Linear Programming

[V. ch9]: Integer Programming

Phillip Keldenich Ahmad Moradi

Department of Computer Science
Algorithms Department
TU Braunschweig

January 23, 2023

Motivation

Definition

Branch and Bound

Branch and Cut

VERTEX COVER

For a given graph $G = (V, E)$, the Vertex Cover problem asks for a minimum-cardinality subset $C \subseteq V$ of vertices such that each edge $vw \in E$ has least one endpoint in C, i.e., $\{v, w\} \cap C \neq \emptyset$.

VERTEX COVER

For a given graph $G = (V, E)$, the Vertex Cover problem asks for a minimum-cardinality subset $C \subseteq V$ of vertices such that each edge $vw \in E$ has least one endpoint in C, i.e., $\{v, w\} \cap C \neq \emptyset$.

Trying to model this as linear program:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$

$$\forall v \in V : 0 \leq x_v \leq 1$$

$$\forall vw \in E : x_v + x_w \geq 1$$

What happens on a triangle (K_3, complete graph on 3 vertices)?

VERTEX COVER

For a given graph $G = (V, E)$, the Vertex Cover problem asks for a minimum-cardinality subset $C \subseteq V$ of vertices such that each edge $vw \in E$ has least one endpoint in C, i.e., $\{v, w\} \cap C \neq \emptyset$.

Trying to model this as linear program:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$

$$\forall v \in V : 0 \leq x_v \leq 1$$

$$\forall vw \in E : x_v + x_w \geq 1$$

What happens on a triangle (K_3, complete graph on 3 vertices)?

Notes: Vertex Cover is NP-hard! LP is not NP-hard unless P = NP.

VERTEX COVER

For a given graph $G = (V, E)$, the Vertex Cover problem asks for a minimum-cardinality subset $C \subseteq V$ of vertices such that each edge $vw \in E$ has least one endpoint in C, i.e., $\{v, w\} \cap C \neq \emptyset$.

Trying to model this as linear program:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$

$$\forall v \in V : 0 \leq x_v \leq 1$$

$$\forall vw \in E : x_v + x_w \geq 1$$

What happens on a triangle (K_3, complete graph on 3 vertices)?

Notes: Vertex Cover is NP-hard! LP is not NP-hard unless P = NP.
LP is in P, even though Simplex is not a polynomial-time algorithm.

Vertex Cover

For a given graph $G = (V, E)$, the Vertex Cover problem asks for a minimum-cardinality subset $C \subseteq V$ of vertices such that each edge $vw \in E$ has least one endpoint in C, i.e., $\{v, w\} \cap C \neq \emptyset$.

Trying to model this as linear program:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$

$$\forall v \in V : 0 \leq x_v \leq 1$$

$$\forall vw \in E : x_v + x_w \geq 1$$

What happens on a triangle (K_3, complete graph on 3 vertices)?

Notes: Vertex Cover is NP-hard! LP is not NP-hard unless P = NP.
LP is in P, even though Simplex is not a polynomial-time algorithm.
Unless P = NP, we thus cannot expect to fully model Vertex Cover as LP!

VERTEX COVER

For a given graph $G = (V, E)$, the Vertex Cover problem asks for a minimum-cardinality subset $C \subseteq V$ of vertices such that each edge $vw \in E$ has least one endpoint in C, i.e., $\{v, w\} \cap C \neq \emptyset$.

Trying to model this as linear program:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$

$$\forall v \in V : 0 \leq x_v \leq 1$$

$$\forall vw \in E : x_v + x_w \geq 1$$

What happens on a triangle (K_3, complete graph on 3 vertices)?

Notes: Vertex Cover is NP-hard! LP is not NP-hard unless $\mathsf{P} = \mathsf{NP}$.
LP is in P, even though Simplex is not a polynomial-time algorithm.
Unless $\mathsf{P} = \mathsf{NP}$, we thus cannot expect to fully model Vertex Cover as LP!
Idea: Extend LP to be able to model NP-hard problems! Any ideas?

Vertex Cover Model

Adapting our model:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$
$$\forall v \in V : 0 \leq x_v \leq 1$$
$$\forall vw \in E : x_v + x_w \geq 1$$

Vertex Cover Model

Adapting our model:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$

$$\forall v \in V : 0 \leq x_v \leq 1$$

$$\forall vw \in E : x_v + x_w \geq 1$$

$$\forall v \in V : x_v \in \mathbb{Z}$$

VERTEX COVER MODEL

Adapting our model:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$
$$\forall v \in V : 0 \leq x_v \leq 1$$
$$\forall vw \in E : x_v + x_w \geq 1$$
$$\forall v \in V : x_v \in \mathbb{Z}$$

Modeling SAT: How can we do that?

VERTEX COVER MODEL

Adapting our model:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$

$$\forall v \in V : 0 \leq x_v \leq 1$$

$$\forall vw \in E : x_v + x_w \geq 1$$

$$\forall v \in V : x_v \in \mathbb{Z}$$

Modeling SAT: How can we do that?
Variable $x_v \in \{0, 1\}$ for each Boolean variable v in the formula $\varphi = \bigwedge_i C_i$.

VERTEX COVER MODEL

Adapting our model:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$

$$\forall v \in V : 0 \leq x_v \leq 1$$

$$\forall vw \in E : x_v + x_w \geq 1$$

$$\forall v \in V : x_v \in \mathbb{Z}$$

Modeling SAT: How can we do that?
Variable $x_v \in \{0, 1\}$ for each Boolean variable v in the formula $\varphi = \bigwedge_i C_i$.

Value of literal $\ell = v$: x_v, $\bar{\ell} =$

Vertex Cover Model

Adapting our model:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$
$$\forall v \in V : 0 \leq x_v \leq 1$$
$$\forall vw \in E : x_v + x_w \geq 1$$
$$\forall v \in V : x_v \in \mathbb{Z}$$

Modeling SAT: How can we do that?
Variable $x_v \in \{0, 1\}$ for each Boolean variable v in the formula $\varphi = \bigwedge_i C_i$.
Value of literal $\ell = v$: x_v, $\bar{\ell} = 1 - x_v$

Vertex Cover Model

Adapting our model:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$
$$\forall v \in V : 0 \leq x_v \leq 1$$
$$\forall vw \in E : x_v + x_w \geq 1$$
$$\forall v \in V : x_v \in \mathbb{Z}$$

Modeling SAT: How can we do that?
Variable $x_v \in \{0, 1\}$ for each Boolean variable v in the formula $\varphi = \bigwedge_i C_i$.

Value of literal $\ell = v$: x_v, $\bar{\ell} = 1 - x_v$
Modeling a clause, e.g., $v_1 \vee \overline{v_2} \vee \overline{v_3}$:

Vertex Cover Model

Adapting our model:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$

$$\forall v \in V : 0 \leq x_v \leq 1$$

$$\forall vw \in E : x_v + x_w \geq 1$$

$$\forall v \in V : x_v \in \mathbb{Z}$$

Modeling SAT: How can we do that?
Variable $x_v \in \{0, 1\}$ for each Boolean variable v in the formula $\varphi = \bigwedge_i C_i$.

