Lecture 6

Instructor: A. Kokossis

Laboratory teaching staff: A. Nikolakopoulos
Lecture 6

MIXED INTEGER PROGRAMMING
Mixed Integer Non-Linear Programming

\[
\text{MIN } f(x^c, x^d)
\]

\[u.\pi.: \quad h(x^c) = 0\]
\[g(x^c, x^d) \leq 0\]
\[x_j^c \in \mathbb{R}^n\]
\[x^d \in \{0, 1\}^m\]

Available methods:

- Branch and Bound
- Generalized Benders Approximation
- Relaxation and outer approximation
Why discrete optimisation

Process synthesis
- System integration (integration of units, flowsheets and networks)
- Knowledge models (i.e. logical constraints, propositional logic)

Discontinuous systems
- Discontinuities (i.e. cost models of economics of scale)
- Procedural models (i.e. if in region A use this model; otherwise use model B)
Modelling with binary variables

Binary variable \( y = \begin{cases} 1 & \text{if } p \text{ True} \\ 0 & \text{if } p \text{ False} \end{cases} \)

- Integer formulation can resolve discontinuities
- Mixed-integer linear programming
Use of integer variables - propositional logic

(a) B is true only if A is true

\[ y_B - y_A \leq 0 \]

\[ y_A = 0 \quad y_B \leq 0 \Rightarrow y_B = 0 \]

\[ y_A = 1 \quad y_B \leq 1 \quad \text{or} \quad y_B = 1 \]

(b) If A is not true, anything related to A must be eliminated.

Let us assume \( \text{Var}_A \geq 0 \)

\[ \text{Var}_A - y_A \cdot UB_{\text{Var}_A} \leq 0 \]

\[ y_A = 0 \]

\[ 0 \leq \text{Var}_A \leq 0 \]

\[ \Rightarrow \text{Var}_A = 0 \]

\[ y_A = 1 \]

\[ \text{Var}_A \leq UB_{\text{Var}_A} \]
Preserve linearity of the model

- The model needs to maintain its linearity

Example

<table>
<thead>
<tr>
<th>Linear formulation</th>
<th>Non linear formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependence (a)</td>
<td>$y_B - y_A \leq 0$</td>
</tr>
<tr>
<td>Dependence (b)</td>
<td>$\text{Var}_A - y_A \cdot UB \leq 0$</td>
</tr>
</tbody>
</table>

- Enough solution methods
- Impossible or very difficult to solve
Modeling of initial problems

Binary variable

\[ y = \begin{cases} 
1 & \text{if } p \text{ True} \\
0 & \text{if } p \text{ False} 
\end{cases} \]

- How do we model in each case;
Solution methods

- “Brute force” approach
- Relaxation method
- Naïve’s method
- Branch and bound
- Benders Decomposition
- Cutting Plane Method
“Brute force” method

Solution of linear problems (LPs) for all combinations of 0-1

\[ y_i \quad i = 1, 2, \ldots n \]

So as to find the global optimum.

Next we select the minimum (or maximum) solution

\[ N_{LP} = 2^n \]

i.e. we examine:

<table>
<thead>
<tr>
<th># variables</th>
<th># LP</th>
<th># combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>~10^{30}</td>
<td>2^n</td>
</tr>
</tbody>
</table>
Relaxation method

All discrete variables are relaxed and assumed to be continuous:

\[ y \in \{0, 1\} \Rightarrow 0 \leq y \leq 1 \]

We solve the LP problem (there will not be necessarily an integer solution).

Integer solution 0 - 1 can only occur in special cases:

- ✓ assignment problem
- ✓ transportation problem
- ✓ transhipment problem
- ✓ Shortest Path
Integer solution – Example: Assignment problem

$m$ machines
$n$ jobs

How do we assign jobs to machines?

$y_{ij}$

Jobs $\rightarrow$ Machines

$\sum_{i} \sum_{j} c_{ij} y_{ij} \rightarrow \text{Max}$

$\sum_{i=1}^{m} y_{ij} = 1, \quad j = 1, \ldots, n$

$\sum_{j=1}^{n} y_{ij} = 1, \quad j = 1, \ldots, m$

$y_{ij} = \{0,1\}^{m} \rightarrow 0 \leq y_{ij} \leq 1$

**Sufficient condition for 0 – 1 solutions:**

For $Bx^d \leq D$

$B$ must be unimodular, i.e. every square, invertible sub-matrix $A(B)$ should have $\det(A) = 1$.

