Mathematics for Decisions

Integer Linear Programming: Cutting Planes, Branch&Cut and the Separation problem

Romeo Rizzi, Alice Raffaele

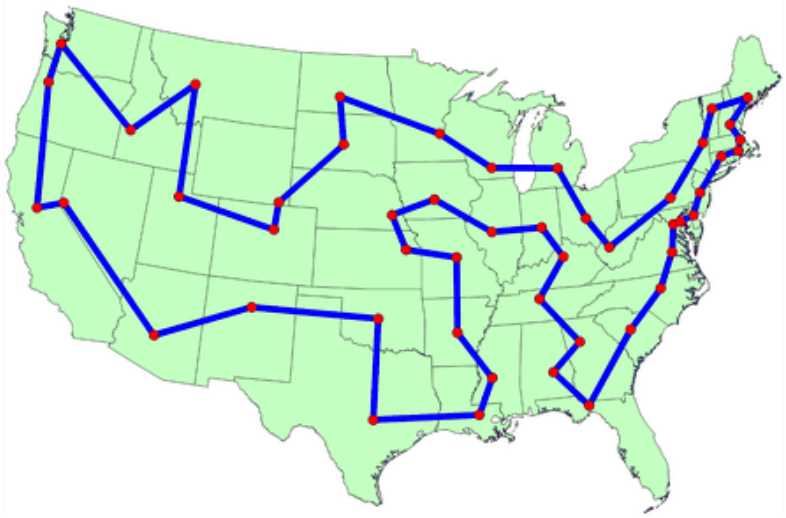
University of Verona

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

December 18, 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)

Problem definition: a salesman must visit n cities exactly once and must return to the original point of departure; the distance (or cost) between each pair of cities (i, j) is known and denoted by c_{ij}. Find the salesman's shortest route by 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 is exactly $(n-1)!/2$, when distances are symmetric $\rightarrow$ It is not so easy to check all possible solutions, even with a computer, especially when solving 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

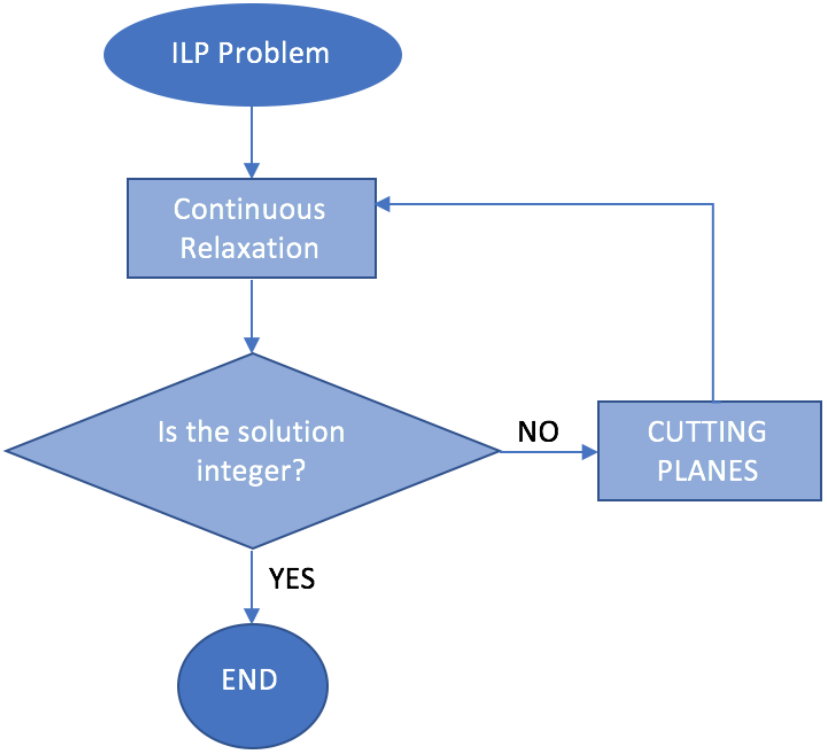
- In 1972, Karp proved the problem to be NP-Hard: this dooms exact methods to take exponential time (only) in the worst case; anyway, there are many heuristics that quickly yield feasible solutions of reasonable quality. Moreover, the metric case allows for approximation algorithms.

