# Optimisation and Operations Research

## Lecture 18: Branch and Bound

Matthew Roughan
<matthew.roughan@adelaide.edu.au>
http://www.maths.adelaide.edu.au/matthew.roughan/notes/OORII/

School of Mathematical Sciences,
University of Adelaide

August 13, 2019

Section 1

# Branch and Bound

# Are "heuristics" the only approach?

- We are solving ILPs (Integer Linear Programs)
- So far have considered heuristics
  - assumption is there is no tractable method to guarantee a solution
  - but complexity analysis is about "worst case"
  - also, we might have

$$O\big(exp(n)\big) = 0.0000000001 \times e^n$$

  - typical cases might be quite tractable
- So can we find an algorithm that works well when the problem is notionally NP-hard, but the particular instance isn't too bad?

# Example ILP

## Example

Consider the Knapsack Problem we considered earlier (which is a Binary Linear Program). A hiker can choose from the following items:

| Item | 1 chocolate | 2 raisins | 3 camera | 4 jumper | 5 drink |
|---|---|---|---|---|---|
| $w_i$ (kg) | 0.5 | 0.4 | 0.8 | 1.6 | 0.6 |
| $v_i$ (value) | 2.75 | 2.5 | 1 | 5 | 3.0 |
| $v_i/w_i$ | 5.5 | 6.25 | 1.25 | 3.125 | 5 |

The hiker wants to maximise the value of the carried items subject to a total weight constraint of 2.5 kg, *i.e.*, in general solve

$$\max \left\{ \sum_i v_i z_i \,\middle|\, \sum_i w_i z_i \leq W, z_i = 0 \text{ or } 1 \right\}$$

where the $z_i$ are binary indicator variables for each item.

# Let's see what AMPL/lpsolve does

INPUT:

```
param n;              # the parameters are set
param w{i in 1..n};   #     in a .dat file
param v{i in 1..n};
param W;

var z{i in 1..n} >= 0 binary;

maximize value:    sum{i in 1..n} v[i]*z[i];
subject to weight: sum{i in 1..n} w[i]*z[i] <= W;
```

OUTPUT:

```
LP_SOLVE 4.0.1.0: optimal, objective 10.25
12 simplex iterations
3 branch & bound nodes: depth 2
```

SOLUTION: $\mathbf{z} = (1, 1, 0, 1, 0)^T$ and the value is 10.25

# Branch and Bound

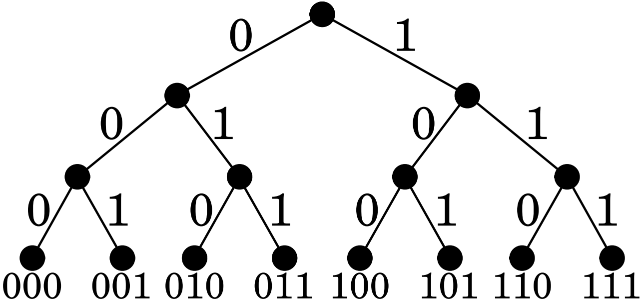
- `lpsolve` is using a method called "Branch & Bound"
  - it found the optimum solution
  - it "knows" it is the correct solution
  - somehow it used Simplex on the way?
- The goal of this lecture is to explain B&B

# Branching

Imagine we are solving a Binary Linear Program, *e.g.*,

$$(BLP) \quad \mathbf{z}^* = \max \left\{ \mathbf{c}^T \mathbf{x} \;\middle|\; A\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge 0, \mathbf{x} \in \{0,1\}^n \right\}$$

Then we can *enumerate* all of the possible solutions on a tree



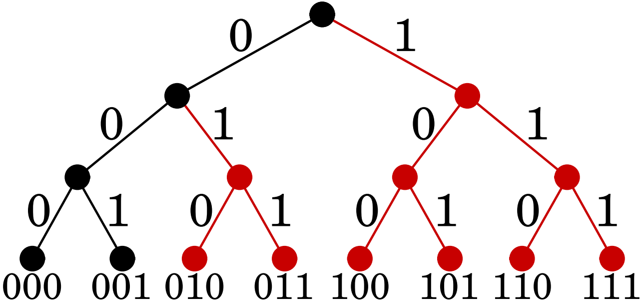

But there are $2^n$ solutions – we can't evaluate them all

# Branching and Pruning

Imagine we are solving a Binary Linear Program, *e.g.*,

$$(BLP) \quad \mathbf{z}^* = \max\left\{\mathbf{c}^T\mathbf{x} \;\middle|\; A\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge 0, \mathbf{x} \in \{0,1\}^n\right\}$$

