Mathematics for Decisions

Integer Linear Programming: TSP and Branch-and-Bound

Romeo Rizzi, Alice Raffaele

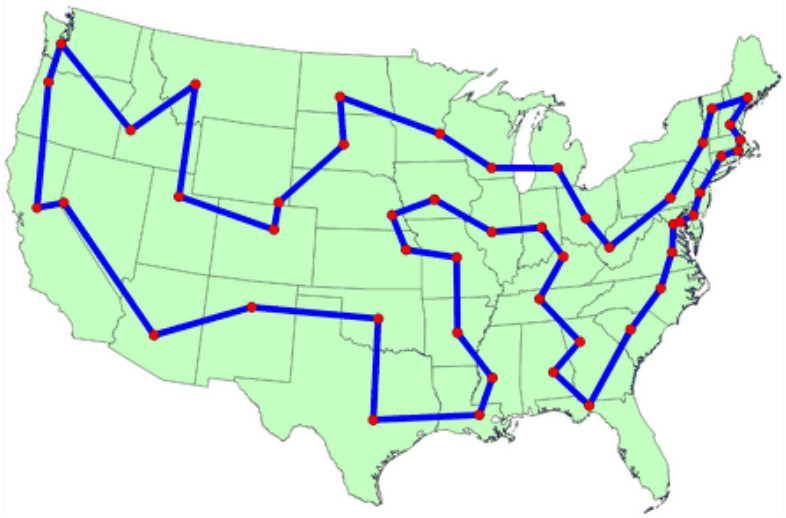
University of Verona

romeo.rizzi@univr.it, alice.raffaele@unitn.it

April 2020

Overview

The Traveling Salesman Problem



"It is shown that a certain tour of 49 cities, one in each of the 48 states and Washington DC, has the shortest road distance"
(Dantzig, Fulkerson and Johnson, 1954)

- **Definition:** a salesman must visit n cities exactly once and must return to the original point of departure; the distance (or cost) between cities i and j is known and values c_{ij}. Find the shortest route computing only one minimum length cycle.

- Since every city must be visited, the solution route will be a cyclic permutation of all cities.
- Given n cities, the number of possible solutions are exactly $(n-1)!/2$, if distances are symmetric $\rightarrow$ It is not so easy to check all possible solutions, even with a computer, especially for large instances:

No. cities	No. tours	Time
5	12	12 microsecs
8	2520	2.5 millisecs
10	181,440	0.18 secs
12	19,958,400	20 secs
15	87,178,291,200	12.1 hours
18	177,843,714,048,000	5.64 years
20	60,822,550,204,416,000	1927 years

- Karp proved in 1972 that the problem is NP-Hard; this dooms exact methods to take exponential time (only) in the worst case. There are many heuristics that quickly yield feasible solutions of reasonable quality. The metric case allows for approximation algorithms.

Some history

- 1766 - Euler studied the **Knight's tour** problem
- 1856 - Kirkman studied the graph of a **polyhedron**, wondering if there could be a circuit which passes through every vertex only once.
- 1856-1859 - Hamilton developed the **Icosian Game**:

- 1884 - Tait offers an elegant proof of the 4-colors Conjecture essentially assuming that Kirkman's conjecture is true for simple polytopes.

- 1930 - Menger was the first who talked about **Hamiltonian path** and studied the general formulation.
- 1946 - William Tutte's counterexample to Tait's (and Kirkman's) conjecture
- 1949 - The **problem name** *Traveling Salesman Problem* appeared for the first time in Julia Robinson's paper "*On the Hamiltonian Game (A Traveling Salesman Problem)*".
- 1954 - Dantzig, Fulkerson and Johnson formulated the **Symmetric TSP**, the first ILP formulation, and then also the **Asymmetric** version.

How to tackle the TSP

There are essentially three categories of algorithms for ILP:

- **Heuristics algorithms**:
  - Approaches guaranteed to yield some output within the allotted time. The quality of the solutions returned and the computational resources used are the main measures compared in the experimental comparisons in the literature, but also the time to design and implement the heuristic makes the difference in the real world.
- **Approximation algorithms**:
  - Polynomial time algorithms that are guaranteed to return feasible solutions scoring objective function values within proven bounds from the optimum.
- **Exact algorithms**:
  - They always return an optimum feasible solution (unless no feasible solution exists).

Heuristics algorithms

- Several heuristics have been developed during the years, among these:
  - **Constructive algorithms** (e.g., various greedy and GRASP approaches like farthest insertion, random insertion, nearest neighbor): these generate feasible solutions starting from the bare instance. Example of *Nearest neighbor*: https://www.youtube.com/watch?v=E85l6euMsd0
  - **Local Search** (e.g., 2-opt, k-opt, Variable Neighborhood Search, Exponential Neighborhood Search). Example of *2-opt*: https://www.youtube.com/watch?v=3zSmjkpvcgw)
  - **Meta-heuristics** (e.g., Simulated Annealing, Tabu Search, Genetic, Memetic and Ant Colony algorithms, etc.)

