Communications Network Design

lecture 12

Matthew Roughan
<matthew.roughan@adelaide.edu.au>

Discipline of Applied Mathematics
School of Mathematical Sciences
University of Adelaide

March 2, 2009

Budget constraint model and branch and bound

Branch and bound is a standard technique for solving integer programs, by relaxing the problem to the non-integer problem to find bounds, and using these to prune a tree of the possible solutions (rather than evaluating all possible solutions).

This lecture introduces an simplified network design problem (the budget constraint model) and the concept of branch and bound optimization.

Budget Constraint Model

- separable linear cost model

$$
\begin{aligned}
C(\mathbf{f}) &= \sum_{e \in L(\mathbf{f})} (\beta_e + \alpha_e f_e) \qquad \text{where } L(\mathbf{f}) = \{e \in E : f_e > 0\} \\
&= \sum_{e \in L(\mathbf{f})} \beta_e + \sum_{\mu \in P} l_\mu(L(\mathbf{f}))\, x_\mu
\end{aligned}
$$

- separate costs into
  - initial **investment** costs (of laying optical fibre)

$$C_{\text{inv}}(L) = \sum_{e \in L} \beta_e$$

  - **operations** cost of lighting up the link

$$C_{\text{op}}(\mathbf{f}, L) = \sum_{e \in L} \alpha_e f_e$$

Budget Constraint Model (BCM)

- ealier, we considered the problem

$$\min C(\mathbf{f}) = \min \left[C_{\text{inv}}(L) + C_{\text{op}}(\mathbf{f}, L)\right]$$

subject to the appropriate constraints

- budget constraint model
  - have a budget constraint on the investment costs

$$C_{\text{inv}}(L) \le B$$

  - consider the optimization problem

$$\min C_{\text{op}}(\mathbf{f}, L) \text{ subject to } C_{\text{inv}}(L) \le B$$

with additional constraints as above.

Given $L(\mathbf{f}) = \{e \in E : f_e > 0\}$

$$
\begin{aligned}
C(\mathbf{f}) &= \sum_{e \in L(\mathbf{f})} (\beta_e + \alpha_e f_e) \\
&= \sum_{e : f_e > 0} (\beta_e + \alpha_e f_e) \\
&= \sum_{e : f_e > 0} c_e(f_e) \\
c_e(f_e) &= \begin{cases} 0 & \text{if } f_e = 0 \\ \beta_e + \alpha_e f_e & \text{if } f_e > 0 \end{cases}
\end{aligned}
$$

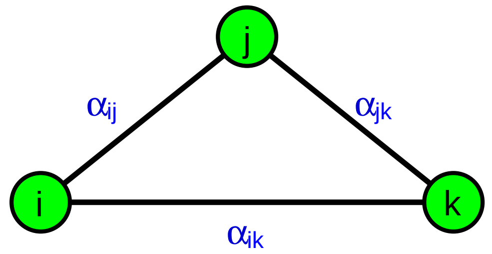
Formulation: of BCM

$$
\begin{aligned}
\text{(P')} \quad \text{min} \quad C(\mathbf{f}) &= \sum_{e \in L} \alpha_e f_e \\
\text{s.t.} \quad f_e &= \sum_{\mu : e \in \mu} x_\mu && \forall e \in E \\
\sum_{\mu : \mu \in P_k} x_\mu &= t_k && \forall k \in K \\
\sum_{e \in E} \beta_e z_e &\leq B \\
x_\mu &\geq 0 && \forall \mu \in P \\
z_e &= 0, \text{ or } 1 && \forall e \in E
\end{aligned}
$$

$$
z_e = \begin{cases} 1 & \text{if link } e \in L \text{ (i.e. we use } e\text{)} \\ 0 & \text{if link } e \notin L \text{ (i.e. we don't use } e\text{)} \end{cases}
$$

BCM and the triangle inequality

- α_e satisfy the triangle inequality



$$\alpha_{ij} < \alpha_{ik} + \alpha_{kj}$$

  - because β_e have been moved into constraints
  - otherwise, link $e = (i, j)$ could be deleted as it is a longer path than $i-k-j$

BCM and Branch and Bound

- this is an old, well studied problem, e.g. see [1]
- NP-hard
- look for heuristic solutions
  - branch and bound [2]
- **Branch and Bound** is the topic of this lecture.

