Lecture

Solving combinatorial optimization problems using mathematical programming

Section 2 : Non-compact MILP and Branch&Cut

Pierre Fouilhoux

Université Sorbonne Paris Nord - LIPN CNRS

Hanoi, Vietnam, 2024

Combinatorial Optimization Problem

To find a greatest (smallest) element within a valuated finite set.

Combinatorial Optimization Problem

To find a greatest (smallest) element within a valuated finite set.

Given :
- a finite subset of elements $E = \{e_1, \ldots, e_n\}$
- a **solution set** $\mathcal{F}$ of subsets of E
- a weight $c = (c(e_1), \ldots, c(e_n))$

a **Combinatorial Optimization Problem** is to find a solution $F \in \mathcal{F}$ whose weight $c(F) = \sum_{e \in F} c(e)$ is maximum (or min.),

$$i.e. \quad \max \{c(F) \mid F \in \mathcal{F}\}.$$

"Natural" MIP formulation

Binary variable $x_e = \begin{cases} 1 & \text{if } e \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$ for every $e \in E$.

"Natural" MIP formulation

Binary variable $x_e = \begin{cases} 1 & \text{if } e \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$ for every $e \in E$.

$$\max \sum_{e \in E} c(e) x_e$$
$$Ax \leq b$$
$$x_e \in \{0,1\} \quad \forall e \in E.$$

We will suppose here that :

- $Ax \leq b$ is known
- the linear relaxation of this MIP can be obtained

"Natural" MIP formulation

Binary variable $x_e = \begin{cases} 1 & \text{if } e \text{ is chosen} \\ 0 & \text{otherwise} \end{cases}$ for every $e \in E$.

$$\max \sum_{e \in E} c(e) x_e$$
$$Ax \leq b$$
$$x_e \in \{0,1\} \quad \forall e \in E.$$

We will suppose here that :
- $Ax \leq b$ is known
- the linear relaxation of this MIP can be obtained

Important remark : $Ax \leq b$ can be **non-compact**,
i.e. can contain an exponential number of inequalities !

1. Compact / non-compact formulations
2. Cutting Plane based algorithm
3. Branch&Cut algorithm
4. Reinforcement of relaxation value
5. The travelling Salesman Problem (TSP)

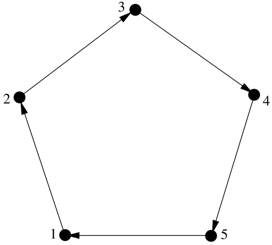
The Acyclic Induced subgraph problem

Let $G = (V, A)$ be a directed graph
with $n = |V|$ nodes and $m = |A|$ arcs.

A **circuit** of G : a sequence of arcs

$$C = (i_1 i_2, i_2 i_3, \dots, i_{k-1} i_1)$$

Notation :
$V(C)$ is the set of nodes of C.

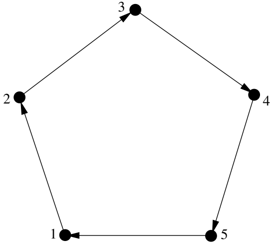


The Acyclic Induced subgraph problem

Let $G = (V, A)$ be a directed graph
with $n = |V|$ nodes and $m = |A|$ arcs.

A **circuit** of G : a sequence of arcs

$$C = (i_1 i_2, i_2 i_3, ..., i_{k-1} i_1)$$



Notation :
$V(C)$ is the set of nodes of C.

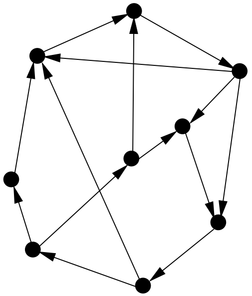
A graph is **acyclic** if it contains no circuit.

