

IE426 – Optimization models and applications

Fall 2008 – Homework #2 – Solutions

1 Modeling (3 pts.)

In an automated library, a robotic bookcase of length l has n shelves (there are n shelves of length l each, on top of each other). The total height of the bookcase is H . The height of each shelf can be changed in order to accommodate books of variable height, given that the total height is at most H . We have a set B of $|B| = m$ books, each with a height h_i , a weight w_i , and a thickness t_i , for each $i \in B$. We have to place all books on the bookcase in such a way that:

- Each shelf holds a maximum weight of W ;
- All books of a shelf must not exceed the length l of the shelf;
- Their height must be below the height of the shelf.

Formulate a Integer Linear Programming model for this problem. The manager of the library wants to know to which shelf each book goes and what is the height of each shelf. Minimize the total weight of the top shelf.

Solution. Consider a set $S = \{1, 2, \dots, n\}$ of shelves, indicating n as the top shelf. Variables:

- binary variables x_{bs} , for each book $b \in B$ and shelf $s \in S$. It is 1 if book b is in shelf s , 0 otherwise.
- Continuous variable y_s , equal to the height of shelf s .

Constraints:

- y_s is no less than the height of any book in shelf s ,

$$y_s \geq h_b x_{bs} \quad \forall b \in B, s \in S;$$

- the total thickness of the books in a shelf cannot be more than the length l of the shelf,

$$\sum_{b \in B} t_b x_{bs} \leq l \quad \forall s \in S;$$

- the total height must be below H ,

$$\sum_{b \in B} y_s \leq H;$$

- the total weight of a shelf must be below W ,

$$\sum_{b \in B} w_b x_{bs} \leq W;$$

- every book must be assigned to a shelf,

$$\sum_{s \in S} x_{bs} = 1 \quad \forall b \in B;$$

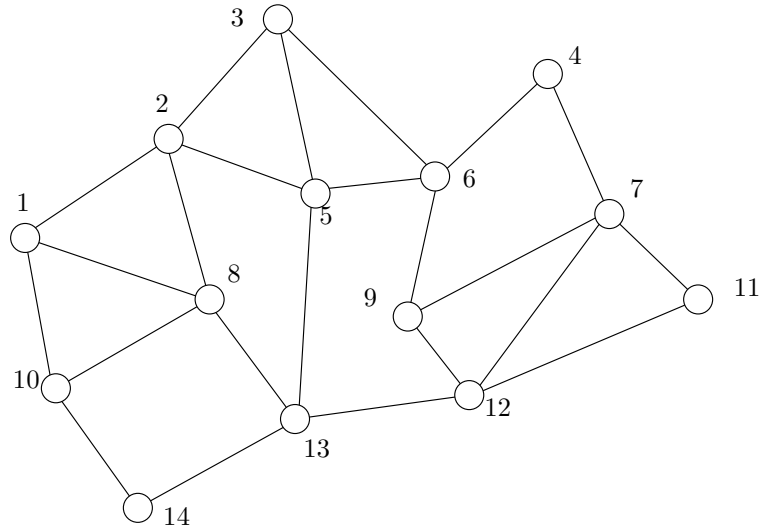
- Domains of the variables,

$$x_{bs} \in \{0, 1\} \quad \forall b \in B, s \in S; \quad y_s \geq 0 \quad \forall s \in S.$$

Objective function: the total weight of the top shelf, $\sum_{b \in B} w_b x_{bn}$.

2 Network flow (4 pts.)

Consider the oil pipeline defined by the following graph $G = (V, E)$:



The channels of the pipeline are traversed by oil in both directions. In order to guarantee a level of safety on any sequence of pipes traversed by the oil from any node i to any node j , it is required that there be, in at least every second node of any path the oil might follow, a special safety equipment that obviously costs a lot of money. This is **equivalent** to guaranteeing, for any pipe in the pipeline, that the safety equipment be present in at least one of its end nodes. For instance, pipe $\{2, 8\}$ is safe if and only if the safety equipment is installed at 2, at 8, or at both.

The company wants to minimize the number of nodes at which this safety equipment is installed, while making sure that the safety requirement above is met. Formulate a Mixed Integer Linear Programming model for this problem, assuming no information is known on G (that is, give a general model, with indices and \sum 's, rather than a model for this specific example).

Translate this model into AMPL, write the `.mod` file and the `.dat` file and submit it to the NEOS server using a MILP solver of your choice (e.g. MINTO). Include in the homework a copy of the output you get from NEOS – there should be no modeling errors. Remember, you need to also upload a `.run` file which should simply contain lines

```
solve;  
display x, y, mycost;
```

where x , y and $mycost$ are simply names for variables and objective function.

