MILP

LP: $\max \; \mathbf{c}'\mathbf{x}$

s.t. $\mathbf{A}\mathbf{x} \le \mathbf{b}$

$\mathbf{x} \ge 0$

MILP: some x_i integer

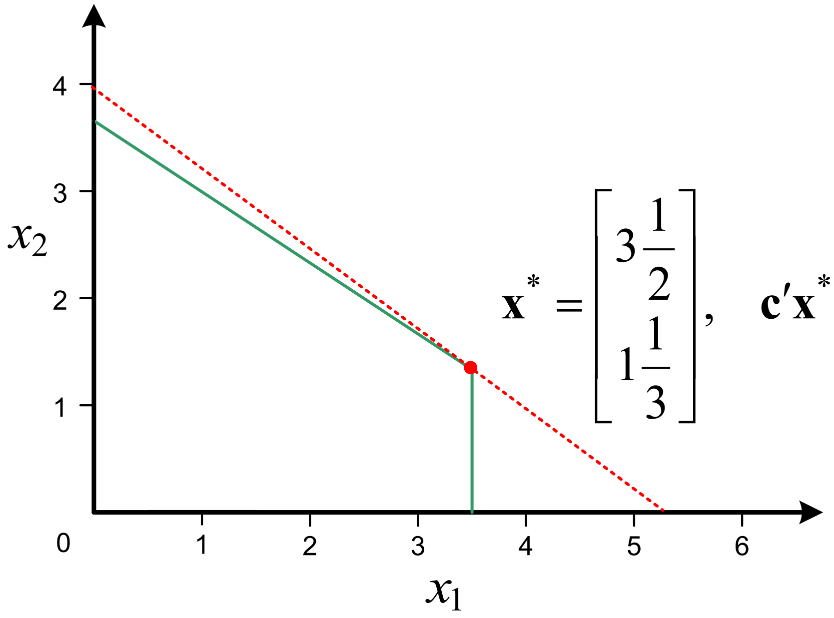
ILP: $\mathbf{x}$ integer

BLP: $\mathbf{x} \in \{0,1\}$

$$\begin{aligned} \max \quad & 6x_1 + 8x_2 \\ \text{s.t.} \quad & 2x_1 + 3x_2 \le 11 \\ & 2x_1 \le 7 \\ & x_1, x_2 \ge 0 \end{aligned}$$

$$\mathbf{c} = \begin{bmatrix} 6 & 8 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 2 & 3 \\ 2 & 0 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 11 \\ 7 \end{bmatrix}$$



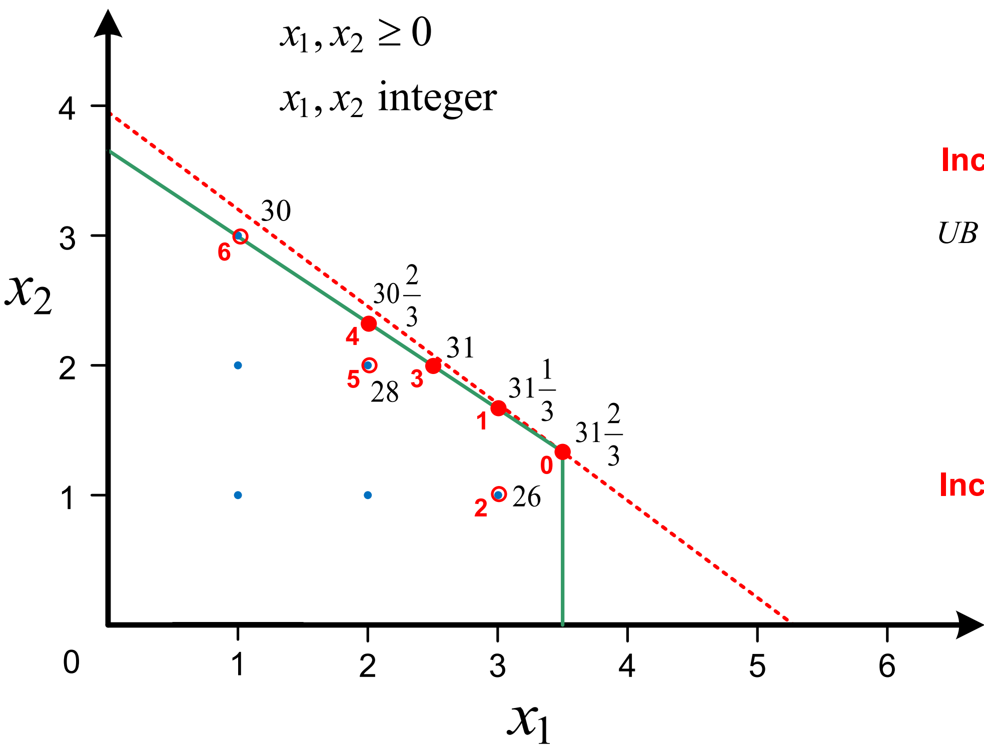
$$\mathbf{x}^* = \begin{bmatrix} 3\frac{1}{2} \\ 1\frac{1}{3} \end{bmatrix}, \quad \mathbf{c}'\mathbf{x}^* = 31\frac{2}{3}$$

