# Lecture 8: Linear and integer programming modelling and tools

- Normal and standard forms
- Modelling
- Tools





## Example. Linear program

- $\min x_2$ s.t.

$$\begin{array}{rcrl} x_1 & & & \geq 2 \\ 3x_1 & - & x_2 & \geq 0 \\ x_1 & + & x_2 & \geq 6 \\ -x_1 & + & 2x_2 & \geq 0 \end{array}$$

- Optimal solution is $(4,2)$ of cost 2.
- If we were maximizing, the linear program would be unbounded.
- If we reversed some of the inequalities, the resulting LP
  $\min x_2$ s.t.

$$\begin{array}{rcrl} x_1 & & & \leq 2 \\ 3x_1 & - & x_2 & \geq 0 \\ x_1 & + & x_2 & \geq 6 \\ -x_1 & + & 2x_2 & \leq 0 \end{array}$$

  would be infeasible.





## General Linear Programs

- A general linear program

$$\begin{array}{rl} & \min cx \\ s.t. & Ax = b \\ & l \leq x \leq u \end{array}$$

  where the "$=$" could be also $\leq$ or $\geq$ and min could also be max
- can be transform to equivalent simpler forms, for instance, 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

- Canonical form
$$\begin{array}{rl} & \min cx \\ s.t. & Ax \geq b \\ & x \geq 0 \end{array}$$
- Standard form
$$\begin{array}{rl} & \min cx \\ s.t. & Ax = b \\ & x \geq 0 \end{array}$$
- Transformations to these forms
  - From maximization to minimization: $\max cx \Leftrightarrow \min -cx$
  - From equality to inequality: $ax = b \Leftrightarrow \begin{cases} ax \geq b \\ -ax \geq -b \end{cases}$
  - From inequality to equality: $ax \leq b \Leftrightarrow ax + s = b, s \geq 0$
  - From non-positivity to non-negativity: to express $x_j \leq 0$, replace $x_j$ everywhere with $-y_j$ and impose $y_j \geq 0$.
  - From unrestricted variable to non-negative: if $x_j$ is unrestricted in sign, replace it everywhere with $x_j^+ - x_j^-$ and impose $x_j^+ \geq 0, x_j^- \geq 0$.

# Modelling

## The diet problem:

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

$$\begin{aligned}
&\min c_1 x_1 + \cdots c_n x_n \text{ s.t.}\\
&a_{1,1}x_1 + \cdots + a_{1,n}x_n \geq r_1\\
&\vdots\\
&a_{m,1}x_1 + \cdots + a_{m,n}x_n \geq r_m\\
&x_1 \geq 0, \ldots, x_n \geq 0
\end{aligned}$$





# Knapsack

- 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: take variables $x_i$ to model whether item $i$ is included ($x_i = 1$) or not ($x_i = 0$)

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





# Warehouse Location 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 $d_{i,j}$ 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 $j$
  $y_{i,j}$ representing the decision to assign customer $i$ to warehouse $j$





# Warehouse Location Problem—cont'd

- Objective function to minimize:

$$\sum_j c_j x_j + \sum_i \sum_j d_{i,j} y_{i,j}$$

- Customers are assigned to exactly one warehouse:

$$\sum_j y_{i,j} = 1 \quad \text{for all } i = 1, \ldots, n$$

- Customers can be assigned only to an open warehouse. Two approaches:
  - If a warehouse is open, it can serve all $n$ customers:

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

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

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

## Resource Constraints

- In a scheduling application typically following types of variables are used:
  $s_j$: starting time for job $j$
  $x_{ij}$: binary variable representing whether job $i$ occurs before job $j$
- Consider now the constraint:
  "If job 2 occurs after job 1, then it starts at least 10 time units after the end of job 1"
- This can be represented by introducing a suitably large constant $M$ ($d_1$ is the duration of job 1):

$$s_2 \geq s_1 + d_1 + 10 - M(1 - x_{12})$$

  - If $x_{12} = 1$: we get $s_2 \geq s_1 + d_1 + 10$ as required.
  - If $x_{12} = 0$: we get $s_2 \geq s_1 + d_1 + 10 - M$, which implies no restriction on $s_2$ if $M$ is sufficiently large.





