Branch-and-Bound

Math 482, Lecture 33

Misha Lavrov

April 27, 2020

Branch-and-bound

The branch-and-bound method is a general strategy for optimization problems.

Branch-and-bound

The branch-and-bound method is a general strategy for optimization problems.

- We **branch** by casework, dividing a problem into several subproblems, and then dividing those subproblems into further subproblems, until they're easy to solve.

Branch-and-bound

The branch-and-bound method is a general strategy for optimization problems.

- We **branch** by casework, dividing a problem into several subproblems, and then dividing those subproblems into further subproblems, until they're easy to solve.
- When a subproblem is too hard to solve directly, we at least put a **bound** on its objective value to let us eliminate branches without having to look at all of them.

Branch-and-bound

The branch-and-bound method is a general strategy for optimization problems.

- We **branch** by casework, dividing a problem into several subproblems, and then dividing those subproblems into further subproblems, until they're easy to solve.
- When a subproblem is too hard to solve directly, we at least put a **bound** on its objective value to let us eliminate branches without having to look at all of them.

For example: if subproblem A definitely achieves an objective value of 100 (and we're maximizing), and subproblem B's objective value is at most 80, we can **prune** subproblem B without breaking it down into further cases.

Branch-and-bound for integer programming

Here is an overview of how we can apply this to integer programs.

Branch-and-bound for integer programming

Here is an overview of how we can apply this to integer programs.

- We can bound the value of an integer program by solving its **linear relaxation**: the LP where we forget about the integer constraints.

Branch-and-bound for integer programming

Here is an overview of how we can apply this to integer programs.

- We can bound the value of an integer program by solving its **linear relaxation**: the LP where we forget about the integer constraints.
- A subproblem is “easy” if the linear relaxation happens to have an integer solution. Otherwise, we will need to branch on it.

Branch-and-bound for integer programming

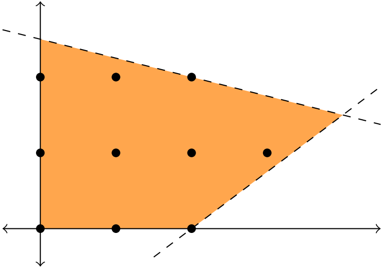
Here is an overview of how we can apply this to integer programs.

- We can bound the value of an integer program by solving its **linear relaxation**: the LP where we forget about the integer constraints.
- A subproblem is "easy" if the linear relaxation happens to have an integer solution. Otherwise, we will need to branch on it.
- To branch on a fractional solution where $x_i = f \notin \mathbb{Z}$, take the following two subproblems:
  - one where we add the constraint $x_i \le \lfloor f \rfloor$, and
  - one where we add the constraint $x_i \ge \lceil f \rceil$.

Branch-and-bound example

We will use branch and bound to solve the following linear program:

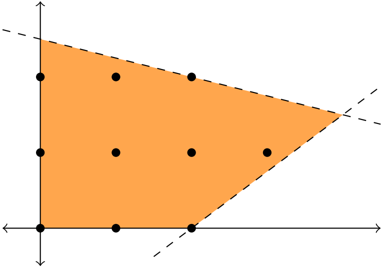
$$\begin{aligned}
\underset{x,y\in\mathbb{Z}}{\text{maximize}} \quad & 4x + 5y \\
\text{subject to} \quad & x + 4y \leq 10 \\
& 3x - 4y \leq 6 \\
& x, y \geq 0
\end{aligned}$$



Branch-and-bound example

We will use branch and bound to solve the following linear program:

$$\begin{aligned} \underset{x,y\in\mathbb{Z}}{\text{maximize}} \quad & 4x + 5y \\ \text{subject to} \quad & x + 4y \le 10 \\ & 3x - 4y \le 6 \\ & x, y \ge 0 \end{aligned}$$



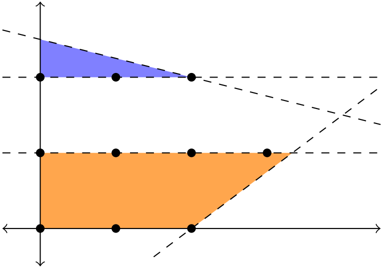
Step 1: solve the LP relaxation. This has optimal solution $(x, y) = (4, 1.5)$ with $4x + 5y = 23.5$.