Solution. This is a very easy Edge Covering problem (we “cover” all nodes by means of edges). Define binary variable x_i for all node $i \in V$, which is 1 if i is included in the solution, 0 otherwise. We just need to make sure that, for each edge $\{i, j\}$, either i or j , or both, are included in the solution:

$$x_i + x_j \geq 1 \quad \forall \{i, j\} \in E.$$

The only other constraint is

$$x_i \in \{0, 1\} \forall i \in V.$$

The objective function, to be minimized, is the number of nodes: $\sum_{i \in V} x_i$.

Model file:

```

param n;

set V=1..n;
set E within {i in V, j in V: i<j};

var x {V} binary;

minimize totnodes: sum {i in V} x [i];

cover {(i,j) in E}: x [i] + x [j] >= 1;

Data file:

param n=14;

set E =
(1,2) (1,8) (1,10) (2,3) (2,5) (2,8) (3,5) (3,6) (4,6) (4,7) (5,6) (5,13) (6,9)
(7,9) (7,11) (7,12) (8,10) (8,13) (9,12) (10,14) (11,12) (12,13) (13,14);

```

The output from NEOS should look like this:

```

----- Begin Solver Output -----
Executing /home/neos/neos-solvers/minto-ampl/minto-ampl-driver.py
File exists
You are using the solver mintoamp.
Executing AMPL.
processing data.
processing commands.

14 variables, all binary
23 constraints, all linear; 46 nonzeros
1 linear objective; 14 nonzeros.

MINTO (AMPL) v3.0: MINTO arg string (from AMPL options): ''

MINTO, a Mixed INTeger Optimizer -- version 3.1.0 (LINUX/COIN-OSI)
Copyright (C) 1992-2004 -- M.W.P. Savelsbergh

MINTO: Solving problem amplprob
MINTO: Problem statistics:

    Number of constraints: 23
    Number of variables:  14 (0)
    Number of nonzero's:  46

    Number of continuous variables: 0
    Number of binary variables:    14
    Number of integer variables:   0

MINTO: Row structure analysis (after preprocessing):
    Number of constraints of type BINSUM1UB:  23

MINTO control parameters:
    Objective sense      : minimization
    Output level        : 1
    Maximum cpu time    : 1000000
    Maximum #nodes      : 1000000

MINTO system function activity levels:
    Bound improvement   : active

```

Branching type : 3
Node selection type : 5
Preprocessing level : 2
Primal heuristic : active
Clique cuts : active
Implication cuts : active
Knapsack covers : active
GUB covers : active
Flow covers : active
Row management : active
Restarts : active
Force branching : 1
Advanced basis : active
Names mode level : 0

MINTO: Updating primal (MINTO heuristic)
Value: -8.00 Elapsed time: 0.00 Node: 1

MINTO: Value of solution: 8.000000
MINTO: Non-zero variables in the solution:
xopt[0] = 1.000000 ()
xopt[1] = 1.000000 ()
xopt[2] = 1.000000 ()
xopt[5] = 1.000000 ()
xopt[6] = 1.000000 ()
xopt[9] = 1.000000 ()
xopt[11] = 1.000000 ()
xopt[12] = 1.000000 ()
zopt = 8.000000
zroot = 7.500000
zinit = 7.000000

MINTO: Branch and bound statistics

Maximum number of unevaluated nodes	=	1
Number of evaluated nodes	=	1
Depth of the tree	=	0
Number of linear programs solved	=	2
Maximum number of rows in active LP	=	28 (23)

MINTO: Cut generation statistics

Number of generated clique inequalities	=	5
Number of generated implication inequalities	=	0
Number of generated lifted knapsack covers	=	0
Knapsack covers	=	0
Knapsack covers with GUBs	=	0
Surrogate knapsack covers	=	0
Number of generated generalized flow covers	=	0

APPL: Timing statistics (in seconds)

appl_bounds:	0.00
appl_constraints:	0.00
appl_delconstraints:	0.00
appl_divide:	0.00
appl_exit:	0.00
appl_fathom:	0.00
appl_feasible:	0.00
appl_init:	0.00

appl_initlp:	0.00
appl_mps:	0.00
appl_node:	0.00
appl_preprocessing:	0.00
appl_primal:	0.00
appl_rank:	0.00
appl_terminatefp:	0.00
appl_terminatenode:	0.00
appl_variables:	0.00

MINTO: Timing statistics (in seconds)

Reading MPS file:	0.00
Restart:	0.00
Preprocessing and probing:	0.00
Primal heuristic:	0.00
Bound improvement:	0.00
Clique cut generation:	0.00
Implication cut generation:	0.00
Knapsack cover generation:	0.00
GUB cover generation:	0.00
Surrogate knapsack cover generation:	0.00
Flow cover generation:	0.00
Cut pool search:	0.00
Cut Management:	0.00
Branching:	0.00
Time LP solving:	0.00
Total elapsed time:	0.00

x [*] :=

```

1 1
2 1
3 1
4 0
5 0
6 1
7 1
8 0
9 0
10 1
11 0
12 1
13 1
14 0
;

```

totnodes = 8

3 Logic and IP (2 pts.)

For each of the following expressions in propositional logic, formulate the Integer Programming models (constraints and binary variables only) to guarantee that the expression is true. “ \vee ” is the logic OR, “ \wedge ” is the logic AND, and “ \neg ” is the logical negation; “ \Rightarrow ” is the logical implication, “ \Leftrightarrow ” is the logical equivalence.

