IE 631 Integer Programming **KAIST, Spring 2023**
Lecture #4: Convex Hull and Branch-and-Cut Methods March 9, 2023
Lecturer: Dabeen Lee

---

1 Outline

In this lecture, we study

- convex hull and reduction to linear programming,
- characterizing the convex hull of a two variable mixed integer linear set,
- methods for solving integer programming.

2 Two-dimensional mixed integer linear set

Consider a mixed integer linear set given by

$$S = \{(x, y) \in \mathbb{Z} \times \mathbb{R}_+ : \ x - y \leq \beta\} \tag{4.1}$$

for some $\beta \in \mathbb{R}$. Note that

$$P = \{(x, y) \in \mathbb{R} \times \mathbb{R}_+ : \ x - y \leq \beta\}$$

corresponds to the LP relaxation of S defined by the two inequalities $y \geq 0$ and $x - y \leq \beta$. Figure 4.1 illustrates the mixed integer linear set S and its relaxation P. Let us characterzie the convex hull

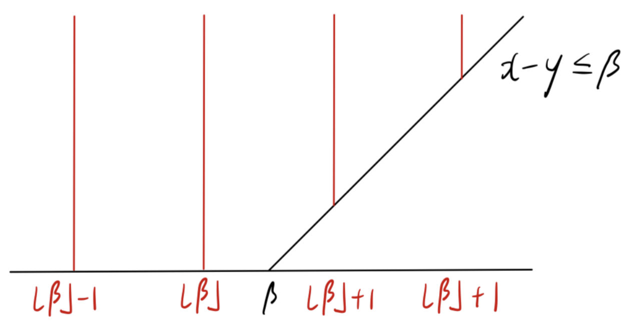

Figure 4.1: Illustration of S and P

of S.

Lemma 4.1. *Let $f = \beta - \lfloor \beta \rfloor$ be the fractional part of β. Then the inequality*

$$x - \frac{1}{1-f} y \leq \lfloor \beta \rfloor \tag{4.2}$$

holds for any $(x, y) \in S$. In other words, the inequality is valid and a valid inequality for S.

The inequality (4.2) in Lemma 4.1 holds at equality when

$$(x, y) = (\lfloor \beta \rfloor, 0), \ (\lfloor \beta \rfloor + 1, 1 - f).$$

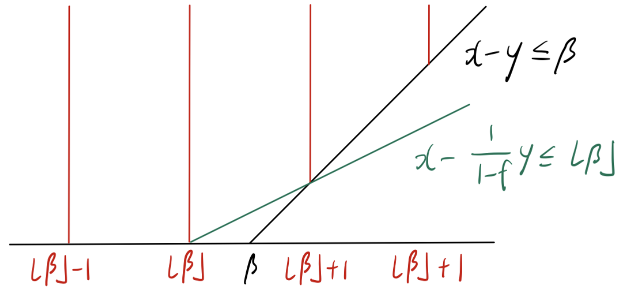

Figure 4.2: Valid inequality for S

Equivalently, the line defined by

$$x - \frac{1}{1-f}y = \lfloor \beta \rfloor$$

go through the two points $(\lfloor \beta \rfloor, 0)$ and $(\lfloor \beta \rfloor + 1, 1-f)$, as shown in Figure 4.2.

We will see later that the **mixed integer rounding (MIR) cuts** by Nemhauser and Wolsey [NW90] are obtained based on the inequality (4.2) that is valid for the mixed integer linear set (4.1).

In fact, the inequality (4.2) together with $y \geq 0$ and $x - y \leq \beta$ describe the convex hull of the mixed integer linear set (4.1).

Proposition 4.2. *Let* $S = \{(x, y) \in \mathbb{Z} \times \mathbb{R}_+ : \ x - y \leq \beta\}$. *Then*

$$\operatorname{conv}(S) = \left\{(x, y) \in \mathbb{R} \times \mathbb{R}_+ : \ x - y \leq \beta, \ x - \frac{1}{1-f}y \leq \lfloor \beta \rfloor \right\}.$$

Proof. Let Q denote the set on the right-hand side. By Lemma 4.1, we know that the inequality (4.2) is valid for S, implying in turn that $\operatorname{conv}(S) \subseteq Q$.

To show that $Q \subseteq \operatorname{conv}(S)$, we will argue that any point $(\bar{x}, \bar{y}) \in Q$ can be expressed as a convex combination of some two points in S. If $\bar{x} \in \mathbb{Z}$, then $(\bar{x}, \bar{y}) \in S \subseteq \operatorname{conv}(S)$. Thus we may assume that $\bar{x} \notin \mathbb{Z}$. Note that one of the following three holds: (1) $\bar{x} < \lfloor \beta \rfloor$, (2) $\lfloor \beta \rfloor < \bar{x} < \lfloor \beta \rfloor + 1$, (3) $\bar{x} > \lfloor \beta \rfloor + 1$.