The Acyclic Induced subgraph problem

Given a node subset $W \subset V$,

$A(W)$: arcs with both endnodes in W

$(W, A(W))$: **subgraph induced by** W

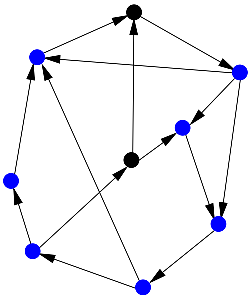


The Acyclic Induced subgraph problem

Given a node subset $W \subset V$,

$A(W)$: arcs with both endnodes in W

$(W, A(W))$: **subgraph induced by** W

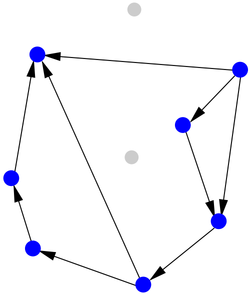


The Acyclic Induced subgraph problem

Given a node subset $W \subset V$,

$A(W)$: arcs with both endnodes in W

$(W, A(W))$: **subgraph induced by** W

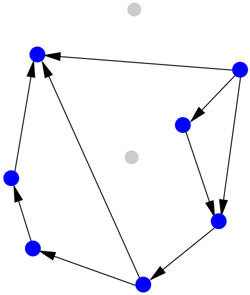


The Acyclic Induced subgraph problem

Given a node subset $W \subset V$,

$A(W)$: arcs with both endnodes in W

$(W, A(W))$: **subgraph induced by** W



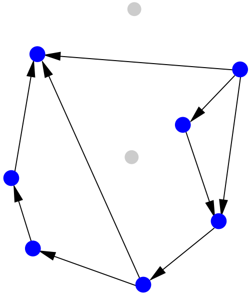
The **Acyclic Induced Subgraph Problem (AISP)** is to find a node subset W inducing an acyclic subgraph with $|W|$ maximum

The Acyclic Induced subgraph problem

Given a node subset $W \subset V$,

$A(W)$: arcs with both endnodes in W

$(W, A(W))$: **subgraph induced by** W



The **Acyclic Induced Subgraph Problem (AISP)** is to find
a node subset W inducing an acyclic subgraph with $|W|$ maximum
or
a node subset W' “breaking” every circuit of G with $|W'|$ min.

The Acyclic Induced subgraph problem

- The AISP is NP-hard.

Indeed :
given a non-directed graph G
construct a directed graph G' by replacing one edge by two arcs

then finding an acyclic subgraph in G' is as hard as finding a maximum stable set in G.

- The AISP is polynomial for graphs of maximum degree 3 [Baiou, Barahona]

A compact formulation

Two types of variables :

- Binary variables $x_i = \begin{cases} 1 & \text{if node } i \text{ is chosen} \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in V.$

- Continuous variables u_i $\forall i \in V$.

A Miller-Tucker-Zemlin (MTZ) formulation :

$$(F_{MTZ}) \begin{cases} \max \sum_{i \in V} x_i & \\ u_i - u_j + 1 \leq n(2 - x_i - x_j) & \forall ij \in A \\ 1 \leq u_i \leq n & \forall i \in V \\ u_i \in \mathbb{R} & \forall i \in V \\ x_i \in \{0, 1\} & \forall i \in V \end{cases}$$

A compact formulation

- The MTZ formulation is equivalent to the AISP.

Indeed :
Given a solution (x, u) of (F_{MTZ}), let $W = \{i \in V \mid x_i = 1\}$.
If W induces a circuit C : $u_i + 1 \leq u_j \;\; \forall ij \in C$, a contradiction.

- (F_{MTZ}) is compact

- n binary variables and n continuous variables
- m inequalities

Let's go with Cplex !

A non-compact formulation

Same binary variables $x_i = \begin{cases} 1 & \text{if node } i \text{ is chosen} \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in V.$

$$(F_C) \begin{cases} \max \sum_{i \in V} x_i & \\ \sum_{i \in V(C)} x_i \leq |C| - 1 & \forall C \text{ circuit of } G \\ x_i \in \{0, 1\} & \forall i \in V \end{cases}$$

These inequalities are called the **circuit inequalities**.

Formulation (F_C) is clearly equivalent to the AISP.

A non-compact formulation

Circuit inequalities are in **exponential number** with respect to the number of nodes.

Formulation (F_C) cannot be directly used as compact formulation in Cplex :

- Is this formulation really better than a compact one ?
- How to use such a non-compact formulation ?

