Discrete Optimization

Quentin Louveaux

ULg - Institut Montefiore

2015

Valid inequalities

We have seen that having a good formulation is crucial to obtain a (fast)-solving problem.
Main issue : how to automatically improve a formulation.

Definition

Let $P \subseteq \mathbb{R}^n$. An inequality $\sum_{j=1}^n a_j x_j \leq b$ is valid if it is satisfied by all points $x \in P$.

Typically, we want to derive valid inequalities for the set of integral solutions and at the same time cut off some part of the linear programming relaxation.

The rounding principle

Let $P = \{x \in \mathbb{Z}^n \mid Ax \leq b\}$ and $P_{LP} = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ be the corresponding linear programming relaxation.

If $x \leq c$ is valid for P_{LP} then $x \leq \lfloor c \rfloor$ is valid for P.

The Chvatal-Gomory procedure

- Compute a nonnegative combination of the rows of the LP formulation

$$(u^T A)x \leq u^T b, \qquad (u \geq 0)$$

- The inequality

$$(\lfloor u^T A \rfloor)x \leq \lfloor u^T b \rfloor$$

  is valid for P.

Gomory's fractional cutting plane algorithm

- Based on the simplex algorithm applied to the linear relaxation of the MIP
- automatically generate and apply cuts until solution is integer
  - if optimal solution is fractional, use the information provided by the optimal basis to generate cuts (apply the Chvatal-Gomory procedure)
- terminates in a finite number of iterations if combined with the right simplex pivoting rule
- not very successful in practice, hence **branch-and-cut**.

The Basic Mixed Integer inequality

2D case

Let $X = \{(x, y) \in \mathbb{R}_+ \times \mathbb{Z}_+ \mid x + y \geq b\}$ and $f = b - \lfloor b \rfloor > 0$.
Then

$$\frac{x}{f} + y \geq \lceil b \rceil$$

is valid for X

Corollary

Let $X = \{(x, y) \in \mathbb{R}_+ \times \mathbb{Z}_+ \mid y \leq b + x\}$ and $f = b - \lfloor b \rfloor > 0$.
Then

$$y \leq \lfloor b \rfloor + \frac{x}{1 - f}$$

is valid for X

Mixed Integer Rounding (MIR) cut

Let

$$X = \{(x, y) \in \mathbb{R}_+ \times \mathbb{Z}_+^2 \mid a_1 y_1 + a_2 y_2 \le b + x\},$$

$$f = b - \lfloor b \rfloor > 0,$$

and

$$f_i = a_1 - \lfloor a_i \rfloor, \ \ i = 1, 2$$

with

$$f_1 \le f \le f_2.$$

Then

$$\lfloor a_1 \rfloor y_1 + \left(\lfloor a_2 \rfloor + \frac{f_2 - f}{1 - f} \right) y_2 \le \lfloor b \rfloor + \frac{x}{1 - f}$$

is valid for X.

Superadditivity : preliminary definitions

Superadditive function

The function $F : D \subseteq \mathbb{R}^m \mapsto \mathbb{R}$ is superadditive if

$$F(a_1) + F(a_2) \leq F(a_1 + a_2)$$

for all $a_1, a_2 \in D$ such that $a_1 + a_2 \in D$.

Remark : F superadditive $\Rightarrow F(0) \leq 0$.

Non-decreasing function

The function $F : D \subseteq \mathbb{R}^m \mapsto \mathbb{R}$ is non-decreasing if

$$F(a_1) \leq F(a_2)$$

for all $a_1, a_2 \in D$ such that $a_1 \leq a_2$.

Superadditivity

If $F : \mathbb{R}^m \mapsto \mathbb{R}$ is superadditive, non-decreasing and satisfies $F(0) = 0$, then the inequality

$$\sum_{j=1}^{n} F(A_j)x_j \leq F(b)$$

is valid for conv(P) with $P = \{x \in \mathbb{Z}^n_+ | Ax \leq b\}$.

Proof, comparison to MIR

Strong inequalities

- Inequalities $\pi x \leq \pi_0$ and $\lambda \pi x \leq \lambda \pi_0$ are identical if $\lambda > 0$.
- An inequality $\pi x \leq \pi_0$ dominates $\mu x \leq \mu_0$ if there exists $u > 0$ with

$$\pi \geq u\mu \quad \text{and} \quad \pi_0 \leq u\mu_0$$

if we work in a polyhedron $P \subset \mathbb{R}^n_+$.

Polyhedra, faces and facets

- n points $x^{(1)}, \ldots, x^{(k)}$ are affinely independent if $x^{(2)} - x^{(1)}, \ldots, x^{(k)} - x^{(1)}$ are linearly independent or equivalently if $(x^{(1)}, 1), \ldots, (x^{(k)}, 1)$ are linearly independent.
- The dimension d of a polyhedron P is the maximum number of affinely independent points in P minus 1.
- F is a face of P if $F = \{x \in P : \pi x = \pi_0\}$ for some valid inequality $\pi x \leq \pi_0$.
- F is a facet if F is a face of P of dimension $\dim(P) - 1$.