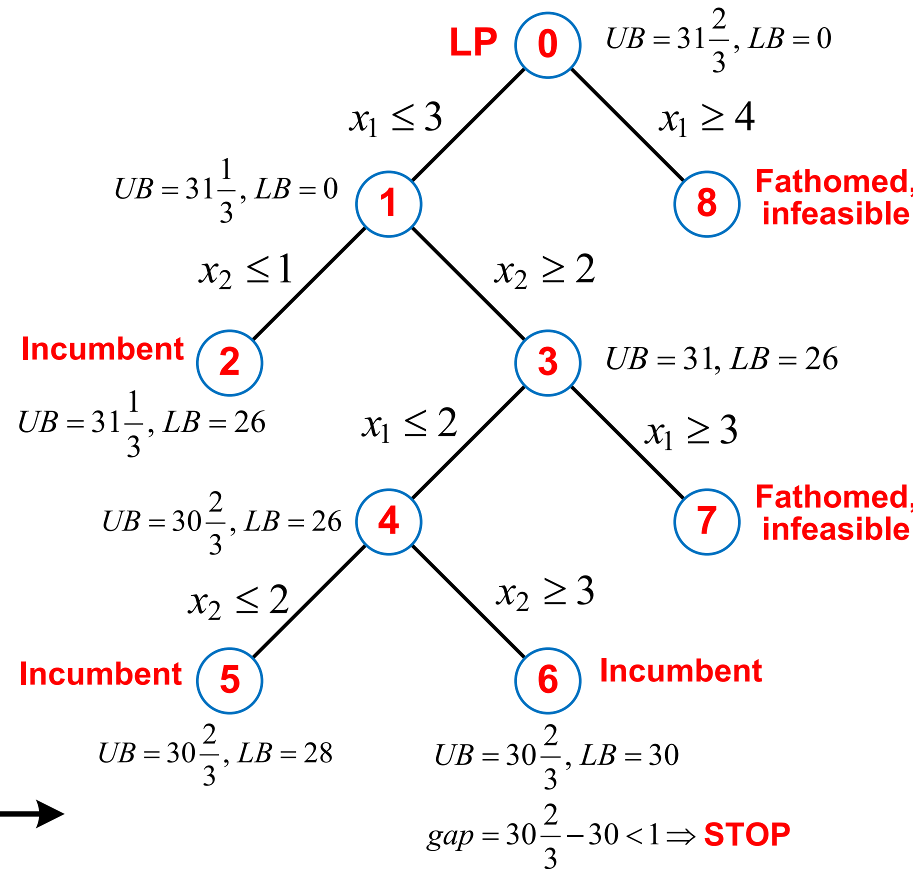
Branch and Bound

$$\begin{aligned} \max \quad & 6x_1 + 8x_2 \\ \text{s.t.} \quad & 2x_1 + 3x_2 \le 11 \\ & 2x_1 \le 7 \\ & x_1, x_2 \ge 0 \\ & x_1, x_2 \text{ integer} \end{aligned}$$

$$\mathbf{c} = \begin{bmatrix} 6 & 8 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} 2 & 3 \\ 2 & 0 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 11 \\ 7 \end{bmatrix}$$





