

# Mixed Integer Linear Programming

---

Which is sort of famous, for valid reasons

# Mixed Integer Linear Programming

A **Mixed Integer Linear Program** is a problem in the form

$$\operatorname{argmin}_x \{ c^T x \mid Ax \geq b, x \geq 0, x_I \in \mathbb{Z} \}$$

- The cost function and all constraints are linear
- All variables are non-negative
- **Some** variables (those with index in ***I***) are **integer**

**MILP is an extremely powerful formalism**

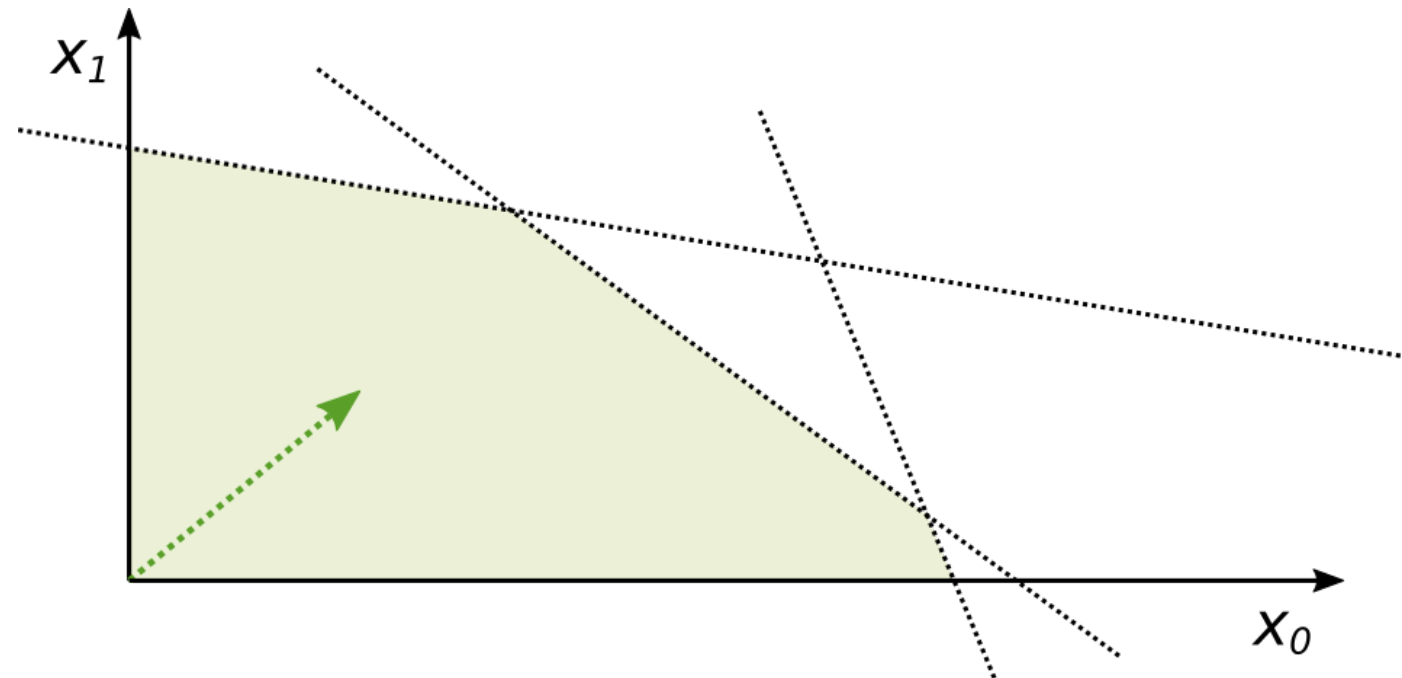
Thanks to the presence of integer variables

- ...Any **combinatorial element** can be modeled
- ...And **non-linearity** can be approximated

