AM 121: Intro to Optimization Models and Methods

Fall 2016

Lecture 14: Branch and Bound (II)

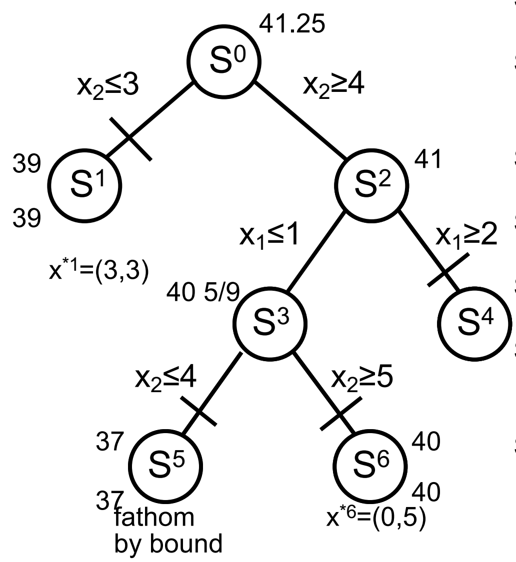
David C. Parkes
SEAS

Lesson Plan

- Branch and Bound review
- Node selection
- Branching decision
- Formulation strength
- CPLEX and “Cut generation”

Review

$$\max\ 5x_1 + 8x_2$$
$$\text{s.t. } x_1 + x_2 \leq 6$$
$$5x_1 + 9x_2 \leq 45$$
$$x_1, x_2 \geq 0, \text{ integer}$$



best soln → upper bound

S^0: (2.25, 3.75) $-\infty$, 41.25
fractional, branch on x_2

S^1: (3,3) 39, 41.25
fathom by integrality
new solution!

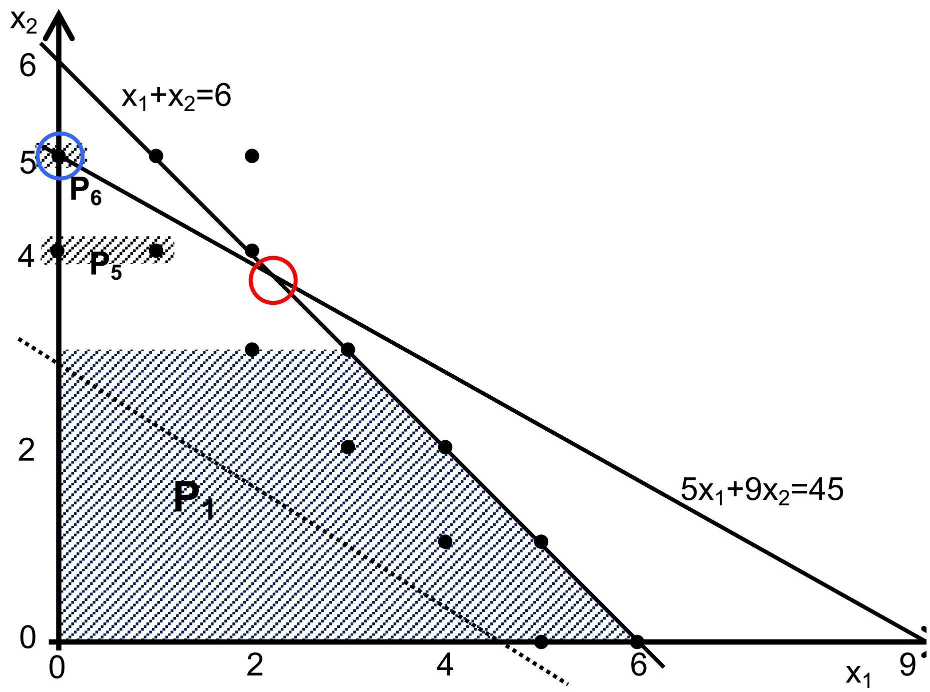
S^2: (1.8, 4) 39, 41
fractional, branch on x_1

S^3: (1, 4 4/9) 39, 41
fractional, branch on x_2

S^5: (1,4) 39, 41
fathom by bound

S^6: (0,5) 40, 41
fathom by integrality
new solution!

