O(n^2 2^n)$ and the memory used in storing the arrays is $M(n) = O(n2^n)$.