Facets of $\text{conv}(P)$ are the valid inequalities that we are looking for !

Knapsack covers

We consider the knapsack set

$$X = \{x \in \{0,1\}^n \mid \sum_{j=1}^{n} a_j x_j \leq b\}.$$

Definition

A set C is a cover if $\sum_{j \in C} a_j > b$.

A cover inequality

If C is a cover, the cover inequality

$$\sum_{j \in C} x_j \leq |C| - 1$$

is valid for X.

Lifting a cover inequality

Consider an inequality $\sum_{j \in C} x_j \leq |C| - 1$.
Consider a variable $i \notin C$ that we would like to lift, namely we want to give it a coefficient in the cover inequality.

$$\begin{aligned}\alpha_i = |C| - 1 - \max & \sum_{j \in C} x_j \\ \text{s. t. } & \sum_{j \in C} a_j x_j \leq b - a_i \\ & x_j \in \{0, 1\}.\end{aligned}$$

Branch-and-cut : used in all MIP solvers nowadays

- Branch-and-bound combined with cutting plane algorithm
- uses several families of cuts, depending on the problem (MIR, covers, ...)
- typically starts as a cutting plane algorithm, then branches
- at each node, decide to branch or to generate and add cuts
- cuts are often node specific, i.e. cannot be imported in other parts of the tree without care.

User cuts callback

- In some cases, the user may want to define problem specific cuts.
- Bad idea : Generate 10000 valid inequalities and add them to the formulation. (Why ?)
- Good idea : Write a separation code.
  The solver then calls the separation routine to cut a fractional LP solution.
- **User cuts callback are called to cut a fractional point !**

Subtour elimination constraints

One example of user cuts callback may relate to subtour elimination constraints (borderline example since it is not really a cut).

Consider the prize-collecting TSP.

$$
\begin{aligned}
\max \ & \sum_{e \in E} c_e x_e + \sum_{j \in V} f_j y_j \\
\text{s.t.} \ & \sum_{e \in \delta(i)} x_e = 2y_i && \text{for all } i \in V \\
& \sum_{e \in E(S)} x_e \leq \sum_{i \in S \setminus \{k\}} y_i && k \in S, S \subseteq V \setminus \{1\} \\
& y_1 = 1 \\
& x \in \{0,1\}^{|E|}, y \in \{0,1\}^{|V|}
\end{aligned}
$$

Problem : There is an exponential number of generalized subtour elimination constraints !
We could use them as user cut callbacks.
We need to be able to separate them.

The separation problem

Definition

Given a convex set $X \subseteq \mathbb{R}^n$ and a point $x \in \mathbb{R}^n$, the separation problem is to determine whether

- $x \in X$ or
- provide a valid inequality $a^T y \leq b$ for X such that $a^T x > b$ proving that $x \notin X$.

In the case of the subtour elimination constraints, the set X is the polyhedron satisfying the exponential number of inequalities.
The separation problem consists in either finding one violated inequality or proving that none is violated.

Generalized subtour elimination constraints

We formulate the separation problem as an integer program for a fixed $k \in V$.

Data : (x^*, y^*) is the value of the LP without the subtour elim. constraints.

Variables : $z_i = 1$, $i \in V$ if i belongs to the subset S in the constraint.

The constraint using S is violated if $\sum_{e \in E(S)} x_e^* > \sum_{i \in S \setminus \{k\}} y_i^*$.

$$\max \sum_{e=(i,j)|i<j} x_e^* z_i z_j - \sum_{i \in V \setminus \{k\}} y_i^* z_i, z_k = 1.$$

This is a quadratic program which can be solved efficiently by linearizing the products $w_e = 1$ if $z_i = 1, z_j = 1$ for $e = (i,j)$.

$$\begin{aligned} \max \sum_{e \in E} x_e^* w_e - & \sum_{i \in V \setminus \{k\}} y_i^* z_i \\ \text{s.t. } w_e \leq z_i & \qquad e = (i,j) \\ w_e \leq z_j & \qquad e = (i,j) \\ w_e \geq z_i + z_j - 1 & \qquad e = (i,j) \\ z_k = 1, w \in \{0,1\}^{|E|}, z \in \{0,1\}^{|V|}.. & \end{aligned}$$

The problem can be solved by relaxing the integrality and the constraints $w_e \geq z_i + z_j - 1$. It provides a separation if its solution is larger than 0.

Lazy constraints

- If a family of constraints is too large, we do not want to add them to the initial formulation.
- We may define them as lazy and provide a separation routine to the solver to cut integral solutions.
- Each time the solver finds an integral solution, it checks whether all lazy constraints are satisfied and if not, add some to the initial formulation.

Subtour elimination constraints (again)

- It is much easier to separate the subtour elimination constraints over integral solutions.
- Visit the graph until a subtour is found.
  $\mathcal{O}(|V|) \rightarrow$ more efficient than the previous user cut callback.
  (Needs a good database for the solution)

Often we need a combination of the two !