MILP Solvers

LP: $\max\ \mathbf{c}'\mathbf{x}$

$$\text{s.t.}\quad \mathbf{A}\mathbf{x} \le \mathbf{b}$$
$$\mathbf{x} \ge 0$$

MILP: some x_i integer

ILP: $\mathbf{x}$ integer

BLP: $\mathbf{x} \in \{0,1\}$

intlinprog: $\min\ \mathbf{c}\quad (\max\ -\mathbf{c})$

$$\text{s.t.}\quad \mathbf{A}_{\text{lt}} \le \mathbf{b}_{\text{lt}}$$
$$\mathbf{A}_{\text{eq}} = \mathbf{b}_{\text{eq}}$$
$$LB \le \mathbf{x} \le UB$$

integer variable indices

cplex: $\mathbf{c}$ (sense *min* or *max*)

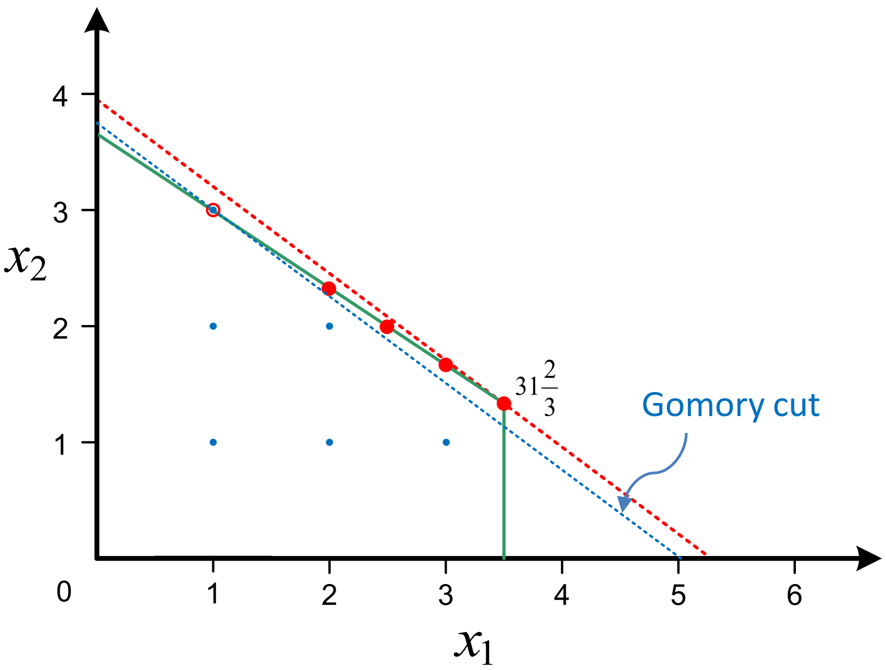
$$\text{s.t.}\quad lhs \le \mathbf{A} \le rhs$$
$$LB \le \mathbf{x} \le UB$$
$$\text{variable:} \begin{cases} C & \text{continuous} \\ B & \text{binary} \\ I & \text{general integer} \end{cases}$$

lhs	rhs		
$-\infty$	$\mathbf{b}$	$\Rightarrow$	$\le$
$\mathbf{b}$	∞	$\Rightarrow$	$\ge$
$\mathbf{b}$	$\mathbf{b}$	$\Rightarrow$	$=$

gurobi: $\mathbf{c}$ (modelsense *min* or *max*)

$$\text{s.t.}\quad \mathbf{A}\begin{Bmatrix} < \\ = \\ > \end{Bmatrix}\mathbf{b}$$
$$LB \le \mathbf{x} \le UB$$
$$\text{variable:} \begin{cases} C & \text{continuous} \\ B & \text{binary} \\ I & \text{general integer} \end{cases}$$