Approximation algorithms

- Given a minimum problem, let x^* and x^H be respectively the optimal solution and the solution obtained with the approximation algorithm.
- We can say that $\frac{x^H - x^*}{x^*} \leq \varepsilon \rightarrow x^H \leq (1 + \varepsilon)x^*$.
- ε represents the error.
- In problems like TSP, given an instance I (i.e., all distances between n nodes), poly-time algorithms exists to generate a feasible solution.

Approximation algorithms

- Given a minimum problem, let x^* and x^H be respectively the optimal solution and the solution obtained with the approximation algorithm.
- We can say that $\frac{x^H - x^*}{x^*} \leq \varepsilon \rightarrow x^H \leq (1 + \varepsilon)x^*$.
- ε represents the error.
- In problems like TSP, given an instance I (i.e., all distances between n nodes), poly-time algorithms exists to generate a feasible solution.

We'll discuss them soon

Exact algorithms

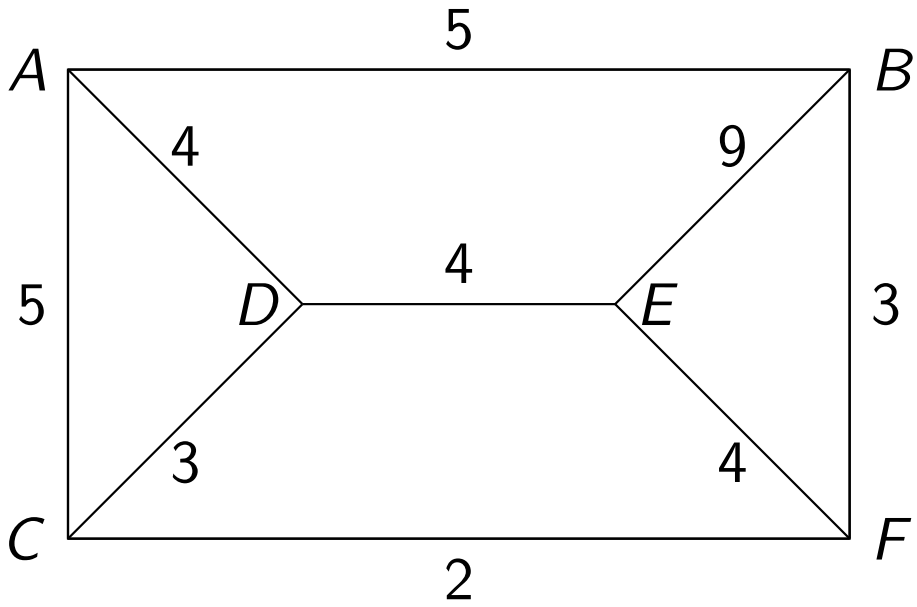
- In operations research, some main general frameworks to design exact algorithms to NP-hard problems are:
  - **Branch-and-Bound**
  - Cutting Planes
  - Branch-and-Cut
- Let's define some basic concepts before discussing Branch-and-Bound.

Lower Bound and Upper Bound

- **Lower Bound (LB)**:
  - Given an instance I, a lower bound to $val(x, I)$ is a real b such that $b \leq val(x, I)$ for every feasible solution x to I.
  - Bound b_2 is tighter than bound b_1 when $b_1 \leq b_2$.
  - A bound b is tight when there exists a feasible solution x such that $val(x, I) = b$.
- **Upper Bound (UB)**:
  - Given an instance I, an upper bound to $val(x, I)$ is a real B such that $B \geq val(opt(I), I)$.
  - Bound B_1 is tighter than bound B_2 when $B_1 \leq B_2$.
  - A bound b is tight when $B = val(opt(I), I)$.

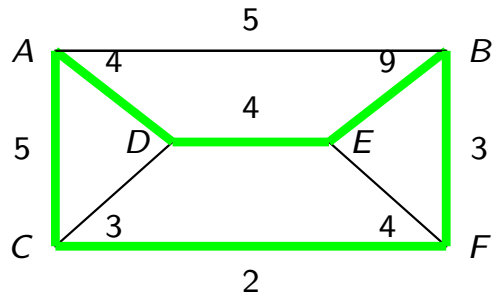
- By a lower bound (upper bound) for a certain problem we often mean a closed formula or a poly-time algorithm that, given I, yields a lower bound (upper bound) for I.
- We will see how lower bounds and upper bounds can both play helpful, especially the tighter they happen to be.
- Heuristics and approximation algorithms provide upper bounds.
- Approximation algorithms also provide lower bounds. Why?