What if we could eliminate some sub-branches




We don't have to search the whole tree

# Branching and Pruning

- Pruning reduces the search space
  - hopefully to the point where we can search the entire space
- Requires
  - a method to branch for general ILPs
    - binary branching, even when the problem isn't binary
  - a method to find "solutions" part way down a branch
  - a method to determine when a branch can be pruned
    - we will use *bounds* created by *relaxations*

# Branching of ILPs

- Branching of Binary IPs
  - pick a variable $z_i$
  - left branch has $z_i = 0$, right branch has $z_i = 1$
  - in either case $z_i$ is no longer a "variable"
  - we have *partitioned* the feasible solutions into two sets
    - divide and conquer
- Generalise the idea for Integer LPs
  - partition the set into two parts
  - pick a variable $x_i$ and a divider $c$ (which is NOT an integer)
  - left branch is $x_i \leq \lfloor c \rfloor$ and right branch is $x_i \geq \lceil c \rceil$

$$\begin{aligned} \lfloor c \rfloor &= \text{the floor of } c \\ \lceil c \rceil &= \text{the ceiling of } c \end{aligned}$$

  - $x_i$ is still a variable, but on a restricted space

# Example ILP

## Example

Consider the Integer Linear Program

$$\begin{array}{rrcrcl} \max z & = & x & + & y & \\ \text{s.t.} & & -x & + & 2y & \leq & 8 \\ & & 23x & + & 10y & \leq & 138 \end{array}$$

for non-negative integers $x$ and $y$.

Branch on $x$ at $c = 3.5$, and we get two new LPs $\max z = x + y$ such that

$$\begin{array}{rcrcr} -x & + & 2y & \leq & 8 \\ 23x & + & 10y & \leq & 138 \\ x & & & \leq & 3 \end{array} \quad \text{and} \quad \begin{array}{rcrcr} -x & + & 2y & \leq & 8 \\ 23x & + & 10y & \leq & 138 \\ x & & & \geq & 4 \end{array}$$

# Relaxation: a reminder

- *Relaxation* means defining a new problem with some of the original constraints dropped
  - in this context, we drop some of the integrality constraints

## Example (continued)

$$
\begin{array}{rrcrcl}
\max z & = & x & + & y & \\
\text{s.t.} & & -x & + & 2y & \leq 8 \\
 & & 23x & + & 10y & \leq 138 \\
 & & & & x, y & \in \mathbb{Z}^+
\end{array}
$$

Relax the integer constraints, *i.e.*, form a new problem $(LP_0)$ with $x, y \in \mathbb{R}^+$. Solving $(LP_0)$ gives the optimal solution as

$$
z_0^* = 9\frac{1}{4} \quad \text{at} \quad (x_0^*, y_0^*)^T = \left(3\frac{1}{2}, 5\frac{3}{4}\right)^T
$$

# Relaxation issues

- *Relaxation* means defining a new problem with some of the original constraints dropped
  - in this context, we drop some of the integrality constraints
- Remember that in relaxing an ILP to a LP
  - the solution to the LP might not be close to that of the ILP
  - a feasible LP might not indicate a feasible ILP
- So relaxation by itself isn't a good approach to solve an ILP
  - but we can use these to generate "partial" solutions to help search for a fully feasible solution

# What can we tell from a relaxation?

For each Integer Linear Program:

$$(\text{ILP}) \qquad \mathbf{z}^* = \max\{\mathbf{c}^T\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0, \mathbf{x} \in \mathbb{Z}^n\}$$

there is an associated relaxed Linear Program:

$$(LP_0) \qquad \mathbf{z}_0^* = \max\{\mathbf{c}^T\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0, \mathbf{x} \in \mathbb{R}^n\}$$

Now, $(LP_0)$ is *less constrained* than the $(ILP)$ so

- If $(LP_0)$ is infeasible, then so is $(ILP)$
- If $(LP_0)$ is optimised by integer variables, then that solution is feasible and optimal for the $(ILP)$
- The optimal objective value for $(LP_0)$ is greater than or equal to the optimal objective for the $(ILP)$

$$\mathbf{z}_0^* \geq \mathbf{z}^*$$

# Relaxation Gives Bounds

- The relaxed problem is a LP
  - we know how to solve this, *e.g.*, Simplex
- The relaxed LP tells us something about the ILP
  - it doesn't give the solution
  - it does provide an *upper bound* on the solution

