Lecture 8: Linear and integer programming modelling and tools

- Normal and standard forms
- Modelling
- Tools

General Linear Programs

- In a general linear program
  \[
  \min \sum_{i=1}^{n} c_i x_i \quad \text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, \ldots, m
  \]
  \[
  l_j \leq x_j \leq u_j
  \]
  inequalities with \( \leq \) or \( \geq \) can occur in addition to equalities, maximization can be used instead of minimization, and some of the variables can be unrestricted (do not have bounds).

- A general LP can be transformed to an equivalent simpler form, for instance, to a canonical or standard form (introduced below).

- Two forms are equivalent if they have the same set of optimal solutions or are both infeasible or both unbounded.

Standard and Canonical Forms

- An LP is in canonical form when
  - the object function is minimized,
  - all constraints are inequalities of the form \( \sum_{j=1}^{n} a_{ij} x_j \geq b_i \), and
  - all variables are non-negative, i.e., bounded by the constraint \( x_j \geq 0 \).
  that is, the LP is in the form
  \[
  \min \sum_{i=1}^{n} c_i x_i \quad \text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, \ldots, m
  \]
  \[
  x_j \geq 0, \quad j = 1, \ldots, n
  \]

- The standard form is similar but all constraints are of the form \( \sum_{j=1}^{n} a_{ij} x_j = b_i \).
Transformations—cont’d

- An unrestricted variable $x_j$ can be eliminated using a pair of non-negative variables $x^+_j, x^-_j$ by replacing $x_j$ everywhere with $x^+_j - x^-_j$ and imposing $x^+_j \geq 0, x^-_j \geq 0$.
- Non-positivity constraints can be expressed as non-negativity constraints: to express $x_j \leq 0$, replace $x_j$ everywhere with $-y_j$ and impose $y_j \geq 0$.
- These transformations are sometimes needed when modeling if the tool used does not support a feature exploited in the LP model, for example, non-positive or unrestricted variables.

Example—cont’d

- Second: eliminate non-positivity constraints and transform inequalities to equalities with slack and surplus variables to obtain:

$$\begin{align*}
&\min -x^+_2 + x^-_2 - y_1 \\
&\text{s.t.}
&-3y_1 - x^+_2 + x^-_2 - s_1 = 0 \\
&-y_1 + x^+_2 - x^-_2 + s_2 = 6 \\
&-y_1 - s_3 = -2 \\
&y_1 \geq 0 \\
&x^+_2 \geq 0, x^-_2 \geq 0 \\
&s_1 \geq 0, s_2 \geq 0, s_3 \geq 0
\end{align*}$$

Modelling

The diet problem: (a typical problem suitable for linear programming)

- Given $a_{ij}$: amount of the $i$th nutrient in a unit of the $j$th food item
- $r_i$: yearly requirement of the $i$th nutrient
- $c_j$: cost per unit of the $j$th food item
- Build a yearly diet (decide yearly consumption of $n$ food items) such that it satisfies the minimal nutritional requirements for $m$ nutrients and is as inexpensive as possible.
- LP solution: take variables $x_j$ to represent yearly consumption of the $j$th food item

$$\begin{align*}
\text{min } c_1x_1 + \cdots + c_nx_n \\
\text{s.t.}
&a_{1,1}x_1 + \cdots + a_{1,n}x_n \geq r_1 \\
&\vdots \\
&a_{m,1}x_1 + \cdots + a_{m,n}x_n \geq r_m \\
x_1 \geq 0, \ldots, x_n \geq 0
\end{align*}$$
Knapsack
(a typical problem suitable for (0-1) integer programming)

▶ Given: a knapsack of a fixed volume $v$ and $n$ objects, each with a volume $a_i$ and a value $b_i$.
▶ Find a collection of these objects with maximal total value that fits in the knapsack.
▶ IP solution: for each item $i$ take a binary variable $x_i$ to model whether item $i$ is included ($x_i = 1$) or not ($x_i = 0$)

$$
\begin{align*}
\text{max} & \quad b_1 x_1 + \cdots + b_n x_n \\
\text{s.t.} & \quad a_1 x_1 + \cdots + a_n x_n \leq v \\
& \quad 0 \leq x_1 \leq 1, \ldots, 0 \leq x_n \leq 1 \\
& \quad x_j \text{ is integer for all } j \in \{1, \ldots, n\}
\end{align*}
$$