Exact algorithms

- Main general frameworks to design exact algorithms for NP-hard problems:
  - **Branch-and-Bound**
  - **Cutting Planes**
  - **Branch-and-Cut**

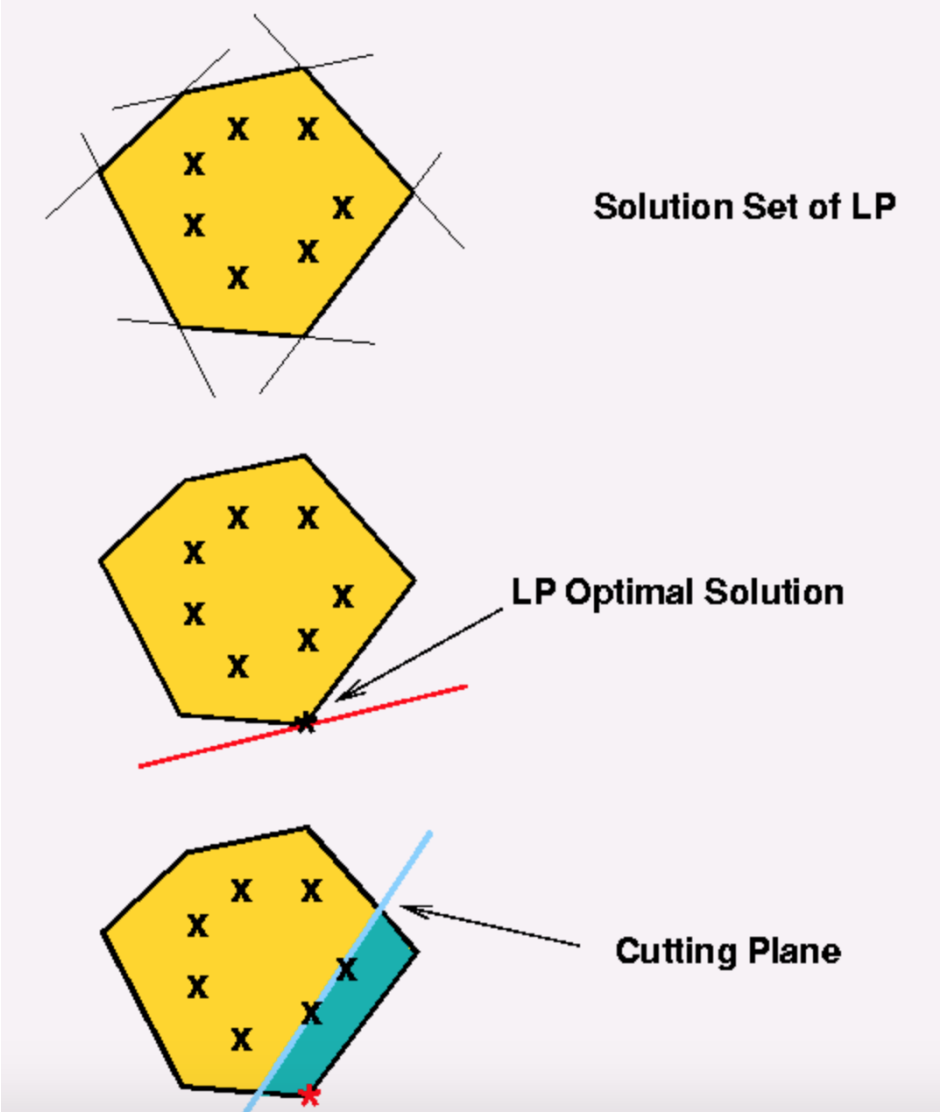
Cutting Planes

Also this method starts from the continuous relaxation of the original ILP to solve it.