MILP Solvers



- **Presolve**: eliminate variables

$$2x_1 + 2x_2 \leq 1,\ x_1, x_2 \geq 0 \text{ and integer}$$

$$\Rightarrow x_1 = x_2 = 0$$

- **Cutting planes**: keeps all integer solutions and cuts off LP solutions (Gomory cut)
- **Heuristics**: find good initial incumbent solution (Hybrid UFL)
- **Parallel**: use separate cores to solve nodes in B&B tree
- **Speedup** from 1990-2014:
  - 320,000 × computer speed
  - 580,000 × algorithm improvements

  $\Rightarrow$ 10 days of 24/7 processing $\rightarrow$ 1 sec

- Cplex (IBM, comm first solver)
- Gurobi (dev Robert Bixby)
- Xpress (used by LLamasoft)
- SAS/OR (part of SAS system)
- Symphony (open source)
- Matlab's `intlinprog`

MILP Formulation of UFL

$$\min \quad \sum_{i \in N} k_i y_i + \sum_{i \in N} \sum_{j \in M} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{i \in N} x_{ij} = 1, \qquad j \in M$$

$$\sout{m y_i \geq \sum_{j \in M} x_{ij}, \qquad i \in N}$$

$$0 \leq x_{ij} \leq 1, \qquad i \in N, j \in M$$

$$y_i \in \{0,1\}, \qquad i \in N$$

```
%% UFL MILP Matlab code, given k and C
mp.addobj('min',k,C)
for j = M
   mp.addcstr(0,{':',j},'=',1)
end
for i = N
   mp.addcstr({m,{i}},'>=',{i,':'})
end
mp.addub(1,1)
mp.addctype('B','C')
```

$$y_i \geq x_{ij}, \qquad i \in N, j \in M$$

where

k_i = fixed cost of NF at site $i \in N = \{1,\ldots,n\}$

c_{ij} = variable cost from i to serve EF $j \in M = \{1,\ldots,m\}$

$$y_i = \begin{cases} 1, & \text{if NF established at site } i \\ 0, & \text{otherwise} \end{cases}$$

x_{ij} = fraction of EF j demand served from NF at site i.

Capacitated Facility Location (CFL)

$$\min \quad \sum_{i\in N} k_i y_i + \sum_{i\in N}\sum_{j\in M} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{i\in N} x_{ij} = 1, \quad j \in M$$

$$K_i y_i \geq \sum_{j\in M} f_j x_{ij}, \ i \in N$$

$$0 \leq x_{ij} \leq 1, \quad i \in N, j \in M$$

$$y_i \in \{0,1\}, \quad i \in N$$

where

$$k_i = \text{fixed cost of NF at site } i \in N = \{1,\ldots,n\}$$

$$c_{ij} = \text{variable cost from } i \text{ to serve EF } j \in M = \{1,\ldots,m\}$$

$$K_i = \text{capacity of NF at site } i \in N = \{1,\ldots,n\}$$

$$f_j = \text{demand EF } j \in M = \{1,\ldots,m\}$$

$$y_i = \begin{cases} 1, & \text{if NF established at site } i \\ 0, & \text{otherwise} \end{cases}$$

$$x_{ij} = \text{fraction of EF } j \text{ demand served from NF at site } i.$$

- CFL does not have simple and effective heuristics, unlike UFL
- Other types of constraints:
  - Fix NF *i* at site *j*: set *LB* and *UB* of x_{ij} to 1
  - Convert UFL to p-Median: set all *k* to 0 and add constraint sum{y_i} = *p*

Matlog's Milp

- Executing `mp = Milp` creates a *Milp* object that can be used to define a MILP model that is then passed to a Solver
  - Similar syntax to math notation for MILP
  - *AMPL* and *OPL* algebraic modeling languages provide similar capabilities, but *Milp* integrated into MATLAB