**Example (continued)**

Solving $(LP_0)$ gives the optimal solution as

$$z_0^* = 9\frac{1}{4} \quad \text{at} \quad (x_0^*, y_0^*)^T = \left(3\frac{1}{2}, 5\frac{3}{4}\right)^T$$

The ILP has solution

$$z^* = 8 \leq z_0^*$$

- We can use the bounds to prune branches

# Branch and Bound

- Keep a list of *subproblems* resulting from branching, and work on these one by one
  - solve relaxed versions to get upper bounds
  - sometimes we might also get an integer solution
- *key:* if upper bound of a subproblem is less than objective for a known integer feasible solution, then
  - the subproblem cannot have a solution greater than the already known solution
  - we can eliminate this solution
  - we can also prune all of the tree below the solution
- it lets us do a *non-exhaustive* search of the subproblems
  - if we get to the end, we have a proof of optimality without exhaustive search

# Branch and Bound: algorithm

1. *Initialization:* initialize variables, in particular, start a list of subproblems, initialized with our original integer program.
2. *Termination:* terminate the program when we reach the optimum (*i.e.*, the list of subproblems is empty).
3. *Problem selection and relaxation:* select the next problem from the list of possible subproblems, and solve a relaxation on it.
4. *Fathoming and pruning:* eliminate branches of the tree once we prove they cannot contain an optimal solution.
5. *Branching:* partition the current problem into subproblems, and add these to our list.

# Branch and Bound: example

Consider the problem (from [LM01])

$$
\text{IP}^0 \left\{ \begin{array}{lrcl} \text{maximize} & 13x_1 + 8x_2 & & \\ \text{subject to} & x_1 + 2x_2 & \leq & 10 \\ & 5x_1 + 2x_2 & \leq & 20 \\ & x_1 \geq 0, x_2 \geq 0 & & \\ & x_1, x_2 \text{ integer} & & \end{array} \right.
$$

# Branch and Bound: algorithm

*Initialization:*

- initialize the **list** of problems $\mathcal{L}$
  - set initially $\mathcal{L} = \{\text{IP}^0\}$, where $\text{IP}^0$ is the initial problem
  - often store/picture $\mathcal{L}$ as a tree
- incumbent objective value $z_{ip} = -\infty$
  - best (integer) solution we have found so far
  - initial value is the worst possible
- initial value of upper bound on problem is $\bar{z}_0 = \infty$
  - If the upper bound of a solution $\bar{z}_i < z_{ip}$ then this problem $\text{IP}^i$ (and its dependent tree) obviously cannot achieve the same objective value that we have already achieved elsewhere in our solutions.
- constraint set of problem $\text{IP}^0$ is set to be

$$S^0 = \{\mathbf{x} \in \mathbb{Z}^n | A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0\}$$

# Branch and Bound: algorithm

*Termination:*

- If $\mathcal{L} = \phi$ then we stop
  - If $z_{ip} = -\infty$ then the integer program is infeasible.
    - our search didn't find an integer feasible solution
  - Otherwise, the subproblem $\text{IP}^i$ which yielded the current value of $z_{ip}$ is optimal gives the optimal solution $\mathbf{x}^*$

We stop branch and bound when we have run out of subproblems (which are listed in $\mathcal{L}$ ) to solve, *i.e.*, when $\mathcal{L}$ is empty.

# Branch and Bound: algorithm

*Problem selection:*

- select a problem from $\mathcal{L}$
  - there are multiple ways to decide which problem to choose from the list
    - ★ the method used can have a big impact on speed
  - once selected, delete the problem from the list

*Relaxation:*

- solve a relaxation of the problem
  - denote the optimal solution by $\mathbf{x}^{iR}$
  - denote the optimal objective value by $z_i^R$
    - ★ $z_i^R = -\infty$ if no feasible solutions exist

