Integer Linear Programming (ILP)

Zdeněk Hanzálek, Přemysl Šůcha
hanzalek@fel.cvut.cz

CTU in Prague

March 8, 2017
## Table of contents

1. **Introduction**
   - Problem Statement
   - Comparison of ILP and LP
   - Examples

2. **Algorithms**
   - Enumerative Methods
   - Branch and Bound Method
   - Special Cases of ILP

3. **Problem Formulation Using ILP**

4. **Conclusion**
Problem Statement

**Integer Linear Programming (ILP)**

The ILP problem is given by matrix $A \in \mathbb{R}^{m \times n}$ and vectors $b \in \mathbb{R}^{m}$ and $c \in \mathbb{R}^{n}$. The goal is to find a vector $x \in \mathbb{Z}^{n}$ such that $A \cdot x \leq b$ and $c^{T} \cdot x$ is the maximum.

Usually, the problem is given as $\max \{ c^{T} \cdot x : A \cdot x \leq b, x \in \mathbb{Z}^{n} \}$.

- A large number of practical optimization problems can be modeled and solved using Integer Linear Programming - ILP.
The ILP problem differs from the LP problem in allowing only integer-valued variables. If some variables can contain real numbers, the problem is called Mixed Integer Programming - MIP. Often MIP is also called ILP, and we will use the term ILP when at least one variable has integer domain.

If we solve the ILP problem by an LP algorithm and then just round the solution, we could not only get the suboptimal solution, we can also obtain a solution which is not feasible.

While the LP is solvable in polynomial time, ILP is NP-hard, i.e. there is no known algorithm which can solve it in polynomial time.

Since the ILP solution space is not a convex set, we cannot use convex optimization techniques.
2-Partition Problem

- **Instance:** Number of banknotes $n \in \mathbb{Z}^+$ and their values $p_1, \ldots, p_n$, where $p_{i \in 1..n} \in \mathbb{Z}^+$.

- **Decision:** Is there a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} p_i = \sum_{i \not\in S} p_i$?

The decision problem, which can be written while using the equation above as an ILP constraint (but we write it differently).

- $x_i = 1$ iff $i \in S$

This is one of the “easiest” NP-complete problems.

\[
\begin{align*}
\text{min} & \quad 0 \\
\text{subject to:} & \quad \sum_{i \in 1..n} x_i \cdot p_i = 0.5 \cdot \sum_{i \in 1..n} p_i \\
\text{parameters:} & \quad n \in \mathbb{Z}^+, \; p_{i \in 1..n} \in \mathbb{Z}^+ \\
\text{variables:} & \quad x_{i \in 1..n} \in \{0, 1\}
\end{align*}
\]
Example ILP1b: Fractional Variant of the 2-Part. Prob.

We allow division of banknotes such that $x_{i \in 1..n} \in \langle 0, 1 \rangle$. The solution space is a convex set - the problem can be formulated by LP:

\[
\begin{align*}
\min & \quad 0 \\
\text{subject to:} & \\
\sum_{i \in 1..n} x_i \cdot p_i & = 0.5 \cdot \sum_{i \in 1..n} p_i \\
\sum_{i \in 1..n} x_i & \leq 1 \\
\text{parameters:} & \\
& n \in \mathbb{Z}_0^+, \quad p_{i \in 1..n} \in \mathbb{Z}_0^+ \\
\text{variables:} & \\
& x_{i \in 1..n} \in \mathbb{R}_0^+
\end{align*}
\]

- For example: $p = [100, 50, 50, 50, 20, 20, 10, 10]$ the fractional variant allows for $x = [0, 0, 0.9, 1, 1, 1, 1, 1]$ and thus divides the banknotes into equal halves $100 + 50 + 5 = 45 + 50 + 20 + 20 + 10 + 10$, but this instance does not have a non-fractional solution.