Warehouse Location Problem
(A more complicated 0-1 IP problem)

▶ There is a set of $n$ customers who need to be assigned to one of the $m$ potential warehouse locations.
▶ Customers can only be assigned to an open warehouse, with there being a cost of $c_j$ for opening warehouse $j$.
▶ Once open, a warehouse can serve as many customers as it chooses (with different costs $d_{ij}$ for each customer-warehouse pair).
▶ Choose a set of warehouse locations that minimizes the overall costs of serving all the $n$ customers.
▶ IP solution: introduce binary variables $x_j$ representing the decision to open warehouse $j$ and $y_{ij}$ representing the decision to assign customer $i$ to warehouse $j$.

Warehouse Location Problem—cont’d

▶ Objective function to minimize:

$$
\sum_{j=1}^{m} c_j x_j + \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} y_{ij}
$$

▶ Customers are assigned to exactly one warehouse:

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

▶ Customers can be assigned only to an open warehouse. Two approaches:

▶ If a warehouse is open, it can serve all $n$ customers:

$$
\sum_{j=1}^{n} x_{ij} \leq nx_j \quad \text{for all } j = 1, \ldots, m
$$

▶ If a customer $i$ is assigned to warehouse $j$, it must be open:

$$
y_{ij} \leq x_j \quad \text{for all } j = 1, \ldots, m \text{ and } i = 1, \ldots, n
$$

Expressing Constraints in MIP

▶ Some constraints cannot be represented straightforwardly using linear constraints.
▶ A frequently occurring situation involves combining constraints “disjunctively”.
▶ An implication is a typical example which can sometimes be encoded by introducing an additional variable and a new large constant.

▶ Example. Consider a binary variable $x$ and the constraint “if $x = 1$ then $\sum_{j=1}^{n} x_j \geq b_i$” where each $x_j$ is non-negative. Using a large constant $M$ this can be expressed as follows:

$$
\sum_{j=1}^{n} x_j \geq b_i - M(1 - x)
$$

Notice that here if $x = 1$, then $\sum_{j=1}^{n} x_j \geq b_i$ must hold but if $x = 0$, then $\sum_{j=1}^{n} x_j \geq b_i - M$ imposes no constraint on variables $x_1, \ldots, x_n$ if we choose some $M \geq b_i$. 
Expressing Constraints—cont’d

Example. Consider a disjunctive constraint “x ≥ 5 or y ≤ 6” where x and y are non-negative and y ≤ 1000. This constraint can be encoded by introducing a new binary variable b and constant M as follows:

\[
x + Mb ≥ 5 \\
y - M(1 - b) ≤ 6
\]

Here if we choose \( M ≥ 994 \), then
- if \( b = 0 \), we have constraints \( x ≥ 5 \) and \( y - M ≤ 6 \) where the latter is satisfied by every value of \( y \) (0 ≤ y ≤ 1000) and
- if \( b = 1 \), we have constraints \( x + M ≥ 5 \) and \( y ≤ 6 \) where the former is satisfied by every value of \( x ≥ 0 \).

Unfortunately, these techniques for expressing disjunctions are not general and, e.g., choosing a value for the constant \( M \) is often non-trivial.

Example: Resource Constraints—cont’d

Disjunctive constraints on binary variables can be expressed straightforwardly.

For example, to enforce that the values of variables \( x_{ij} \) are assigned consistently according to their intuitive meaning following constraints need to be added.