# Branch and Bound: algorithm

For the example

$$
\text{IP}^0 \left\{ \begin{array}{lrcl} \text{maximize} & 13x_1 + 8x_2 & & \\ \text{subject to} & x_1 + 2x_2 & \leq & 10 \\ & 5x_1 + 2x_2 & \leq & 20 \\ & x_1 \geq 0, x_2 \geq 0 & & \\ & x_1, x_2 \text{ integer} & & \end{array} \right.
$$

the relaxation is

$$
\text{LP}^0 \left\{ \begin{array}{lrcl} \text{maximize} & z = 13x_1 + 8x_2 & & \\ \text{subject to} & x_1 + 2x_2 & \leq & 10 \\ & 5x_1 + 2x_2 & \leq & 20 \\ & x_1 \geq 0, x_2 \geq 0 & & \end{array} \right.
$$

which has solutions $x_1^{0R} = 2.5$ and $x_2^{0R} = 3.75$ with $z_0^R = 62.5$

# Branch and Bound: algorithm

*Fathoming :*

- we say branch of the tree is **fathomed** if
  - infeasible
  - feasible solution, and $z_i^R \leq z_{ip}$
  - integral feasible solution
    - set $z_{ip} \leftarrow \max\{z_{ip}, z_i^R\}$

*Pruning:*

- in any of the cases above, we need not investigate any more subproblems of the current problem
  - subproblems have more constraints
  - their $z$ must lie under the upper bound
- Prune any subtrees with $z_j^R \leq z_{ip}$
- If we pruned *Goto step 2*

We don't prune the example yet (see later for complete example).

# Branch and Bound: algorithm

*Branching:*

- also called partitioning
- want to partition the current problem into subproblems
  - there are several ways to perform partitioning
- If $S^i$ is the current constraint set, then we need a disjoint partition $\{S^{ij}\}_{j=1}^{k}$ of this set
  - we add problems $\{\mathrm{IP}^{ij}\}_{j=1}^{k}$ to $\mathcal{L}$
  - typically $k = 2$ for binary branching
  - $\mathrm{IP}^{ij}$ is just $\mathrm{IP}^{i}$ with its feasible region restricted to $S^{ij}$
- *Goto step 2*

# Branch and Bound: example

Consider the problem (from [LM01])

$$\text{IP}^0 \begin{cases} \text{maximize} & 13x_1 + 8x_2 \\ \text{subject to} & x_1 + 2x_2 \leq 10 \\ & 5x_1 + 2x_2 \leq 20 \\ & x_1 \geq 0, x_2 \geq 0 \\ & x_1, x_2 \text{ integer} \end{cases}$$

with relaxation

$$\text{LP}^0 \begin{cases} \text{maximize} & z = 13x_1 + 8x_2 \\ \text{subject to} & x_1 + 2x_2 \leq 10 \\ & 5x_1 + 2x_2 \leq 20 \\ & x_1 \geq 0, x_2 \geq 0 \end{cases}$$

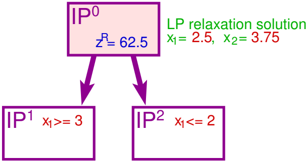
which has solutions $x_1^0 = 2.5$ and $x_2^0 = 3.75$ with $z_0^R = 62.5$

# Branch and Bound: algorithm

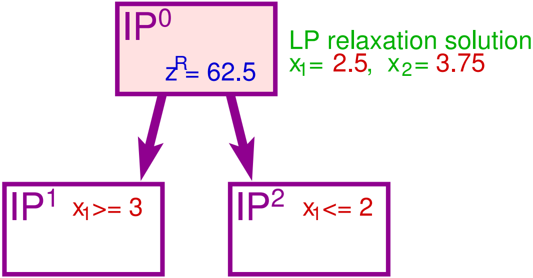
In the example we partition on $x_1$

- this is the "most infeasible"
  - furthest from an integral value (because $x_1^0 = 2.5$)
- partition into two subproblems around $c = 2.5$
  - $\text{IP}^1$ has $x_1 \geq 3$
  - $\text{IP}^2$ has $x_1 \leq 2$