**MILP solvers classically rely on three main techniques**

# Linear Relaxation

If we remove the integrality constraints from a MILP we obtain an LP



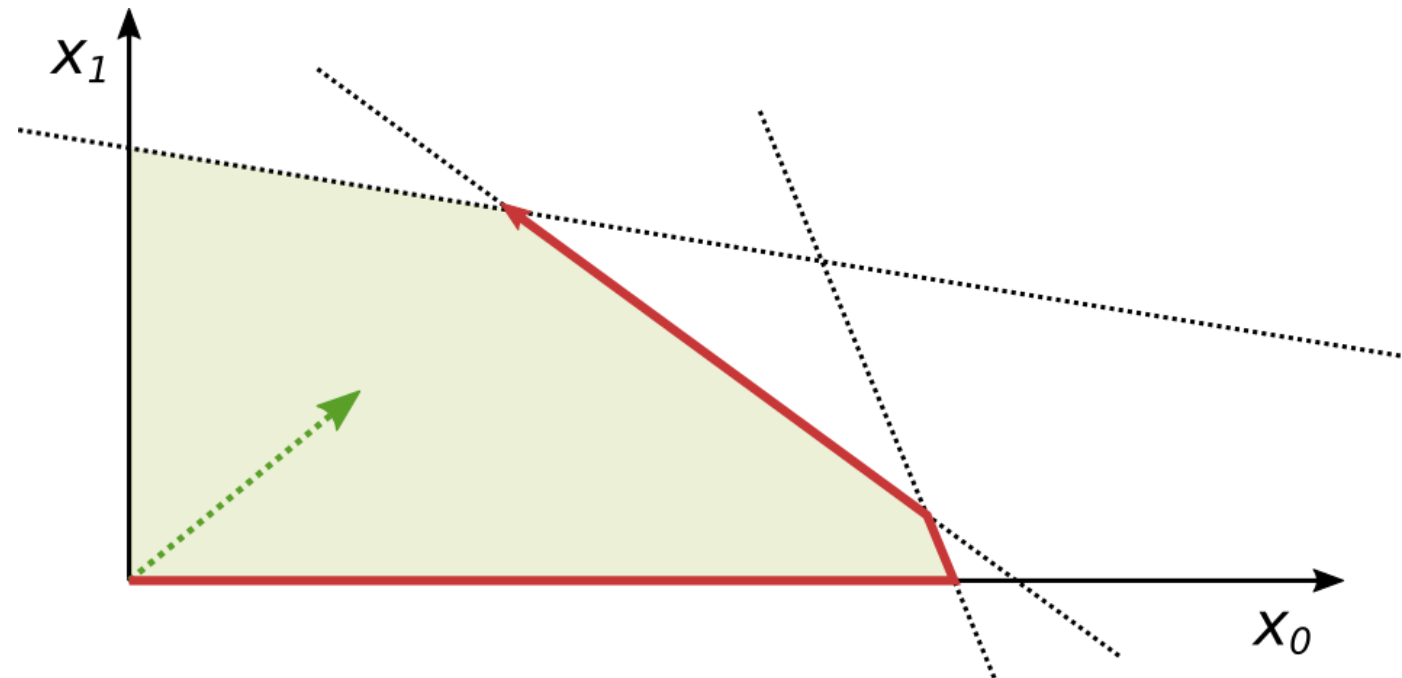
$$\operatorname{argmin}_x \{c^T x \mid Ax \geq b, x \geq 0\}$$

This is called the **linear (or LP) relaxation** of the MILP

- The feasible space is defined via linear constraints  $\Rightarrow$  it is a **polytope**
- The cost vector  $c$  is also the **gradient** and determines an optimization direction