- For some non-fractional instances we can easily find that they cannot be partitioned (e.g. when the sum of all values divided by the greatest common divisor is not an even number), however we do not know any alg that can do it in polynomial time for any non-fractional instance.
Example ILP1c: 2-Partition Prob. - Optimization Version

- The decision problem can be solved by an optimization algorithm while using a threshold value (here $0.5 \times \sum_{i=1..n} p_i$) and comparing the optimal solution with the threshold.
- Moreover, when the decision problem has no solution, the optimization algorithm returns a value that is closest to the threshold.

$$
\begin{align*}
\text{min} & \quad C_{max} \\
\text{subject to:} & \quad \sum_{i=1..n} x_i \cdot p_i \leq C_{max} \\
& \quad \sum_{i=1..n} (1 - x_i) \cdot p_i \leq C_{max} \\
\text{parameters:} & \quad n \in \mathbb{Z}^+_0, \quad p_i \in 1..n \in \mathbb{Z}^+_0 \\
\text{variables:} & \quad x_i \in 1..n \in \{0, 1\}, \quad C_{max} \in \mathbb{R}^+_0
\end{align*}
$$

Application: the scheduling of nonpreemptive tasks $\{T_1, T_2, \ldots, T_n\}$ with processing times $[p_1, p_2, \ldots, p_n]$ on two parallel identical processors and minimization of the completion time of the last task (i.e. maximum completion time $C_{max}$) - $P2\|C_{max}$. The fractional variant of 2-partition problem corresponds to the preemptive scheduling problem.
Example ILP2a: Shortest Paths

Shortest Path in directed graph

- **Instance:** digraph $G$ with $n$ nodes, distance matrix $c : V \times V \rightarrow \mathbb{R}_0^+$ and two nodes $s, t \in V$.
- **Goal:** find the shortest path from $s$ to $t$ or decide that $t$ is unreachable from $s$.

LP formulation using a physical analogy:

- node = ball
- edge = string (we consider a symmetric distance matrix $c$)
- node $s$ is fixed, other nodes are pulled by gravity
- tightened string = shortest path

\[
\begin{align*}
\text{max} & \quad l_t \\
\text{subject to:} & \quad l_s = 0 \\
& \quad l_j \leq l_i + c_{i,j} \quad i \in 1..n, j \in 1..n \\
\text{parameters:} & \quad n \in \mathbb{Z}_0^+, \; c_{i\in1..n,j\in1..n} \in \mathbb{R}_0^+ \\
\text{variables:} & \quad l_{i\in1..n} \in \mathbb{R}_0^+ 
\end{align*}
\]

Is it a polynomial problem?

Z. Hanzálek (CTU) Integer Linear Programming (ILP) March 8, 2017 8 / 43
Asymmetric Traveling Salesman Problem

- **Instance:** complete digraph $K_n$ ($n \geq 3$), distance matrix $c : V \times V \rightarrow \mathbb{Q}^+$.
- **Goal:** find the shortest Hamiltonian cycle (i.e. a closed oriented walk going through all nodes).

$x_{i,j} = 1$ iff node $i$ is in the cycle just before node $j$

The enter and leave constraints do not capture the TSP completely, since any disjoint cycle (i.e. consisting of several sub-tours) will satisfy them. We use $s_i$, the “time” of entering node $i$, to **eliminate the sub-tours**.

\[
\begin{align*}
\text{min} & \quad \sum_{i \in 1..n} \sum_{j \in 1..n} c_{i,j} * x_{i,j} \\
\text{subject to:} & \quad \sum_{i \in 1..n} x_{i,j} = 1 \quad j \in 1..n \quad \text{enter once} \\
& \quad \sum_{j \in 1..n} x_{i,j} = 1 \quad i \in 1..n \quad \text{leave once} \\
& \quad s_i + c_{i,j} - (1 - x_{i,j}) * M \leq s_j \quad i \in 1..n, j \in 2..n \quad \text{cycle indivisibility}
\end{align*}
\]

parameters: $M \in \mathbb{Z}_0^+$, $n \in \mathbb{Z}_0^+$, $c_{i\in 1..n,j\in 1..n} \in \mathbb{Q}^+$

variables: $x_{i\in 1..n,j\in 1..n} \in \{0, 1\}$, $s_i \in 1..n \in \mathbb{R}_0^+$
The most successful methods to solve the ILP problem are:

- Enumerative Methods
- Branch and Bounds
- Cutting Planes Methods

Ralph Gomory and Vašek Chvátal are prominent personalities in the field of ILP. Some of the solution methods are called: Gomory’s Cuts or Chvátal-Gomory’s cuts.
Enumerative Methods

- Based on the idea of **inspecting all possible solutions**.
- Due to the integer nature of the variables, the number of solutions is countable, but their number is huge. So this method is usually suited only for smaller instances with a small number of variables.
- The LP problem is solved for every combination of discrete variables.
max $-2x_1 + x_2$

s.t.

\[9x_1 - 3x_2 \geq 11\]
\[x_1 + 2x_2 \leq 10\]
\[2x_1 - x_2 \leq 7\]

\[x_1, x_2 \geq 0, \quad x_1, x_2 \in \mathbb{Z}^+_0\]

The figure below shows 10 feasible solutions. The optimal solution is $x_1 = 2, x_2 = 2$ with an objective function value of $-2$. 
The method is based on splitting the solution space into disjoint sets. It starts by relaxing on the integrality of the variables and solves the LP problem.

If all variables $x_i$ are integers, the computation ends. Otherwise one variable $x_i \notin \mathbb{Z}$ is chosen and its value is assigned to $k$.

Then the solution space is divided into two sets - in the first one we consider $x_i \leq \lfloor k \rfloor$ and in the second one $x_i \geq \lceil k \rceil + 1$.

The algorithm recursively repeats computation for the both new sets till feasible integer solution is found.
By branching the algorithm creates a solution space which can be depicted as a tree.

A node represents the partial solution.

A leaf determines some (integer) solution or “bounded” branch (infeasible solution or the solution which does not give a better value than the best solution found up to now).

As soon as the algorithm finds an integer solution, its objective function value can be used for bounding.

The node is discarded whenever $z$, its (integer or real) objective function value, is worse than $z^*$, the value of the best known solution.

The ILP algorithm often uses an LP simplex method because after adding a new constraint it is not needed to start the algorithm again, but it allows one to continue the previous LP computation while solving the dual simplex method.
Branch and Bound Algorithm - ILP maximization problem

function \( z, x = \text{ILP}(A, b, c, z^b, x^b) \)

- \( z^{LP}, x^{LP} := \text{solution to LP problem} \)

- \( z, x := -\infty, [\ ] \)
  - return

Is the solution to LP infeasible?

- yes
  - return

- no
  
  select some non-integer \( x_i \)
  
  \( k := x_i \)
  
  solve recursively two problems:
  
  - the first one extended with \( x_i \leq k \)
    
    \( z', x' := \text{ILP}(A', b', c, z^b, x^b) \)
    
    if \( z' > z^b \) then \( z^b, x^b := z', x' \)
    
  - the second one extended with \( x_i \geq k+1 \)
    
    \( z'', x'' := \text{ILP}(A'', b'', c, z^b, x^b) \)
    
    if \( z'' > z^b \) then \( z^b, x^b := z'', x'' \)

Are all variables integer?

- yes
  
  \( z, x := z^b, x^b \)
  
  return

- no
  
  \( z, x := z^{LP}, x^{LP} \)
  
  return
Branch and Bound - Example

\[
\begin{align*}
\text{max} & \quad -x_1 + 2x_2 \\
& \quad 2x_1 - x_2 \leq 5 \\
& \quad -4x_1 + 4x_2 \leq 5 \\
& \quad x_1, x_2 \geq 0 \\
& \quad x_1, x_2 \in \mathbb{Z}
\end{align*}
\]

Search direction

\[
\begin{align*}
& x_1 \leq 1 \\
& x_1 \geq 2
\end{align*}
\]

Infeasible solution

It is not needed to continue since \( z < z^* \).

Since the search space has no other solution with \( z > z^* \), algorithm terminates.