So now $\mathcal{L} = \{\text{IP}^1, \text{IP}^2\}$



# Branch and Bound: example




$\mathcal{L} = \{\text{IP}^1, \text{IP}^2\}$

# Branch and Bound: example

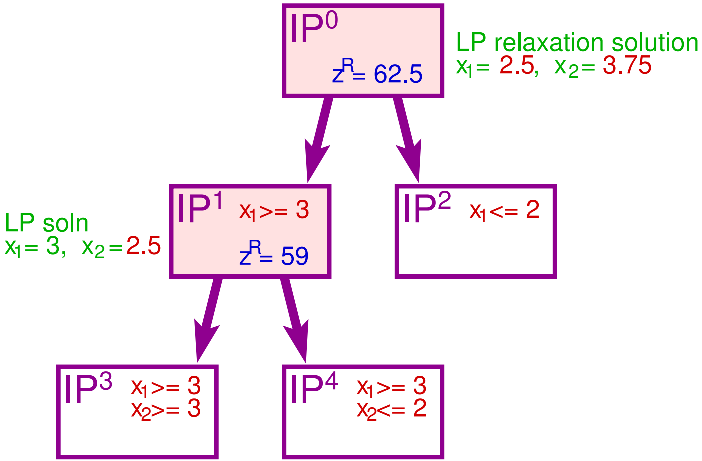
Problem selection (just chose in order) of $\text{IP}^1$

$$\text{IP}^1 \begin{cases} \text{maximize} & 13x_1 + 8x_2 & & \\ \text{subject to} & x_1 + 2x_2 & \leq & 10 \\ & 5x_1 + 2x_2 & \leq & 20 \\ & x_1 & \geq & 3 \\ & x_1 \geq 0, x_2 \geq 0 & & \\ & x_1, x_2 \text{ integer} & & \end{cases}$$

The relaxation (to a LP) has solutions

- $x_1^1 = 3$ and $x_2^1 = 2.5$ with $z_1^R = 59$
- we will next partition on $x_2$
  - $\text{IP}^3$ has $x_2 \leq 2$
  - $\text{IP}^4$ has $x_2 \geq 3$

# Branch and Bound: example




$\mathcal{L} = \{\text{IP}^2, \text{IP}^3, \text{IP}^4\}$

# Branch and Bound: example

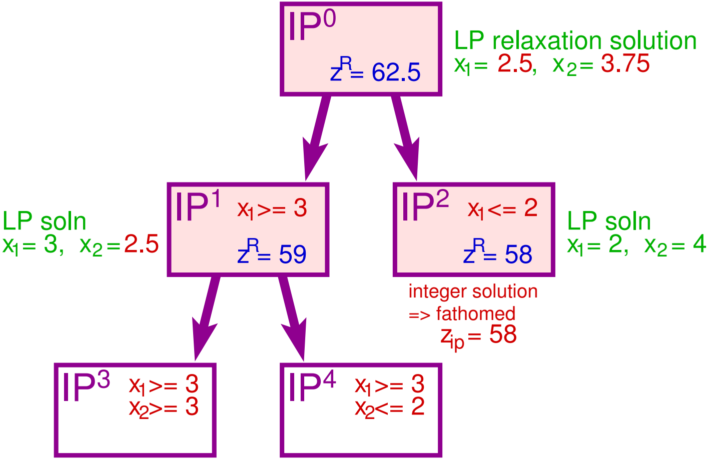
Problem selection (best bound) of $IP^2$

$$
IP^2 \begin{cases}
\text{maximize} & 13x_1 + 8x_2 & & \\
\text{subject to} & x_1 + 2x_2 & \leq & 10 \\
 & 5x_1 + 2x_2 & \leq & 20 \\
 & x_1 & \leq & 2 \\
 & x_1 \geq 0, x_2 \geq 0 & & \\
 & x_1, x_2 \text{ integer} & &
\end{cases}
$$

The relaxation (to a LP) has solutions

- $x_1^2 = 2$ and $x_2^2 = 4$ with $z_2^R = 58$
- *integral feasible*
- So set $z_{ip} = 58$
- And $IP^2$ is *fathomed*
    - no more subproblems

# Branch and Bound: example




$\mathcal{L} = \{\text{IP}^3, \text{IP}^4\}$

# Branch and Bound: example

Problem selection (order) of $IP^3$

$$
IP^3 \left\{ \begin{array}{lrcl}
\text{maximize} & 13x_1 + 8x_2 & & \\
\text{subject to} & x_1 + 2x_2 & \leq & 10 \\
 & 5x_1 + 2x_2 & \leq & 20 \\
 & x_1 & \geq & 3 \\
 & x_2 & \geq & 3 \\
 & x_1 \geq 0, x_2 \geq 0 & & \\
 & x_1, x_2 \text{ integer} & &
\end{array} \right.
$$