Value of literal $\ell = v$: x_v, $\bar{\ell} = 1 - x_v$
Modeling a clause, e.g., $v_1 \vee \overline{v_2} \vee \overline{v_3}$: $x_{v_1} + (1 - x_{v_2}) + (1 - x_{v_3}) \geq 1$

VERTEX COVER MODEL

Adapting our model:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$

$$\forall v \in V : 0 \leq x_v \leq 1$$

$$\forall vw \in E : x_v + x_w \geq 1$$

$$\forall v \in V : x_v \in \mathbb{Z}$$

Modeling SAT: How can we do that?
Variable $x_v \in \{0, 1\}$ for each Boolean variable v in the formula $\varphi = \bigwedge_i C_i$.

Value of literal $\ell = v$: x_v, $\bar{\ell} = 1 - x_v$
Modeling a clause, e.g., $v_1 \vee \overline{v_2} \vee \overline{v_3}$: $x_{v_1} + (1 - x_{v_2}) + (1 - x_{v_3}) \geq 1$

Notes: Obviously, LP with integer variables is NP-hard.

VERTEX COVER MODEL

Adapting our model:

$$\min \sum_{v \in V} x_v \text{ s.t.}$$

$$\forall v \in V : 0 \leq x_v \leq 1$$

$$\forall vw \in E : x_v + x_w \geq 1$$

$$\forall v \in V : x_v \in \mathbb{Z}$$

Modeling SAT: How can we do that?

Variable $x_v \in \{0, 1\}$ for each Boolean variable v in the formula $\varphi = \bigwedge_i C_i$.

Value of literal $\ell = v$: x_v, $\bar{\ell} = 1 - x_v$

Modeling a clause, e.g., $v_1 \vee \overline{v_2} \vee \overline{v_3}$: $x_{v_1} + (1 - x_{v_2}) + (1 - x_{v_3}) \geq 1$

Notes: Obviously, LP with integer variables is NP-hard.

Even deciding feasibility is NP-hard (see SAT example).

INTEGER PROGRAM

A linear program where all variables are restricted to $\mathbb{Z}$ is called *integer program* (IP).

A linear program where some (but not all) variables are restricted to $\mathbb{Z}$ is called *mixed integer program* (MIP).

A linear program where all variables are restricted to $\{0, 1\}$ is called 0-1-program or binary program.

0-1-programs, IP and MIP are NP-complete.
They can be used to straightforwardly model many NP-complete problems.
Good solvers exist that can solve small to moderate size instances of many NP-hard problems.

Motivation

Definition

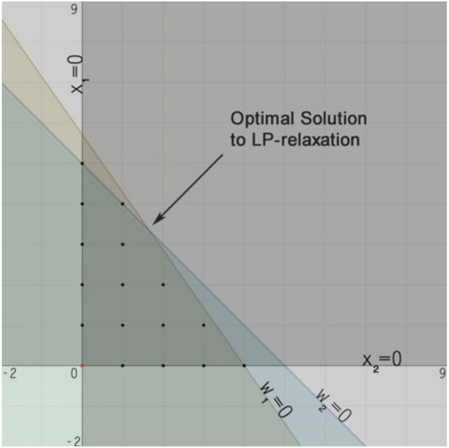
Branch and Bound

Branch and Cut

SOLVING IPS

How do we solve integer programs?
Using a technique called Branch & Bound, or an extension of that; let's show an example.

$$\max 17x_1 + 12x_2 \text{ s.t.}$$
$$10x_1 + 7x_2 \leq 40$$
$$x_1 + x_2 \leq 5$$
$$x_1, x_2 \geq 0$$
$$x_1, x_2 \in \mathbb{Z}$$



Solving the LP relaxation (of subproblem P_0, the original problem) gives us
$\zeta^0 = 68 + 1/3, x_1^0 = 5/3, x_2^0 = 10/3$.
This tells us the optimal (integer) solution is not better than ζ^0.

BRANCH & BOUND: EXAMPLE

How do we continue when the LP relaxation of a subproblem P_i has a non-integral optimal solution?

BRANCH & BOUND: EXAMPLE

How do we continue when the LP relaxation of a subproblem P_i has a non-integral optimal solution?

- We branch, i.e., split P_i into (at least two) new subproblems.

Branch & Bound: Example

How do we continue when the LP relaxation of a subproblem P_i has a non-integral optimal solution?

- We branch, i.e., split P_i into (at least two) new subproblems.
- Together, the subproblems must cover all possible integer solutions in P_i.

Branch & Bound: Example

How do we continue when the LP relaxation of a subproblem P_i has a non-integral optimal solution?

- We branch, i.e., split P_i into (at least two) new subproblems.
- Together, the subproblems must cover all possible integer solutions in P_i.
- None of the subproblems must contain the non-integral optimal solution of P_i.

BRANCH & BOUND: EXAMPLE

How do we continue when the LP relaxation of a subproblem P_i has a non-integral optimal solution?

- We branch, i.e., split P_i into (at least two) new subproblems.
- Together, the subproblems must cover all possible integer solutions in P_i.
- None of the subproblems must contain the non-integral optimal solution of P_i.
- Ideally, they should also be disjoint, i.e., each solution is contained in at most one of the new problems.

BRANCH & BOUND: EXAMPLE

How do we continue when the LP relaxation of a subproblem P_i has a non-integral optimal solution?

- We branch, i.e., split P_i into (at least two) new subproblems.
- Together, the subproblems must cover all possible integer solutions in P_i.
- None of the subproblems must contain the non-integral optimal solution of P_i.
- Ideally, they should also be disjoint, i.e., each solution is contained in at most one of the new problems.
- In the example, $x_1^0 = 5/3$; in any integer solution, we must have $x_1 \leq 1$ or $x_1 \geq 2$. We create two new subproblems P_1 (by adding $x_1 \leq 1$) and P_2 (by adding $x_1 \geq 2$) to the original constraints.

BRANCH & BOUND: EXAMPLE

How do we continue when the LP relaxation of a subproblem P_i has a non-integral optimal solution?

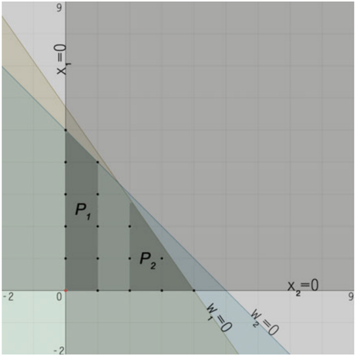
- We branch, i.e., split P_i into (at least two) new subproblems.
- Together, the subproblems must cover all possible integer solutions in P_i.
- None of the subproblems must contain the non-integral optimal solution of P_i.
- Ideally, they should also be disjoint, i.e., each solution is contained in at most one of the new problems.
- In the example, $x_1^0 = 5/3$; in any integer solution, we must have $x_1 \leq 1$ or $x_1 \geq 2$. We create two new subproblems P_1 (by adding $x_1 \leq 1$) and P_2 (by adding $x_1 \geq 2$) to the original constraints.
- In general, we can take any integer variable x with non-integral value θ and use $x \leq \lfloor \theta \rfloor$ and $x \geq \lceil \theta \rceil$ as new constraints.

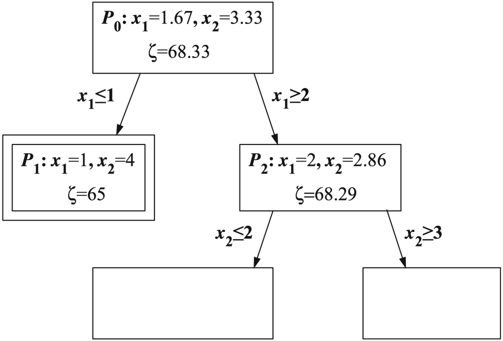
Branch & Bound: Example

How do we continue when the LP relaxation of a subproblem P_i has a non-integral optimal solution?