First, consider the case where $\bar{x} < \lfloor \beta \rfloor$, as shown in Figure 4.3. Then both $\lfloor \bar{x} \rfloor$ and $\lceil \bar{x} \rceil$ are less

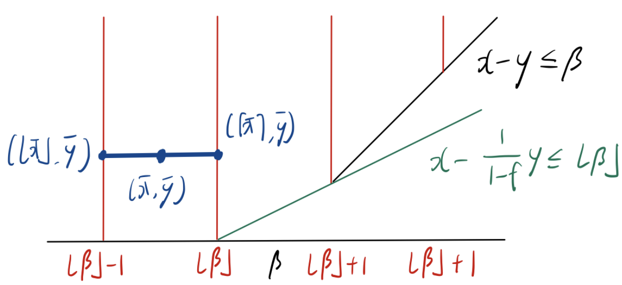

Figure 4.3: The case $\bar{x} < \lfloor \beta \rfloor$

than or equal to $\lfloor \beta \rfloor$, so $(\lfloor \bar{x} \rfloor, \bar{y})$ and $(\lceil \bar{x} \rceil, \bar{y})$ belong to S. Here, $(\bar{x}, \bar{y})$ is a convex combination of $(\lfloor \bar{x} \rfloor, \bar{y})$ and $(\lceil \bar{x} \rceil, \bar{y})$.

Second, we consider the case where $\lfloor \beta \rfloor < \bar{x} < \lfloor \beta \rfloor + 1$, as shown in Figure 4.4. Let us give a

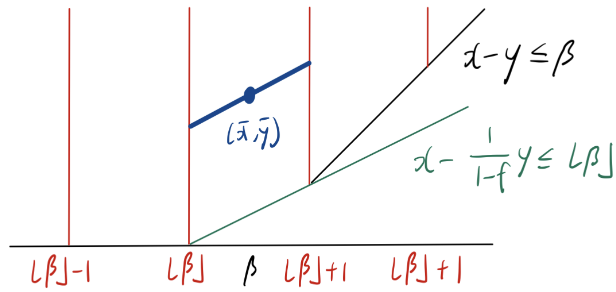

Figure 4.4: The case $\lfloor\beta\rfloor < \bar{x} < \lfloor\beta\rfloor + 1$

pictorial proof. Draw a line segment that goes through $(\bar{x}, \bar{y})$ and is parallel to the line defined by

$$x - \frac{1}{1-f} y = \lfloor\beta\rfloor.$$

The line segment crosses $x = \lfloor\beta\rfloor$ and $x = \lfloor\beta\rfloor + 1$. The intersection points belong to S, and $\bar{x}$ is a convex combination of them.

Lastly, we consider the case where $\lfloor\beta\rfloor + 1 < \bar{x}$, as shown in Figure 4.5. Let us give a pictorial

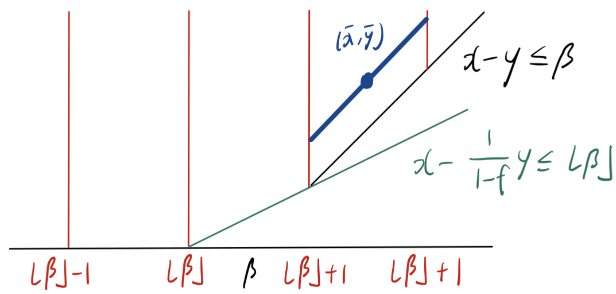

Figure 4.5: The case $\lfloor\beta\rfloor + 1 < \bar{x}$

proof. Draw a line segment that goes through $(\bar{x}, \bar{y})$ and is parallel to the line defined by $x - y = \beta$. The line segment crosses $x = \lfloor\bar{x}\rfloor$ and $x = \lceil\bar{x}\rceil$. The intersection points belong to S, and $\bar{x}$ is a convex combination of them. □

One can find an algebraic proof of Proposition 4.2 from [CCZ14, Proposition 1.5].

3 Methods for solving integer programming

The most successful algorithmic frameworks for integer programming are the **branch-and-bound** and the **cutting-plane** methods. Let us discuss the outlines and the ideas behind the methods. Again, we consider an integer program

$$z^* = \max\left\{c^\top x + h^\top y : \ (x, y) \in S\right\} \tag{MILP$_0$}$$

where

$$S = \left\{(x, y) \in \mathbb{Z}^d \times \mathbb{R}^p : \ Ax + Gy \le b\right\}.$$

The first common step is to solve the LP relaxation given by

$$z_0 = \max\left\{c^\top x + h^\top y : \ (x, y) \in P_0\right\} \tag{LP$_0$}$$

where

$$P_0 = \left\{(x, y) \in \mathbb{R}^d \times \mathbb{R}^p : \ Ax + Gy \leq b\right\}.$$

The modern software uses the **barrier method** and the **simplex method**[1]. Let z_0 be the optimal value of the LP relaxation, and assume that z_0 is finite. For rational data, z_0 being finite implies that the optimal value z^* of the integer program is also finite (we will see this later).