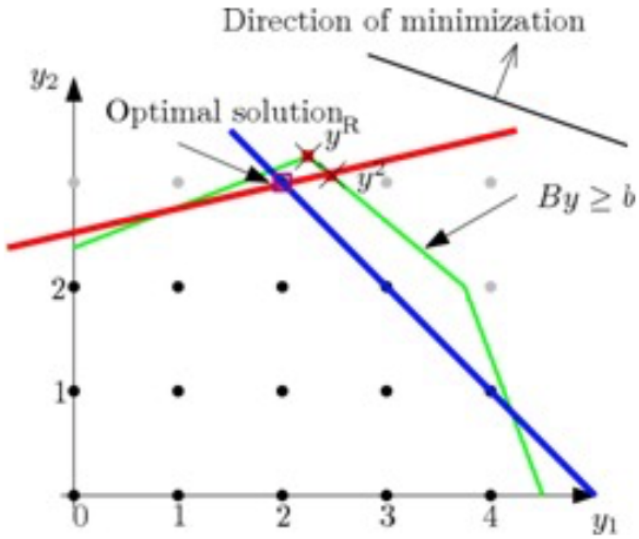


- If the solution found is not integer, the method looks for any linear inequality that divides integer solutions from the fractional obtained $\rightarrow$ The **Separation problem** is solved to find such an inequality (i.e., a cut) and add it to the current problem. Then, the procedure is repeated.
- Otherwise, stop.

- The *cutting plane* removes the (non-integer) optimal solution from the relaxed linear program.
- The inequality is **valid** if it does not change the set of integer solutions of the problem.
- **Note:** if the process does not finish, it does not return a valid solution.



Branch-and-Cut



Let's put together these two methods: we design a Branch-and-Bound where, in every node t of the tree, some cuts are generated in order to obtain:

- an integer solution of the continuous relaxation;
- a tighter lower bound.

The branching operation is done only when cuts are ineffective; thus, tailing off is avoided.

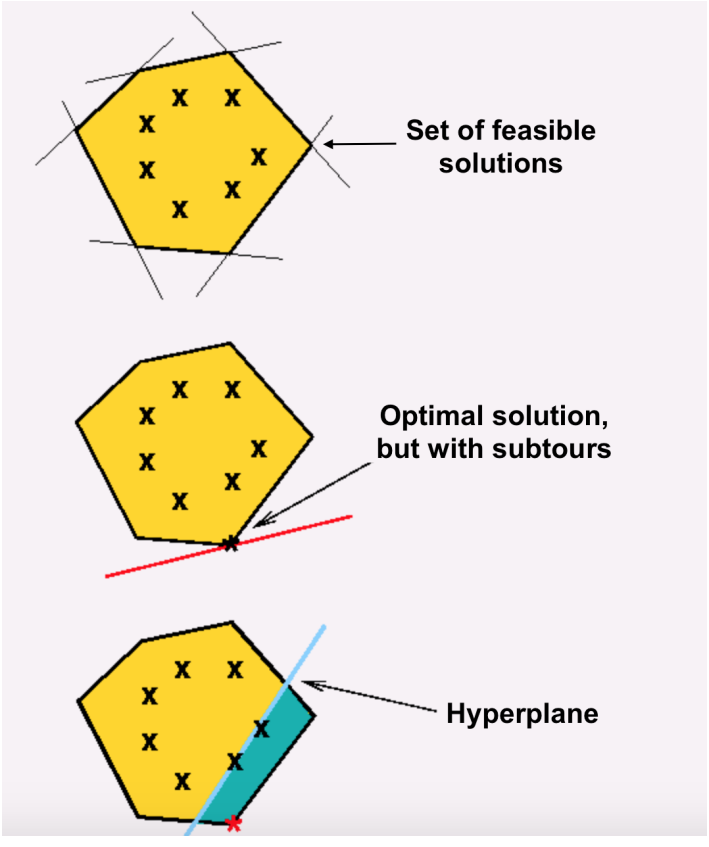
How Branch-and-Cut works

In every node t of the tree, before branching:

1. It solves the continuous relaxation of the problem, finding x^*.
2. If x^* is fractional, it looks for any violated constraint and adds it to the problem, then solving it again.
3. It repeats Step 2 until all violated constraints have been added; if the solution is still fractional, it uses a *separation process* to find new cuts or it branches and restarts the procedure.