# Linear Relaxation

If we remove the integrality constraints from a MILP we obtain an LP



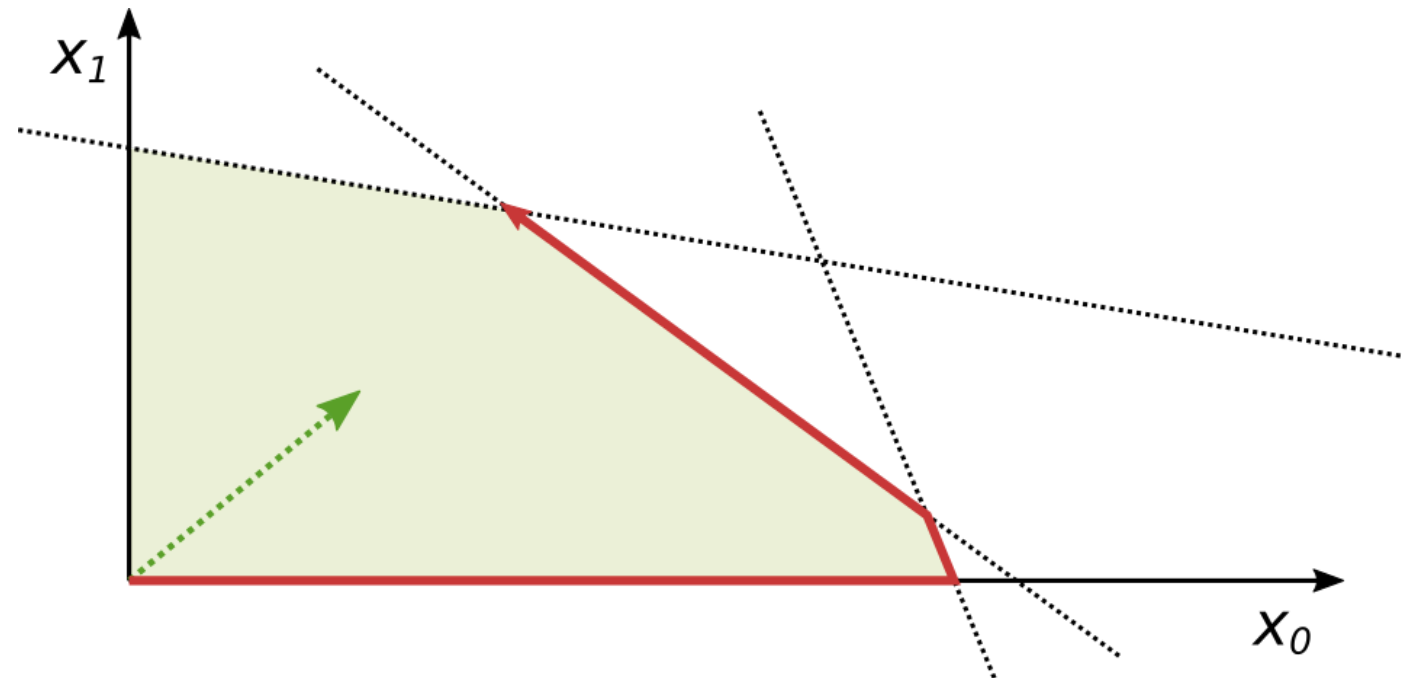
$$\operatorname{argmin}_x \{c^T x \mid Ax \geq b, x \geq 0\}$$

LPs can be solved in pseudo-polynomial time via the Simplex method

- The method start from a polytope vertex
- ...And then moves between adjacent vertexes until the optimum is reached

# Linear Relaxation

If we remove the integrality constraints from a MILP we obtain an LP



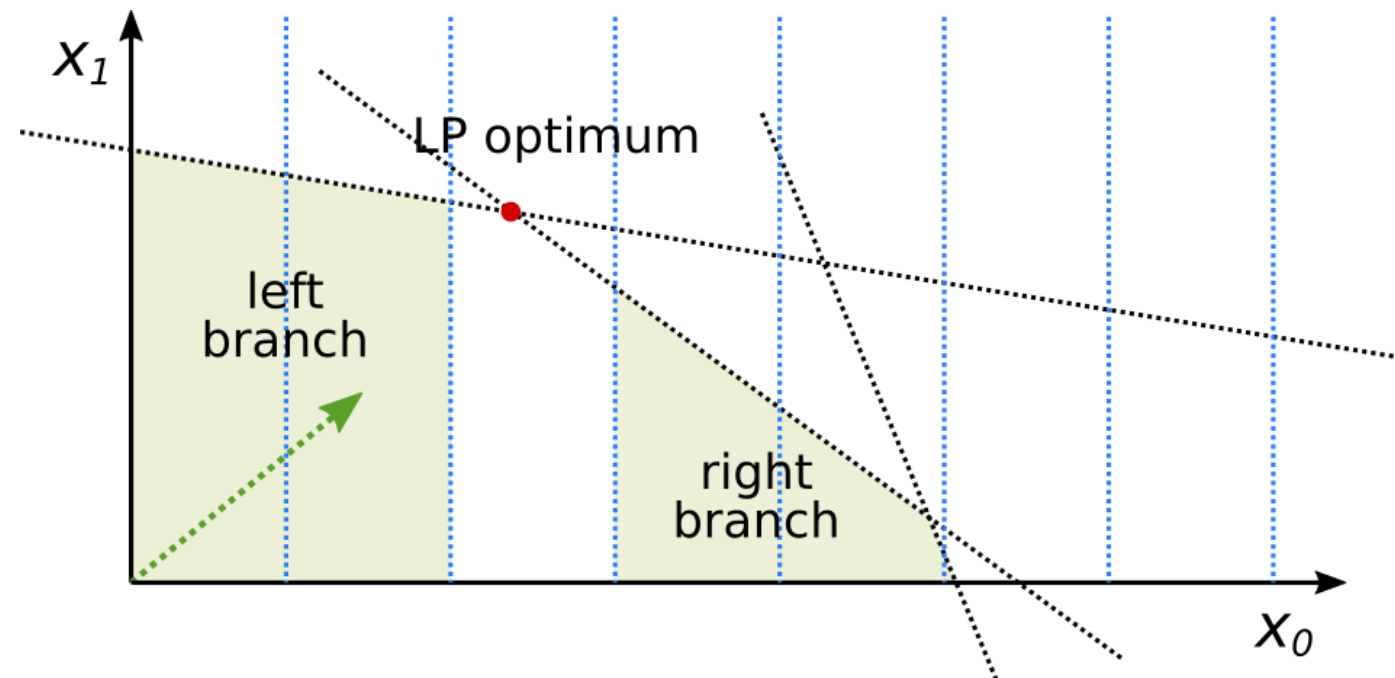
$$\operatorname{argmin}_x \{ c^T x \mid Ax \geq b, x \geq 0 \}$$

