Wednesday, November 5

Program of the day:
- Solving MIP models by branch-and-bound (Wolsey chapter 7)
- Applications: Knapsack Problem (demo)
- Design issues in branch-and-bound
- Relaxations

Techniques for MIP
- Preprocessing
- Branch-and-bound
- Valid cuts

Development
1960 Breakthrough: branch-and-bound
1970 Small problems \((n < 100)\) may be solved. Exponential growth, many important problems cannot be solved.
1983 Crowder, Johnson, Padberg: new algorithm for pure BIP. Sparse matrices up to \((n = 2756)\).
1985 Johnson, Kostreva, Sahl: further improvements.
1987 Van Roy, Wolsey: Mixed IP. Up to 1000 binary variables, several continuous variables.

Now Preprocessing, addition of cuts, good branching strategies

Solving IP by enumeration
- Binary IP
  \[
  \begin{align*}
  \text{maximize } & \quad 2x_1 + 3x_2 - 1x_3 + 5x_4 \\
  \text{subject to } & \quad 4x_1 + 1x_2 + 2x_3 + 3x_4 \leq 8 \\
  & \quad x_1, x_2, x_3, x_4 \in \{0, 1\}
  \end{align*}
  \]

- Integer IP
  \[
  \begin{align*}
  \text{maximize } & \quad 2x_1 + 3x_2 - 1x_3 + 5x_4 \\
  \text{subject to } & \quad 4x_1 + 1x_2 + 2x_3 + 3x_4 \leq 8 \\
  & \quad x_1, x_2, x_3, x_4 \in \mathbb{N}
  \end{align*}
  \]

- Mixed integer IP
  \[
  \begin{align*}
  \text{maximize } & \quad 2x_1 + 3x_2 - 1x_3 + 5x_4 \\
  \text{subject to } & \quad 4x_1 + 1x_2 + 2x_3 + 3x_4 \leq 8 \\
  & \quad x_1, x_2 \in \mathbb{R} \\
  & \quad x_3, x_4 \in \{0, 1\}
  \end{align*}
  \]

Elements of Branch-and-bound

Problem
\[
\begin{align*}
\text{maximize } & \quad cx \\
\text{subject to } & \quad x \in S
\end{align*}
\]

- **Divide and conquer** (Wolsey prop. 7.1)
  \[
  \begin{align*}
  S &= S_1 \cup S_2 \cup \ldots \cup S_K \quad \text{and} \quad z^k = \max\{cx : x \in S_k\} \\
  z &= \max_{k=1,\ldots,K} z^k
  \end{align*}
  \]
  
  Often: decomposition by splitting on decision variable
  
  Overlap between \(S_i\) and \(S_j\) is allowed
Elements of Branch-and-bound

- **Upper bound function** (Wolsey prop. 7.2)
  \[ z^k = \sup \{ cx : x \in S_k \} \]
  then
  \[ z = \max z^k \]
  is an upper bound on \( S \)

- **Lower bound** (so far best solution) \( \overline{z} \)

- **Upper bound test**
  if \( z^k \leq \overline{z} \) then \( x^* \not\in S_k \)

Example: Knapsack Problem

Given \( n \) items and a knapsack

- Item \( j \) has the weight \( w_j \)
- Profit of item \( j \) is \( p_j \)
- The capacity of the knapsack is \( c \)

\[
\begin{align*}
\text{maximize} & \quad \sum_{j=1}^{n} p_j x_j \\
\text{subject to} & \quad \sum_{j=1}^{n} w_j x_j \leq c \\
& \quad x_j \in \{0,1\}, \quad j = 1, \ldots, n.
\end{align*}
\]

Important problem
- Budgeting
- Transportation
- Subproblem (e.g. separation of valid inequalities)

Branch-and-bound

A systematical enumeration technique for solving IP/MIP problems, which apply bounding rules to avoid to examine specific parts of the solution space.

\[
\begin{align*}
\text{maximize} & \quad cx \\
\text{subject to} & \quad Ax \leq b \\
& \quad x^t \geq 0 \\
& \quad x^n \geq 0, \text{integer}
\end{align*}
\]