S^4: infeasible 40, 40
fathom by infeasibility



Branch and Bound Method

- Maintain a list of **open subproblems**, an **incumbent** $\underline{x}$ with value $\underline{z}$, and an upper bound $\bar{z}$

- **Node selection decision**: pick open subproblem and solve LP relaxation
- **Branching decision**: if can't fathom the node, then pick a fractional variable and branch

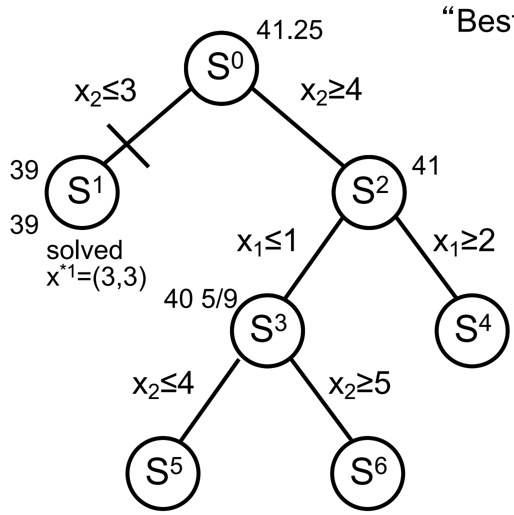
- Continue until all open subproblems are fathomed, or "optimality gap" is acceptable.

Node selection Decision

- Depth-first search (solve a node just generated)
  - + Find an integer solution quickly. This way, open subproblems can be "fathomed by bound."
  - + Can obtain next LP solution via **dual simplex pivots**, since branching adds or modifies a constraint.
- Best-bound first (solve node *k* with highest LPR $\underline{z}_k$.)
  - (Upper bound on subproblem is inherited from parent)
  - + Never solve a subproblem with an upper-bound less than the value of **optimal** integer solution.
  - + Improve upper bounds quickly, try to prove optimality of current incumbent.

- In practice: initial DFS ("diving") followed by a mix of best-bound and DFS is effective.

DFS will solve one of S^5 or S^6
"Best-bound first" will solve S^4
(since 41 > 40 5/9)



Example: Dual Pivots

- Optimal tableau for LPR of IP is

$$\begin{aligned} z \qquad\qquad & + 1.25x_3 + 0.75x_4 = 41.25 \\ x_1 \qquad & + 2.25x_3 - 0.25x_4 = 2.25 \\ x_2 & - 1.25x_3 + 0.25x_4 = 3.75 \qquad (1) \end{aligned}$$

- Consider LPR of $S^2 = S^0 \cap \{x_2 \geq 4\}$
- Introduce excess variable $x_5 \geq 0$, and write $x_2 \geq 4$ as

$$x_2 \qquad\qquad - x_5 = 4 \qquad (2)$$

- Establish basis $B=\{x_1,x_2,x_5\}$ by (2)' = (1) - (2)

$$-1.25x_3 + 0.25x_4 + x_5 = -0.25 \qquad (2')$$

Using Dual Pivots in BnB

- Initial tableau for LPR of S^2 is **dual feasible**:

$$\begin{aligned} z \qquad\qquad & + 1.25x_3 + 0.75x_4 = 41.25 \\ x_1 \qquad & + 2.25x_3 - 0.25x_4 = 2.25 \\ x_2 & - 1.25x_3 + 0.25x_4 = 3.75 \\ \leftarrow \qquad & \boxed{-1.25x_3} + 0.25x_4 + x_5 = -0.25 \\ & \quad \uparrow \end{aligned}$$

- Dual Pivot (x_5 out, x_3 in). Get:

$$\begin{aligned} z \qquad\qquad & + x_4 + x_5 = 41 \\ x_1 \qquad & + 0.2x_4 + 1.8x_5 = 1.8 \\ x_2 & \qquad\qquad - x_5 = 4 \\ & x_3 - 0.2x_4 - 0.8x_5 = 0.2 \end{aligned}$$

- $B=\{x_1,x_2,x_3\}$, $x_1^*=1.8$, $x_2^*=4$, $z^*=41$
- **One pivot!** In general will take more than one pivot, but can often find new LP solution quickly.