LPs can be solved in **polynomial** time via Interior Point methods

- These used to be slower in practice than the Simplex, but not anymore
- In a MILP complex, the Simplex is still preferred (later we will see why)

# Technique #1: Branching

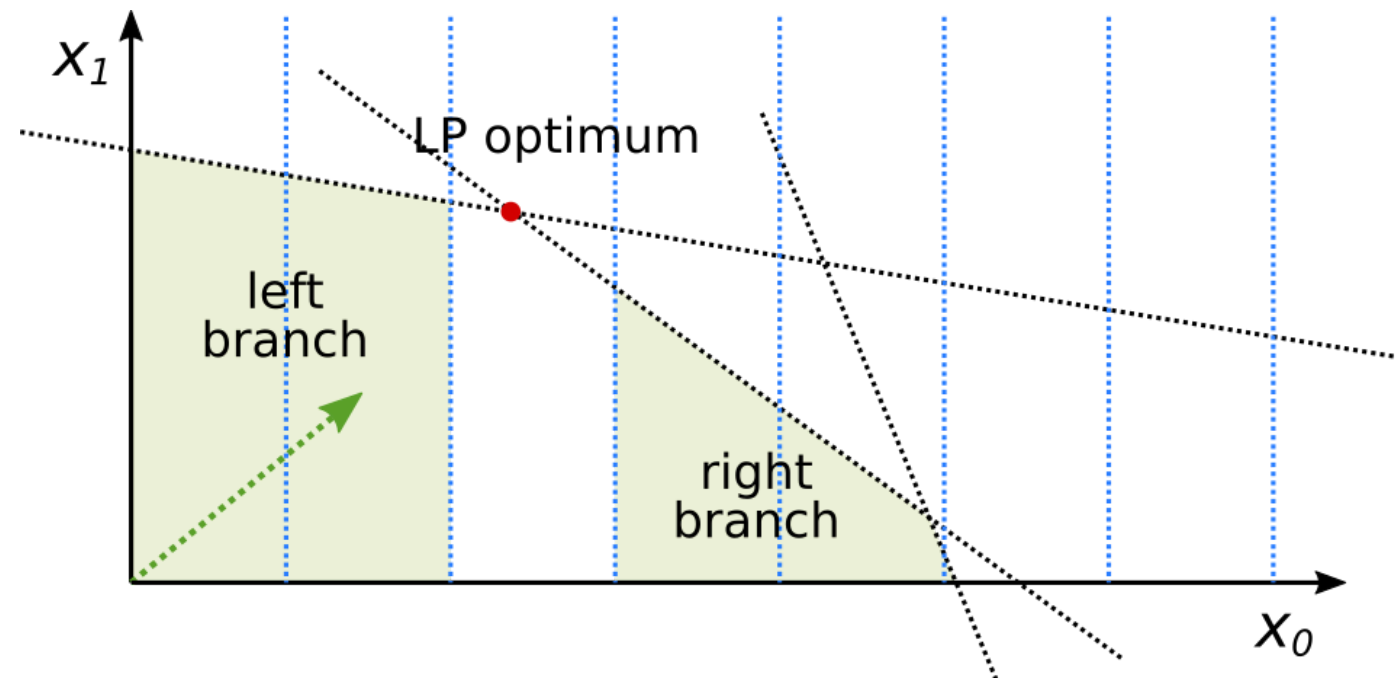
When tackling a MILP, we start by **solving its LP relaxation**