Notation

We can write an optimization problem several different ways

- integer linear programming problem, called (IP)

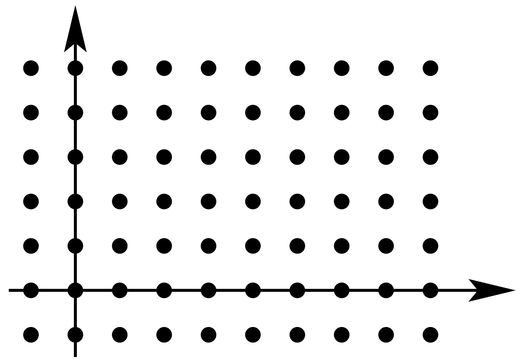
$$
(\mathrm{IP}) \left\{ \begin{array}{llcl} \text{maximize} & \mathbf{c}^T\mathbf{x} & & \\ \text{subject to} & A\mathbf{x} & \leq & \mathbf{b} \\ & \mathbf{x} & \geq & 0 \\ & \mathbf{x} & \in & \mathbb{Z}^n \end{array} \right.
$$

- short form

$$\max\{\mathbf{c}^T\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0, \mathbf{x} \in \mathbb{Z}^n\}$$

Communications Network Design: lecture 12 – p.7/38

$\mathbb{Z}$ is the set of integers $\{\ldots,-3,-2,-1,0,1,2,3,\ldots\}$
$\mathbb{Z}^n$ is the n-dimensional integer lattice, e.g. a segment of Z^2 is shown at the points in the figure below.



Communications Network Design: lecture 12 – p.8/38

Integer programming

- Take an integer linear programming problem

$$\max\{\mathbf{c}^T\mathbf{x} \mid A\mathbf{x} \le \mathbf{b}, \mathbf{x} \ge 0, \mathbf{x} \in \mathbb{Z}^n\}$$

  - some of our variables are real (e.g. link loads)
    - we have a **mixed**-integer linear programming problem
  - $\mathbb{Z}^n$ is the set of n-dimensional vectors of integers
    - we will restrict to $\mathbf{x} \in \{0,1\}^n$
  - Many other classic examples
    - travelling salesman problem
    - knapsack problem
    - set covering problem
    - machine scheduling problem

Converting BCM into integer program

Variables are

$$z_e = \begin{cases} 1 & \text{if link } e \in L \text{ (i.e. we use } e) \\ 0 & \text{if link } e \notin L \text{ (i.e. we don't use } e) \end{cases}$$

Write optimization objective

$$\begin{aligned} C(\mathbf{f}) &= \sum_e \alpha_e f_e && (1) \\ &= \sum_e \alpha_e \sum_{\mu: e \in \mu} x_\mu && (2) \\ &= \sum_e \sum_\mu \alpha_e A(e,\mu) x_\mu && (3) \\ &= [\alpha^t A]\,\mathbf{x} && (4) \end{aligned}$$

More information on integer programming can be found at
`http://mat.gsia.cmu.edu/orclass/integer/integer.html`

Communications Network Design: lecture 12 – p.9/38

(2) follows from the fact that in the BCM $f_e = \sum_{\mu: e \in \mu} x_\mu, \forall e \in E$.

(3) we defined the routing matrix A by

$$A(e,\mu) = \begin{cases} 1, & \text{if path } \mu \text{ uses link } e \text{, i.e. } e \in \mu \\ 0, & \text{otherwise} \end{cases}$$

(4) is just a vector/matrix representation of (3), and can be rewritten in the familiar form

$$C(\mathbf{f}) = \sum_{\mu \in P} l_\mu x_\mu = \mathbf{l}^t \mathbf{x}$$

Communications Network Design: lecture 12 – p.10/38

Converting BCM into integer program

We derive the routing vector $\mathbf{x}$ from the $\mathbf{z}$ by solving the shortest path problem (with linear costs) on the graph determined by the $\mathbf{z}$.

Converting BCM into integer program

Obvious constraints given in the BCM are

$$\sum_{\mu:\mu\in P_k} x_\mu = t_k, \quad \forall k \in K \tag{5}$$

$$\sum_{e\in E} \beta_e z_e \leq B \tag{6}$$

we just need to write these in matrix form, but there is a less obvious contraint

$$(1-z_e) f_e = (1-z_e) \sum_{\mu:e\in\mu} x_\mu = 0 \tag{7}$$

which says we cannot put traffic on absent links.

(7) says that $x_\mu = 0$ for any path μ that uses a link e that is not present in the network we build, i.e. we can't route traffic along links that don't exist.