For general MILP problems the solution of the relaxed problem is not integer.
Naïve relaxation

**Basic ideal:** Rounding to the closest integer.

\[ y_i^* = 0.8 \rightarrow y_i^* = 1.0 \quad y_i^* = 0.1 \rightarrow y_i^* = 0 \]

**Example:**

\[
\begin{align*}
\min \quad & z = -12y_1 - y_2 \\
y_1 + y_2 & \leq 1 \\
1.2y_1 + 0.5y_2 & \leq 1 \\
y_1, y_2 & \in \{0, 1\}
\end{align*}
\]

Optimum: \((y_1^*, y_2^*) = (0, 1)\)

**Relaxation**

\[
\begin{align*}
y_1^* & = 0.715 \rightarrow 1.0 \\
y_2^* & = 0.285 \rightarrow 0.0
\end{align*}
\]

\[
\begin{align*}
\downarrow
\end{align*}
\]

\[
(y_1^*, y_2^*) = (1, 0) \quad \text{Rounded solution}
\]

Infeasible!
Branch and Bound

Initial node

Parent node

Offspring node 4 5 6 7

Terminal nodes
Solution search

Basic idea: bypass exclusive evaluation of all \{0,1\} combination making use of properties.

- Let \( z^* \) be the optimal solution in the binary tree. Then \( z^* \) will be one of \( z_8^*, z_9^*, \ldots z_{15}^* \).

- In each node \( i \) the relaxed LP \( z_i^* \) can be solved assuming the integer variables to be continuous. If \( i \) precedes \( j \) then
  \[ z_i^* \leq z_j^* \] (assuming minimization)

Thus
- If \( z_i^* \) is not feasible \( \Rightarrow \) anything under \( i \) is infeasible
- We assume that we solve the relaxed LPs 1, 3, 6, 12 and 2 and that
  \[ z_1^* \leq z_3^* \leq z_6^* \leq z_{12}^* \]
  \[ z_1^* \leq z_2^* \]

Then
  \[ z_1^* \leq z^* \leq z_{12}^* \]

Assume now \[ z_{12}^* \leq z_2^* \]
Bounding the tree

Redundant part of the tree
Observations

• Properties are not always enough to define next moves. Usually heuristic rules are followed to conclude the algorithmic procedure.

Usual questions:

- From which node do we branch next;
  - Depth first: branch from the most recent
  - Breadth first: from that with the smallest lower bound

- Which variables must be kept integer?
  - Those closer to 0.5
  - Those that come first in the list
Benders Decomposition

General formulation

\[
\begin{align*}
\text{min} & \quad f(x, y) = c^T \cdot y + F(x) \\
x, y & \quad h(x, y) = A \cdot y + H(x) = 0 \\
& \quad g(x, y) = \beta y + G(x) \leq 0 \\
x & \in \mathbb{R}, y \in \{0, 1\}
\end{align*}
\]

Basic idea

We “Decompose” the problem in

- A **primal problem** which provides an upper bound (ONLY CONTINUOUS VARIABLES)
- A **master problem** which gives a lower bound (ONLY INTEGER VARIABLES)
- We repeat (improving lower and upper bounds) until convergence.
Solution Approximation

- **MI(N)LP model**
  - **Primal problem** ⇒ **Upper bound**
  - **Master** ⇒ **Κάτω όριο**
  - **Termination**

- **(Constant binary, solves for the rest and calculates Lagrange multipliers (typical NLP))**
  - **Formulates Lagrange, solves for binaries with constant marginal values (typical MILP)**
DICOPT++

- Discrete and Continuous OPTimiser (Viswanathan και Grossman, EDRC/CMU)

- The algorithms can deal with non-convexity, but cannot find always the optimum

- Can handle only binary and continuous variables - not integer

- Give good results for small number of integer variables (<50) and smooth enough models (equations)