For an integer linear formulation

$$(F) \quad \begin{cases} \max c^T x \\ \quad Ax \leq b \\ \quad x \in \mathbb{Z} \end{cases}$$

with n variables
with a exponential number of inequalities with respect to n.

For an integer linear formulation

$$(F) \quad \begin{cases} \max c^T x \\ \quad Ax \leq b \\ \quad x \in \mathbb{Z} \end{cases} \qquad\qquad (\tilde{F}) \quad \begin{cases} \max c^T x \\ \quad Ax \leq b \\ \quad \cancel{x \in \mathbb{Z}} \end{cases}$$

with n variables
with a exponential number of inequalities with respect to n.

How can we solve the linear relaxation $(\tilde{F})$ of (F) ?

Polyhedron

A *hyperplane* of $\boldsymbol{R}^n$ is the set of points $\tilde{x} \in \boldsymbol{R}^n$ satisfying a linear equality $ax = \alpha$.

A *halfspace* of $\boldsymbol{R}^n$ is the set of points $\tilde{x} \in \boldsymbol{R}^n$ satisfying a linear equality $ax \leq \alpha$.

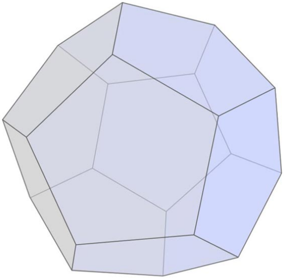
A **polyhedron** P is the intersection of a finite number of halfspaces
i.e.

$$P = \{\tilde{x} \in \boldsymbol{R}^n \mid A\tilde{x} \leq b\}$$

with $Ax \leq b$ system of linear inequalities.

Such a system $Ax \leq b$ **characterizes** a polyhedron.

A **polytope** is a bounded polyhedron.



Extreme point

- An **extreme point** (or vertex) of a polytope P is a point $x \in P$ s.t. there is no solutions $x^1 \neq x^2$ in P with $x = \frac{1}{2}x^1 + \frac{1}{2}x^2$.

- Solving a (bounded) linear formulation

$$(\tilde{F}) \begin{cases} \max \ c^T x \\ \quad Ax \leq b \end{cases}$$

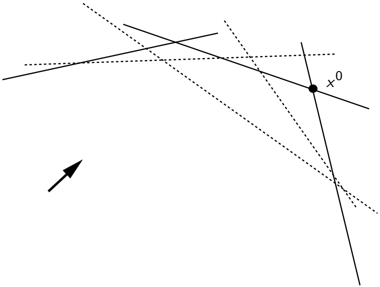
reduces to find an optimal extreme point of polytope

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$$

Initialisation

Let $A_0 x \leq b_0$ be a subset of inequalities of $Ax \leq b$
and $(\tilde{F}^0)$ the linear program restricted to $A_0 x \leq b_0$.
Let x^0 the solution of $(\tilde{F}^0)$.

$$(\tilde{F}^0) \begin{cases} \max c^T x \\ A_0 x \leq b_0 \end{cases}$$

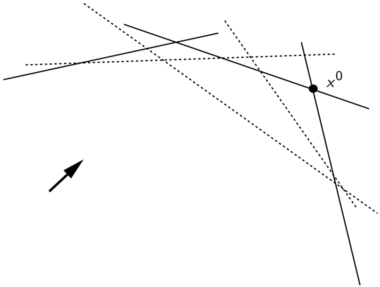


Solution x^0 is an extreme point of the polytope characterized by $A_0 x \leq b_0$.

Initialisation

Let $A_0x \leq b_0$ be a subset of inequalities of $Ax \leq b$
and $(\tilde{F}^0)$ the linear program restricted to $A_0x \leq b_0$.
Let x^0 the solution of $(\tilde{F}^0)$.

$$(\tilde{F}^0) \begin{cases} \max c^T x \\ A_0 x \leq b_0 \end{cases}$$



Solution x^0 is an extreme point of the polytope characterized by $A_0x \leq b_0$.

Note that :
if x^0 satisfies every inequality of $Ax \leq b$
x^0 is be an extreme point of the polytope characterized by $Ax \leq b$
and x^0 will an optimal solution !