Aside: Weak Dual Pairs

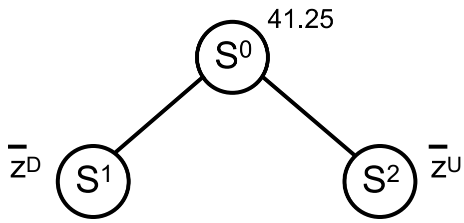
- Consider
  Primal: max{ c(x) : x∈X}
  Dual: min { w(y) : y∈Y}
- Form a **weak dual pair** if $c(x) \le w(y), \forall x \in X, \forall y \in Y$

- **Proposition**. max $\{c^T x: Ax \le b, x \ge 0, \text{integer}\}$ and
  min $\{b^T y: A^T y \ge c, y \in R^m_{\ge 0}\}$ form a weak dual pair.
- => any feasible solution to the dual of current (primal) LPR provides a valid upper bound.
- => means that don't even need to solve dual to optimality. Can prune by bound earlier!

Branching Decision

- Most-fractional variable:
  - try to make progress towards integer soln quickly
- User priorities (e.g., "big decisions" first):
  - "the location of a facility is more consequential than the districts it serves" for example
- Strong branching:
  - "look ahead" before making a commitment to a branching decision
- Pseudocost method:
  - *estimate* effect on objective value of LPR of branching (approx form of strong branching, looks at what has happened earlier in problem)

Strong Branching: Example



S^0: (2.25, 3.75)   41.25

which var branch at S^0?

$x_1 \leq 2$ $\bar{z}^D = 41\ 1/9$
$x_1 \geq 3$ $\bar{z}^U = 39$

$x_2 \leq 3$ $\bar{z}^D = 39$
$x_2 \geq 4$ $\bar{z}^U = 41$

Which most quickly improves the bound?
Answer: branching on x_2 since it minimizes the max bound

Strong Branching

Let *C* denote set of integer vars in current subproblem with a fractional assignment

LP solution

For each $j \in C$:
(a) solve subproblem with $x_j \leq \lfloor \bar{x_j} \rfloor$ and $x_j \geq \lceil \bar{x_j} \rceil$
(b) let $\bar{z}_j^D$ and $\bar{z}_j^U$ denote the value of LP solns

Branch on $j^* = \arg\min_{j \in C} \max[\bar{z}_j^D, \bar{z}_j^U]$

Can also solve subproblems approximately, just using a few dual simplex pivots.

Unboundedness in IPs

(advanced material)

- If an LP relaxation is unbounded:
  - if the integer variables take on integer values, then IP is unbounded. Else, we can branch.
- But, BnB may not terminate on a problem that is infeasible or unbounded.
- Will terminate if feasible, bounded, or if all integer variables bounded.

$$\min\ 0$$
$$1 \leq 3x - 3y \leq 2$$
$$x, y \in \mathbb{Z}$$

Infeasible, but B&B keeps searching

Formulation Strength

Recall: Firehouse Location

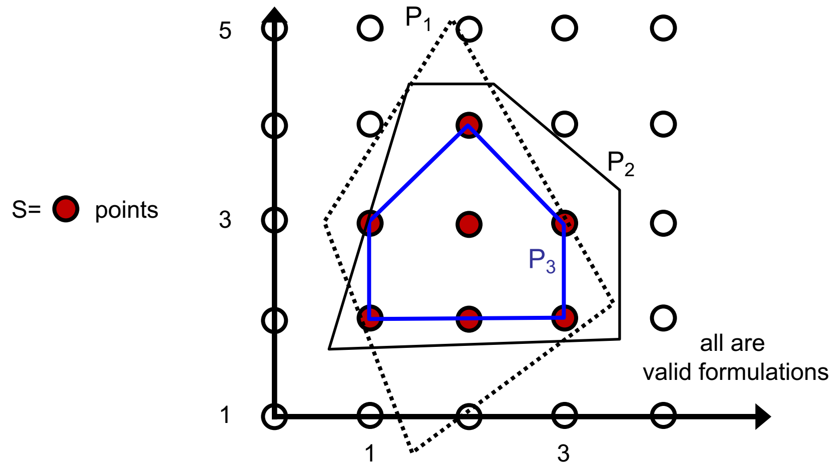
Formulation (P_1):
•If m=2 sites, and n=4 districts:
•$x_{11} + x_{21} + x_{31} + x_{41} <= 4y_1$
•$x_{12} + x_{22} + x_{32} + x_{42} <= 4y_2$
•m constraints