Milp

```
Milp Mixed-integer linear programming model.
 This class stores Milp models and provides methods to create the models
 and format solutions for output.
 Milp Properties:
    Model        Milp model (same structure as Cplex model).
 Milp Methods:
    Milp         Constructor for Milp objects.
    addobj       Add variable cost arrays to objective function.
    addcstr      Add constraint to model.
    addlb        Add lower bounds for each variable array.
    addub        Add upper bounds for each variable array.
    addctype     Specify type of each variable array.
    namesolution Convert solution to named field arrays.
    dispmodel    Display matrix view of model.
    lp2milp      Convert LP model to Milp model.
    milp2lp      Convert Milp model to LINPROG inputs.
    milp2ilp     Convert Milp model to INTLINPROG inputs.
    milp2gb      Convert Milp model to Gurobi input structure.
```

Illustrating Milp syntax

```
c = [1:4], C = reshape(5:10,2,3)
mp = Milp('Example');
mp.addobj('min',c,C)

mp.addcstr(0,1,'=',100)
mp.addcstr(c,-C,'>=',0)
mp.addcstr(c,'>=',C)
mp.addcstr([c; 2*c],repmat(C(:)',2,1),'<=',[400 500])
mp.addcstr({3},{2,2},'<=',600)
mp.addcstr({2,{3}},{3*3,{2,2}},'<=',700)
mp.addcstr({[2 3],{[3 4]}},{4,{2,':'}},'=',800)
mp.addcstr(0,{C(:,[2 3]),{':',[2 3]}},'>=',900)

mp.addlb(-10,0)
mp.addub(10,Inf)
mp.addctype('B','C')
mp.dispmodel
```

$$\text{addobj}\left(\text{'min'}, \mathbf{k}, \mathbf{C}\right)$$

$$\text{addobj}\left(\text{'min'}, \mathbf{y}, \mathbf{X}\right)$$

$$\text{addcstr}\left(my_3, nx_{2,4}, \text{'='}, 7\right)$$

$$\text{addcstr}\left(\{m,\{3\}\}, \{n,\{2,4\}\}, \text{'='}, 7\right)$$

$$\text{addcstr}\left(0\mathbf{y}, 1x_{2,4}, \text{'='}, 7\right)$$

$$\text{addcstr}\left(0, \{2,4\}, \text{'='}, 7\right)$$

```
% c =   1     2     3     4
% C =
%       5     7     9
%       6     8    10
%
% Example:  lhs  B    B    B    B    C    C    C    C    C    C  rhs
% -------:------------------------------------------------------------
%    Min:        1    2    3    4    5    6    7    8    9   10
%      1:  100   0    0    0    0    1    1    1    1    1    1  100
%      2:    0   1    2    3    4   -5   -6   -7   -8   -9  -10  Inf
%      3:    0   1    2    3    4   -5   -6   -7   -8   -9  -10  Inf
%      4: -Inf   1    2    3    4    5    6    7    8    9   10  400
%      5: -Inf   2    4    6    8    5    6    7    8    9   10  500
%      6: -Inf   0    0    1    0    0    0    0    1    0    0  600
%      7: -Inf   0    0    2    0    0    0    0    9    0    0  700
%      8:  800   0    0    2    3    0    4    0    4    0    4  800
%      9:  900   0    0    0    0    0    0    7    8    9   10  Inf
%     lb:      -10  -10  -10  -10    0    0    0    0    0    0
%     ub:       10   10   10   10  Inf  Inf  Inf  Inf  Inf  Inf
```

Ex 10: UFL MILP

```
k = [8      8     10      8      9      8];
C = [0      3      7     10      6      4
     3      0      4      7      6      7
     7      4      0      3      6      8
    10      7      3      0      7      8
     6      6      6      7      0      2
     4      7      8      8      2      0];
mp = Milp('UFL');
mp.addobj('min',k,C)
[n m] = size(C);
for j = 1:m
   mp.addcstr(0,{':',j},'=',1)
end
for i = 1:n
   mp.addcstr({m,{i}},'>=',{i,':'})  % Weak formulation
end
mp.addub(Inf,1)
mp.addctype('B','C')
[x,TC,nevals,XFlg] = milplog(mp); TC,nevals,XFlg
x = mp.namesolution(x), xC = x.C
TC = k*x.k' + sum(sum(C.*xC))
```

$$
\begin{aligned}
\min \quad & \sum_{i \in N} k_i y_i + \sum_{i \in N} \sum_{j \in M} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{i \in N} x_{ij} = 1, && j \in M \\
& m y_i \geq \sum_{j \in M} x_{ij}, && i \in N \\
& 0 \leq x_{ij} \leq 1, && i \in N, j \in M \\
& y_i \in \{0,1\}, && i \in N
\end{aligned}
$$

```
TC =
   31.0000
nevals =
    67
XFlg =
     1
x =
  struct with fields:

    k: [0 0 1 0 0 1]
    C: [6×6 double]
xC =
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     1     1     1     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     1     0     0     0     1     1
TC =
    31
```