Separation problem

Definition (Separation problem)

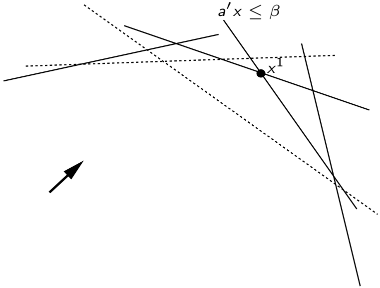
Given a point $\tilde{x} \in \boldsymbol{R}^n$,
the **separation problem** associated to $Ax \leq b$ and $\tilde{x}$ is
- to determine whether $\tilde{x}$ satisfies every inequality of $Ax \leq b$
- or to produce an inequality $ax \leq \alpha$ of $Ax \leq b$ violated by $\tilde{x}$.

An inequality $ax \leq \alpha$ of $Ax \leq b$ is **violated** by x^0 if $ax^0 > \alpha$.

Iteration

If the separation problem for x^0 produces an inequality $a'x \leq \beta$ violated by x^0

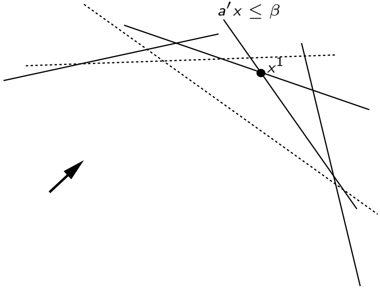
$$(\tilde{F}_1) \begin{cases} \max c^T x \\ A_0 x \leq b_0 \\ a'x \leq \beta \end{cases}$$



Iteration

If the separation problem for x^0 produces an inequality $a'x \leq \beta$ violated by x^0

$$(\tilde{F}_1) \begin{cases} \max c^T x \\ A_0 x \leq b_0 \\ a'x \leq \beta \end{cases}$$



And so on !

Cutting plane based algorithm

Definition (Cutting-plane based "method")

While there exists an inequality $ax \leq \alpha$ of $Ax \leq b$ violated by x^i

$(\tilde{F}_{i+1}) \leftarrow (\tilde{F}_i) + ax \leq \alpha$

Solve the linear program $\tilde{F}_{i+1}$

$x_{i+1} \leftarrow$ solution of $\tilde{F}_{i+1}$

$i \leftarrow i + 1$

- An algorithm which solves the separation problem is called a **separation algorithm** (the whole method is to reiterate the separation algorithm).

- An inequality which is violated a solution x^i is called a **cut** because the inequality **separates** an useless part of $\boldsymbol{R}^n$ from the polytope characterized by $Ax \leq b$.

Validity of the method

- *Ending*
In the worst case, the algorithm enumerates every inequality of $Ax \leq b$.

- *Validity*
At the end of the loop, the final solution x^* is an extreme point of the whole system $Ax \leq b$ and maximizes $c^T x$:
then x^* is an optimal solution of $(\tilde{F})$.

Complexity

Theorem (Grötschel, Lovász, Schrijver, 1981)

A cutting plane based method of rational system $Ax \leq b$ is polynomial
if and only if
the separation algorithm associated to $Ax \leq b$ is polynomial.

This fundamental results is :

Optimize $\Leftrightarrow$ Separate

Separation algorithm for the circuit inequalities

Theorem

The circuit inequalities

$$\sum_{i \in V(C)} x_i \leq |C| - 1 \quad \forall \text{ circuit } C$$

can be separated in polynomial time.

Note that, by setting $x' = 1 - x$,
a circuit inequality can be rewritten

$$\sum_{i \in V(C)} x'_i \geq 1$$

Separation algorithm for the circuit inequalities

Given a point $\tilde{x} \in [0,1]^n$
The separation problem for the circuit inequalities is to determine whether or not there exists a circuit inequality violated by $\tilde{x}$
(and in the last case, to produce one violated inequality).