TSP Example

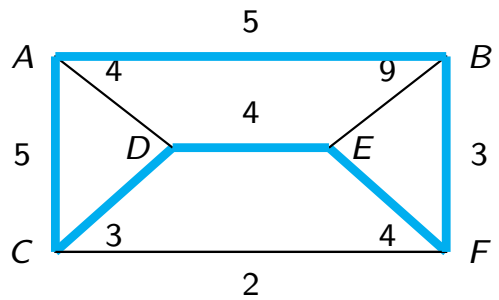


What is a LB for this simple graph? And an UB?

- This is a feasible solution of value 27:



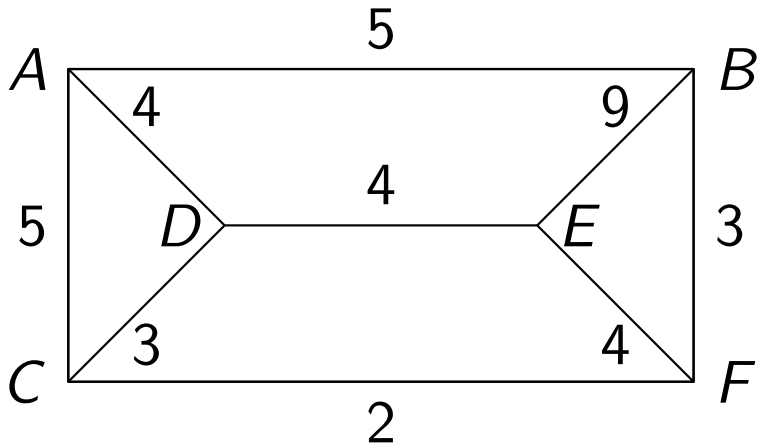
- It certifies that 27 is an upper bound on the minimum cost. Once we know it, we will never be willing to pay more than 27.
- The following solution of cost 24 provides a tighter (more interesting) upper bound:



- Thus for minimization problems, feasible solutions represent upper bounds.

Lower bound for the TSP

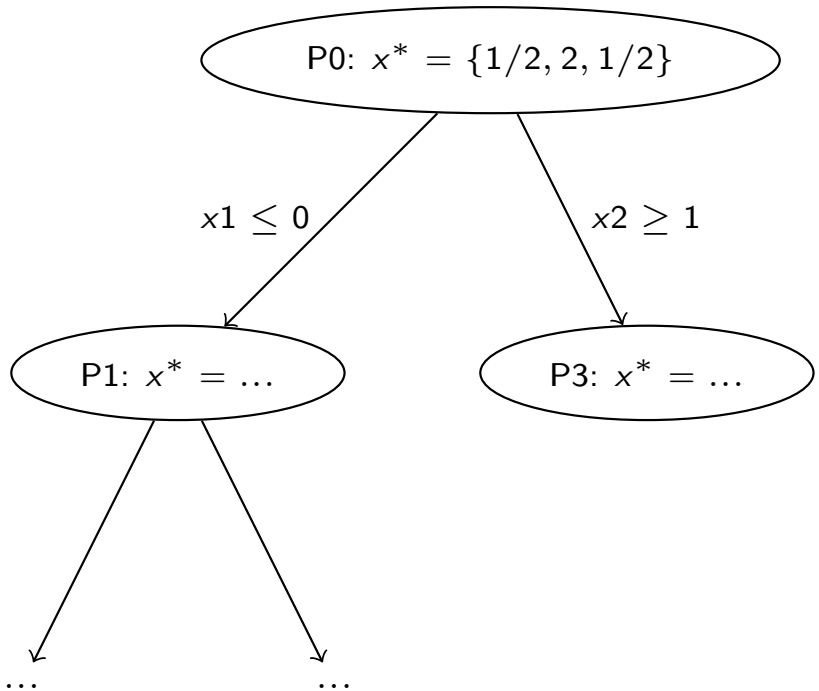
- The cost of any tour is $\frac{1}{2}\sum$ (sum of the costs of the two tour edges adjacent to v).
- The sum of the two tour edges adjacent to a given vertex $v \geq$ sum of the two edges of least cost adjacent to v.
- The cost of any tour is $\geq \frac{1}{2}\sum$ (sum of the costs of the two least cost edges adjacent to v).
- In our example, LB $= \frac{1}{2}$ [(4+5) + (3+5) + (2+3) + (3+4) + (4+4) + (2+3)] $= \frac{1}{2}$ 42 $= 21$.



- Let's introduce an exact method that exploits bounds...