The Separation problem

- Given an instance of an integer programming problem and a point x, determine whether x is in the convex hull of the feasible integral points. Moreover, if it is not in the convex hull, find a separating hyperplane that cuts off x from the convex hull.
- The algorithm that solves the Separation problem is called **Separation oracle**.



The Separation Oracle

- We need an algorithm that takes a solution to an ILP problem as input and either certifies if it is feasible (in the case of TSP, that it does not contain any subtours) or it returns as output the hyperplane corresponding to the cut.

The Separation Oracle

- We need an algorithm that takes a solution to an ILP problem as input and either certifies if it is feasible (in the case of TSP, that it does not contain any subtours) or it returns as output the hyperplane corresponding to the cut.

What is separation here, in the case of TSP?

By writing the connectivity constraints, we realize that separation can be seen as a *minimum cut problem*.

Minimum Cut

- Let's consider x^* as our current solution.
- We can use a **support graph** G^*: the graph with $V^* = V$ and $E^* = \{e \in E : x_e^* > 0\}$.
- We assign to each $e \in E^*$ the weight x_e^* .
- A violated SEC exists if and only if there is a cut in G^* whose x^*-weight is less than 2.
- To compute the min cut, we can use Karger's algorithm or we can take advantage of the **Max-flow Min-Cut theorem**.

Max Flow

- We want to determine the max flow that can be sent from a source node $s \in S$ to a terminal node $t \in V \backslash S$.
- $\sum_{e \in \delta(S)} x_e$ is precisely the value of a maximum flow from a node in S to a node in $V \backslash S$.
- We repeat this for each vertex as node s. If we find a flow smaller than 2, then we find a subtour and which constraint is violated, which will be added to the formulation.
- A famous greedy algorithm that solves Max-Flow problems is **Ford-Fulkerson**, which is polynomial.