- “Either \( i \) occurs before \( j \) or the reverse but not both”
  This is an exclusive-or constraint which can be encoded directly:
  \[
x_{ij} + x_{ji} = 1 \quad (i ≠ j)
  \]

- “If \( i \) occurs before \( j \) and \( j \) before \( k \), then \( i \) occurs before \( k \).”
  This can be seen as a disjunction \( x_{ij} \lor x_{jk} \lor x_{ik} \) of binary variables \( x_{ij}, x_{jk}, x_{ik} \):
  \[
x_{ij} + x_{jk} - x_{ik} ≤ 1
  \]

A potential problem: \( O(n^3) \) constraints are needed where \( n \) is the number of jobs.
Hamiltonian Cycle

- However, the constraints above are not sufficient.
- Consider, for example, a graph with 6 nodes such that variables $x_{1,2}, x_{2,3}, x_{3,1}, x_{4,5}, x_{5,6}, x_{6,4}$ are set to 1 and all others to 0. This solution satisfies the constraints but does not represent a Hamiltonian cycle (two separate cycles).
- Enforcing a single cycle is non-trivial.
- A solution for small graphs is to require that the cycle leaves every proper subset of the nodes, that is, to have a constraint

$$\sum_{(i,j) \in E, i \in s, j \notin s} x_{i,j} \geq 1$$

for every proper subset $s$ of the nodes $V$.
- In the example above, this constraint would be violated for $s = \{1, 2, 3\}$.
- A potential problem for bigger graphs: $O(2^n)$ constraints needed where $n$ is the number of nodes.

Hamiltonian Cycle–cont’d

- For condition ‘if $p_i = n$, then $p_j \geq 2$” we can use the technique for implications:

$$p_j \geq 2 - (n - p_i)$$

Notice that
- if $n = p_i$, then we get $p_j \geq 2$ and
- if $n > p_i$, then the constraint is satisfied for all value of $p_j$ ($1 \leq p_j \leq n$).
- To complete the encoding in IP we need to express disequality ($\neq$)

Expressing Disequality

- For expressing an arbitrary disequality $x \neq y$ of two bounded integer variables $x$ and $y$ we reformulate the disequality as “$x > y$ or $y > x$” or equivalently “$x - y \geq 1$ or $x - y \leq -1$”.
- Now we can model the disjunction using a binary variable $b$ and a large constant $M$ and the constraints

$$x - y + Mb \geq 1$$
$$x - y - M(1 - b) \leq -1$$

Notice that
- if $b = 0$, then we get $x - y \geq 1, x - y \leq M - 1$ and
- if $b = 1$, then we get $x - y + M \geq 1, x - y \leq -1$

where the constraints involving $M$ are satisfied by all values of $x, y$ given large enough $M$ w.r.t. to the bounds on the values of $x, y$.
MIP Tools

- There are several efficient commercial MIP solvers.
- Also public domain systems exist but these are not as efficient as the commercial ones.
- See, for example, http://www-unix.mcs.anl.gov/otc/Guide/faq/linear-programming-faq.html for MIP systems and other information and frequently asked questions.

MIP Solvers

- A MIP solver can typically take its input via an input file and an API.
- There is a number of widely used input formats (like MPS) and tool specific formats (lp_solve, CPLEX, LINDO, GNU MathProg, LPFML XML, ...).
- MIP solvers do not require the input program to be in a standard form but typically quite general MIPs are allowed, that is
  - both minimization and maximization are supported and
  - operators “=”, “≤”, and “≥” can all be used.

lp_solve

- In the third home assignment a public domain MIP solver, lp_solve is employed.
- See the newest version (5.5) at http://lpsolve.sourceforge.net/5.5/
- lp_solve accepts a number of input formats
  
  **Example.** lp_solve native format
  
  \[
  \begin{align*}
  \text{min: } & x_1 + x_2 + 3x_3; \\
  & x_1 - x_2 \leq 1; \\
  & 2x_2 - 2.5x_3 \geq 1; \\
  & -7x_3 + x_2 = 3; \\
  \end{align*}
  \]
  
  > lp_solve < example
  
  Value of objective function: 3
  
  \begin{align*}
  x_1 & = 0 \\
  x_2 & = 3 \\
  x_3 & = 0 \\
  \end{align*}