- We branch, i.e., split P_i into (at least two) new subproblems.
- Together, the subproblems must cover all possible integer solutions in P_i.
- None of the subproblems must contain the non-integral optimal solution of P_i.
- Ideally, they should also be disjoint, i.e., each solution is contained in at most one of the new problems.
- In the example, $x_1^0 = 5/3$; in any integer solution, we must have $x_1 \leq 1$ or $x_1 \geq 2$. We create two new subproblems P_1 (by adding $x_1 \leq 1$) and P_2 (by adding $x_1 \geq 2$) to the original constraints.
- In general, we can take any integer variable x with non-integral value θ and use $x \leq \lfloor \theta \rfloor$ and $x \geq \lceil \theta \rceil$ as new constraints.
- The optimal integer solution to P_i is the best integer solution found recursively in the subproblems.

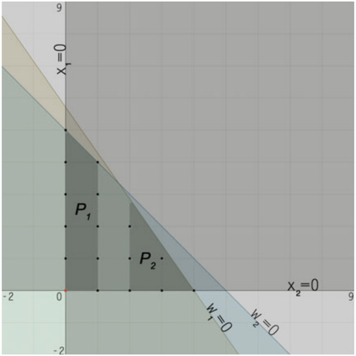
RESULT OF FIRST BRANCHING

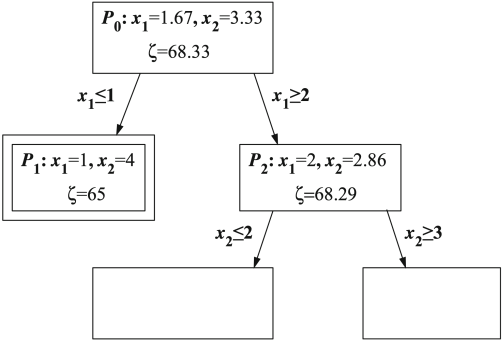




The subproblems form a *search tree*. The relaxation of the left child problem P_1 has an integral solution. It does not need another branch and becomes a leaf of the search tree (double box).

RESULT OF FIRST BRANCHING

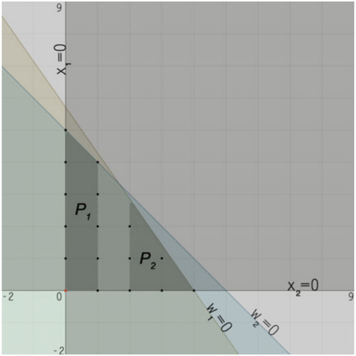


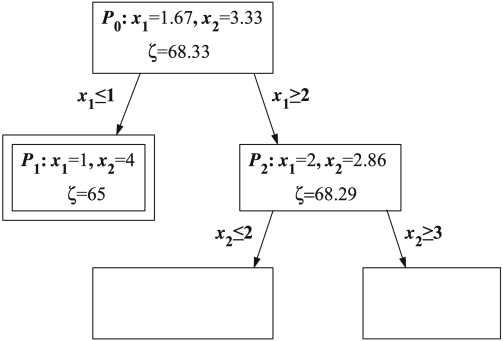


The subproblems form a *search tree*. The relaxation of the left child problem P_1 has an integral solution. It does not need another branch and becomes a leaf of the search tree (double box).

What if the right child had an objective value $\zeta \leq 65$?

Result of First Branching

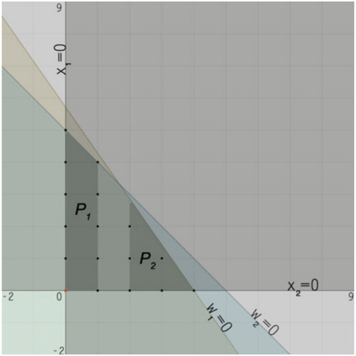


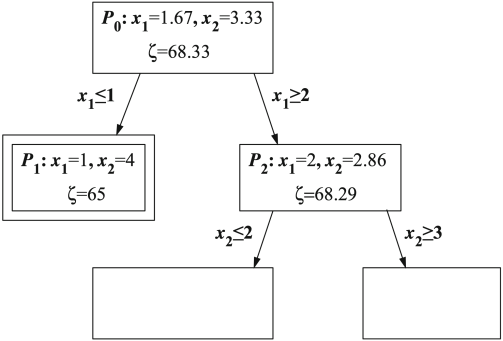


The subproblems form a *search tree*. The relaxation of the left child problem P_1 has an integral solution. It does not need another branch and becomes a leaf of the search tree (double box).

What if the right child had an objective value $\zeta \leq 65$?
We could make it a leaf because its bound is not better than a solution we already found!

RESULT OF FIRST BRANCHING

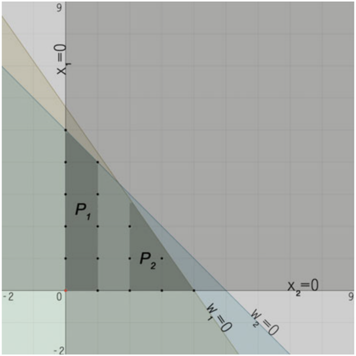


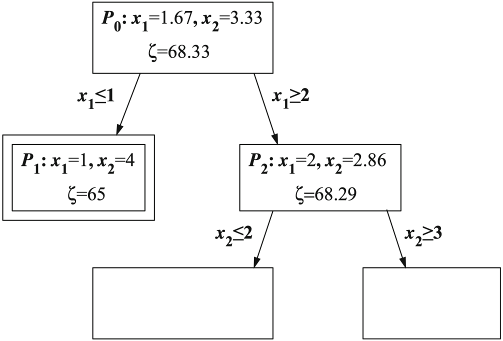


The subproblems form a *search tree*. The relaxation of the left child problem P_1 has an integral solution. It does not need another branch and becomes a leaf of the search tree (double box).

What if the right child had an objective value $\zeta \leq 65$?
We could make it a leaf because its bound is not better than a solution we already found!
This is called pruning and important for making Branch & Bound efficient in practice.

Result of First Branching





The subproblems form a *search tree*. The relaxation of the left child problem P_1 has an integral solution. It does not need another branch and becomes a leaf of the search tree (double box).

What if the right child had an objective value $\zeta \leq 65$?
We could make it a leaf because its bound is not better than a solution we already found!
This is called pruning and important for making Branch & Bound efficient in practice.
Pruning relies on good bounds, i.e., strong LP relaxations. If optimal solutions are much worse than the bounds we obtain, pruning can only be applied rarely and the number of subproblems rises.

CONTINUING

Exploring a node of the search tree means:

- solving the LP relaxation (most expensive step),
- deciding whether and how to branch.

Continuing

Exploring a node of the search tree means:

- solving the LP relaxation (most expensive step),
- deciding whether and how to branch.

We usually explore nodes in a (sort of) depth-first order. This has several advantages:

- Memory requirements: DFS needs essentially $O(\text{depth})$.
  BFS needs to store a level of the tree (often $\Omega(\text{nodes})$).

Continuing

Exploring a node of the search tree means:

- solving the LP relaxation (most expensive step),
- deciding whether and how to branch.

We usually explore nodes in a (sort of) depth-first order. This has several advantages:

- Memory requirements: DFS needs essentially $O(\text{depth})$.
  BFS needs to store a level of the tree (often $\Omega(\text{nodes})$).
- Integer solutions are often deep in the tree. We need them to prune; earlier is better. When aborting the search, e.g., due to a timeout, we want to have a good solution.