Branch-and-Bound

- It essentially explores all the solutions but, to avoid seeing all of them (like a brute-force algorithm), it implicitely **enumerates** them exploring a tree and **branching** on possible elementary choices.
- It avoids exploring entire subtrees comparing **lower bounds and upper bounds**, pruning some branches.

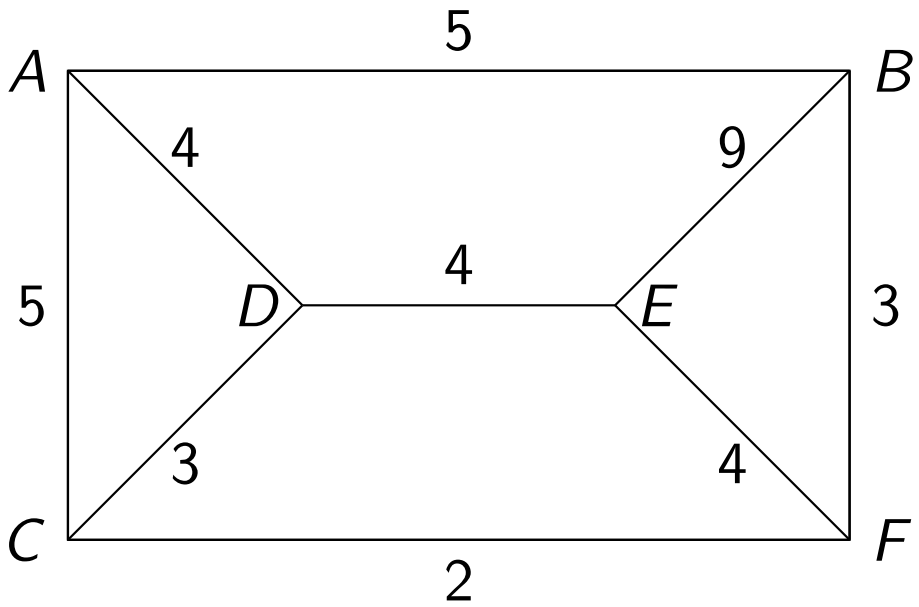


- At any node of the tree, the algorithm makes a finite decision and sets one of the unbound variables.
- A node (sub-problem) is *solved* when:
  - its optimal solution is found;
  - it admits no feasible solution;
  - it has become apparent that all possible solutions to the subproblem are worse then the best one found so far.

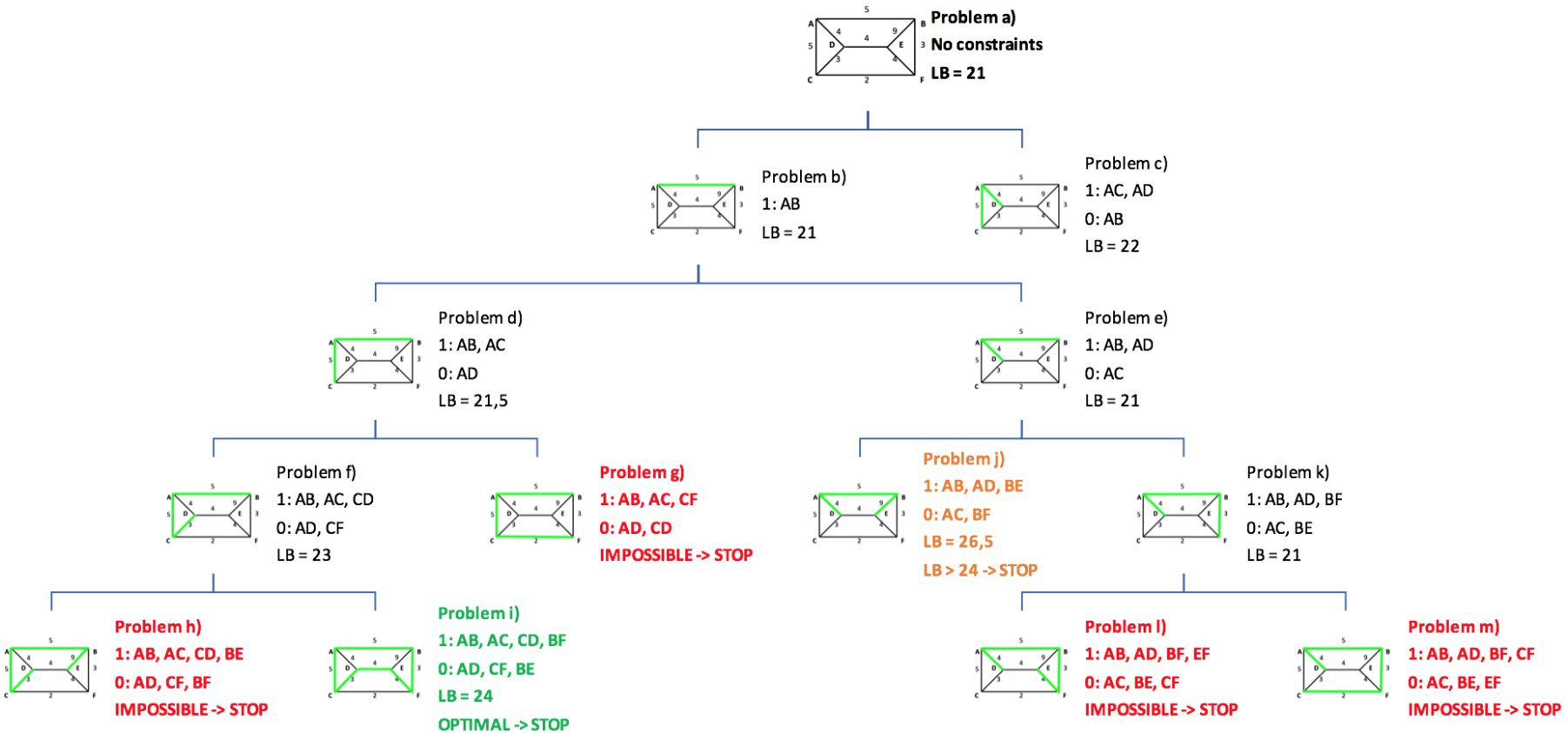
  If it is not possible to find its optimal solution, then branch.