Relationship to linear programming

For each integer program:

(IP) $\max\{\mathbf{c}^T\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0, \mathbf{x} \in \mathbb{Z}^n\}$

there is an associated linear program:

(LP) $\max\{\mathbf{c}^T\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0\}$

Now (LP) is less constrained than (IP) so

- If (LP) is infeasible, then so is (IP)
- If (LP) is optimized by integer variables, then that solution is feasible and optimal for (IP)
- The optimal objective value for (LP) is greater than or equal to the optimal objective for (IP)

Bounds

- call the (LP) a **relaxation**
  - because we have relaxed some constraints
- it is easy to solve (usually)
  - its a standard linear program
  - can use simplex, or interior point methods
- rounding off the solution to the relaxation might work badly
  - it could even produce a partitioned graph
  - not all traffic can get through!
- but the (LP) relaxation does provide a **bound**
  - we can use this to **prune branches**

Remember we can always convert a constraint such as

$$A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0.$$

into an equality by including slack variables s such that

$$A\mathbf{x} + \mathbf{s} = \mathbf{b}, \mathbf{x} \geq 0, \mathbf{s} \geq 0.$$

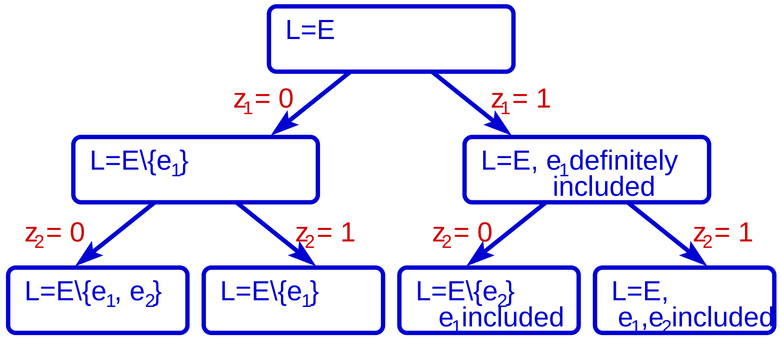
Branching

- the above gives us bounds for solutions
- we also need to **branch**
  - at each point where we don't have an integer solution, we can branch by splitting the possible solutions into two partitions
  - for example, we require $x_1 \in \{0,1\}$, but the relaxation solution was $x_1 = 0.2$, we then subdivide the problem into two parts
    - ⋆ $x_1 = 0$
    - ⋆ $x_1 = 1$
  - then solve each of these subproblems

Branching example

For the network problem, we have decision variables

$$z_e = \begin{cases} 1 & \text{if link } e \in L \text{ (i.e. we use } e) \\ 0 & \text{if link } e \notin L \text{ (i.e. we don't use } e) \end{cases}$$



Communications Network Design: lecture 12 – p.15/38

Communications Network Design: lecture 12 – p.16/38

Branch and Bound

- **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

More information and examples of Branch and bound can be found at
http://mat.gsia.cmu.edu/orclass/integer/integer.html
http://en.wikipedia.org/wiki/Branch_and_bound
http://mathworld.wolfram.com/BranchandBoundAlgorithm.html

An instructive paper is
http://www.rpi.edu/~mitchj/papers/leeejem.html

A list of implementations can be found at
http://www.mat.univie.ac.at/~neum/glopt/software_g.html

Communications Network Design: lecture 12 – p.17/38

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.

Communications Network Design: lecture 12 – p.18/38

Branch and Bound: example

Consider the problem (from [2])

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

Branch and Bound: algorithm

Initialization:

- initialize the **list** of problems $\mathcal{L}$
  - set initially $\mathcal{L} = \{\mathrm{IP}^0\}$, where IP^0 is the initial problem
  - often store/picture $\mathcal{L}$ as a tree
- incumbent objective value $z_{ip} = -\infty$
- initial value of upper bound on problem is $\bar{z}_0 = \infty$
- constraint set of problem IP^0 is set to be $S^0 = \{\mathbf{x} \in \mathbb{Z}^n | A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0\}$

The incumbent objective value z_{ip} represents the "best" solution we have found so far. If any solution does worse than this we can ignore it. Its intial value $-\infty$ is chosen to be the worst possible, so that any feasible solution will be better than this.

The bound gives us an upper bound on the solution of an integer program. Initially we don't know anything, and so the upper bound is effectively undefined by setting it to be $\bar{z}_0 = \infty$. When we solve the relaxation, we will find out this value.