CONTINUING

Exploring a node of the search tree means:

- solving the LP relaxation (most expensive step),
- deciding whether and how to branch.

We usually explore nodes in a (sort of) depth-first order. This has several advantages:

- Memory requirements: DFS needs essentially $O(\text{depth})$.
  BFS needs to store a level of the tree (often $\Omega(\text{nodes})$).
- Integer solutions are often deep in the tree. We need them to prune; earlier is better. When aborting the search, e.g., due to a timeout, we want to have a good solution.
- Warm Starting: In DFS, the next problem we solve is very often only one added constraint away from the previously solved one. We can hope that we can use the previous optimal basis as a starting point for solving the next problem with much fewer iterations than starting from scratch. Let's see how that could be done!

Dual Simplex Warm Starting

Consider our original problem P_0 and its related problem P_2 (P_0 with $x_1 \geq 2$).
Optimal dictionary for P_0:

$$\begin{array}{rcrcrcr} \zeta = & \frac{205}{3} & - & \frac{5}{3}w_1 & - & \frac{1}{3}w_2 \\ \hline x_1 = & \frac{5}{3} & - & \frac{1}{3}w_1 & + & \frac{7}{3}w_2 \\ x_2 = & \frac{10}{3} & + & \frac{1}{3}w_1 & - & \frac{10}{3}w_2 \end{array}$$

What happens when we add $x_1 \geq 2$?

Dual Simplex Warm Starting

Consider our original problem P_0 and its related problem P_2 (P_0 with $x_1 \geq 2$).
Optimal dictionary for P_0:

$$
\begin{array}{rrrr}
\zeta = & \frac{205}{3} - & \frac{5}{3}w_1 - & \frac{1}{3}w_2 \\
\hline
x_1 = & \frac{5}{3} - & \frac{1}{3}w_1 + & \frac{7}{3}w_2 \\
x_2 = & \frac{10}{3} + & \frac{1}{3}w_1 - & \frac{10}{3}w_2
\end{array}
$$

What happens when we add $x_1 \geq 2$? We get a slack variable $g_1 = x_1 - 2 = -{}^1\!/_3 - {}^{w_1}\!/_3 + {}^{7w_2}\!/_3$.

Dual Simplex Warm Starting

Consider our original problem P_0 and its related problem P_2 (P_0 with $x_1 \geq 2$).
Optimal dictionary for P_0:

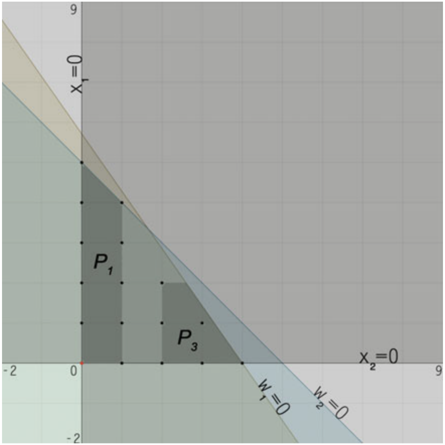
$$\begin{array}{rrrr} \zeta = & \frac{205}{3} - & \frac{5}{3}w_1 - & \frac{1}{3}w_2 \\ \hline x_1 = & \frac{5}{3} - & \frac{1}{3}w_1 + & \frac{7}{3}w_2 \\ x_2 = & \frac{10}{3} + & \frac{1}{3}w_1 - & \frac{10}{3}w_2 \end{array}$$

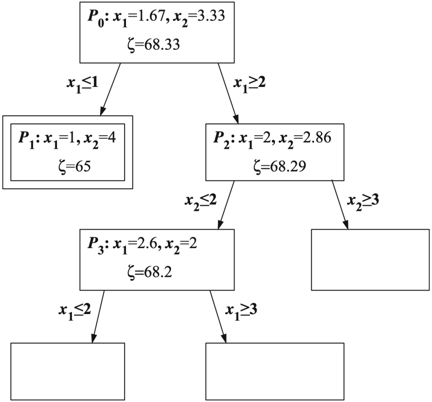
What happens when we add $x_1 \geq 2$? We get a slack variable $g_1 = x_1 - 2 = -1/3 - w_1/3 + 7w_2/3$. That variable is non-basic (because the constraint is violated) and makes the new dictionary primally infeasible. It is dually feasible however, so we can use dual Simplex.

$$\begin{array}{rrrr} \zeta = & \frac{205}{3} - & \frac{5}{3}w_1 - & \frac{1}{3}w_2 \\ \hline x_1 = & \frac{5}{3} - & \frac{1}{3}w_1 + & \frac{7}{3}w_2 \\ x_2 = & \frac{10}{3} + & \frac{1}{3}w_1 - & \frac{10}{3}w_2 \\ g_1 = & -\frac{1}{3} - & \frac{1}{3}w_1 + & \frac{7}{3}w_2 \end{array}$$

Continuing Our Example

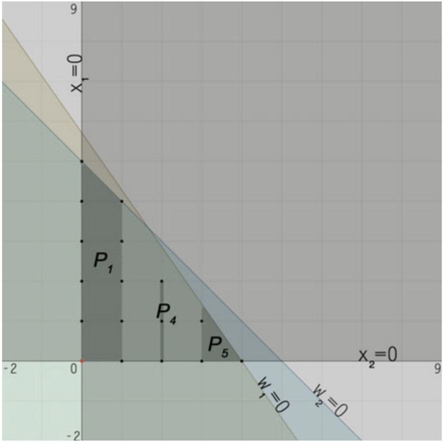
After exploring P_3:

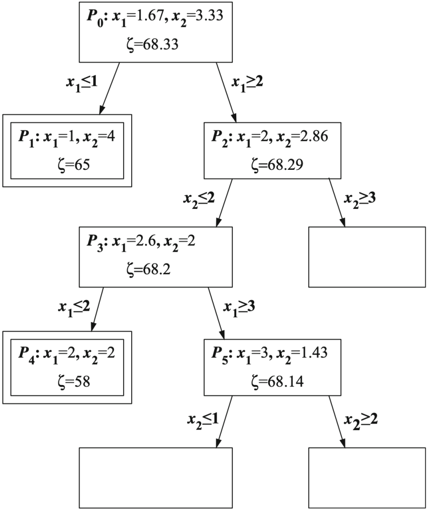




Continuing Our Example

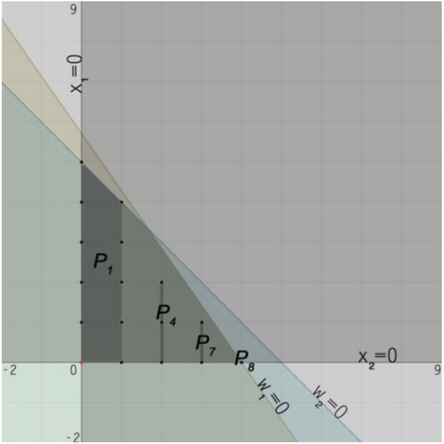
After exploring P_4, P_5:

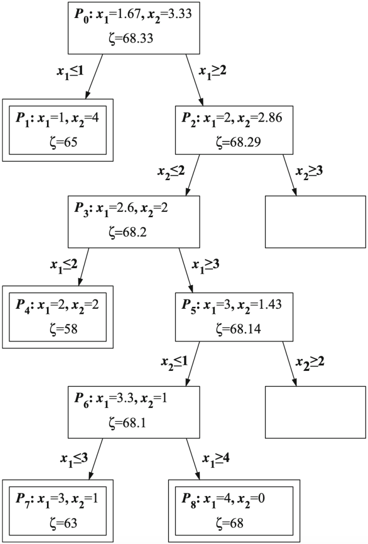




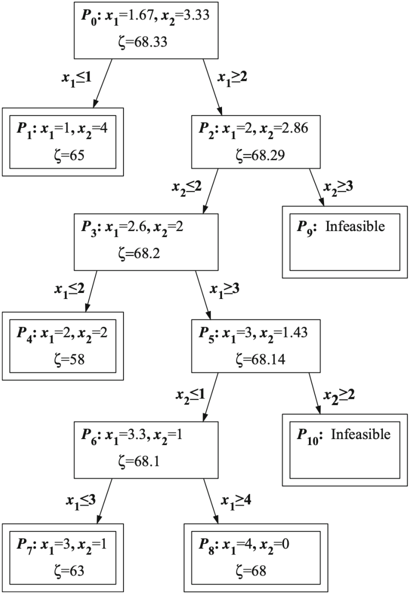
Continuing Our Example

After exploring P_6, P_7, P_8:





Final Search Tree



Branch & Bound Algorithm

We maintain a stack (or (priority) queue) Q of unexplored search nodes, and a best current solution B with value v_B and assume maximization (minimization is analogous).

Branch & Bound Algorithm

We maintain a stack (or (priority) queue) Q of unexplored search nodes, and a best current solution B with value v_B and assume maximization (minimization is analogous).

- Initialize Q with P_0, the original problem.

Branch & Bound Algorithm

We maintain a stack (or (priority) queue) Q of unexplored search nodes, and a best current solution B with value v_B and assume maximization (minimization is analogous).

- Initialize Q with P_0, the original problem.
- Initialize B, v_B with the best known solution (or set $v_B = -\infty, B = \perp$).

BRANCH & BOUND ALGORITHM

We maintain a stack (or (priority) queue) Q of unexplored search nodes, and a best current solution B with value v_B and assume maximization (minimization is analogous).

- Initialize Q with P_0, the original problem.
- Initialize B, v_B with the best known solution (or set $v_B = -\infty, B = \perp$).
- While Q is non-empty:

Branch & Bound Algorithm

We maintain a stack (or (priority) queue) Q of unexplored search nodes, and a best current solution B with value v_B and assume maximization (minimization is analogous).

- Initialize Q with P_0, the original problem.
- Initialize B, v_B with the best known solution (or set $v_B = -\infty, B = \perp$).
- While Q is non-empty:
  - Take the next P_i out of Q.

BRANCH & BOUND ALGORITHM

We maintain a stack (or (priority) queue) Q of unexplored search nodes, and a best current solution B with value v_B and assume maximization (minimization is analogous).

- Initialize Q with P_0, the original problem.
- Initialize B, v_B with the best known solution (or set $v_B = -\infty, B = \perp$).
- While Q is non-empty:
  - Take the next P_i out of Q.
  - Compute the optimal solution x^i with value ζ^i for the LP relaxation of P_i.

Branch & Bound Algorithm

We maintain a stack (or (priority) queue) Q of unexplored search nodes, and a best current solution B with value v_B and assume maximization (minimization is analogous).

- Initialize Q with P_0, the original problem.
- Initialize B, v_B with the best known solution (or set $v_B = -\infty, B = \perp$).
- While Q is non-empty:
  - Take the next P_i out of Q.
  - Compute the optimal solution x^i with value ζ^i for the LP relaxation of P_i.
  - If P_i is infeasible or $\zeta^i \leq v_B$, continue with next P_i.

Branch & Bound Algorithm

We maintain a stack (or (priority) queue) Q of unexplored search nodes, and a best current solution B with value v_B and assume maximization (minimization is analogous).

- Initialize Q with P_0, the original problem.
- Initialize B, v_B with the best known solution (or set $v_B = -\infty, B = \perp$).
- While Q is non-empty:
  - Take the next P_i out of Q.
  - Compute the optimal solution x^i with value ζ^i for the LP relaxation of P_i.
  - If P_i is infeasible or $\zeta^i \leq v_B$, continue with next P_i.
  - If x^i is integral, update $B = x^i, v_B = \zeta^i$, and continue with next P_i.

Branch & Bound Algorithm

We maintain a stack (or (priority) queue) Q of unexplored search nodes, and a best current solution B with value v_B and assume maximization (minimization is analogous).

- Initialize Q with P_0, the original problem.
- Initialize B, v_B with the best known solution (or set $v_B = -\infty, B = \perp$).
- While Q is non-empty:
  - Take the next P_i out of Q.
  - Compute the optimal solution x^i with value ζ^i for the LP relaxation of P_i.
  - If P_i is infeasible or $\zeta^i \leq v_B$, continue with next P_i.
  - If x^i is integral, update $B = x^i, v_B = \zeta^i$, and continue with next P_i.
  - Select non-integral variable x with value θ from x^i.

BRANCH & BOUND ALGORITHM

We maintain a stack (or (priority) queue) Q of unexplored search nodes, and a best current solution B with value v_B and assume maximization (minimization is analogous).

- Initialize Q with P_0, the original problem.
- Initialize B, v_B with the best known solution (or set $v_B = -\infty, B = \bot$).
- While Q is non-empty:
  - Take the next P_i out of Q.
  - Compute the optimal solution x^i with value ζ^i for the LP relaxation of P_i.
  - If P_i is infeasible or $\zeta^i \leq v_B$, continue with next P_i.
  - If x^i is integral, update $B = x^i, v_B = \zeta^i$, and continue with next P_i.
  - Select non-integral variable x with value θ from x^i.
  - Add $P_i \cup \{x \leq \lfloor\theta\rfloor\}$ and $P_i \cup \{x \geq \lceil\theta\rceil\}$ to Q.

Branch & Bound Algorithm

We maintain a stack (or (priority) queue) Q of unexplored search nodes, and a best current solution B with value v_B and assume maximization (minimization is analogous).

- Initialize Q with P_0, the original problem.
- Initialize B, v_B with the best known solution (or set $v_B = -\infty, B = \perp$).
- While Q is non-empty:
  - Take the next P_i out of Q.
  - Compute the optimal solution x^i with value ζ^i for the LP relaxation of P_i.
  - If P_i is infeasible or $\zeta^i \leq v_B$, continue with next P_i.
  - If x^i is integral, update $B = x^i, v_B = \zeta^i$, and continue with next P_i.
  - Select non-integral variable x with value θ from x^i.
  - Add $P_i \cup \{x \leq \lfloor\theta\rfloor\}$ and $P_i \cup \{x \geq \lceil\theta\rceil\}$ to Q.
- If $B = \perp$, report infeasibility. Otherwise, return optimal solution B.

Cutting Planes

There are several ways to extend Branch & Bound, usually with the goal of making it faster, at least for many interesting and practically relevant NP-hard problems.

CUTTING PLANES

There are several ways to extend Branch & Bound, usually with the goal of making it faster, at least for many interesting and practically relevant NP-hard problems.
One extremely important extension, implemented by all serious (M)IP solvers, is the addition of *cutting planes*.

Cutting Planes

There are several ways to extend Branch & Bound, usually with the goal of making it faster, at least for many interesting and practically relevant NP-hard problems.
One extremely important extension, implemented by all serious (M)IP solvers, is the addition of *cutting planes*.