- If all integrality constraints are satisfied, we have found the true optimum
- If some  $x_j$  has a fraction value  $v_j$ , we **split the problem** in two:
  - In the first subproblem, we add the constraint  $x_j \leq \lfloor v_j \rfloor$
  - In the second subproblem, we add  $x_j \geq \lceil v_j \rceil$
- Then we can repeat the whole process

# Technique #1: Branching

This approach is referred to as **branching**



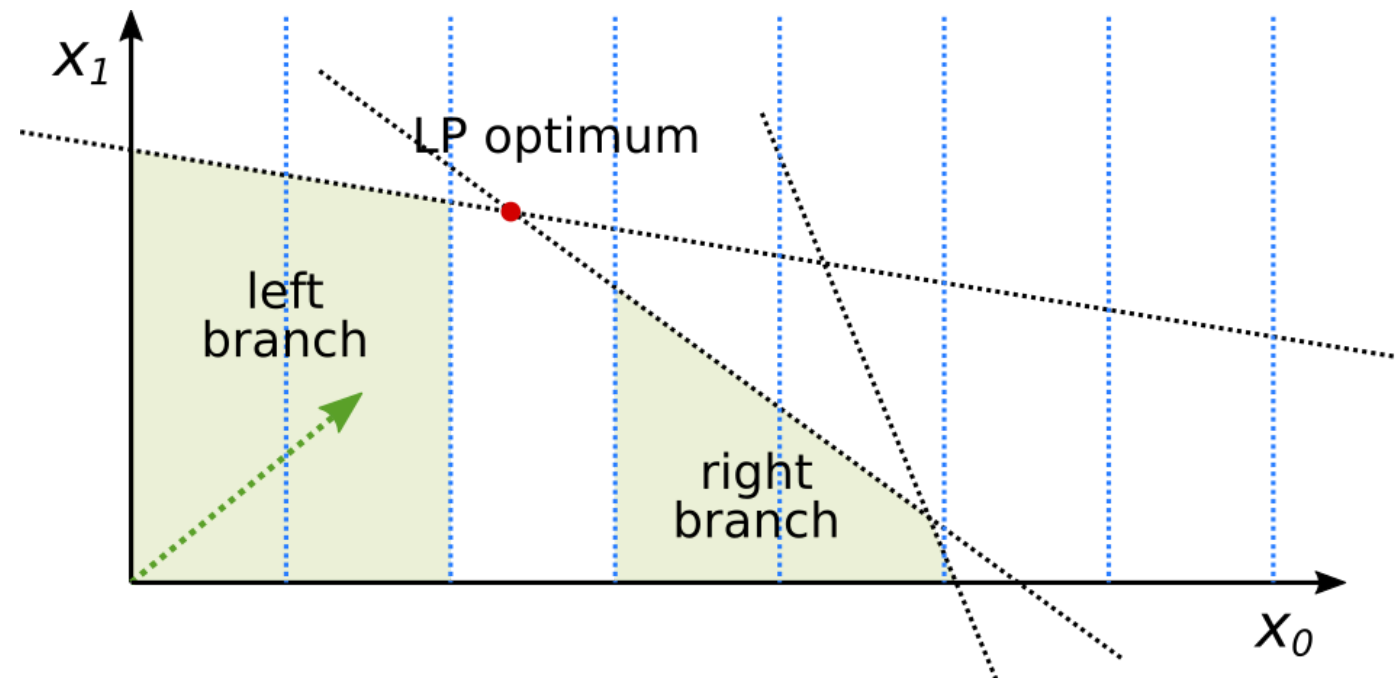
- The first subproblem is also known as the **left branch**
- The second as the **right branch**