The branch step, geometrically

Since the optimal solution has $y = 1.5 \notin Z$, we can consider two cases that both eliminate this point: $y \leq 1$, or $y \geq 2$.

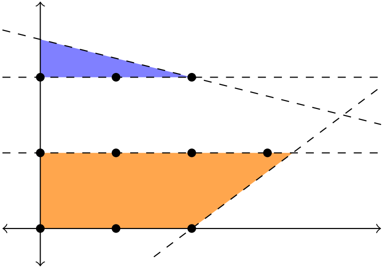
The branch step, geometrically

Since the optimal solution has $y = 1.5 \notin Z$, we can consider two cases that both eliminate this point: $y \leq 1$, or $y \geq 2$.



The branch step, geometrically

Since the optimal solution has $y = 1.5 \notin Z$, we can consider two cases that both eliminate this point: $y \leq 1$, or $y \geq 2$.



(Note: we must get rid of the point $(4, 1.5)$ in future cases we consider, or we'll just get it back as the optimal solution again!)

The branch step in the simplex tableau

We already know how to use the simplex method. But it's important to note that we don't have to solve the new LPs from scratch.

The branch step in the simplex tableau

We already know how to use the simplex method. But it's important to note that we don't have to solve the new LPs from scratch.

The general method:

1. Take the optimal simplex tableau for the previous subproblem.

The branch step in the simplex tableau

We already know how to use the simplex method. But it's important to note that we don't have to solve the new LPs from scratch.

The general method:

1. Take the optimal simplex tableau for the previous subproblem.
2. Add a new row (and slack variable) for the new constraint we add.

The branch step in the simplex tableau

We already know how to use the simplex method. But it's important to note that we don't have to solve the new LPs from scratch.

The general method:

1. Take the optimal simplex tableau for the previous subproblem.
2. Add a new row (and slack variable) for the new constraint we add.
3. Row-reduce the resulting tableau.

The branch step in the simplex tableau

We already know how to use the simplex method. But it's important to note that we don't have to solve the new LPs from scratch.

The general method:

1. Take the optimal simplex tableau for the previous subproblem.
2. Add a new row (and slack variable) for the new constraint we add.
3. Row-reduce the resulting tableau.
4. Solve with the dual simplex method.

The branch step: an example

Here's how we do this to add a $y \geq 2$ constraint to the LP that gave us $(x, y) = (4, 1.5)$.

The branch step: an example

Here's how we do this to add a $y \geq 2$ constraint to the LP that gave us $(x, y) = (4, 1.5)$.

Step 1: take the optimal tableau

	x	y	s_1	s_2	
y	0	1	$3/16$	$-1/16$	$3/2$
x	1	0	$1/4$	$1/4$	4
$-z$	0	0	$-31/16$	$-11/16$	$-47/2$

The branch step: an example

Here's how we do this to add a $y \geq 2$ constraint to the LP that gave us $(x, y) = (4, 1.5)$.

Step 2: Add a new row for "$-y + s_3 = -2$"

	x	y	s_1	s_2	s_3	
y	0	1	$3/16$	$-1/16$	0	$3/2$
x	1	0	$1/4$	$1/4$	0	4
s_3	0	-1	0	0	1	-2
$-z$	0	0	$-31/16$	$-11/16$	0	$-47/2$

The branch step: an example

Here's how we do this to add a $y \geq 2$ constraint to the LP that gave us $(x, y) = (4, 1.5)$.

Step 3: Row-reduce this tableau

	x	y	s_1	s_2	s_3	
y	0	1	$^3/_{16}$	$-^1/_{16}$	0	$^3/_2$
x	1	0	$^1/_4$	$^1/_4$	0	4
s_3	0	0	$^3/_{16}$	$-^1/_{16}$	1	$-^1/_2$
$-z$	0	0	$-^{31}/_{16}$	$-^{11}/_{16}$	0	$-^{47}/_2$

The branch step: an example

Here's how we do this to add a $y \geq 2$ constraint to the LP that gave us $(x, y) = (4, 1.5)$.