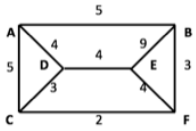
TSP Example with Branch-and-Bound

Let's try to solve this instance:

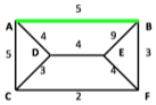


A tree search (Branch-and-Bound)

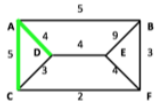




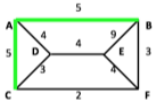
Problem a)
No constraints
LB = 21



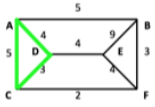
Problem b)
1: AB
LB = 21



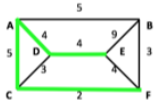
Problem c)
1: AC, AD
0: AB
LB = 22



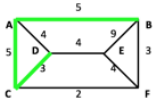
Problem d)
1: AB, AC
0: AD
LB = 22,5



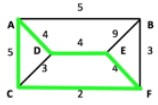
Problem n)
1: AC, AD, CD
0: AB, CF
IMPOSSIBLE -> STOP



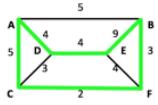
Problem o)
1: AC, AD, CF, DE
0: AB, CD
LB = 22,5



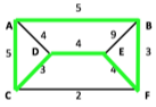
Problem f)
1: AB, AC, CD
0: AD, CF
LB = 23



Problem p)
1: AC, AD, CF, EF
0: AB, CD, BF
IMPOSSIBLE -> STOP

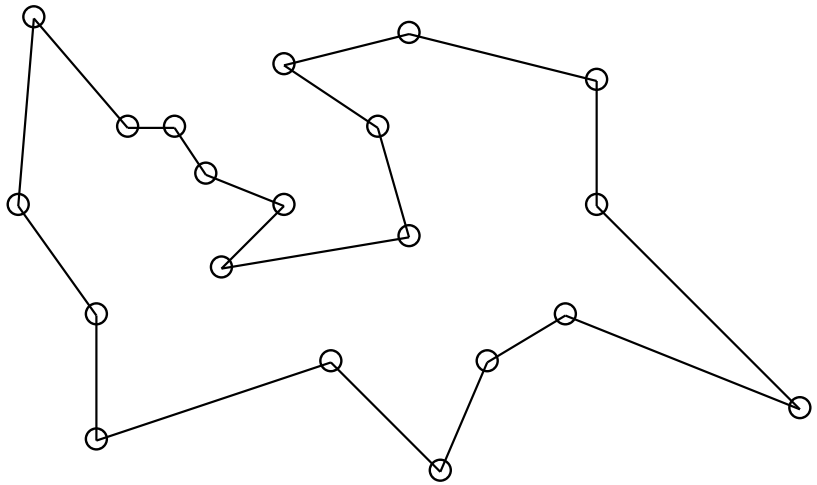


Problem q)
1: AC, AD, CF, DE, EF, BE
0: AB, CD, BF
LB = 27
LB > 24 -> STOP



Problem i)
1: AB, AC, CD, BF
0: AD, CF, BE
LB = 24
OPTIMAL -> STOP

TSP Mathematical Formulation



- The distance between two vertices is the same in both directions (**undirected graph**).
- Let V be the set of vertices and E the set of edges.
- We introduce a binary variable x_e for every edge $e \in E$: $x = 1$ iff the edge e is included in the tour.
- The objective function is to minimize the total cost of edges selected, i.e. traveled by the salesman.