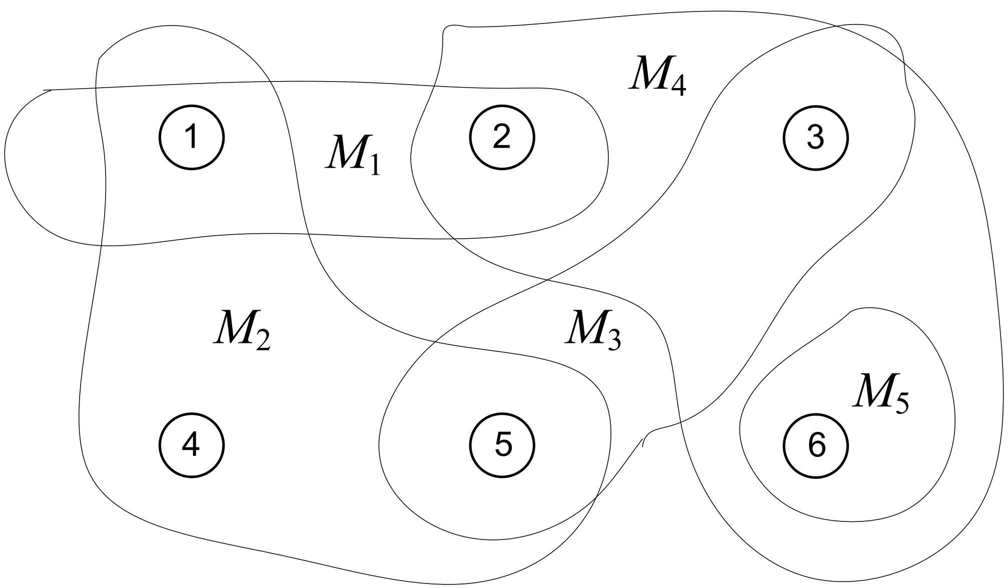
(Weighted) Set Covering

$M = \{1,\ldots,m\}$, objects to be covered

$M_i \subseteq M$, $i \in N = \{1,\ldots,n\}$, subsets of M

$c_i =$ cost of using M_i in cover

$I^* = \arg\min_I \left\{ \sum_{i\in I} c_i : \bigcup_{i\in I} M_i = M \right\}$, min cost covering of M



$M = \{1,\ldots,6\}$

$i \in N = \{1,\ldots,5\}$

$M_1 = \{1,2\}, M_2 = \{1,4,5\}, M_3 = \{3,5\}$

$M_4 = \{2,3,6\}, M_5 = \{6\}$

$c_i = 1$, for all $i \in N$

$$I^* = \arg\min_I \left\{ \sum_{i\in I} c_i : \bigcup_{i\in I} M_i = M \right\}$$

$$= \{2,4\}$$

$$\sum_{i\in I^*} c_i = 2$$

(Weighted) Set Covering

$M = \{1, \ldots, m\}$, objects to be covered

$M_i \subseteq M, i \in N = \{1, \ldots, n\}$, subsets of M

$c_i =$ cost of using M_i in cover

$I^* = \arg\min_I \left\{ \sum_{i \in I} c_i : \bigcup_{i \in I} M_i = M \right\}$, min cost covering of M

$$\begin{aligned} \min \quad & \sum_{i \in N} c_i x_i \\ \text{s.t.} \quad & \sum_{i \in N} a_{ji} x_i \geq 1, \qquad j \in M \\ & x_i \in \{0,1\}, \qquad i \in N \end{aligned}$$