Set $x' = 1 - \tilde{x}$.
Find a circuit $\tilde{C}$ of minimal weight with respect to x'.
(This can be done in polynomial time since $x' \geq 0$)

- If $\sum_{i \in V(\tilde{C})} x'_i < 1$: $\sum_{i \in V(\tilde{C})} x_i \leq |\tilde{C}| - 1$ is violated by $\tilde{x}$.
- If $\sum_{i \in V(\tilde{C})} x'_i \geq 1$: there is no circuit ineq. violated by $\tilde{x}$. □

Branch-and-Cut algorithm

The cutting plane based method gives in polynomial time the relaxation value of an integer formulation.

At the end, the solution is not integer (unless if $P = NP$).

Branch-and-Cut algorithm

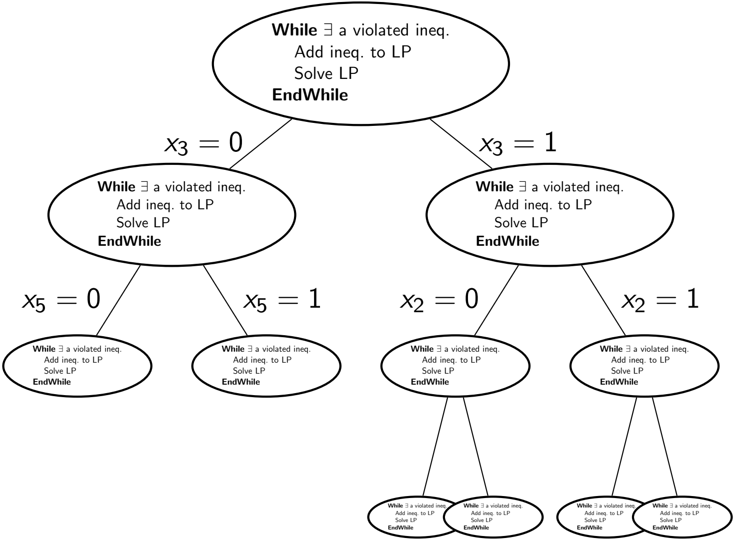
The cutting plane based method gives in polynomial time the relaxation value of an integer formulation.

At the end, the solution is not integer (unless if $P = NP$).

The only known general framewok to solve integer problem is... Branch&Bound !

A Branch&Bound that uses a cutting plane based algorithm in every node of the Branch&Bound tree
is a **Branch&Cut algorithm**.

Branch-and-cut algorithm



Formulation comparison

There is no generic method to compare the relaxation values of two formulations.

Formulation comparison

There is no generic method to compare the relaxation values of two formulations.

Theorem

The relaxation value of the MTZ formulation (F_{MTZ})
$\leq$ *the relaxation value of the circuit formulation* (F_C).

Sketch of the proof.
First note that by summing MTZ inequalities over a circuit C, we obtain

$$\sum_{i \in V(C)} x_i \leq \left(1 - \frac{1}{2n}\right) |C|$$

Let $\tilde{x}$ be a solution of the relaxation $(\tilde{F}_C)$, then $\tilde{x}$ satisfies

$$\sum_{i \in V(C)} \tilde{x}_i \leq |C| - 1 \leq \left(1 - \frac{1}{2n}\right) |C|$$

By setting appropriate values u_i, we get a solution $(\tilde{x}, u)$ of (F_{MTZ}). □

Reinforcement

Given an integer formulation (F) $\begin{cases} \max c^T x \\ \quad Ax \leq b \\ \quad x \in \mathbb{Z} \end{cases}$

Definition

An inequality $ax \leq \alpha$ is **valid** for an integer formulation (F) if every integer solution of (F) satisfies $ax \leq \alpha$.

Adding a valid inequality to formulation (F) does not change the solution space of (F).

But valid inequalities can improve the relaxation value !

Obtaining valid inequalities

- Summing inequalities : Chvátal-Gomory rounding method :
- Multiplying inequalities : lift-and-project Lovász-Shrivjer meth.
- Finding particular sub-structures (stable set, knapsack,...)
- Lifting coefficients of inequalities (to obtain stronger ones)
- Disjunctive cuts, local branching inequalities,...
- ... and many others techniques ...