Step 4: Solve using the dual simplex method

	x	y	s_1	s_2	s_3	
y	0	1	0	0	-1	2
x	1	0	1	0	4	2
s_2	0	0	-3	1	-16	8
$-z$	0	0	-4	0	-11	-18

Solving the first two subproblems

What we get when we branch on $y \leq 1$ versus $y \geq 2$:

$$(x, y) = (4, 1.5)$$
$$z = 23.5$$

$y \geq 2$:

$$(x, y) = (2, 2)$$
$$z = 18$$

$y \leq 1$:

$$(x, y) = (3.\overline{3}, 1)$$
$$z = 18.\overline{3}$$

Solving the first two subproblems

What we get when we branch on $y \leq 1$ versus $y \geq 2$:

$$\begin{array}{c} (x, y) = (4, 1.5) \\ z = 23.5 \end{array}$$

$$y \geq 2 \qquad\qquad y \leq 1$$

$$\begin{array}{c} (x, y) = (2, 2) \\ z = 18 \end{array} \qquad\qquad \begin{array}{c} (x, y) = (3.\overline{3}, 1) \\ z = 18.\overline{3} \end{array}$$

- The left node is an integer solution, giving us a **lower bound** of 18.

Solving the first two subproblems

What we get when we branch on $y \leq 1$ versus $y \geq 2$:

$$\begin{tikzcd}
& \boxed{\begin{array}{c}(x, y) = (4, 1.5)\\ z = 23.5\end{array}} \arrow[dl, "y \geq 2"'] \arrow[dr, "y \leq 1"] & \\
\boxed{\begin{array}{c}(x, y) = (2, 2)\\ z = 18\end{array}} & & \boxed{\begin{array}{c}(x, y) = (3.\overline{3}, 1)\\ z = 18.\overline{3}\end{array}}
\end{tikzcd}$$

- The left node is an integer solution, giving us a **lower bound** of 18.
- The right node is a fractional solution with $z > 18$, so it's still worth exploring.

Solving the first two subproblems

What we get when we branch on $y \leq 1$ versus $y \geq 2$:

$$(x, y) = (4, 1.5) \qquad z = 23.5$$

$y \geq 2$:

$$(x, y) = (2, 2) \qquad z = 18$$

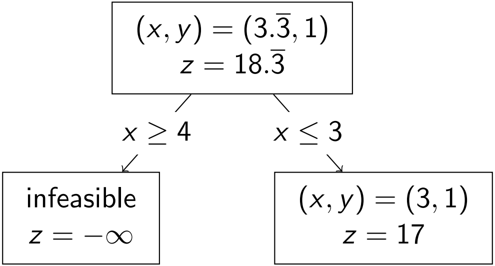
$y \leq 1$:

$$(x, y) = (3.\overline{3}, 1) \qquad z = 18.\overline{3}$$

- The left node is an integer solution, giving us a **lower bound** of 18.
- The right node is a fractional solution with $z > 18$, so it's still worth exploring.
- We can branch on x: add $x \leq 3$ or $x \geq 4$ as a constraint.

Solving the next two subproblems

What we get when we branch on $x \leq 3$ versus $x \geq 4$ (from the node where we already had $y \leq 1$ as an extra constraint):



Solving the next two subproblems

What we get when we branch on $x \leq 3$ versus $x \geq 4$ (from the node where we already had $y \leq 1$ as an extra constraint):

$$
\begin{tikzcd}
& \boxed{\begin{array}{c}(x,y) = (3.\overline{3}, 1) \\ z = 18.\overline{3}\end{array}} \arrow[dl, "x \geq 4"'] \arrow[dr, "x \leq 3"] & \\
\boxed{\begin{array}{c}\text{infeasible} \\ z = -\infty\end{array}} & & \boxed{\begin{array}{c}(x,y) = (3, 1) \\ z = 17\end{array}}
\end{tikzcd}
$$

- The left node is infeasible, so we ignore it completely.