- For every vertex v, precisely two of the edges incident with v are selected.
- For any subset $S \subset V$, let $\delta(S) =$ be the edges with one end-vertex in S.
- We obtain the following initial formulation, where constraints (2), which are called *assignment constraints*, ensure that every vertex is visited exactly twice:

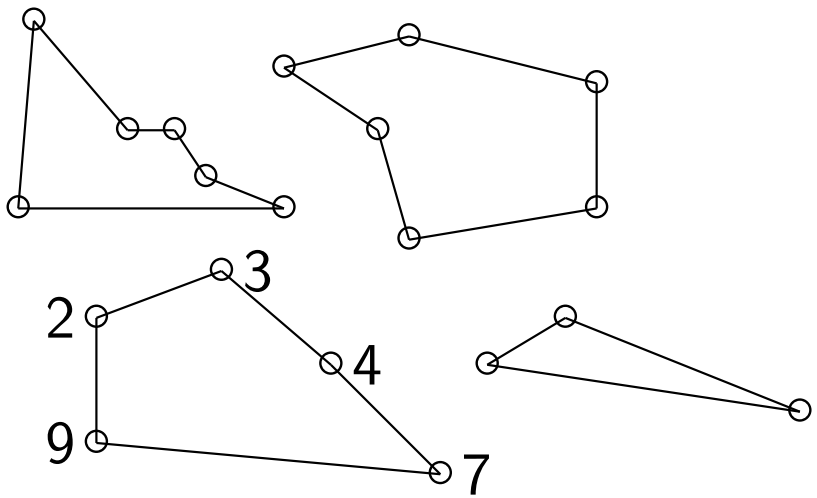
$$\textit{min} \quad \sum_{e \in E} c_e x_e \tag{1}$$

$$\text{subject to} \quad \sum_{e \in \delta(v)} x_e = 2, \ \forall v \in V \tag{2}$$

$$x_e \in \{0, 1\}. \tag{3}$$

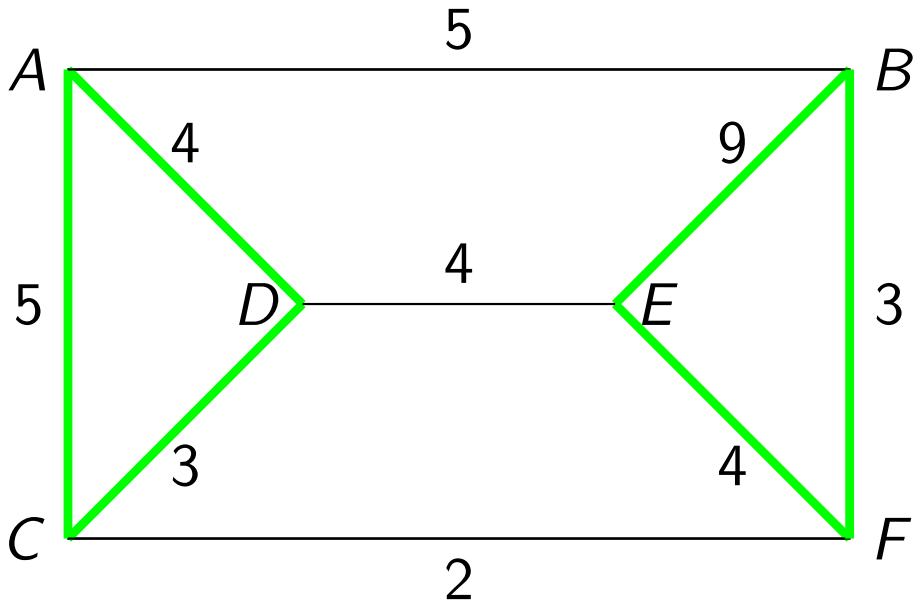
...but it's not enough!

A solution like the following in the picture would be feasible according to the above formulation:



But we do not want our salesman to "teleport" or "jump" from one city to another $\rightarrow$ Our formulation is hence incomplete.

In fact, in the example before, we could have obtained this:



This solution can be used as a lower bound for the problem.
But, in order to formulate the correct model for TSP, we need to add more constraints...