If the upper bound of a solution $\bar{z}_i < z_{ip}$ then this problem IP^i obviously cannot achieve the same objective value that we have already achieved elsewhere in our solutions.

The constraint set S^i defines the set of possible solutions for the the particular problem IP^i under consideration.

Branch and Bound: algorithm

Termination:

- If $\mathcal{L} = \phi$ then we stop
  - If $z_{ip} = -\infty$ then the integer program is infeasible.
  - Otherwise, the subproblem IP^i which yielded the current value of z_{ip} is optimal gives the optimal solution $\mathbf{x}^*$

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

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.

For the example

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

the relaxation is

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

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

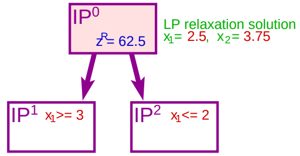
Branching:

- 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 $\{\text{IP}^{ij}\}_{j=1}^k$ to $\mathcal{L}$
  - IP^{ij} is just IP^i with its feasible region restricted to S^{ij}
- **Goto step 2**

In the example we partition on x_1

- this is the "most infeasible"
  - furthest from an integral value
- partition into two subproblems by adding an extra constraint
  - IP^1 has $x_1 \geq 3$
  - IP^2 has $x_1 \leq 2$

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



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 [2])

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

with relaxation

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

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

We can think of “branch and bound” as a meta-heuristic – there are many ways to do each step in the above algorithm, and our choice will build a particular form of branch and bound.

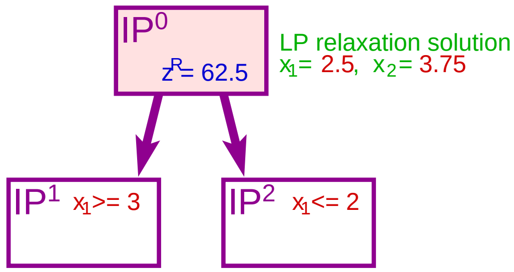
Communications Network Design: lecture 12 – p.25/38

Communications Network Design: lecture 12 – p.26/38

Branch and Bound: example

- we will partition on x_1
  - this is the "most infeasible"
    - furthest from an integral value
- we will partition on x_1
  - partition into two subproblems by adding an extra constraint
    - IP^1 has $x_1 \geq 3$
    - IP^2 has $x_1 \leq 2$
- $\mathcal{L} = \{\mathrm{IP}^1, \mathrm{IP}^2\}$

Branch and Bound: example



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

Communications Network Design: lecture 12 – p.27/38

Communications Network Design: lecture 12 – p.28/38

Branch and Bound: example

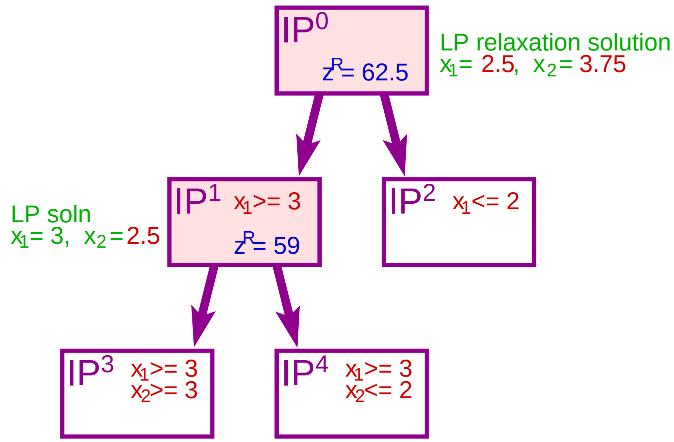
Problem selection (just chose in order) of IP^1

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

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
  - IP^3 has $x_2 \leq 2$
  - IP^4 has $x_2 \geq 3$

Branch and Bound: example



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

Communications Network Design: lecture 12 – p.29/38

Communications Network Design: lecture 12 – p.30/38

Branch and Bound: example

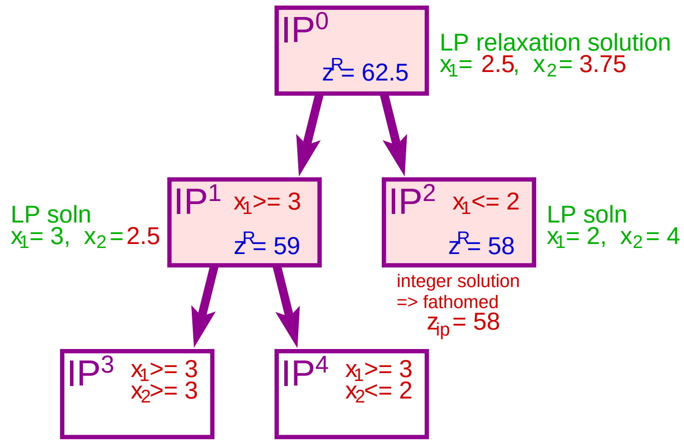
Problem selection (best bound) of IP^2

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

The relaxation (to a LP) has solutions

- $x_1^2 = 2$ and $x_2^2 = 4$ with $z_2^R = 58$
- integral feasible
- $z_{ip} = 58$
- IP^2 is fathomed