The first feasible solution

The second feasible solution with a better value

The third feasible solution

Infeasible solution with a better value

Search direction

\[
\begin{align*}
& x_1 \leq 0 \\
& x_1 \geq 1
\end{align*}
\]
ILP Solution Space

\[ \text{max } z = 3x_1 + 4x_2 \]
\[ \text{s.t. } 5x_1 + 8x_2 \leq 24 \]
\[ x_1, x_2 \in \mathbb{Z}_0^+ \]

- What is optimal solution?
- Can we use LP to solve ILP problem?
Rounding is not always good choice

\[ \text{max } z = 3x_1 + 4x_2 \]

- LP solution \( z = 14.4 \) for \( x_1 = 4.8, x_2 = 0 \)
- Round, get infeasible solution \( x_1 = 5, x_2 = 0 \)
- Truncate, get \( z = 12 \) for \( x_1 = 4, x_2 = 0 \)
- Optimal solution is \( z = 13 \) for \( x_1 = 3, x_2 = 1 \)
Advantages of using integer variables

- more realistic (it does not make sense to produce 4.3 cars)
- flexible - e.g. binary variable can be used to model the decision (logical expression)
- we can formulate NP-hard problems

Drawbacks

- harder to create a model
- usually suited to solve the problems with less than 1000 integer variables
Shortest path in a graph

**Instance:** digraph $G$ given by incidence matrix $W : V \times E \rightarrow \{-1, 0, +1\}$ (such that $w_{ij} = +1$ when edge $e_j$ leaves vertex $i$), distance vector $c \in \mathbb{R}^+_0$ and two nodes $s, t \in V$.

**Goal:** find the shortest path from $s$ to $t$ or decide that $t$ is unreachable from $s$.

LP formulation:

- $x_j = 1$ iff edge $j$ is chosen
- For every node except $s$ and $t$ we enter the node as many times as we leave it

```
\begin{align*}
\text{min } & \sum_{j=1}^{m} c_j \cdot x_j \\
\text{subject to: } & \sum_{j=1}^{m} w_{s,j} \cdot x_j = 1 \quad \text{source } s \\
& \sum_{j=1}^{m} w_{t,j} \cdot x_j = -1 \quad \text{sink } t \\
& \sum_{j=1}^{m} w_{i,j} \cdot x_j = 0 \quad i \in V \setminus \{s, t\} \\
\text{pars: } & w_{i,j} \in \{-1, 0, 1\}, \ c_j \in \mathbb{R}^+_0 \\
\text{vars: } & x_j \in \mathbb{R}^+_0
\end{align*}
```

The returned values of $x_i$ are integers (binary) even though it is LP. Why?
Polynomial time algorithm for general ILP is not known, however there are special cases which can be solved in polynomial time.

**Definition - Totally unimodular matrix**

Matrix $A = [a_{ij}]$ of size $m/n$ is totally unimodular if the determinant of every square submatrix of matrix $A$ is equal 0, +1 or -1.

Necessary condition: if $A$ is totally unimodular then $a_{ij} \in \{0, 1, -1\}$ $\forall i, j$.

**Example:** ILP constraints of Shortest Paths problem are $W \times x = b$

\[
W = \begin{bmatrix}
    e_1 & e_2 & e_3 & e_4 & e_5 \\
    v_1 & 1 & 0 & 1 & 0 & 1 \\
    v_2 & -1 & 1 & 0 & 0 & 0 \\
    v_3 & 0 & 0 & -1 & 1 & 0 \\
    v_4 & 0 & -1 & 0 & -1 & -1 \\
\end{bmatrix}
\]

$x = \begin{bmatrix}
    x_1 \\
    x_2 \\
    x_3 \\
    x_4 \\
    x_5 \\
\end{bmatrix}$

$b = \begin{bmatrix}
    1 \\
    0 \\
    0 \\
    -1 \\
\end{bmatrix}$
Lemma - Sufficient Condition

Let $A$ be a matrix of size $m/n$ such that

1. $a_{ij} \in \{0, 1, -1\}$, $i = 1, \ldots, m$, $j = 1, \ldots, n$

2. each column in $A$ contains one non-zero element or exactly two non-zero elements $+1$ and $-1$

Then matrix $A$ is totally unimodular.