## Resource Constraints—cont'd

- To enforce that the values of variables $x_{ij}$ are assigned consistently according to their intuitive meaning further constraints need to be added.
- Either $i$ occurs before $j$ or the reverse but not both:

$$x_{ij} + x_{ji} = 1 \quad (i \neq j)$$

- If $i$ occurs before $j$ and $j$ before $k$, then $i$ occurs before $k$.

$$x_{ij} + x_{jk} - x_{ik} \leq 1$$

  A potential problem: $\mathrm{O}(n^3)$ constraints are needed where $n$ is the number of jobs.





## Routing Constraints

- Consider the Hamiltonian cycle problem:
  INSTANCE: A graph $(V, E)$.
  QUESTION: Is there a simple cycle visiting all nodes of the graph?
- Introduce a binary variable $x_{i,j}$ for each edge $(i, j) \in E$ indicating whether the edge is included in the cycle ($x_{i,j} = 1$) or not ($x_{i,j} = 0$).
- Constraints:
  - The cycle leaves each node $i$ through exactly one edge:

$$\sum_j x_{i,j} = 1$$

  - The cycle enters each node $i$ through exactly one edge:

$$\sum_j x_{j,i} = 1$$





## 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: $\mathrm{O}(2^n)$ constraints needed where $n$ is the number of nodes.

## Hamiltonian Cycle–cont'd

- Another approach, where the number of constraints remains polynomial, is to introduce an integer variable $p_i$ for each node $i = 1, \ldots, n$ in the graph to represent the position of the node $i$ in the cycle, that is, $p_i = k$ means that node $i$ is $k$th node visited in the cycle.
- In order to enforce a single cycle we need to enforce the following conditions.
- Each $p_i$ has a value in $\{1, \ldots, n\}$:

$$1 \leq p_i \leq n$$

- This value is unique, that is, for all pairs of nodes $i$ and $j$ with $i \neq j$, $p_j \neq p_i$ holds.
- For all pairs of nodes $i$ and $j$ with $i \neq j$ such that $(i, j) \notin E$, node $j$ cannot be the next node after $i$, that is,
  - $p_j \neq p_i + 1$ holds and
  - if $p_i = n$, then $p_j \neq 1$.





## Expressing Disequality

- In order to obtain a MIP we need to be able to express disequality ($\neq$) constraints.
- Because for every $p_i$, $1 \leq p_i \leq n$ holds, condition "if $p_i = n$, then $p_j \neq 1$" can be expressed as

$$1 - (n - p_i) \leq p_j - 1$$

- For expressing an arbitrary disequality $x \neq y$, we introduce a binary integer variable $b$ and a large constant $M$ and the constraints

$$x - y + Mb \geq 1$$
$$x - y + Mb \leq M - 1$$

Notice that
  - if $b = 0$, then we get $x - y \geq 1, x - y \leq M - 1$ which can be satisfied only if $x > y$ and
  - if $b = 1$, then we get $x - y + M \geq 1, x - y \leq -1$ which can be satisfied only if $x < y$.





## MIP Tools

- There are several efficient commercial MIP solvers.
- Also public domain systems exists 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 a number of wide 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 "$=$", "$\leq$", and "$\geq$" 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

```
min: x1 + x2 + 3x3;
     x1 - x2 <= 1;
     2x2 - 2.5x3  >= 1;
     -7x3 + x2 = 3;
> lp_solve < example
Value of objective function:                3
x1                         0
x2                         3
x3                         0
```



T–79.4201 Search Problems and Algorithms

## Instructions for home assignment round three

- The goal is to solve two optimization problems by encoding them as MIP problems which are then solved lp_solve.
- The task is to write a (Java) program that takes as input an instance of the problem, generates a MIP encoding, runs lp_solve on the encoding, and transforms the output of lp_solve to the required format.
- Further information can be found on the home page of the course (the problems, general instructions, lp_solve binaries, format for MIP programs, Java libraries to translate the format to lp_solve native format, results back to the required format, reading input, …).

I.N. & P.O. Spring 2006