Solving the next two subproblems

What we get when we branch on $x \leq 3$ versus $x \geq 4$ (from the node where we already had $y \leq 1$ as an extra constraint):

$(x, y) = (3.\overline{3}, 1)$
$z = 18.\overline{3}$

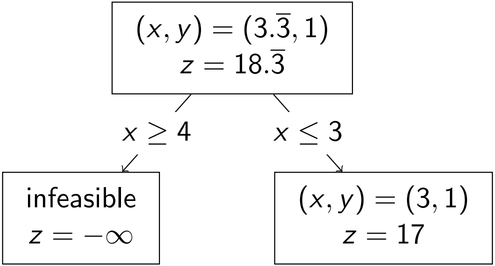
$x \geq 4$ → infeasible, $z = -\infty$

$x \leq 3$ → $(x, y) = (3, 1)$, $z = 17$

- The left node is infeasible, so we ignore it completely.
- The right node is another integer solution, but it has $z = 17 < 18$, so it's not as good as the first. (Even if it were a fractional solution, we wouldn't branch on it.)

Solving the next two subproblems

What we get when we branch on $x \le 3$ versus $x \ge 4$ (from the node where we already had $y \le 1$ as an extra constraint):



- The left node is infeasible, so we ignore it completely.
- The right node is another integer solution, but it has $z = 17 < 18$, so it's not as good as the first. (Even if it were a fractional solution, we wouldn't branch on it.)
- We have no more nodes worth exploring, so we're done.

A formal description

Formally, the branch-and-bound algorithm works as follows.

We maintain:

- A list $\mathcal{L}$ of "nodes": linear programs to solve.

A formal description

Formally, the branch-and-bound algorithm works as follows.

We maintain:

- A list $\mathcal{L}$ of "nodes": linear programs to solve.

  (Initially, $\mathcal{L}$ only contains one node: the LP relaxation of our original problem.)

A formal description

Formally, the branch-and-bound algorithm works as follows.

We maintain:

- A list $\mathcal{L}$ of "nodes": linear programs to solve.

  (Initially, $\mathcal{L}$ only contains one node: the LP relaxation of our original problem.)

- A point $\mathbf{x}^*$, the best integer solution found so far, and its objective value z^*.

A formal description

Formally, the branch-and-bound algorithm works as follows.

We maintain:

- A list $\mathcal{L}$ of "nodes": linear programs to solve.

  (Initially, $\mathcal{L}$ only contains one node: the LP relaxation of our original problem.)

- A point $\mathbf{x}^*$, the best integer solution found so far, and its objective value z^*.

  (Initially, there is no $\mathbf{x}^*$, and we set $z^* = -\infty$.)

A formal description

Formally, the branch-and-bound algorithm works as follows.

We maintain:

- A list $\mathcal{L}$ of "nodes": linear programs to solve.

  (Initially, $\mathcal{L}$ only contains one node: the LP relaxation of our original problem.)

- A point $\mathbf{x}^*$, the best integer solution found so far, and its objective value z^*.

  (Initially, there is no $\mathbf{x}^*$, and we set $z^* = -\infty$.)

At each step, we pick a node, remove it from $\mathcal{L}$, solve the LP, and do something based on the solution.

A formal description

Formally, the branch-and-bound algorithm works as follows.

We maintain:

- A list $\mathcal{L}$ of "nodes": linear programs to solve.

  (Initially, $\mathcal{L}$ only contains one node: the LP relaxation of our original problem.)

- A point $\mathbf{x}^*$, the best integer solution found so far, and its objective value z^*.

  (Initially, there is no $\mathbf{x}^*$, and we set $z^* = -\infty$.)

At each step, we pick a node, remove it from $\mathcal{L}$, solve the LP, and do something based on the solution. **Repeat until $\mathcal{L}$ is empty.**

Handling a new node

Suppose the node we look at has optimal solution $\mathbf{x}$ with objective value z. Then, **in order**:

Handling a new node

Suppose the node we look at has optimal solution $\mathbf{x}$ with objective value z. Then, **in order**:

1. If $z \leq z^*$, do nothing; the node is **pruned by bound**.

Handling a new node

Suppose the node we look at has optimal solution $\mathbf{x}$ with objective value z. Then, **in order**:

1. If $z \leq z^*$, do nothing; the node is **pruned by bound**.
2. If $z > z*$ and $\mathbf{x}$ is an integer solution, set $\mathbf{x}^* = \mathbf{x}$ and $z^* = z$; the node is **pruned by integrality**.