The relaxation (to a LP) is infeasible

- $z_3^R = -\infty$
- $IP^3$ is fathomed
- $\mathcal{L} = \{IP^4\}$

# Branch and Bound: example

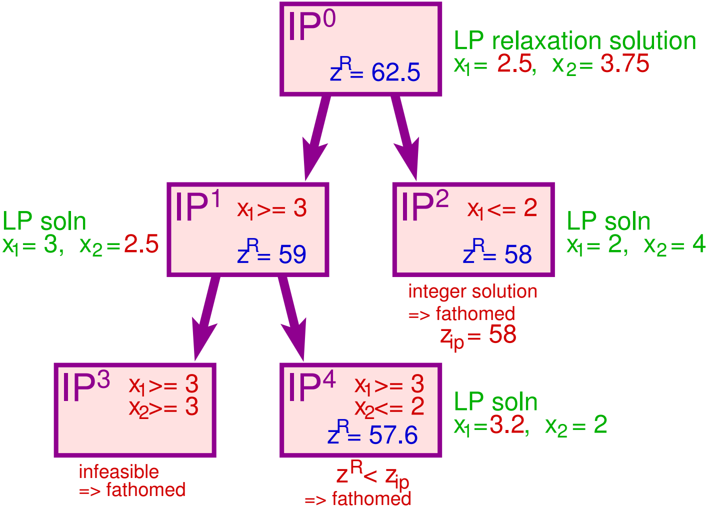
Problem selection (only possible one) of $\text{IP}^4$

$$
\text{IP}^4 \left\{ \begin{array}{lrcl} \text{maximize} & 13x_1 + 8x_2 & & \\ \text{subject to} & x_1 + 2x_2 & \leq & 10 \\ & 5x_1 + 2x_2 & \leq & 20 \\ & x_1 & \geq & 3 \\ & x_2 & \leq & 2 \\ & x_1 \geq 0, x_2 \geq 0 & & \\ & x_1, x_2 \text{ integer} & & \end{array} \right.
$$

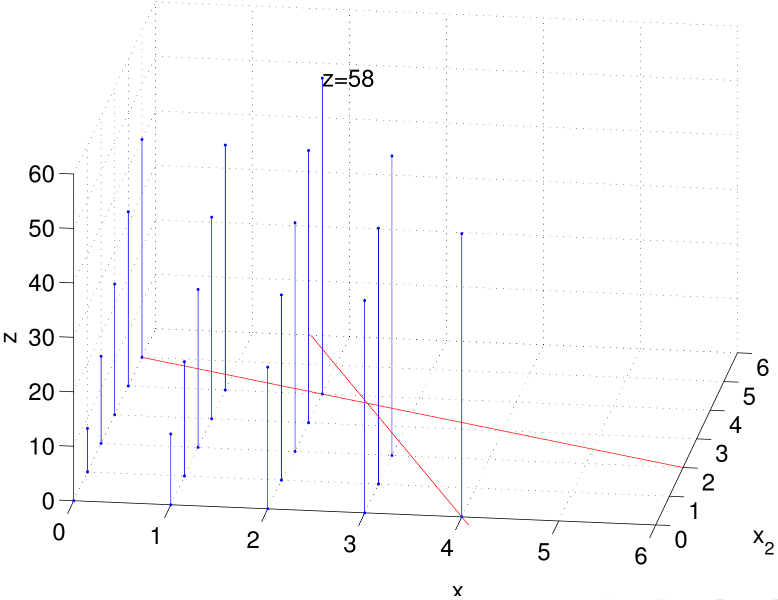
The relaxation (to a LP) has solution

- $x_1^2 = 3.2$ and $x_2^2 = 2$ with $\boxed{z_4^R = 57.6 < z_{ip}}$
- $\text{IP}^4$ is fathomed

# Branch and Bound: example



# Branch and Bound: example



# Takeaways

- B&B uses pruning to perform a non-exhaustive search
  - we can prune branches when they are
    - infeasible
    - integer feasable
    - their upper bound (on their relaxation) is less than an existing solution
- More on B&B in the next lecture

# Further reading I

Eva K. Lee and John Mitchell, *Encyclopedia of optimization*, ch. Branch-and-bound methods for integer programming, Kluwer Academic Publishers, 2001, http://www.rpi.edu/~mitchj/papers/leeejem.html.