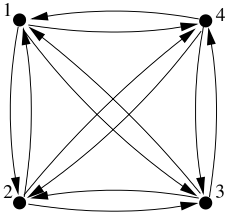
Clique inequalities

In a directed graph, a clique is a subset K of nodes inducing a complete subgraph.

The **clique inequalities**

$$\sum_{i \in K} x_i \leq 1 \text{ pour every clique } K \text{ of } G$$

are valid for the formulation.



Indeed, at most one node can be taken among a clique.

Clique inequalities

Lemma

The separation problem for the clique inequalities is NP-complete.

Indeed, proving that there is no violated clique inequality is equivalent to find a maximal weighted clique in G, which is a famous NP-hard problem □

Clique inequalities

Lemma

The separation problem for the clique inequalities is NP-complete.

Indeed, proving that there is no violated clique inequality is equivalent to find a maximal weighted clique in G, which is a famous NP-hard problem □

- However, a **heuristic separation algorithm** can be used !

For instance, a simple but efficient greedy heuristic :

Given a linear relaxation value x^* :

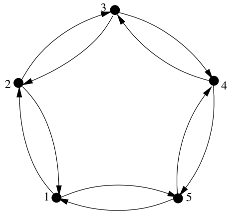
- sort the nodes with respect to decreasing values x_i^*
- $K \leftarrow \emptyset$
- iteratively try to add a node in K such that K stays a clique

Chvàtal sum techniques

For a "directed cycle" D like this one.

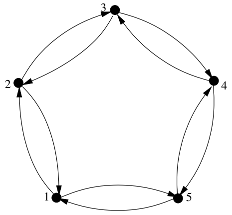
For each consecutive nodes i and $i+1$ there is a circuit inequality

$$x_i + x_{i+1} \leq 1$$



Chvàtal sum techniques

For a "directed cycle" D like this one.



For each consecutive nodes i and $i+1$ there is a circuit inequality

$$x_i + x_{i+1} \leq 1$$

By summing these inequalities :

$$\begin{array}{ccccccccc} x_1 & + & x_2 & & & & & \leq & 1 \\ & & x_2 & + & x_3 & & & \leq & 1 \\ & & & & & .... & & & \\ x_1 & & & & & + & x_{|D|} & \leq & 1 \\ \hline \end{array}$$

$$\sum_{i \in V(C)} x_i \leq \frac{|D|}{2}$$

odd cycle inequalities

The left part is integer, let's round it down

$$\sum_{i \in V(C)} x_i \leq \frac{|D|}{2}$$

odd cycle inequalities

The left part is integer, let's round it down

$$\sum_{i \in V(C)} x_i \leq \left\lfloor \frac{|D|}{2} \right\rfloor$$

odd cycle inequalities

The left part is integer, let's round it down

$$\sum_{i \in V(C)} x_i \leq \left\lfloor \frac{|D|}{2} \right\rfloor$$

- If $|D|$ is even, nothing new is obtained.
- If $|D|$ is odd, we obtain a new valid inequality

$$\sum_{i \in V(C)} x_i \leq \frac{|D|-1}{2}$$

Such odd cycle inequalities can be found in the stable set polytope.

They can be separated in polynomial time.

What is inside a MIP solver ?

A MIP solver like CPLEX, GUROBI, XPRESS, SCIP,... are “Automatic Branch-and-Cut process.

- Strong preprocessing phase
- Automatic use of generic valid inequalities through efficient cutting plane based methods
- Automatic lifting operations to reinforce known inequalities and produce nex ones
- Automatic logical inference to break the symmetry of the branching tree
- Generic rounding heuristics

And it is more and more easy to add your own valid inequalities to these frameworks !

1. Compact / non-compact formulations
2. Cutting Plane based algorithm
3. Branch&Cut algorithm
4. Reinforcement of relaxation value
5. The travelling Salesman Problem (TSP)

The travelling salesman problem (TSP)

n cities
c_{ij} the transportation cost between i and j
Find a Hamiltonian "tour" that visits each cities exactly once

Let $x_{ij} = 1$ if edge ij is chosen and 0 otherwise.