Separation = Optimization

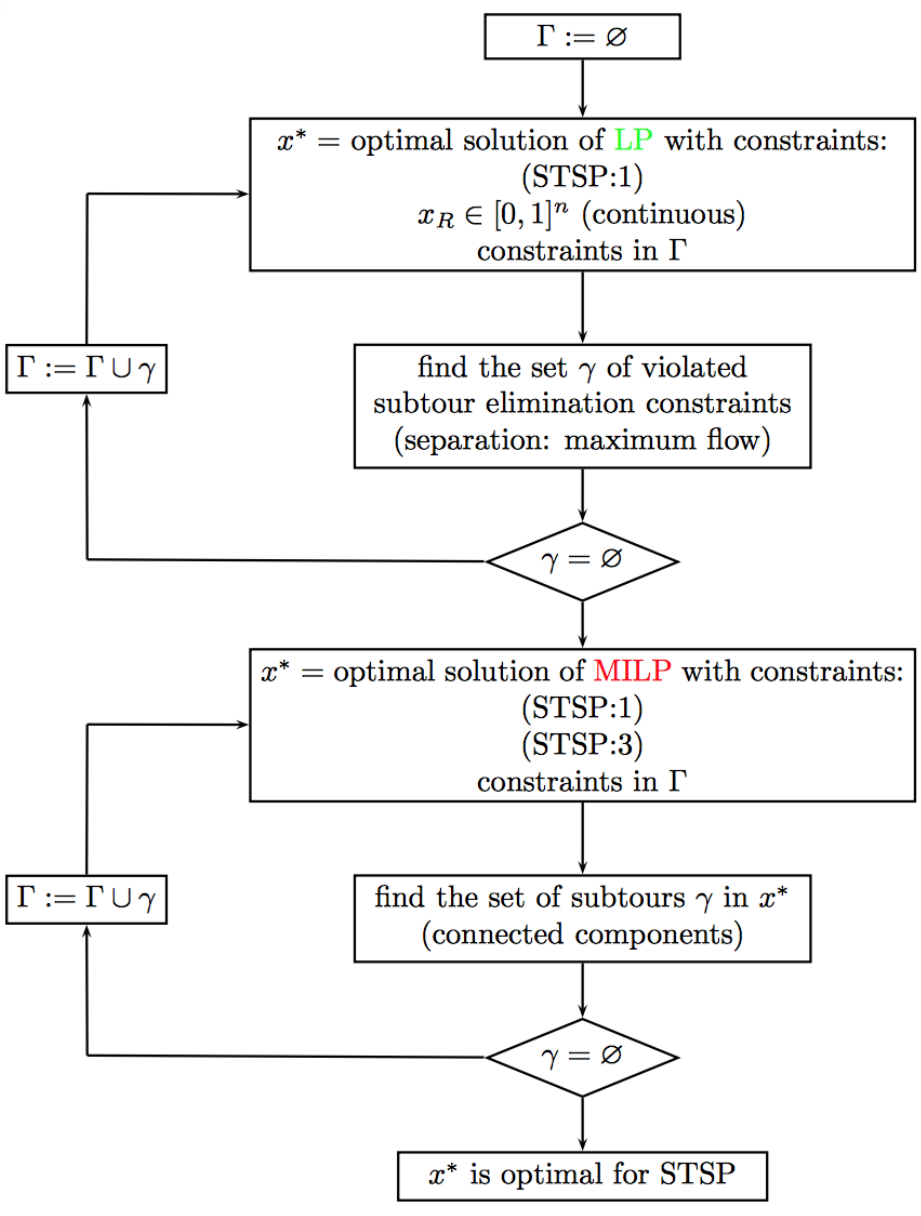
Theorem

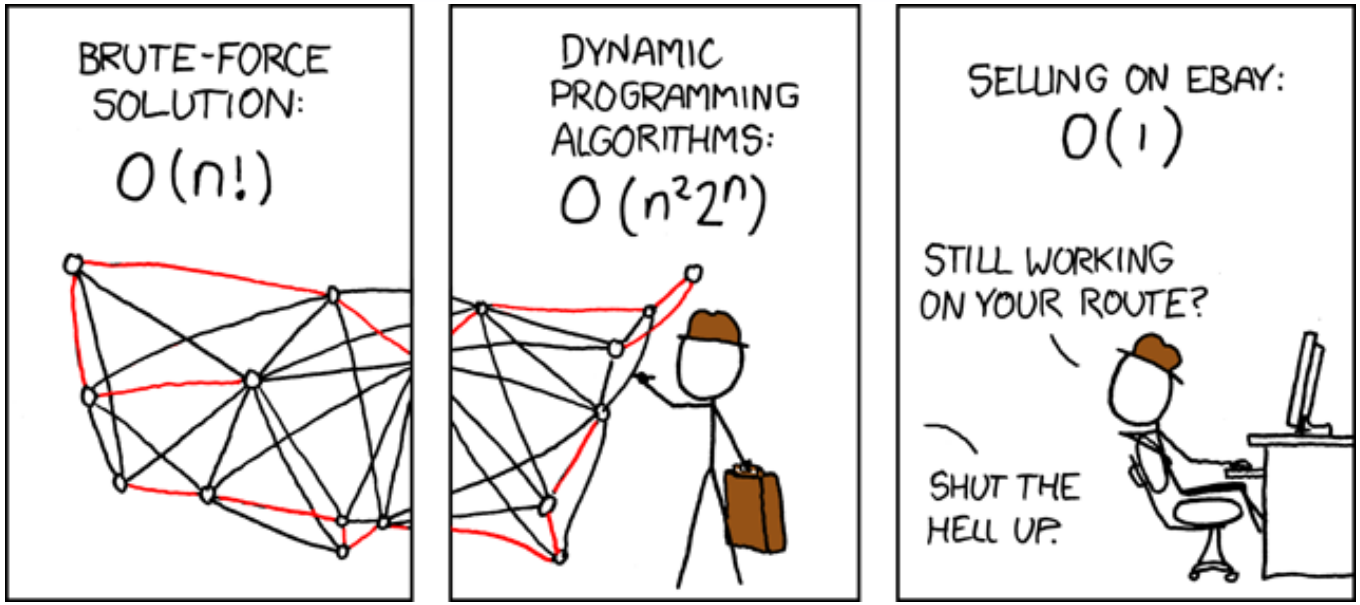
Given a linear program, if we can solve the separation problem in polynomial time, then we can solve the optimization problem in polynomial time too, by using the Ellipsoid method (1979).