Branch and Bound: example



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

Communications Network Design: lecture 12 – p.31/38

Communications Network Design: lecture 12 – p.32/38

Branch and Bound: example

Problem selection (order) of IP^3

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

The relaxation (to a LP) is infeasible

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

Branch and Bound: example

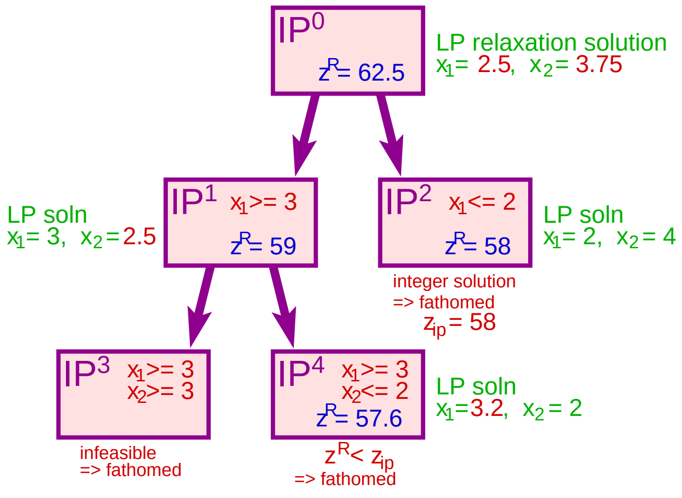
Problem selection (only possible one) of IP^4

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

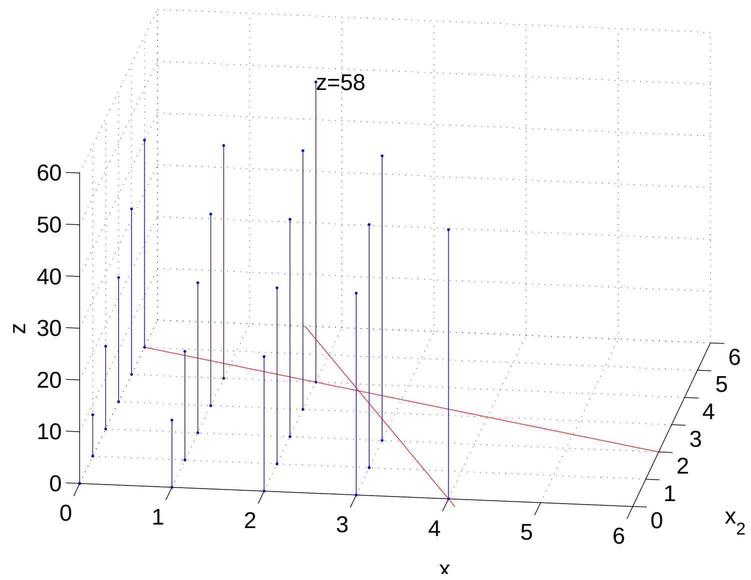
The relaxation (to a LP) has solution

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

Branch and Bound: example



Branch and Bound: example



Communications Network Design: lecture 12 – p.35/38

Communications Network Design: lecture 12 – p.36/38

Branch and Bound

- B&B is a very general algorithm
  - as described above we seek the optimum
  - can also be used as a heuristic
- different strategies available for each step above
  - can use heuristics inside B&B
  - pre-processing of the problem can be good
- no single strategy stands out as best for all problems
  - but sometimes we can exploit properties of a particular problem to do better

References

[1] D. S. Johnson, J. K. Lenstra, and A. H. G. R. Kan, "The complexity of the network design problem," Networks, vol. 8, pp. 279–285, 1978.

[2] E. K. Lee and J. Mitchell, Encyclopedia of Optimization, ch. Branch-and-bound methods for integer programming. Kluwer Academic Publishers, 2001. http://www.rpi.edu/~mitchj/papers/leeejem.html.