where

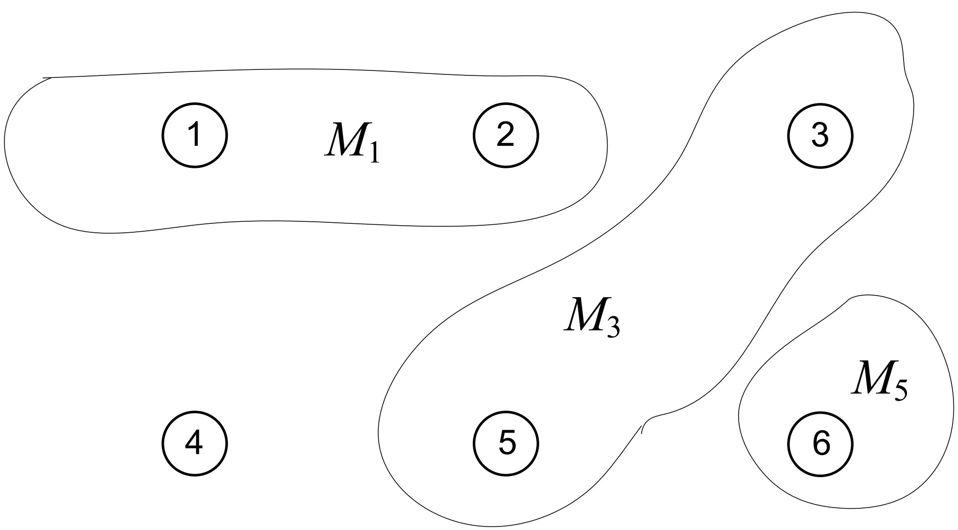
$$x_i = \begin{cases} 1, & \text{if } M_i \text{ is in cover} \\ 0, & \text{otherwise} \end{cases}$$

$$a_{ji} = \begin{cases} 1, & \text{if } j \in M_i \\ 0, & \text{otherwise.} \end{cases}$$

```
%% Set Covering BLP Matlab code,
%  given c and A
mp = Milp('Set Cover')
mp.addobj('min',c)
mp.addcstr(A,'>=',1)
mp.addctype('B')
```

Set Packing

- Maximize the number of mutually disjoint sets
  - Dual of Set Covering problem
  - Not all objects required in a packing
  - Limited logistics engineering application (c.f. bin packing)



$$
\begin{aligned}
\max \quad & \sum_{i \in N} x_i \\
\text{s.t.} \quad & \sum_{i \in N} a_{ji} x_i \le 1, \quad && j \in M \\
& x_i \in \{0,1\}, \quad && i \in N
\end{aligned}
$$

Bin Packing

$$M = \{1, \ldots, m\}, \quad \text{objects to be packed}$$

$$v_j = \text{volume of object } j$$

$$V = \text{volume of each bin } B_i \left(\max v_j \le V \right)$$

$$B^* = \arg\min_B \left\{ |B| : \sum_{j \in B_i} v_j \le V, \bigcup_{B_i \in B} B_i = M \right\}, \quad \text{min packing of } M$$

$$\begin{aligned}
\min \quad & \sum_{i \in M} y_i \\
\text{s.t.} \quad & V y_i \ge \sum_{j \in M} v_j x_{ij}, && i \in M \\
& \sum_{i \in M} x_{ij} = 1, && j \in M \\
& y_i \in \{0,1\}, && i \in M \\
& x_{ij} \in \{0,1\}, && i \in M, j \in M
\end{aligned}$$

where

$$y_i = \begin{cases} 1, & \text{if bin } B_i \text{ is used in packing} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{if object } j \text{ packed into bin } B_i \\ 0, & \text{otherwise.} \end{cases}$$

```
%% Bin Packing BLP Matlab code,
%  given v and V
mp = Milp('Bin Packing')
mp.addobj('min',ones(1,m),zeros(m))
for i = M
   mp.addcstr({V,{i}},'>=',{v,{i,':'}})
end
for j = M
   mp.addcstr(0,{':',j},'=',1)
end
mp.addctype('B','B')
```