- Branching tree enumerates all integer variables.
- Once all integer variables are fixed, remaining problem is solved by LP.
- General MIP algorithm does not know structure of problem
- Upper bounds \( \overline{z} \) are derived in each node by LP-relaxation.
- If \( \overline{z} \leq \overline{z} \) then descendant nodes need not to be examined

Branch-and-bound for MIP

Recursive procedure which at each node:

- If infeasible, backtrack
- Solve LP-relaxation, getting \( \overline{x} \) and \( \overline{z} \)
- If \( \overline{z} \leq \overline{z} \) then backtrack
- If all \( x \) are integral: update \( \overline{z} \), backtrack
- Choose a fractional variable \( x_i = d \)
- Branch on

\[
\begin{align*}
x_i & \leq \lfloor d \rfloor \\
x_i & \geq \lceil d \rceil
\end{align*}
\]

Where

- objective function should be maximized
- \( \overline{z} \) is so far best solution (incumbent solution)
- \( \overline{z} \) is upper bound at node
- \( \overline{x} \) is LP-solution to current problem
Branch-and-bound for MIP

Example:

maximize \( x_1 + x_2 \)
subject to
\[
\begin{align*}
  x_1 + 5x_2 &\leq 20 \\
  5x_1 + 3x_2 &\leq 20 \\
  6x_1 + x_2 &\leq 21 \\
  x_1, x_2 &\geq 0, \text{ integer}
\end{align*}
\]

Branch on most fractional variable, best-first search

Root node
- LP-solution \( x_1 = \frac{20}{11}, x_2 = \frac{40}{11} = 3.636. \)
- Lower bound \( z = -\infty. \)
- Two nodes: \( x_2 \leq 3 \) and \( x_2 \geq 4 \) with upper bounds \( \bar{z} = 5.2 \) and \( \bar{z} = 4. \)

Node 1
- Add constraint \( x_2 \leq 3 \), getting LP-solution \( x_1 = \frac{14}{7} = 2.2 \) and \( x_2 = 3. \)
- Two nodes: \( x_1 \leq 2 \) and \( x_1 \geq 3 \) with upper bounds \( z = 5 \) and \( z = \frac{14}{7} = 4.6667. \)

Node 2
- Add constraint \( x_1 \leq 2 \), getting LP-solution \( x_1 = 2 \) and \( x_2 = 3. \) Upper bound \( \bar{z} = 5. \) Feasible solution \( z = 5. \)

Node 3
- Add constraint \( x_1 \geq 3 \), getting LP-solution \( x_1 = 3 \) and \( x_2 = \frac{5}{2} = 1.6667. \) Upper bound \( \bar{z} = 4.6667 < \bar{z}. \)

Node 4
- Add constraint \( x_2 \geq 4 \), getting LP-solution \( x_1 = 0 \) and \( x_2 = 4. \) Upper bound \( \bar{z} = 4 < \bar{z}. \)

Design issues

maximize \( cx \)
subject to \( x \in S \)

Pruning rules (Wolsey 7.2)
- Prune by optimality \( z^k = \max \{ cx : x \in S_k \} \)
- Prune by bound \( z_k \leq \bar{z} \)
- Prune by infeasibility \( S_k = \emptyset \)

Branching rules (Wolsey 7.4)
- most fractional variable \( j \) i.e. \( x_j = [x_j] \) close to \( \frac{1}{2} \)
- least fractional variable
- greedy approach

Selecting next problem
- Depth-first-search (quickly find solution, small changes in LP, space)
- Best-first-search (greedy approach)

Design issues

Relaxation (Wolsey 2.1)

\[
\begin{align*}
\max & \{ cx : x \in S \} \quad (IP) \\
\max & \{ f(x) : x \in T \} \quad (RP)
\end{align*}
\]

RP is a relaxation of IP if
- \( S \subseteq T \)
- \( f(x) \geq cx \) for all \( x \in S \)

Which constraints should be relaxed
- Quality of bound (tightness of relaxation)
- Remaining problem can be solved efficiently
- Constraints difficult to formulate mathematically
- Constraints which are too expensive to write up