Lemma - Integer solution by simplex algorithm

The ILP problem with a totally unimodular matrix $A$ and integer vector $b$ can be solved by a simplex algorithm and the solution will be an integer.

Lemma - Polynomial time complexity

The ILP problem with a totally unimodular matrix $A$ and integer vector $b$ can be solved in polynomial time.
We consider 6 buildings for investment. The price and rental income for each of them are listed in the table.

<table>
<thead>
<tr>
<th>building</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>price[mil. Kč]</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>income[thousands Kč]</td>
<td>16</td>
<td>22</td>
<td>12</td>
<td>8</td>
<td>11</td>
<td>19</td>
</tr>
</tbody>
</table>

Goal:
- maximize income

Constraints:
- investment budget is 14 mil Kč
- each building can be bought only once
We consider 6 buildings for investment. The price and rental income for each of them are listed in the table.

<table>
<thead>
<tr>
<th>building</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>price [mil. Kč]</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>income [thousands Kč]</td>
<td>16</td>
<td>22</td>
<td>12</td>
<td>8</td>
<td>11</td>
<td>19</td>
</tr>
</tbody>
</table>

Goal:
- maximize income

Constraints:
- investment budget is 14 mil Kč
- each building can be bought only once

Formulation
- \( x_i = 1 \) if we buy building \( i \)

\[
\begin{align*}
\max \quad z & = 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6 \\
\text{s.t.} \quad 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 & \leq 14 \\
x_i \in \{0, 1\} & \quad i \in 1, \ldots, 6
\end{align*}
\]
Adding Logical Formula $x_1 \Rightarrow \overline{x_3}$

Another constraint:

- If building 1 is selected, then building 3 is not selected.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_3$</td>
<td>$x_1 \Rightarrow \overline{x_3}$</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Adding Logical Formula \( x_1 \Rightarrow \overline{x_3} \)

Another constraint:
- If building 1 is selected, then building 3 is not selected.

\[
\begin{array}{c|c|c}
  x_1 & x_3 & x_1 \Rightarrow \overline{x_3} \\
  \hline
  0 & 0 & 1 \\
  0 & 1 & 1 \\
  1 & 0 & 1 \\
  1 & 1 & 0 \\
\end{array}
\]
Adding Logical Formula $x_1 \Rightarrow \overline{x_3}$

Another constraint:

- If building 1 is selected, then building 3 is not selected.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_3$</th>
<th>$x_1 \Rightarrow \overline{x_3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$$\begin{align*}
\text{max} & \quad z = 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6 \\
\text{s.t.} & \quad 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14 \\
& \quad x_1 + x_3 \leq 1 \\
x_i \in 1 \ldots 6 & \in \{0, 1\}
\end{align*}$$
Adding Logical Formula $x_2 \Rightarrow x_1$

Another constraint:
- if building 2 is selected, then building 1 is selected too

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_2 \Rightarrow x_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Adding Logical Formula $x_2 \Rightarrow x_1$

Another constraint:

- if building 2 is selected, then building 1 is selected too

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_2 \Rightarrow x_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Adding Logical Formula $x_2 \Rightarrow x_1$

Another constraint:
- if building 2 is selected, then building 1 is selected too

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_2 \Rightarrow x_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

max $z = 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6$

s.t. $5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14$

$x_{i\in 1\ldots 6} \in \{0, 1\}$
Adding Logical Formula $x_4 \text{ XOR } x_5$

Another constraint:
- either building 4 is chosen or building 5 is chosen, but not both

<table>
<thead>
<tr>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_4 \text{ XOR } x_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Another constraint:

- either building 4 is chosen or building 5 is chosen, but not both

<table>
<thead>
<tr>
<th>$x_4$</th>
<th>$x_5$</th>
<th>( x_4 \text{ XOR } x_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Adding Logical Formula $x_4 \text{ XOR } x_5$

Another constraint:

- either building 4 is chosen or building 5 is chosen, but not both

<table>
<thead>
<tr>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_4 \text{ XOR } x_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$max \quad z = 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6$

$s.t. \quad 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14$

$x_4 + x_5 = 1$

$x_i \in \{0, 1\}$
Formulate:

- building 1 must be chosen but building 2 can not
- at least 3 estates must be chosen
- exactly 3 estates must be chosen
- if estates 1 and 2 have been chosen, then estate 3 must be chosen too \((x_1 \text{ AND } x_2) \Rightarrow x_3\)
- exactly 2 estates can not be chosen
Labor costs, material demand and profit are listed in the table

<table>
<thead>
<tr>
<th>product</th>
<th>T-shirt</th>
<th>shirt</th>
<th>trousers</th>
<th>capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>labor costs</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>150</td>
</tr>
<tr>
<td>material</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>160</td>
</tr>
<tr>
<td>profit</td>
<td>6</td>
<td>4</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Goal:
- maximize the profit

Constraints:
- labor capacity is 150 person-hours
- material capacity is 160 meters
Labor costs, material demand and profit are listed in the table

<table>
<thead>
<tr>
<th>product</th>
<th>T-shirt</th>
<th>shirt</th>
<th>trousers</th>
<th>capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>labor costs</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>150</td>
</tr>
<tr>
<td>material</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>160</td>
</tr>
<tr>
<td>profit</td>
<td>6</td>
<td>4</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Goal:
- maximize the profit

Constraints:
- labor capacity is 150 person-hours
- material capacity is 160 meters

Formulation
- $x_i$ is the amount of product $i$

$$
\begin{align*}
\text{max} \quad & z = 6x_1 + 4x_2 + 7x_3 \\
\text{s.t.} \quad & 3x_1 + 2x_2 + 6x_3 \leq 150 \\
& 4x_1 + 3x_2 + 4x_3 \leq 160 \\
& x_i \in \mathbb{Z}_0^+ 
\end{align*}
$$
Another constraint:

- the fixed cost has to be covered to rent the machine

<table>
<thead>
<tr>
<th>product</th>
<th>T-shirt</th>
<th>shirt</th>
<th>trousers</th>
</tr>
</thead>
<tbody>
<tr>
<td>machine cost</td>
<td>200</td>
<td>150</td>
<td>100</td>
</tr>
</tbody>
</table>
Another constraint:

- the fixed cost has to be covered to rent the machine

<table>
<thead>
<tr>
<th>product</th>
<th>T-shirt</th>
<th>shirt</th>
<th>trousers</th>
</tr>
</thead>
<tbody>
<tr>
<td>machine cost</td>
<td>200</td>
<td>150</td>
<td>100</td>
</tr>
</tbody>
</table>

Formulation

- add binary variable $y_i$ such that $y_i = 1$ when the machine producing product $i$ is the rent
- the objective function will be changed to
  $\max z = 6x_1 + 4x_2 + 7x_3 − 200y_1 − 150y_2 − 100y_3$
- join binary $y_i$ with integer $x_i$
Machine \( i \) is rented when product \( i \) is produced. We join binary \( y_i \) with integer \( x_i \) such that:

- \( y_i = 0 \) iff \( x_i = 0 \)
- \( y_i = 1 \) iff \( x_i \geq 1 \)
Machine \( i \) is rented when product \( i \) is produced. We join binary \( y_i \) with integer \( x_i \) such that:

- \( y_i = 0 \) iff \( x_i = 0 \)
- \( y_i = 1 \) iff \( x_i \geq 1 \)

For range \( x_i \in \langle 0, 100 \rangle \) these relations can be written as inequalities

- \( x_i \leq 100 \ast y_i \)
- \( x_i \geq y_i \) ...we can omit this inequality due to the objective function
  \((x_i = 0, y_i = 1 \) will not be chosen because we want \( x_i \) to be maximal and \( y_i \) minimal\)
Another constraint:

- Total amount of work can be 40, 80 or 120 hours in order to better fit the work contract

We only want some values of person-hours to be available:

\[ 3x_1 + 2x_2 + 6x_3 = \text{either 40 or 80 or 120} \]