Alternate formulation (P_2):
•$x_{11} <= y_1$; $x_{21} <= y_1$; $x_{31} <= y_1$; $x_{41} <= y_1$
•$x_{12} <= y_2$; $x_{22} <= y_2$; $x_{32} <= y_2$; $x_{42} <= y_2$
•mn constraints

- **Definition**. The **LP relaxation** (LPR) of an IP replaces all integer variables with continuous variables.
- **Definition**. The **polyhedron of an IP** is the feasible region of the LPR.

Valid formulations

- Consider (IP) max $\{c^T x: x \in S \subseteq Z^n\}$
- **Defn.** Polyhedron P is a **valid formulation** for the IP if $P \cap Z^n = S$



Strong Formulations

- **Proposition.** Consider two valid formulations P_1 and P_2, with $P_2 \subset P_1 \subseteq R^n$. Then $z_2^{LP} \leq z_1^{LP}$.
- **Proof.** Suppose $z_2^{LP} > z_1^{LP}$. But x* that is best in P_2 is also feasible in P_1. Contradiction!

- Say that "P_2 is **stronger** than P_1."

Example

Tight "big M":

- $x_{11}+x_{21}+x_{31}+x_{41} \leq 4y_1$ P_1

Loose "big M":

- $x_{11}+x_{21}+x_{31}+x_{41} \leq 10y_1$ P_1'

- Both are valid formulations.
- But P_1' has additional fractional solutions, e.g. x=(0.5, 0.5, 0.5, 0.5), y=(0.25)
- => P_1 is stronger than P_1'

Importance of Strong Formulations

Benefit 1:

Improve LP bounding → more fathoming of open problems by bound. (main advantage).

Benefit 2:

If search is best-bound first, better guidance in regard to node selection.

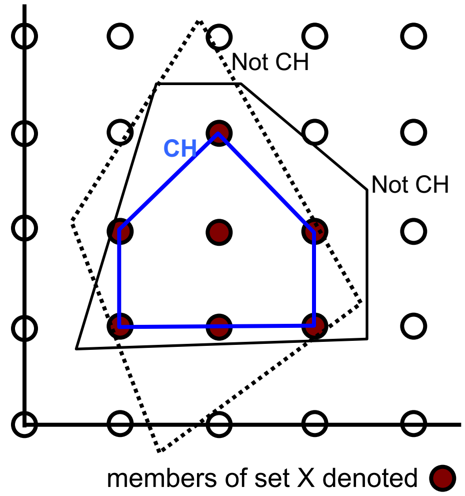
Benefit 3:

Fewer optimal, non-integral solutions.

Convex Hull

- **Definition.** Given set $X \subseteq Z^n$, the **convex hull** of $X=\{x^1,\ldots,x^t\}$ is *conv*(X)=$\{x: x=\sum_{k=1}^{t} \lambda_k x^k, \sum_{k=1}^{t}\lambda_k=1, \lambda_k \geq 0 \text{ for all } k\}$

- **Prop.** *conv*(X) is a polyhedron.
- **Prop.** Extreme points of *conv*(X) all lie in X.
- **Prop.** Can solve IP via solving LP on *conv*(X).



An "Ideal" formulation!

Replace IP $\max\{c^T x: x \in X\}$ with the **LP** $\{\max c^T x: x \in conv(X)\}$

- **Problem**: can require an exponential number of inequalities to define *conv*(X)
  - If $|X|=q$, may need as many as 2^q inequalities.
- Better be the case, else we'd have P=NP!