Let (x^0, y^0) be an optimal solution to the LP relaxation. What if $x^0 \in \mathbb{Z}^d$? Then $(x^0, y^0) \in S$. This implies the following

$$\begin{aligned}\max\left\{c^\top x + h^\top y : \ (x, y) \in S\right\} &\leq \max\left\{c^\top x + h^\top y : \ (x, y) \in P_0\right\} \\ &= c^\top x^0 + h^\top y^0 \\ &\leq \max\left\{c^\top x + h^\top y : \ (x, y) \in S\right\}.\end{aligned}$$

Here, the left-hand side and the right-most side coincide, so the equalities hold throughout, which implies that (x^0, y^0) is an optimal solution to the integer program!

In general, the LP relaxation does not necessarily return an integer solution, and x^0 may have some fractional components. Here, the branch-and-bound and the cutting-plane methods provide two natural strategies to deal with the situation where x^0 has some fractional component.

3.1 Branch-and-bound method

Suppose that component x_j^0 is fractional for some $1 \leq j \leq d$. Then we know that

$$x_j \geq \lceil x_j^0 \rceil \quad \text{or} \quad x_j \leq \lfloor x_j^0 \rfloor.$$

Based on this, we define

$$S_1 = S \cap \left\{(x, y) : \ x_j \geq \lceil x_j^0 \rceil\right\} \quad \text{and} \quad S_2 = S \cap \left\{(x, y) : \ x_j \geq \lceil x_j^0 \rceil\right\},$$

and in fact, $S = S_1 \cup S_2$. Moreover, we create two subproblems

$$\max\left\{c^\top x + h^\top y : \ (x, y) \in S_1\right\}, \tag{MILP$_1$}$$

$$\max\left\{c^\top x + h^\top y : \ (x, y) \in S_2\right\}. \tag{MILP$_2$}$$

Here, the maximum of the optimal values of (MILP$_1$) and (MILP$_2$) would be the optimal value of the origirnal integer program (MILP$_0$).

Note that starting from (MILP$_0$), we have generated two subproblems (MILP$_1$) and (MILP$_2$). We can represent this as a tree structure as in Figure 4.6

For (MILP$_1$) and (MILP$_2$), we solve their LP relaxations,

$$\max\left\{c^\top x + h^\top y : \ (x, y) \in P_1\right\}, \tag{LP$_1$}$$

$$\max\left\{c^\top x + h^\top y : \ (x, y) \in P_2\right\} \tag{LP$_2$}$$

where

$$P_1 = P_0 \cap \left\{(x, y) : \ x_j \geq \lceil x_j^0 \rceil\right\} \quad \text{and} \quad P_2 = P_0 \cap \left\{(x, y) : \ x_j \leq \lfloor x_j^0 \rfloor\right\}.$$

[1] https://www.gurobi.com/documentation/9.5/refman/choosing_the_right_algorit.html

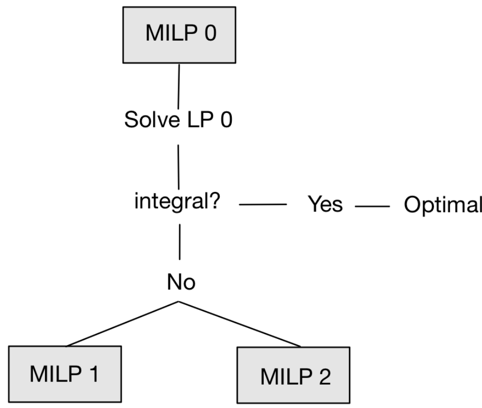

Figure 4.6: Generating two subproblems

- If (LP_1) is infeasible, then it means ($MILP_1$) is infeasible. Then we can remove it from the search tree (**Prune by infeasibility**).
- If (LP_1) returns an integral solution, it means ($MILP_1$) is solved. Then we keep the integral solution but remove ($MILP_1$) from the search tree (**Prune by integrality**).
- If (LP_1) returns a **fractional** solution while the value of (LP_1) is less than or equal to the value of the current best integral solution (possibly from solving ($MILP_2$)), then we remove ($MILP_1$) from the search tree (**Prune by value**).
- If (LP_1) returns a **fractional** solution while the value of (LP_1) is greater than the value of the current best integral solution, then we apply the **branching** procedure on a fractional component.