This is exactly what we are doing here, by applying Max-Flow Min-Cut theorem and a polynomial algorithm like Ford-Fulkerson's to solve the Separation problem.

A possible procedure

- We can remove the $O(2^n)$ connectivity constraints and add only the ones violated by the current solution $\rightarrow$ We need to check if the graph has **only one connected component**.
- By applying Branch-and-Bound:
  1. Relax integrality constraints and remove SECs.
  2. At every iteration, solve a linear programming problem and obtain the current solution (and a LB for the integer problem).
  3. If the solution does not contain any subtours (i.e., if it is composed of only one connected component):
     - 3.1 If the solution is integer, end;
     - 3.2 Otherwise, branch on a fractional variable.
  4. Else, if the solution contains any subtours, solve the **Separation problem**, add the violated constraints to the problem formulation and go back to Step 2.


$\Gamma := \varnothing$
x^* = optimal solution of LP with constraints:
(STSP:1)
$x_R \in [0,1]^n$ (continuous)
constraints in Γ
find the set γ of violated
subtour elimination constraints
(separation: maximum flow)
$\Gamma := \Gamma \cup \gamma$
$\gamma = \varnothing$
x^* = optimal solution of MILP with constraints:
(STSP:1)
(STSP:3)
constraints in Γ
find the set of subtours γ in x^*
(connected components)
$\Gamma := \Gamma \cup \gamma$
$\gamma = \varnothing$
x^* is optimal for STSP



"*What's the complexity class of the best linear programming cutting-plane techniques? I couldn't find it anywhere. Man, the Garfield guy doesn't have these problems...*'

In 1987, Chvátal, Cook and Hartmann showed that a strong variant of Branch-and-Cut requires at least $\frac{2^{n/72}}{n^2}$ operations to solve a specially constructed nasty instance of the Hamiltonian-circuit problem; other TSP instances may require even more steps.

Conclusions

- We studied another exact method (Cutting Planes) which, together with Branch&Bound, allowed us to define another powerful method (Branch&Cut).
- We discussed about **Separation and Optimization**, by introducing a *Separation oracle*.
- We applied this together with the Max-Flow Min-Cut theorem.
- Now we're missing only the fun part: the implementation with Gurobi!

References

- William J. Cook, *In Pursuit of the Traveling Salesman – Mathematics at the Limits of Computation*, https://press.princeton.edu/books/paperback/9780691163529/in-pursuit-of-the-traveling-salesman
- Explain xkcd, *399: Travelling Salesman Problem*, https://www.explainxkcd.com/wiki/index.php/399:_Travelling_Salesman_Problem
- Renata Mansini, *Algoritmi di Ottimizzazione*, University of Brescia, https://www.unibs.it/ugov/degreecourse/65037
- University of Waterloo, *TSP*, http://www.math.uwaterloo.ca/tsp/
- Wikipedia, *Traveling Salesman Problem*, https://en.wikipedia.org/wiki/Travelling_salesman_problem

Appendix A – The Cutting Plane Game

https://gonzalomunoz.org/the-cutting-plane-game/