AMPL Example

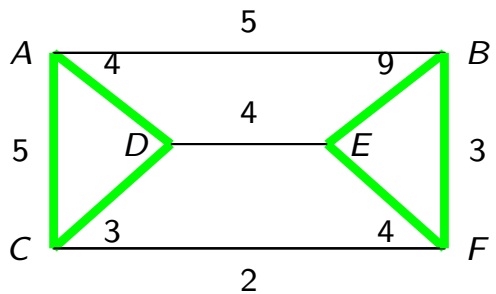
Consider the same example but change costs of (B,E) from 9 to 2 and (C,F) from 2 to 8:

Model:

```
set V ordered;
set E within {i in V, j in V: ord(i) < ord(j)};
param cost{E};
var X{E} binary;
minimize TourCost: sum {(i,j) in E} cost[i,j] * X[i,j];
subject to VisitAllVertices {i in V}:
sum {(i,j) in E} X[i,j] + sum {(j,i) in E} X[j,i] = 2;
```

Data:

```
set V := A B C D E F;
set E := (A,B) (A,C) (A,D)
         (B,E) (B,F) (C,D)
         (C,F) (D,E) (E,F);
param cost :=
[A,B] 5
[A,C] 5
[A,D] 4
[B,E] 2
[B,F] 3
[C,D] 3
[C,F] 8
[D,E] 4
[E,F] 4;
```



The AMPL solver will find the optimal solution with value 21, choosing edges (A,C), (A,D), (B,E), (B,F), (C,D), (E,F).
But it is not what we want...

Subtour Elimination Constraints

- TSP asks for an **Hamiltonian circuit**: a circuit passing through each vertex exactly once, without subtours.
- Given $S \subsetneq V$, $S \neq \emptyset$, the cut with shores S and V/S, denoted $\delta(S)$, is the set of those edges with one endpoint in S and the other in V/S.
- We must impose that the solution is **connected**. This means that, for every proper subset of vertices, at least two selected edges have one endpoint in S and the other outside S:

$$\sum_{e \in \delta(S)} x_e \geq 2, \forall S \subset V : 2 \leq |S| \leq |V| - 2$$

- Violated constraints will have the following formulation:

$$\sum_{e \in \delta(S)} x_e^* < 2.$$

- These are called *Subtour Elimination Constraints* (SEC).

The SEC problem

- Introducing this family of constraints, we are adding an **exponential** number of constraints, since we are considering the power set of V (except the empty set, sets composed of single vertices and the set with all vertices).
- This approach is not polynomial for a double reason:
  - we adopted an ILP model;
  - its formulation has an exponential number of variables and constraints.

TSP Formulation

- Putting together the objective function (1) and constraints (2), (3) and (4) , we get to the correct formulation:

$$min \sum_{e \in E} c_e x_e \tag{1}$$

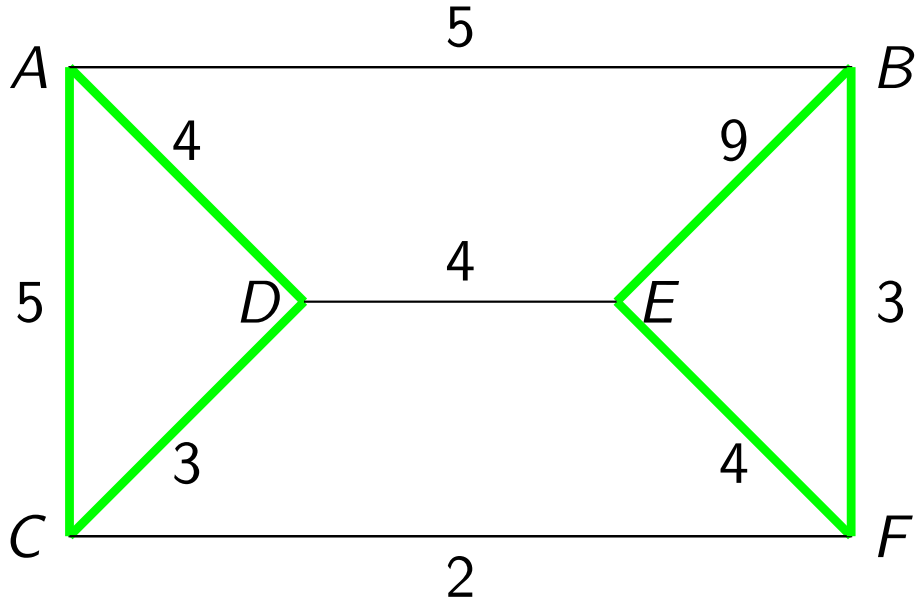
$$\text{subject to} \quad \sum_{e \in \delta(v)} x_e = 2, \ \forall v \in V \tag{2}$$

$$\sum_{e \in \delta(S)} x_e \geq 2, \forall S \subset V : 2 \leq |S| \leq |V| - 2 \tag{3}$$

$$x_e \in \{0, 1\}. \tag{4}$$

Another approach considering flow constraints

- Let's think about another approach that does not imply adding an exponential number of constraints.
- Consider the wrong solution with subtours we obtained before:



- Suppose we want to send some quantity of flow from vertex A to another vertex:
  - since A is connected to C and D, there is no problem to send it to them;
  - instead, if we want to send the flow to F, we cannot.