Computational Complexity (Brief!)

- **Decision problems**: Is there a TSP tour cost ≤ 10?
- **P**: class of decision problems that can be solved in polynomial time ("easy")
- **NP**: class of problems for which when answer is YES there is easy (poly time) proof (e.g., TSP)

- A problem is **NP-complete** if it is in **NP** and any problem in **NP** can be "*reduced*" (in poly time) to the problem; e.g. 0/1 integer programs.

Widely conjectured that P≠NP

Complexity of LP

(advanced material)

- The simplex method is fast in practice, but not worst-case polynomial time.

- First **polynomial-time** LP algorithm was devised in 1979 by Khachian (made headlines!).
- Khachian's **Ellipsoid method** is an **interior point method**. Does not rely on vertex solutions. Fits an increasingly good ellipsoid approximation. Poly time

- In 1984, Karmarkar announced a poly-time interior-point ,method with solution times 50x better than simplex. Again made headlines!

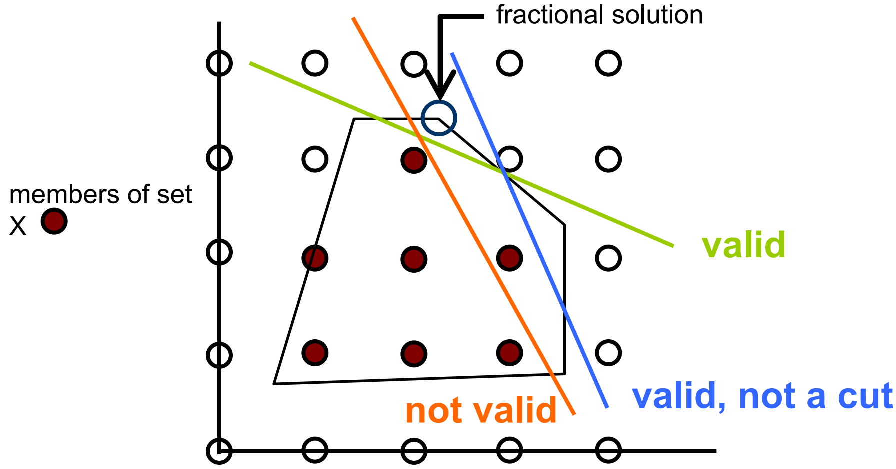
- LP is in class **P**, however 0/1 IP is **NP-complete**.
- If a polynomially-sized description of *conv*(X) could be obtained for every IP, we could solve IPs in poly-time via reduction to LPs.

- Eureka! We would prove **P=NP.**

Alternative Goal

- Q: What else can we do (other than formulate the convex hull) to improve the strength of formulations?
- A: *Try to automatically approximate conv(X) on a given instance, strengthening the formulation.*

- **Defn**. A **valid inequality** may remove some fractional solns, but removes no integer solns.
- **Defn**. A **cut** is a valid inequality that **removes** the current fractional solution x^*.



Looking at CPLEX

- IBM's Ilog CPLEX solver is used by AMPL
- Routinely used to solve real world problems of large economic significance

Some CPLEX features

- Automated Cut Generation
  - At “root” node (Global cuts)
  - At search nodes (Local cuts)
- Automated bound strengthening
  - Tighten right-hand side
- Primal heuristics
  - Look for integer feasible solutions that are close to current fractional solution

Example

```
var X1 integer >= 0;
var X2 integer >= 0;
maximize Obj: 5 * X1 + 8 * X2;
subject to C1: X1 + X2 <= 6;
subject to C2: 5 * X1 + 9 * X2 <= 45;
end;

ampl: model simple.mod;
ampl: option solver cplex;
ampl: solve;
```