We apply the same rule for ($MILP_2$) and repeat the procedure for other subproblems remaining in the tree. The tree that this process generates is called the **branch-and-bound** tree.

3.2 Cutting-plane method

Remember our example of the two-dimensional mixed integer linear set. The inequality (4.2) is valid for the mixed-integer set but it is violated by the point $(\beta, 0)$. So, if we take the set of solutions satisfying (4.2), the point $(\beta, 0)$ is removed. Here, we also say that the inequality **cuts off** the point and that the point is **separated** from the mixed-integer set. In this sense, we say that (4.2) is a **cutting-plane** or a **cut**.

Likewise, we take the solution (x^0, y^0) that is optimal to the LP relaxation (LP_0), and we find a cutting-plane that separates (x^0, y^0) from S. Here, an inequality $\alpha^\top x + \gamma^\top y \leq \beta$ is a cutting-plane that separates (x^0, y^0) from S if

$$\alpha^\top x + \gamma^\top y \leq \beta \quad \forall (x, y) \in S \quad \text{and} \quad \alpha^\top x^0 + \gamma^\top y^0 > \beta.$$

Then we get a new relaxation

$$\max \left\{ c^\top x + h^\top y : \ (x, y) \in P_1 \right\}$$

where

$$P_1 = P_0 \cap \left\{ (x, y) : \ \alpha^\top x + \gamma^\top y \leq \beta \right\}.$$

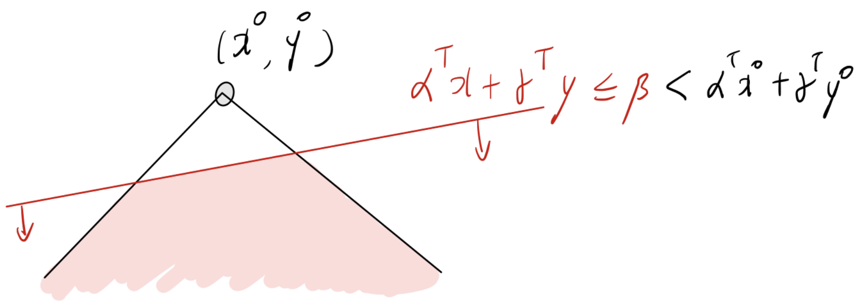

Figure 4.7: Applying a cutting-plane

Repeating the process, we obtain the following cutting-plane algorithm.

- Set $t = 0$.
- $P_0 = \left\{(x, y) \in \mathbb{R}^d \times \mathbb{R}^p : \ Ax + Gy \leq b\right\}$ be the solution set of the LP relaxation (LP_0).
- Until we find an integral solution, repeat the following procedure.
  1. Solve and obtain an optimal solution (x^t, y^t) to $\max\left\{c^\top x + h^\top y : \ (x, y) \in P_t\right\}$.
  2. If (x^t, y^t) is integral, then we stop and (x^t, y^t) is the optimal integral solution.
  3. If (x^t, y^t) is not integral, then find a cutting-plane $\alpha^\top x + \gamma^\top y \leq \beta$ that separates (x^t, y^t) and set
  $$P_{t+1} = P_t \cap \left\{(x, y) : \ \alpha^\top x + \gamma^\top y \leq \beta\right\}$$
  and $t \leftarrow t + 1$.

3.3 Branch-and-cut method

The Branch-and-cut method is basically combining the branch-and-bound method and the cutting-plane algorithm. While running the branch-and-bound procedure, we may find and apply a cutting-plane that separates a fractional solution obtained from solving a subproblem.

The state-of-the-art integer programming software such as CPLEX and Gurobi implements the branch-and-cut method. They apply some specially designed branching rules and cut generation schemes.

One may want to use some problem specific cuts, such as subtour elimination inequalities for TSP and odd set inequalities for the matching problem. In that case, we can use the **(cut) callback** feature within CPLEX and Gurobi to apply user-defined cuts. Basically, the callback feature invokes a node in the branch-and-bound tree and apply the user-defined cut.

- CPLEX: https://www.ibm.com/docs/en/icos/22.1.1?topic=legacy-cut-callback
- Gurobi: https://www.gurobi.com/documentation/10.0/refman/cpp_cb_addcut.html

It is often the case that adding problem-specific cuts leads to a significant improvement in solution time.

How do you find problem specific cuts? One common way is to use the software **PORTA**.

- PORTA: https://porta.zib.de

Given a finite set of vectors as input, PORTA computes the convex hull. Conversely, given a set of linear inequalities, PORTA computes the **extreme points** and the **extreme rays**.

References

[CCZ14] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer Programming*. Springer, 2014. 2

[NW90] George L. Nemhauser and Laurence A. Wolsey. A recursive procedure to generate all cuts for 0-1 mixed integer programs. *Mathematical Programming*, 46:379–390, 1990. 2