The idea is to analyze the solutions of linear relaxations, and to dynamically identify certain types of linear constraints that

Cutting Planes

There are several ways to extend Branch & Bound, usually with the goal of making it faster, at least for many interesting and practically relevant NP-hard problems.
One extremely important extension, implemented by all serious (M)IP solvers, is the addition of *cutting planes*.

The idea is to analyze the solutions of linear relaxations, and to dynamically identify certain types of linear constraints that

- are satisfied by all integral solutions, but

CUTTING PLANES

There are several ways to extend Branch & Bound, usually with the goal of making it faster, at least for many interesting and practically relevant NP-hard problems.
One extremely important extension, implemented by all serious (M)IP solvers, is the addition of *cutting planes*.

The idea is to analyze the solutions of linear relaxations, and to dynamically identify certain types of linear constraints that

- are satisfied by all integral solutions, but
- are not satisfied by the solution to the current linear relaxation.

Cutting Planes

There are several ways to extend Branch & Bound, usually with the goal of making it faster, at least for many interesting and practically relevant NP-hard problems.
One extremely important extension, implemented by all serious (M)IP solvers, is the addition of *cutting planes*.

The idea is to analyze the solutions of linear relaxations, and to dynamically identify certain types of linear constraints that

- are satisfied by all integral solutions, but
- are not satisfied by the solution to the current linear relaxation.

Such inequalities can be dynamically added to and removed from the problem (without changing the set of integral solutions). They are called *cutting planes* or simply *cuts*. They can often drastically improve the quality of the bounds given by linear relaxations, help prune nodes of the search tree and identify integral solutions earlier.

Cutting Planes

There are several ways to extend Branch & Bound, usually with the goal of making it faster, at least for many interesting and practically relevant NP-hard problems.
One extremely important extension, implemented by all serious (M)IP solvers, is the addition of *cutting planes*.

The idea is to analyze the solutions of linear relaxations, and to dynamically identify certain types of linear constraints that

- are satisfied by all integral solutions, but
- are not satisfied by the solution to the current linear relaxation.

Such inequalities can be dynamically added to and removed from the problem (without changing the set of integral solutions). They are called *cutting planes* or simply *cuts*. They can often drastically improve the quality of the bounds given by linear relaxations, help prune nodes of the search tree and identify integral solutions earlier.

Cuts are usually found by heuristic procedures. Modern solvers already contain a set of such procedures that have proven useful for many practical problems. Implementing such procedures efficiently and balancing the additional effort put into finding cuts against the runtime benefits they provide is an important part of engineering a good solver.

Cutting Planes

There are several ways to extend Branch & Bound, usually with the goal of making it faster, at least for many interesting and practically relevant NP-hard problems.
One extremely important extension, implemented by all serious (M)IP solvers, is the addition of *cutting planes*.

The idea is to analyze the solutions of linear relaxations, and to dynamically identify certain types of linear constraints that

- are satisfied by all integral solutions, but
- are not satisfied by the solution to the current linear relaxation.

Such inequalities can be dynamically added to and removed from the problem (without changing the set of integral solutions). They are called *cutting planes* or simply *cuts*. They can often drastically improve the quality of the bounds given by linear relaxations, help prune nodes of the search tree and identify integral solutions earlier.

Cuts are usually found by heuristic procedures. Modern solvers already contain a set of such procedures that have proven useful for many practical problems. Implementing such procedures efficiently and balancing the additional effort put into finding cuts against the runtime benefits they provide is an important part of engineering a good solver.

Furthermore, many problems allow the implementation of problem-specific cuts that are not part of general-purpose solvers. These often require additional knowledge about the problem or are too expensive or too specialized to be included in general-purpose solvers.

Gomory Cuts

A very important family of cuts are the so-called *Gomory cuts*.
Consider an (optimal) basic solution to a linear relaxation. In dictionary form, we have m equations of the form (which are valid constraints)

$$x_i = x_i^* - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j \Leftrightarrow x_i^* = x_i + \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j$$

GOMORY CUTS

A very important family of cuts are the so-called *Gomory cuts*.
Consider an (optimal) basic solution to a linear relaxation. In dictionary form, we have m equations of the form (which are valid constraints)

$$x_i = x_i^* - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j \Leftrightarrow x_i^* = x_i + \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j$$

Consider the case where all x_j with non-zero coefficients are integer variables. Is that case rare?

Gomory Cuts

A very important family of cuts are the so-called *Gomory cuts*.
Consider an (optimal) basic solution to a linear relaxation. In dictionary form, we have m equations of the form (which are valid constraints)

$$x_i = x_i^* - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j \Leftrightarrow x_i^* = x_i + \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j$$

Consider the case where all x_j with non-zero coefficients are integer variables. Is that case rare?
No! Many slack variables are integral, e.g., if all coefficients in their constraint are integral.

Gomory Cuts

A very important family of cuts are the so-called *Gomory cuts*.
Consider an (optimal) basic solution to a linear relaxation. In dictionary form, we have m equations of the form (which are valid constraints)

$$x_i = x_i^* - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j \Leftrightarrow x_i^* = x_i + \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j$$

Consider the case where all x_j with non-zero coefficients are integer variables. Is that case rare?
No! Many slack variables are integral, e.g., if all coefficients in their constraint are integral.
Split into integral and fractional part:

$$\lfloor x_i^* \rfloor + (x_i^* - \lfloor x_i^* \rfloor) = x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j + \sum_{j \in \mathcal{N}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j$$

GOMORY CUTS

A very important family of cuts are the so-called *Gomory cuts*.
Consider an (optimal) basic solution to a linear relaxation. In dictionary form, we have m equations of the form (which are valid constraints)

$$x_i = x_i^* - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j \Leftrightarrow x_i^* = x_i + \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j$$

Consider the case where all x_j with non-zero coefficients are integer variables. Is that case rare?
No! Many slack variables are integral, e.g., if all coefficients in their constraint are integral.
Split into integral and fractional part:

$$\lfloor x_i^* \rfloor + (x_i^* - \lfloor x_i^* \rfloor) = x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j + \sum_{j \in \mathcal{N}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j$$

Separate integral (left-hand side) and fractional (right-hand side):

$$\underbrace{x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j - \lfloor x_i^* \rfloor}_{\in \mathbb{Z}} = \underbrace{(x_i^* - \lfloor x_i^* \rfloor)}_{<1} - \underbrace{\sum_{j \in \mathcal{N}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j}_{\geq 0 \text{ for } x \geq 0}$$

Gomory Cuts

A very important family of cuts are the so-called *Gomory cuts*.
Consider an (optimal) basic solution to a linear relaxation. In dictionary form, we have m equations of the form (which are valid constraints)

$$x_i = x_i^* - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j \Leftrightarrow x_i^* = x_i + \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j$$

Consider the case where all x_j with non-zero coefficients are integer variables. Is that case rare?
No! Many slack variables are integral, e.g., if all coefficients in their constraint are integral.
Split into integral and fractional part:

$$\lfloor x_i^* \rfloor + (x_i^* - \lfloor x_i^* \rfloor) = x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j + \sum_{j \in \mathcal{N}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j$$

Separate integral (left-hand side) and fractional (right-hand side):

$$\underbrace{x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j - \lfloor x_i^* \rfloor}_{\in \mathbb{Z}} = \underbrace{(x_i^* - \lfloor x_i^* \rfloor)}_{<1} - \underbrace{\sum_{j \in \mathcal{N}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j}_{\geq 0 \text{ for } x \geq 0}$$

Therefore, $x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j - \lfloor x_i^* \rfloor \leq 0 \Leftrightarrow x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j \leq \lfloor x_i^* \rfloor$ holds for all integer solutions.

