Constraint Generation

for Travelling Salesman Problem

Travelling Salesman Problem

A travelling salesman must visit 7 customers in 7 different locations whose (symmetric) distance matrix is:

	1	2	3	4	5	6	7
1	-	86	49	57	31	69	50
2		-	68	79	93	24	5
3			-	16	7	72	67
4				-	90	69	1
5					-	86	59
6						-	81

Formulate a mathematical program to determine a visit sequence, minimizing the travelled distance.

Distances obey a triangular inequality and are symmetric.

Every costumer is visited only once, thus the solution is also a **Hamilton tour**.

Travelling Salesman Problem

Data:

Variables:

Objective function:

Constraints:

	1	2	3	4	5	6	7
1	-	86	49	57	31	69	50
2		-	68	79	93	24	5
3			-	16	7	72	67
4				-	90	69	1
5					-	86	59
6						-	81

Travelling Salesman Problem

Data: G=(V,E) complete graph, $n = |V|$ nodes, for each edge $(i,j) \in E$ c_{ij} cost

Variables: $x_{ij} = 1$ if edge (i,j) is selected for the tour, 0 otherwise.

Objective function:

$$\min \sum_{ij} c_{ij}\, x_{ij}$$

Constraints: $\sum_i x_{ij} = 1 \quad \forall j$: exactly one outgoing edge is selected

$\sum_j x_{ij} = 1 \quad \forall i$: exactly one incoming edge is selected

$$\sum_{\substack{i \in S \\ j \in V \setminus S}} x_{ij} = 1 \quad \forall S \subset V, |S| \geq 1 \qquad OR \qquad \sum_{i,j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subset V, |S| \geq 2$$

From all subset S, there must be an outgoing edge to V\S

OR in all subset S, the number of edges is less than |S| (it is not a subtour)

Constraint Generation

- Having an exponential number of constraints makes any problem hard to solve.
- However, usually only a few of them are really needed.
- Thus, if we solve the problem without these constraints, and adding only those which are not satisfied can speed up the method considerably.
- Here, we only include a subtour elimination constraint, if the solution contains that subtour.

Pseudo-code for constraint generation

Step 1. Solve the problem without the exponentially many constraints (just some of them are included)

Step 2. Check if the problem is feasible:

If feasible: stop, we have the solution of the original problem

If infeasible: find the constraint which is not fulfilled, and include it in the model. Go to **Step 1.**

As almost the same problem is solved each time, one can make a warm start from the last solution and use the dual-simplex method.