**Branching is the main method that makes MILP solvers complete**

- In the worst case, we end up with a **search tree** having an exponential number of nodes
- ...But that's somewhat unavoidable, since **solving MILP is NP-hard**

# Technique #1: Branching

This approach is referred to as **branching**



**Branching is also the reason why the Simplex method is preferred to MILPs**

- The Simplex method has a "dual" version
  - ...Whose optimum can be updated efficiently when new constraints are added
- ...And you can guess that's a pretty common operation ;-)



## Technique #2: Bounding

Let's look again at the LP relaxation

$$\operatorname{argmin}_x \{c^T x \mid Ax \geq b, x \geq 0\}$$

- The problem has the same structure
- ...But a larger feasible space (that's why it is called a relaxation)

Hence, its optimal cost will be a **lower bound** (say  $lb$ ) for the MILP

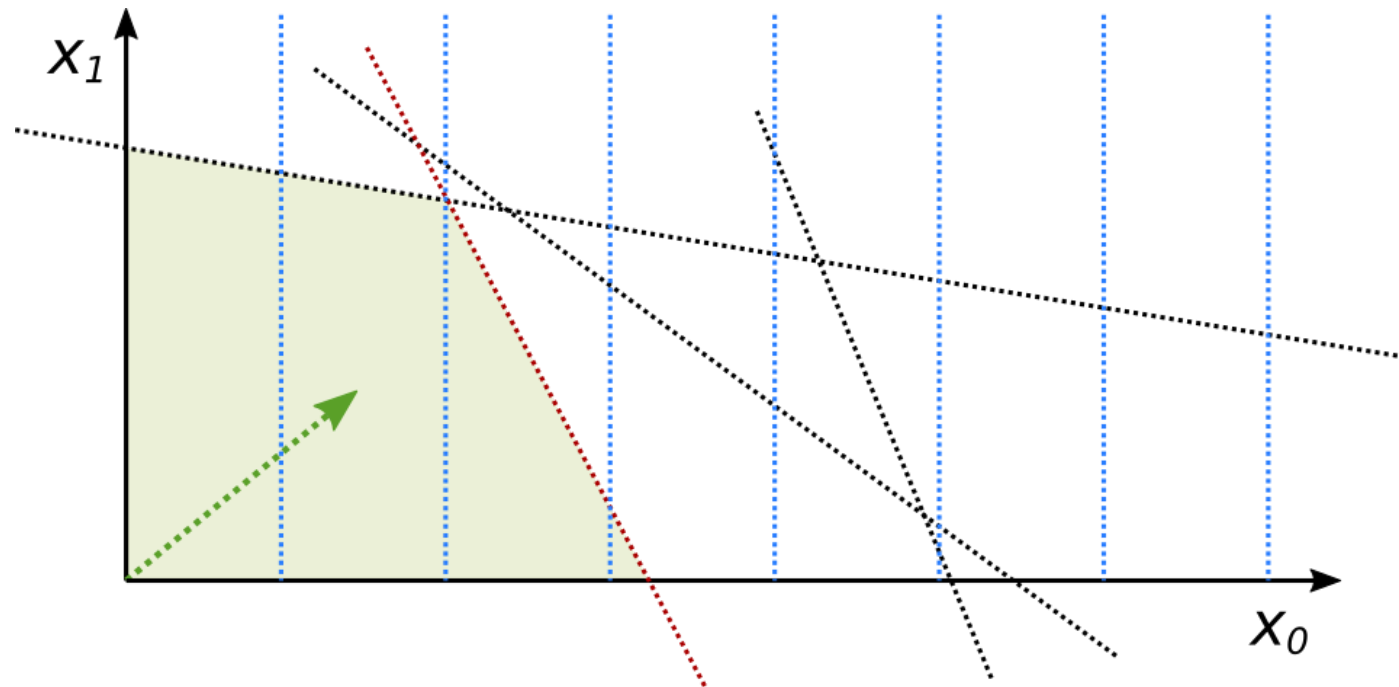
**We can use this bound as an early stopping criterion**