Adding flow constraints

- We introduce two variables $y_{(u,v)}$ and $y_{(v,u)}$ for every edge $e = (u, v) \in E$ and we allow to transmit the flow only through edges selected in the cycle:
  - $y_{(u,v)} \leq x_{(u,v)}$
  - $y_{(v,u)} \leq x_{(u,v)}$.
- We consider the vertices A and F respectively as the source and destination; we want that the outgoing flow from A is equal to 1, whereas the incoming flow is 0:
  - $y(\delta^+(A)) = 1$
  - $y(\delta^-(A)) = 0$.
- We want all other vertices but F to conserve the flow:
  - $y(\delta^-(u)) = y(\delta^+(u)), \forall u \neq A, F$

- In this way, we are forcing the flow to arrive exactly in the only vertex left, i.e. F, connecting A and F.
- This could be repeated for every source vertex and for every destination vertex.
- This, since the number of vertices is n, does not seem exponential at all, but we are adding a polynomial number of constraints and variables.

Conclusions

- We defined the TSP, using one easy formulation and one more complete and beautiful but, unfortunately, with an exponential number of constraints.
- We saw several methods to tackle a ILP problem (heuristics, approximation and exact algorithms), according to the purpose.
- We solved an easy instance of TSP using Branch-and-Bound.
- In Appendix you can find the description of Asymmetric TSP.

References


University of Waterloo, *TSP*
http://www.math.uwaterloo.ca/tsp/


R.Mansini, *Algoritmi di Ottimizzazione*, University of Brescia
https://www.unibs.it/ugov/degreecourse/65037

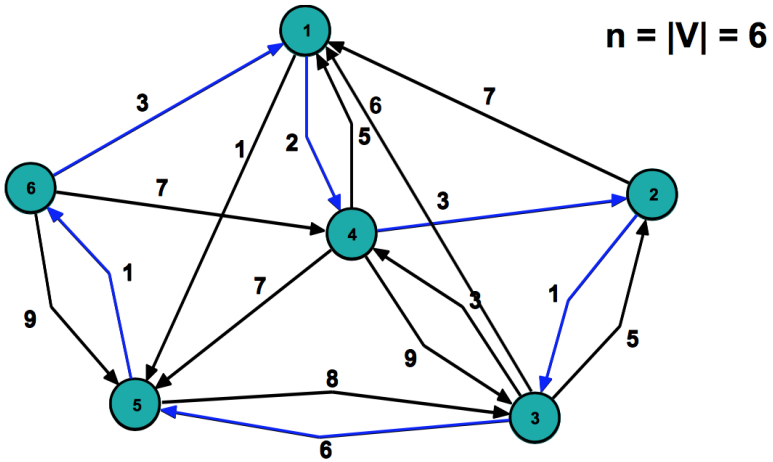

Wikipedia, *Traveling Salesman Problem*
https://en.wikipedia.org/wiki/Travelling_salesman_problem


Gilbert Laporte, *A Short History of the Traveling Salesman Problem*
http://neumann.hec.ca/chairedistributique/common/laporte-short.pdf


Springer, *Encyclopedia of Optimization*
http://www.springer.com/it/book/9780387747583


Operations Research Group, *Asymmetric TSP*, University of Bologna
http://www.or.deis.unibo.it/algottm/files/8_ATSP.pdf

Appendix A - The Asymmetric TSP



- There can be cases where going from vertex i to vertex j does not cost the same then going from vertex j to vertex i or maybe it would not be possible to go through both directions.
- We need to use a set of arcs A, instead of the set of edges E (**directed graph**): this variant is called **asymmetric** TSP.
- Let V be the set of vertices.
- We introduce a binary variable $x_{i,j}$ for every arc $(i, j) \in A$: if $x_{i,j} = 1$, then arc (i, j) appears on the tour.

ATSP Formulation

- Keeping in mind what we said about symmetric TSP, we can formulate the asymmetric TSP as follows:

$$min \sum_{(i,j)\in A} c_{i,j}x_{i,j} \tag{1}$$

$$\text{subject to} \quad \sum_{j\neq i} x_{i,j} = 1,\ \forall i \in V \tag{2}$$

$$\sum_{i\neq j} x_{i,j} = 1,\ \forall j \in V \tag{3}$$

$$\sum_{i\in S, j\in S} x_{i,j} \leq |S| - 1, \forall S \subset V, S \neq \emptyset \tag{4}$$

$$x_{i,j} \in \{0,1\}. \tag{5}$$

- With constraints (2) and (3), we are distinguishing arcs that enter in a certain vertex and arcs that exit from it.

Appendix B

From **EURO 2019**: William Cook,
The TSP: postcards from the edge of impossibility