Enabling feedback to AMPL

```
option cplex_options
   'timing = 1'        # display timing info
   'mipdisplay=2'      # display MIP information
   'mipinterval=1';   # node interval
```

timing=1: show how much cpu time used to solve the problem
mipdisplay=2: show # of open nodes
mipinterval=n: display information every n nodes and whenever it finds an integer solution

```
Root relaxation solution time = 0.00 sec. (0.00 ticks)

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                            0.0000       70.0000        2     ---
      0     0       41.2500     2        0.0000       41.2500        2     ---
*     0+    0                           39.0000       41.2500        2    5.77%
*     0+    0                           40.0000       41.2500        2    3.12%
      0     0        cutoff             40.0000                      2    0.00%
Elapsed time = 0.01 sec. (0.03 ticks, tree = 0.00 MB)

Root node processing (before b&c):
  Real time             =    0.01 sec. (0.03 ticks)
Parallel b&c, 16 threads:
  Real time             =    0.00 sec. (0.00 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =    0.01 sec. (0.03 ticks)

Times (seconds):
Input =  0.000782
Solve =  0.018569
Output = 0.000492
CPLEX 12.6.0.0: optimal integer solution; objective 40
2 MIP simplex iterations
0 branch-and-bound nodes
```

Node: current node
Nodes left: # of open nodes
Objective: value of current LPR
IInf: # integer infeasible variables
Best Integer: best solution found
Best Bound: current upper bound
ItCnt: total #pivots so far
Gap: relative gap between the best integer solution and best bound

Disabling Presolve

```
option presolve 0;
option cplex_options
      'timing = 1'
      'mipdisplay=2'
      'mipinterval=1'
      'boundstr=0'
      'dependency=0'
      'coeffreduce=0'
      'presolve=0'
      'cutpass= -1'
      'scale= -1'
      'prerelax= 0'
      'presolvenode = -1';
```

boundstr=0:	no bound strengthening
dependency=0:	no dependency checker in presolve
coeffreduce=0:	do not do coefficient reduction
cutpass =-1:	stop all cut generation
scale = -1:	do not scale the problem
prerelax = 0:	do not presolve at the initial LPR
presolvenode=-1:	do not presolve at each node

Disabling Primal Heuristics

```
option cplex_options
      'fpheur = -1'
      'heurfreq = -1'
      'rinsheur = -1';
      …
```

fpheur: Whether to use the feasibility pump heuristic on MIP problems (find initial feasible):
- -1 = no
- 0 = automatic choice (default)

heurfreq: How often to apply "node heuristics" for MIPS
- -1= no
- 20=every twenty nodes

rinsheur Relaxation INduced neighborhood Search HEURistic for MIP problems:
- -1 = none
- 0 = automatic choice of interval (default)
- n (for n > 0) = every n nodes.

```
Root relaxation solution time = 0.00 sec. (0.00 ticks)

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

      0     0       41.2500     2                     41.2500        2
      0     2       41.2500     2                     41.2500        2
Elapsed time = 0.01 sec. (0.05 ticks, tree = 0.01 MB)
      1     2       41.0000     1                     41.2500        3
*     2     1      integral     0       39.0000       41.2500        4    5.77%
      3     1       40.5556     1       39.0000       41.2500        5    5.77%
*     4     0      integral     0       40.0000       41.2500        6    3.12%

Root node processing (before b&c):
  Real time             =    0.00 sec. (0.01 ticks)
Parallel b&c, 16 threads:
  Real time             =    0.02 sec. (0.05 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =    0.02 sec. (0.05 ticks)

Times (seconds):
Input =  0.000825
Solve =  0.091536
Output = 0.000714
CPLEX 12.6.0.0: optimal integer solution; objective 40
6 MIP simplex iterations
5 branch-and-bound nodes
```

Node: current node
Nodes left: # of open nodes
Objective: value of current LPR
IInf: # integer infeasible variables
Best Integer: best solution found
Best Bound: current upper bound
ItCnt: total #pivots so far
Gap: relative gap between the best integer solution and best bound

Summary: Branch and Bound

- Node Selection
  - DFS together with dual pivots
  - Followed by best-first search (use LP bounds)
- Branching decision
  - Most-fractional, **pseudo-cost based**, user priorities, strong branching.
- Formulation strength
  - More fathoming by bound, better node selection guidance. Fewer fractional optimal solns
- Cut generation
  - Automatically tighten the formulation as search