Can be formulated using a set of additional variables \( v_{i \in 1...3} \in \{0, 1\} \) as follows:

\[
3x_1 + 2x_2 + 6x_3 = 40v_1 + 80v_2 + 120v_3 \\
\sum_{i=1}^{3} v_i = 1
\]
While modeling problems using ILP, we often need to express that the first, the second or both constraints hold. For example, $x_{i \in 1...4} \in \langle 0, 5 \rangle$, $x_{i \in 1...4} \in \mathcal{R}$ holds

$$2x_1 + x_2 \leq 5$$

or

$$2x_3 - x_4 \leq 2$$

or both

This can be modeled by a big $M$, i.e. big positive number (here 15), and variable $y \in \{0, 1\}$ so it can “switch off” one of the inequalities.

$$2x_1 + x_2 \leq 5 + M \cdot y$$

$$2x_3 - x_4 \leq 2 + M \cdot (1 - y)$$
At least One of Two Constraints Must be Valid

for $y = 0$ inequalities:

$$2x_1 + x_2 \leq 5 + M \cdot y$$
$$2x_3 - x_4 \leq 2 + M \cdot (1 - y)$$

reduce to:

$$2x_1 + x_2 \leq 5$$
At least One of Two Constraints Must be Valid

for \( y = 0 \) inequalities:

\[
2x_1 + x_2 \leq 5 + M \cdot y
\]
\[
2x_3 - x_4 \leq 2 + M \cdot (1 - y)
\]

reduce to:

\[
2x_1 + x_2 \leq 5
\]

for \( y = 1 \) inequalities:

\[
2x_1 + x_2 \leq 5 + M \cdot y
\]
\[
2x_3 - x_4 \leq 2 + M \cdot (1 - y)
\]

reduce to:

\[
2x_3 - x_4 \leq 2
\]
In 2D draw the solution space of the system of inequalities:

\[
\begin{align*}
2x_1 + x_2 & \leq 5 + M \cdot y \\
2x_1 - x_2 & \leq 2 + M \cdot (1 - y)
\end{align*}
\]

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

In 2D draw the solution space of the system of inequalities. Note that the equations correspond to parallel lines. Is it possible to find \(x_1, x_2\) such that both equations are valid simultaneously?

\[
\begin{align*}
2x_1 + x_2 & \leq 5 + M \cdot y \\
M \cdot (1 - y) + 2x_1 + x_2 & \geq 10
\end{align*}
\]

\[y \in \{0, 1\}\]
At least One of Two Constraints Must be Valid
Example: Non-preemptive Scheduling

| $r_j$, $\tilde{d}_j$ | $C_{\text{max}}$ | ... NP-hard problem |

- **Instance:** A set of non-preemptive tasks $\mathcal{T} = \{T_1, \ldots, T_i, \ldots, T_n\}$ with release date $r$ and deadline $\tilde{d}$ should be executed on one processor. The processing times are given by vector $p$.

- **Goal:** Find a feasible schedule represented by start times $s$ that minimizes completion time $C_{\text{max}} = \max_{i \in \{1,n\}} s_i + p_i$ or decide that it does not exist.

  - $T_i$ - chair to be produced by a joiner
  - $r_i$ - time, when the material is available
  - $\tilde{d}_i$ - time when the chair must be completed
  - $s_i$ - time when the chair production starts
  - $s_i + p_i$ - time when the chair production ends

Example:
At least One of Two Constraints Must be Valid
Example: Non-preemptive Scheduling

Since at the given moment, at most, one task is running on a given resource, therefore, for all task pairs \( T_i, T_j \) it must hold:

1. \( T_i \) precedes \( T_j \) \((s_j \geq s_i + p_i)\)
2. or \( T_j \) precedes \( T_i \) \((s_i \geq s_j + p_j)\)

Note that (for \( p_i > 0 \)) both inequalities can’t hold simultaneously.
We need to formulate that at least one inequality holds. We will use variable \( x_{ij} \in \{0, 1\} \) such that \( x_{ij} = 1 \) if \( T_i \) preceds \( T_j \).