Handling a new node

Suppose the node we look at has optimal solution $\mathbf{x}$ with objective value z. Then, **in order**:

1. If $z \leq z^*$, do nothing; the node is **pruned by bound**.
2. If $z > z*$ and $\mathbf{x}$ is an integer solution, set $\mathbf{x}^* = \mathbf{x}$ and $z^* = z$; the node is **pruned by integrality**.
3. If $z > z^*$ but $x_i = f \notin \mathbb{Z}$ for some i, we **branch on $\boldsymbol{x_i}$**.

   Add new nodes to $\mathcal{L}$ based on this node: one where we add the constraint $x_i \leq \lfloor f \rfloor$, and one where we add $x_i \geq \lceil f \rceil$.

Handling a new node

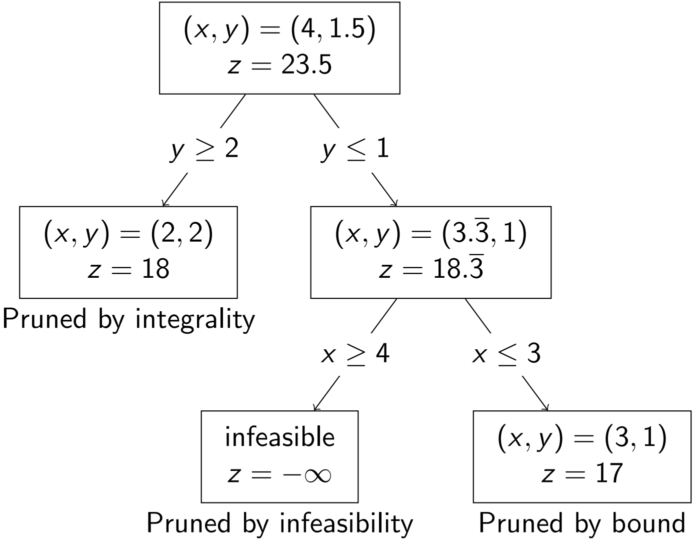
Suppose the node we look at has optimal solution $\mathbf{x}$ with objective value z. Then, **in order**:

1. If $z \leq z^*$, do nothing; the node is **pruned by bound**.
2. If $z > z*$ and $\mathbf{x}$ is an integer solution, set $\mathbf{x}^* = \mathbf{x}$ and $z^* = z$; the node is **pruned by integrality**.
3. If $z > z^*$ but $x_i = f \notin \mathbb{Z}$ for some i, we **branch on x_i**.

   Add new nodes to $\mathcal{L}$ based on this node: one where we add the constraint $x_i \leq \lfloor f \rfloor$, and one where we add $x_i \geq \lceil f \rceil$.

If the node we look at has no feasible solution, we also do nothing; the node is **pruned by infeasibility**.

Branching in our example



Further considerations

There are several places where we have some freedom to choose how to branch-and-bound.

Further considerations

There are several places where we have some freedom to choose how to branch-and-bound.

- **Which node from $\mathcal{L}$ do we look at first?**

Further considerations

There are several places where we have some freedom to choose how to branch-and-bound.

- **Which node from $\mathcal{L}$ do we look at first?**

  Nodes whose parent had a larger z are more promising. We might also want to try to get a few integer solutions as quickly as possible.

Further considerations

There are several places where we have some freedom to choose how to branch-and-bound.

- **Which node from $\mathcal{L}$ do we look at first?**

  Nodes whose parent had a larger z are more promising. We might also want to try to get a few integer solutions as quickly as possible.

- **Which fractional variable do we branch on, if we have a choice?**

Further considerations

There are several places where we have some freedom to choose how to branch-and-bound.

- **Which node from $\mathcal{L}$ do we look at first?**

  Nodes whose parent had a larger z are more promising. We might also want to try to get a few integer solutions as quickly as possible.

- **Which fractional variable do we branch on, if we have a choice?**

  We might care if x_i is very close to an integer or far from one. We might also care if x_i has a high coefficient in the objective function.