The travelling salesman problem (TSP)

n cities
c_{ij} the transportation cost between i and j
Find a Hamiltonian "tour" that visits each cities exactly once

Let $x_{ij} = 1$ if edge ij is chosen and 0 otherwise.

The following linear program is integer

$$\begin{aligned} \min \sum_{i,j} c_{ij} x_{ij} & \\ \sum_{j \in V} x_{ij} = 2 & \quad \forall i \in V, \\ x_{ij} \geq 0 & \quad \forall (i,j) \in V \times V. \end{aligned}$$

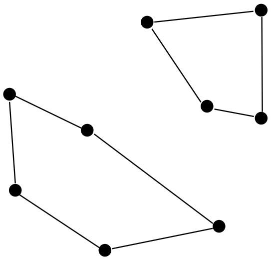
The travelling salesman problem (TSP)

n cities
c_{ij} the transportation cost between i and j
Find a Hamiltonian "tour" that visits each cities exactly once

Let $x_{ij} = 1$ if edge ij is chosen and 0 otherwise.

The following linear program is integer

$$\min \sum_{i,j} c_{ij} x_{ij}$$
$$\sum_{j \in V} x_{ij} = 2 \qquad \forall i \in V,$$
$$x_{ij} \geq 0 \quad \forall (i,j) \in V \times V.$$



Unfortunately, the integer solutions contain "subtours".

Eliminating subtours

Menger's theorem :
a graph is connected if and only if every cut contains at least one edge.

Then "cut" inequalities

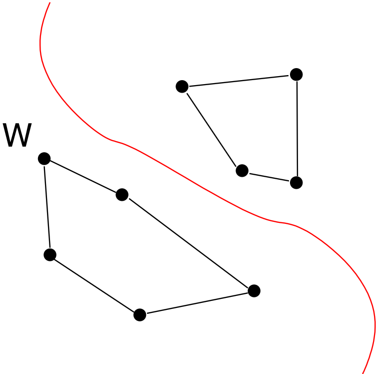
$$\sum_{e \in \delta(W)} x(e) \geq 1 \quad \forall W \subsetneq V \text{ and } W \neq \emptyset$$

Eliminating subtours

Menger's theorem :
a graph is connected if and only if every cut contains at least one edge.

Then "cut" inequalities

$$\sum_{e \in \delta(W)} x(e) \geq 1 \quad \forall W \subsetneq V \text{ and } W \neq \emptyset$$

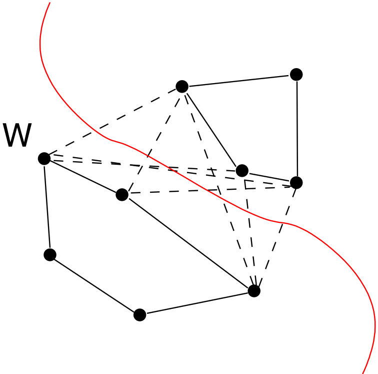


Eliminating subtours

Menger's theorem :
a graph is connected if and only if every cut contains at least one edge.

Then "cut" inequalities

$$\sum_{e \in \delta(W)} x(e) \geq 1 \quad \forall W \subsetneq V \text{ and } W \neq \emptyset$$



The TSP formulation

$$
\begin{aligned}
\min \sum_{e \in E} c(e)x(e) & \\
\sum_{e \in \delta(v)} x(e) = 2 & \quad \forall u \in V, \\
\sum_{e \in \delta(W)} x(e) \geq 2 & \quad \forall W \subsetneq V \text{ and } W \neq \emptyset, \\
x(e) \in \{0,1\} & \quad \forall e \in E.
\end{aligned}
$$

The TSP formulation

$$\min \sum_{e \in E} c(e)x(e)$$
$$\sum_{e \in \delta(v)} x(e) = 2 \quad \forall u \in V,$$
$$\sum_{e \in \delta(W)} x(e) \geq 2 \quad \forall W \subsetneq V \text{ and } W \neq \emptyset,$$
$$x(e) \in \{0, 1\} \quad \forall e \in E.$$

With (a lot of) additional facet defining inequalities,
this formulation succeed to solve instances with more than 200 000 cities.