For every pair \( T_i, T_j \) we introduce inequalities:

\[
\begin{align*}
    s_j + M \cdot (1 - x_{ij}) & \geq s_i + p_i \quad \text{“switched off” when } x_{ij} = 0 \\
    s_i + M \cdot x_{ij} & \geq s_j + p_j \quad \text{“switched off” when } x_{ij} = 1
\end{align*}
\]
Scheduling - Illustration of Non-convex Space

\[ s_i \geq r_i \quad i \in 1..n \quad \text{release date} \]
\[ \tilde{d}_i \geq s_i + p_i \quad i \in 1..n \quad \text{deadline} \]
\[ s_j + M \cdot (1 - x_{ij}) \geq s_i + p_i \quad i \in 1..n, j < i \quad T_i \text{ precedes } T_j \]
\[ s_i + M \cdot x_{ij} \geq s_j + p_j \quad i \in 1..n, j < i \quad T_j \text{ precedes } T_i \]

For example: \( p_i = 2, p_j = 3, r_i = r_j = 0, \tilde{d}_i = 9, \tilde{d}_j = 10 \)

Non-convex 2D space is a projection of two cuts of a 3D polytope (determined by the set of inequalities) in planes \( x = 0 \) and \( x = 1 \).
At least $K$ of $N$ Constraints Must Hold

We have $N$ constraints and we need at least $K$ of them to hold. Constraints are of type:

\[
\begin{align*}
f(x_1, x_2, \ldots, x_n) &\leq b_1 \\
f(x_1, x_2, \ldots, x_n) &\leq b_2 \\
& \vdots \\
f(x_1, x_2, \ldots, x_n) &\leq b_N
\end{align*}
\]

Can be solved by introducing $N$ variables $y_{i \in 1 \ldots N} \in \{0, 1\}$ such that

\[
\begin{align*}
f(x_1, x_2, \ldots, x_n) &\leq b_1 + M \cdot y_1 \\
f(x_1, x_2, \ldots, x_n) &\leq b_2 + M \cdot y_2 \\
& \vdots \\
f(x_1, x_2, \ldots, x_n) &\leq b_N + M \cdot y_N \\
\sum_{i=1}^{N} y_i &= N - K
\end{align*}
\]

If $K = 1$ and $N = 2$ we can use just one variable $y_i$ and represent its negation as a $(1 - y_i)$, see above slides for details.
ILP Solvers

- LP_SOLVE - free [http://groups.yahoo.com/group/lp_solve/](http://groups.yahoo.com/group/lp_solve/)

YALMIP - Matlab toolbox for modelling ILP problems
Another group of algorithms are cutting planes methods. Its general idea is (similarly to the branch and bound method) to repeat the solution of LP problems. It iteratively adds a constraint that cuts off part of the solution space. The new constraint must fulfill these conditions:

- The solution found by LP becomes infeasible
- All integer solutions feasible in the last step have to remain feasible.

Among the best known methods are Dantzig Cuts, Gomory Cuts and Chvátal-Gomory Cuts.
Gomory Cuts

Algorithm

1. (Initialization) Solve the problem as an LP by a simplex algorithm
2. (Optimality test) If the solution is an integer, the computation ends
3. (Reduction) Add new constraint (Gomory cut) into the simplex table. Optimize the problem by dual LP, then goto 2

\[
\begin{align*}
\text{min} & \quad x_1 + 2x_2 \\
\text{s.t.} & \quad -3x_1 + 4x_2 \leq 6 \\
& \quad 4x_1 + 3x_2 \leq 12 \\
& \quad x_1, x_2 \geq 0, x_1, x_2 \in \mathbb{Z}
\end{align*}
\]
ILP - Conclusion

- NP-hard problem.
- Used to formulate majority of combinatorial problems.
- Often solved by branch and bound method.
B. H. Korte and Jens Vygen.  

James B. Orlin.  
15.053 Optimization Methods in Management Science.  
MIT OpenCourseWare, 2007.