GOMORY CUTS

A very important family of cuts are the so-called *Gomory cuts*.
Consider an (optimal) basic solution to a linear relaxation. In dictionary form, we have m equations of the form (which are valid constraints)

$$x_i = x_i^* - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j \Leftrightarrow x_i^* = x_i + \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j$$

Consider the case where all x_j with non-zero coefficients are integer variables. Is that case rare?
No! Many slack variables are integral, e.g., if all coefficients in their constraint are integral.
Split into integral and fractional part:

$$\lfloor x_i^* \rfloor + (x_i^* - \lfloor x_i^* \rfloor) = x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j + \sum_{j \in \mathcal{N}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j$$

Separate integral (left-hand side) and fractional (right-hand side):

$$\underbrace{x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j - \lfloor x_i^* \rfloor}_{\in \mathbb{Z}} = \underbrace{(x_i^* - \lfloor x_i^* \rfloor)}_{<1} - \underbrace{\sum_{j \in \mathcal{N}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j}_{\geq 0 \text{ for } x \geq 0}$$

Therefore, $x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j - \lfloor x_i^* \rfloor \leq 0 \Leftrightarrow x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j \leq \lfloor x_i^* \rfloor$ holds for all integer solutions.
This constraint is always violated in the current basic solution if $x_i^* \notin \mathbb{Z}$. Why?

Gomory Cut Example

With a given optimal dictionary, equivalent cuts (to the general scheme introduced before) can be found like in the following example.

$$
\begin{array}{rrrr}
\zeta = & \frac{179}{3} - & \frac{7}{27}w_1 - & \frac{73}{54}w_2 \\
\hline
x_1 = & \frac{11}{3} - & \frac{5}{54}w_1 - & \frac{1}{54}w_2 \\
x_2 = & \frac{7}{3} + & \frac{1}{27}w_1 + & \frac{5}{54}w_2 \\
w_3 = & 13 - & \frac{5}{9}w_1 - & \frac{8}{9}w_2
\end{array}
$$

Gomory Cut Example

With a given optimal dictionary, equivalent cuts (to the general scheme introduced before) can be found like in the following example.

$$
\begin{array}{rrrr}
\zeta = & \frac{179}{3} - & \frac{7}{27}w_1 - & \frac{73}{54}w_2 \\
\hline
x_1 = & \frac{11}{3} - & \frac{5}{54}w_1 - & \frac{1}{54}w_2 \\
x_2 = & \frac{7}{3} + & \frac{1}{27}w_1 + & \frac{5}{54}w_2 \\
w_3 = & 13 - & \frac{5}{9}w_1 - & \frac{8}{9}w_2
\end{array}
$$

x_1 is not integral. Reorganize equation so all variables are on one side:

$$x_1 + \frac{5}{54}w_1 + \frac{1}{54}w_2 = \frac{11}{3}.$$

Gomory Cut Example

With a given optimal dictionary, equivalent cuts (to the general scheme introduced before) can be found like in the following example.

$$\begin{array}{rcccc}
\zeta = & \frac{179}{3} - & \frac{7}{27}w_1 - & \frac{73}{54}w_2 \\
\hline
x_1 = & \frac{11}{3} - & \frac{5}{54}w_1 - & \frac{1}{54}w_2 \\
x_2 = & \frac{7}{3} + & \frac{1}{27}w_1 + & \frac{5}{54}w_2 \\
w_3 = & 13 - & \frac{5}{9}w_1 - & \frac{8}{9}w_2
\end{array}$$

x_1 is not integral. Reorganize equation so all variables are on one side:

$$x_1 + \frac{5}{54}w_1 + \frac{1}{54}w_2 = \frac{11}{3}.$$

Rounding the left-hand side coefficients makes the left-hand side smaller and integral:

$$x_1 + 0w_1 + 0w_2 \leq \lfloor 11/3 \rfloor = 3 \Rightarrow x_1 \leq 3.$$

Gomory Cut Example Continued

$$
\begin{array}{rrrr}
\zeta = & \frac{179}{3} - & \frac{7}{27} w_1 - & \frac{73}{54} w_2 \\
\hline
x_1 = & \frac{11}{3} - & \frac{5}{54} w_1 - & \frac{1}{54} w_2 \\
x_2 = & \frac{7}{3} + & \frac{1}{27} w_1 + & \frac{5}{54} w_2 \\
w_3 = & 13 - & \frac{5}{9} w_1 - & \frac{8}{9} w_2
\end{array}
$$

Gomory Cut Example Continued

$$
\begin{array}{rcrcrcr}
\zeta & = & \frac{179}{3} & - & \frac{7}{27}w_1 & - & \frac{73}{54}w_2 \\
\hline
x_1 & = & \frac{11}{3} & - & \frac{5}{54}w_1 & - & \frac{1}{54}w_2 \\
x_2 & = & \frac{7}{3} & + & \frac{1}{27}w_1 & + & \frac{5}{54}w_2 \\
w_3 & = & 13 & - & \frac{5}{9}w_1 & - & \frac{8}{9}w_2
\end{array}
$$

Adding $x_1 \leq 3$ adds a (basic, integral!) slack variable $w_4 = 3 - x_1 = 3 - \frac{11}{3} + \frac{5}{54}w_1 + \frac{1}{54}w_2$:

Gomory Cut Example Continued

$$
\begin{array}{rcrcrcr}
\zeta & = & \frac{179}{3} & - & \frac{7}{27} w_1 & - & \frac{73}{54} w_2 \\
\hline
x_1 & = & \frac{11}{3} & - & \frac{5}{54} w_1 & - & \frac{1}{54} w_2 \\
x_2 & = & \frac{7}{3} & + & \frac{1}{27} w_1 & + & \frac{5}{54} w_2 \\
w_3 & = & 13 & - & \frac{5}{9} w_1 & - & \frac{8}{9} w_2
\end{array}
$$

Adding $x_1 \leq 3$ adds a (basic, integral!) slack variable $w_4 = 3 - x_1 = 3 - \frac{11}{3} + \frac{5}{54} w_1 + \frac{1}{54} w_2$:

$$
\begin{array}{rcrcrcr}
\zeta & = & \frac{179}{3} & - & \frac{7}{27} w_1 & - & \frac{73}{54} w_2 \\
\hline
x_1 & = & \frac{11}{3} & - & \frac{5}{54} w_1 & - & \frac{1}{54} w_2 \\
x_2 & = & \frac{7}{3} & + & \frac{1}{27} w_1 & + & \frac{5}{54} w_2 \\
w_3 & = & 13 & - & \frac{5}{9} w_1 & - & \frac{8}{9} w_2 \\
w_4 & = & -\frac{2}{3} & + & \frac{5}{54} w_1 & + & \frac{1}{54} w_2
\end{array}
$$

Gomory Cut Example Continued

$$\begin{array}{rrrr}
\zeta = & \frac{179}{3} - & \frac{7}{27}w_1 - & \frac{73}{54}w_2 \\
\hline
x_1 = & \frac{11}{3} - & \frac{5}{54}w_1 - & \frac{1}{54}w_2 \\
x_2 = & \frac{7}{3} + & \frac{1}{27}w_1 + & \frac{5}{54}w_2 \\
w_3 = & 13 - & \frac{5}{9}w_1 - & \frac{8}{9}w_2
\end{array}$$

Adding $x_1 \leq 3$ adds a (basic, integral!) slack variable $w_4 = 3 - x_1 = 3 - \frac{11}{3} + \frac{5}{54}w_1 + \frac{1}{54}w_2$:

$$\begin{array}{rrrr}
\zeta = & \frac{179}{3} - & \frac{7}{27}w_1 - & \frac{73}{54}w_2 \\
\hline
x_1 = & \frac{11}{3} - & \frac{5}{54}w_1 - & \frac{1}{54}w_2 \\
x_2 = & \frac{7}{3} + & \frac{1}{27}w_1 + & \frac{5}{54}w_2 \\
w_3 = & 13 - & \frac{5}{9}w_1 - & \frac{8}{9}w_2 \\
w_4 = & -\frac{2}{3} + & \frac{5}{54}w_1 + & \frac{1}{54}w_2
\end{array}$$

We can continue with dual Simplex.