- Let  $x^*$  be the best (mixed-integer) solution we have found so far
- If for some node of the search tree we have  $lb > c^T x^*$
- Then we have no hope of beating  $x^*$  and we can destroy (**fathom**) the node

**Branching + Bounding = Branch & Bound**

## Technique #3: Cutting Planes

It is also common to speed-up MILP solution by using **cutting planes**



**Cutting planes are linear inequalities **inferred** by relying on some property**

- In MILP they are typically inferred based on integrality constraint
- They must be **valid** for any feasible solution
- They are useful if they force a fractional solution to become closer to integer

## Technique #3: Cutting Planes

A common example is that of **Gomory Cuts**

While solving the simplex, we end up with many equalities in the form:

$$x_i + \sum_{j \in \bar{B}} \bar{a}_{ij} x_j = \bar{b}_i$$

- Where  $x_i > 0$  and  $x_j = 0, \forall j \in \bar{B}$
- $B$  = the set of indexes of **non-zero** variables in the current LP solution (base)
- $\bar{B}$  = the set of indexes of **zero** variables in the current LP solution
- We will assume all variables are integer, for simplicity

**We can rewrite the equation as**

$$x_i + \sum_{j \in \bar{B}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor + \lfloor \bar{a}_{ij} \rfloor) x_j = \bar{b}_i - \lfloor \bar{b}_i \rfloor + \lfloor \bar{b}_i \rfloor$$

## Technique #3: Cutting Planes

By simple algebraic manipulation we can then get:

$$x_i + \sum_{j \in \bar{B}} \lfloor \bar{a}_{ij} \rfloor x_j - \lfloor \bar{b}_i \rfloor = - \sum_{j \in \bar{B}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j + (\bar{b}_i - \lfloor \bar{b}_i \rfloor)$$

We will build an inequality that is valid for **any feasible, integer** point:

- The right-most part is necessarily  $< 1$ , since:
  - $\bar{b}_i - \lfloor \bar{b}_i \rfloor$  is positive and fractional
  - Each  $\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor$  is positive (and fractional)
  - Each  $x_j$  must be  $\geq 0$
- The left-most part is necessarily an integer, since:
  - $\lfloor \bar{b}_i \rfloor$  is integer and each  $\lfloor \bar{a}_{ij} \rfloor$  is integer
  - Variables are integer as per our assumption

## Technique #3: Cutting Planes

By simple algebraic manipulation we can then get:

$$x_i + \sum_{j \in \bar{B}} \lfloor \bar{a}_{ij} \rfloor x_j - \lfloor \bar{b}_i \rfloor = - \sum_{j \in \bar{B}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j + (\bar{b}_i - \lfloor \bar{b}_i \rfloor)$$

- Hence, the right-most part should be  $< 1$  and integer
- ...Meaning that it must be  $\leq 0$

$$- \sum_{j \in \bar{B}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j + (\bar{b}_i - \lfloor \bar{b}_i \rfloor) \leq 0$$

And from here:

$$\sum_{j \in \bar{B}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j \geq (\bar{b}_i - \lfloor \bar{b}_i \rfloor)$$

## Technique #3: Cutting Planes

This inequality is the Gomory Cut

$$\sum_{j \in \bar{B}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j \geq (\bar{b}_i - \lfloor \bar{b}_i \rfloor)$$

- Now, if we target a  $x_i$  that should be integer, i.e.  $i \in I$
- ...But it's fractional in the current solution

Then we have  $\bar{b}_i - \lfloor \bar{b}_i \rfloor > 0$

- Combined with the fact that  $x_j = 0, \forall j \in \bar{B}$  in the current solution
- We have that the cut is actually making the solution no longer feasible

**Branching + Bounding + Cutting Planes = Branch & Cut**

- Using cutting planes can speed up the solution process considerably
- But it's best not to overdo it, since subsequent cuts may become weaker

# Some Considerations

## We have just scratched the surface with MILP

Modern MILP solver do much more:

- Presolving
- Constraint propagations
- Symmetry breaking
- ...

## MILP methods have a long history

- There is a **huge gap** between the solver performance
- The best solvers (Gurobi, Cplex, Mosek) are commercial (free for academics)
- Then you have a single semi-free solver (SCIP)
- ...A good free solver (CBC)
- ...And finally there is stuff you **should not** touch (glpk, lpsolve)