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

May 2020

Overview

Recap

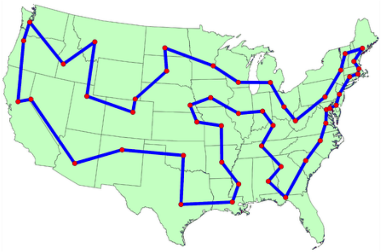
Cutting Planes

Branch-and-Cut

Separation and Optimization

Conclusions

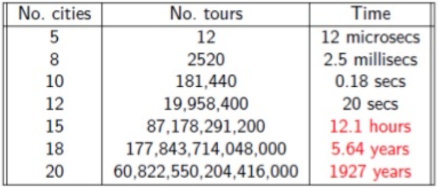
The Traveling Salesman Problem



"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

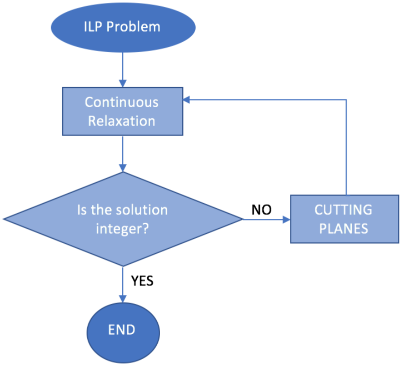
- Karp proved in 1972 that the problem is NP-Hard; this dooms exact methods to take exponential time (only) in the worst case. There are many heuristics that quickly yield feasible solutions of reasonable quality. The metric case allows for approximation algorithms.

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**

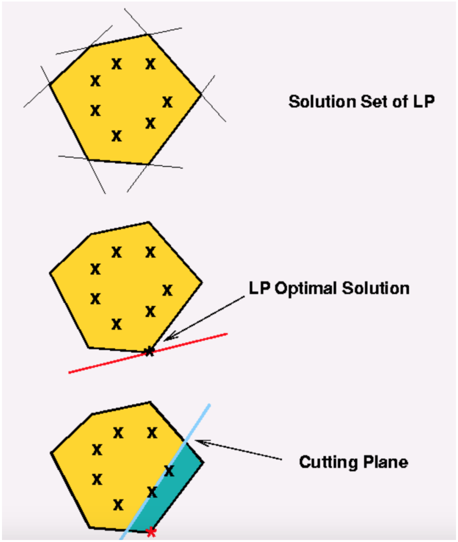
Cutting Planes

Also this method starts from the continuous relaxation of the problem and solves it, obtaining a solution.

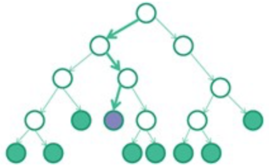


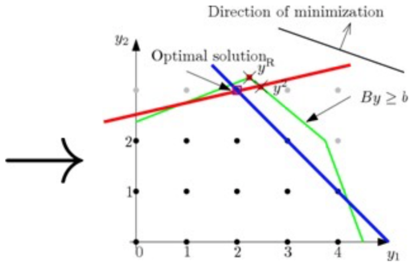
- If the obtained solution is not integer, it looks for a linear inequality that divides integer solutions from fractional ones $\rightarrow$ The **Separation problem** has to be solved, to find the inequality, which is a cut to add to the current problem. Then the procedure is repeated.
- Otherwise, the process ends.

- 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: 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; therefore tailing off is avoided.

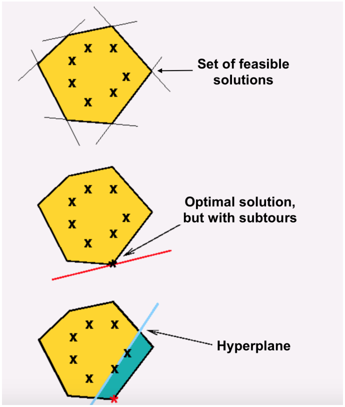
How does Branch-and-Cut work

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 the violated constraint and adds it to the problem, 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 feasible integral points. Further, 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**.


Set of feasible solutions
Optimal solution, but with subtours
Hyperplane

The Separation Oracle

- We need an algorithm that takes the solution as input and either certifies if it is feasible (in this case, it means that it does not contain subtours) or it gives as output the hyperplane corresponding to the cut.

What is separation here?

- Writing the connectivity constraints, we see that it can be seen as a **minimum cut problem**.

Minimum Cut

- Let's consider x^* as our current solution.
- We can use the **support graph** G^*: the graph with $V^* = V$ and $E^* = \{e \in E : x_e^* > 0\}$.
- We give 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.
- We can use an algorithm such as Karger's or we can take advantage of the **Max-flow Min-Cut theorem**.

Max Flow

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

Separation = Optimization

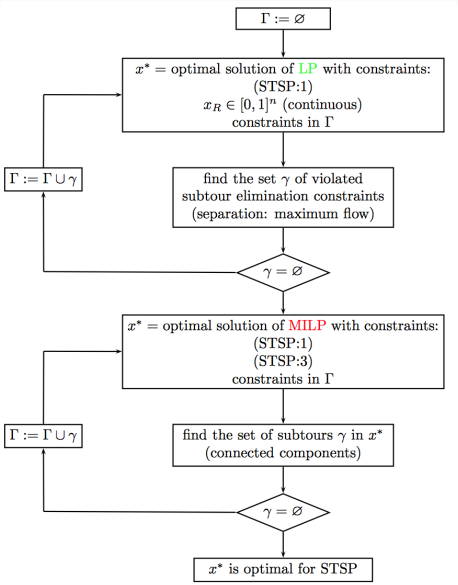
Theorem

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, using the Ellipsoid method (1979).

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

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**.
- Possible procedure, 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 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 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
$\Gamma := \Gamma \cup \gamma$
find the set γ of violated subtour elimination constraints (separation: maximum flow)
$\gamma = \varnothing$
x^* = optimal solution of MILP with constraints:
(STSP:1)
(STSP:3)
constraints in Γ
$\Gamma := \Gamma \cup \gamma$
find the set of subtours γ in x^* (connected components)
$\gamma = \varnothing$
x^* is optimal for STSP

Conclusions

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