Gomory Cut Example Continued

After one dual Simplex pivot:

$$
\begin{array}{rcrcrcr}
\zeta & = & \frac{179}{3} & - & \frac{7}{27}w_4 & - & \frac{73}{54}w_2 \\
\hline
x_1 & = & 3 & - & w_4 & & \\
x_2 & = & \frac{13}{5} & + & \frac{2}{5}w_4 & - & \frac{1}{10}w_2 \\
w_3 & = & 9 & - & 6w_4 & + & w_2 \\
w_1 & = & \frac{36}{5} & + & \frac{54}{5}w_4 & - & \frac{1}{5}w_2
\end{array}
$$

Gomory Cut Example Continued

After one dual Simplex pivot:

$$
\begin{array}{rrrr}
\zeta = & \frac{179}{3} - & \frac{7}{27}w_4 - & \frac{73}{54}w_2 \\
\hline
x_1 = & 3 - & w_4 & \\
x_2 = & \frac{13}{5} + & \frac{2}{5}w_4 - & \frac{1}{10}w_2 \\
w_3 = & 9 - & 6w_4 + & w_2 \\
w_1 = & \frac{36}{5} + & \frac{54}{5}w_4 - & \frac{1}{5}w_2
\end{array}
$$

Gomory cut on $x_2 - \frac{2}{5}w_4 + \frac{1}{10}w_2 = \frac{13}{5}$: $\quad x_2 - w_4 \leq 2.$

Gomory Cut Example Continued

After one dual Simplex pivot:

$$
\begin{array}{rcccc}
\zeta = & \frac{179}{3} - & \frac{7}{27} w_4 - & \frac{73}{54} w_2 \\
\hline
x_1 = & 3 - & w_4 & \\
x_2 = & \frac{13}{5} + & \frac{2}{5} w_4 - & \frac{1}{10} w_2 \\
w_3 = & 9 - & 6 w_4 + & w_2 \\
w_1 = & \frac{36}{5} + & \frac{54}{5} w_4 - & \frac{1}{5} w_2
\end{array}
$$

Gomory cut on $x_2 - \frac{2}{5} w_4 + \frac{1}{10} w_2 = \frac{13}{5}$: $\quad x_2 - w_4 \leq 2.$

$$
\begin{array}{rcccc}
\zeta = & \frac{179}{3} - & \frac{7}{27} w_4 - & \frac{73}{54} w_2 \\
\hline
x_1 = & 3 - & w_4 & \\
x_2 = & \frac{13}{5} + & \frac{2}{5} w_4 - & \frac{1}{10} w_2 \\
w_3 = & 9 - & 6 w_4 + & w_2 \\
w_1 = & \frac{36}{5} + & \frac{54}{5} w_4 - & \frac{1}{5} w_2 \\
w_5 = & -\frac{3}{5} + & \frac{3}{5} w_4 + & \frac{1}{10} w_2
\end{array}
$$

GOMORY CUT EXAMPLE CONTINUED

After one final dual Simplex pivot:

$$
\begin{array}{rrrr}
\zeta = & \frac{179}{3} - & \frac{7}{27} w_5 - & \frac{73}{54} w_2 \\
\hline
x_1 = & 2 - & \frac{5}{3} w_5 + & \frac{1}{6} w_2 \\
x_2 = & 3 + & \frac{2}{3} w_5 - & \frac{1}{6} w_2 \\
w_3 = & 3 - & 10 w_5 + & 2 w_2 \\
w_1 = & 18 + & 18 w_5 - & 2 w_2 \\
w_4 = & 1 + & \frac{5}{3} w_5 - & \frac{1}{6} w_2
\end{array}
$$

Gomory Cut Example Continued

After one final dual Simplex pivot:

$$
\begin{array}{rrrr}
\zeta = & \frac{179}{3} - & \frac{7}{27}w_5 - & \frac{73}{54}w_2 \\
\hline
x_1 = & 2 - & \frac{5}{3}w_5 + & \frac{1}{6}w_2 \\
x_2 = & 3 + & \frac{2}{3}w_5 - & \frac{1}{6}w_2 \\
w_3 = & 3 - & 10w_5 + & 2w_2 \\
w_1 = & 18 + & 18w_5 - & 2w_2 \\
w_4 = & 1 + & \frac{5}{3}w_5 - & \frac{1}{6}w_2
\end{array}
$$

We found the optimal integral solution without branching!

Gomory Cut Example Continued

After one final dual Simplex pivot:

$$
\begin{array}{rrrr}
\zeta = & \frac{179}{3} - & \frac{7}{27} w_5 - & \frac{73}{54} w_2 \\
\hline
x_1 = & 2 - & \frac{5}{3} w_5 + & \frac{1}{6} w_2 \\
x_2 = & 3 + & \frac{2}{3} w_5 - & \frac{1}{6} w_2 \\
w_3 = & 3 - & 10 w_5 + & 2 w_2 \\
w_1 = & 18 + & 18 w_5 - & 2 w_2 \\
w_4 = & 1 + & \frac{5}{3} w_5 - & \frac{1}{6} w_2
\end{array}
$$

We found the optimal integral solution without branching!

In theory, we can always solve integer programs like this only by adding cutting planes. However, for numerical and efficiency reasons, this is not really practical.

Gomory Cut Example Continued

After one final dual Simplex pivot:

$$
\begin{array}{rrrr}
\zeta = & \dfrac{179}{3} - & \dfrac{7}{27}w_5 - & \dfrac{73}{54}w_2 \\
\hline
x_1 = & 2 - & \dfrac{5}{3}w_5 + & \dfrac{1}{6}w_2 \\
x_2 = & 3 + & \dfrac{2}{3}w_5 - & \dfrac{1}{6}w_2 \\
w_3 = & 3 - & 10w_5 + & 2w_2 \\
w_1 = & 18 + & 18w_5 - & 2w_2 \\
w_4 = & 1 + & \dfrac{5}{3}w_5 - & \dfrac{1}{6}w_2
\end{array}
$$

We found the optimal integral solution without branching!

In theory, we can always solve integer programs like this only by adding cutting planes. However, for numerical and efficiency reasons, this is not really practical.

Instead, cutting planes are incorporated into a Branch & Bound solver by adding a limited number of cutting planes after solving a linear relaxation when it seems beneficial. Algorithms that follow this paradigm are called *Branch & Cut* algorithms and are the basis of modern MIP solvers.