1. $(a \vee b \vee (c \wedge \neg d))$
2. $((a \Rightarrow b) \Rightarrow c)$
3. $(a \Rightarrow (b \Rightarrow c))$

4. $\neg(a \vee b \vee \neg c)$
5. $(a \vee b) \Rightarrow (a \wedge \neg c)$
6. $(a \vee (b \wedge c)) \Leftrightarrow (\neg a \vee b)$
7. $(a \Rightarrow \neg a)$

Solution.

1. $x_a + x_b + x_1 \geq 1$; $x_1 \leq x_c$; $x_1 \leq 1 - x_c$; $x_1 \geq x_c + (1 - x_c) - 1$.
2. This is a bit complicated. Let's define z for $(a \Rightarrow b)$ with an associated binary variable x_1 , which means that $z \Leftrightarrow (a \Rightarrow b)$, or $x_1 = 1 \Leftrightarrow (x_a \leq x_b)$, or equivalently $x_1 = 0 \Leftrightarrow (x_a = 1, x_b = 0)$. This means $1 - x_1 = 1 \Rightarrow (x_a = 1, 1 - x_b = 1)$ and $1 - x_1 = 1 \Leftarrow (x_a = 1, 1 - x_b = 1)$:

$$\begin{aligned} 1 - x_1 &\leq x_a \\ 1 - x_1 &\leq 1 - x_b \\ 1 - x_1 &\geq x_a + (1 - x_b) - 1 \\ x_1 &\leq x_c \end{aligned}$$

3. Similar to 2. Define x_1 for $(b \Rightarrow c)$.

$$\begin{aligned} 1 - x_1 &\leq x_b \\ 1 - x_1 &\leq 1 - x_c \\ 1 - x_1 &\geq x_b + (1 - x_c) - 1 \\ x_a &\leq x_1 \end{aligned}$$

4. $x_1 \geq x_a$, $x_1 \geq x_b$, $x_1 \geq 1 - x_c$; $x_1 \leq (x_a + x_b + 1 - x_c)$; $x_1 = 0$, or more simply:

$$x_a + x_b + (1 - x_c) = 0$$

5. $x_1 \geq x_a$, $x_1 \geq x_b$, $x_1 \leq x_a + x_b$; $x_2 \geq x_a + (1 - x_c) - 1$, $x_2 \leq x_a$, $x_2 \leq 1 - x_c$; $x_1 \leq x_2$.
6. Assign x_1 to $b \wedge c$, x_2 to $(a \vee (b \wedge c))$, and x_3 to $(\neg a \vee b)$.

$$\begin{aligned} x_1 &\leq x_b \\ x_1 &\leq x_c \\ x_1 &\geq x_b + x_c - 1 \\ x_2 &\geq x_a \\ x_2 &\geq x_1 \\ x_3 &\geq 1 - x_a \\ x_3 &\geq x_b \\ x_3 &\leq 1 - x_a + x_b \\ x_2 &= x_3 \end{aligned}$$

7. This is only true when a is false. In fact, $x_a \leq 1 - x_a$, or $2x_a \leq 1$ and therefore $x_a \leq \frac{1}{2}$. Since $x_a \in \{0, 1\}$, we have $x_a = 0$.

4 Mixed-Integer Linear Programming (3 pts.)

Consider the following MILP:

$$\begin{array}{ll}
 \min x_1 & \\
 ax \geq a_0 & (A) \\
 bx \geq b_0 & (B) \\
 cx \geq c_0 & (C) \\
 dx \geq d_0 & (D) \\
 ex \geq e_0 & (E) \\
 fx \geq f_0 & (F) \\
 gx \geq g_0 & (G) \\
 hx \geq h_0 & (H) \\
 ix \geq i_0 & (I) \\
 jx \geq j_0 & (J) \\
 kx \geq k_0 & (K) \\
 lx \geq l_0 & (L) \\
 mx \geq m_0 & (M) \\
 nx \geq n_0 & (N) \\
 ox \geq o_0 & (O) \\
 px \geq p_0 & (P),
 \end{array}$$

where x is a vector of n variables, a, b, \dots, p are coefficient vectors with n elements, and a_0, b_0, \dots, p_0 are constants. Constraints (A) to (P) are special: we don't impose that constraints (A) to (F) be **all** satisfied, but exactly four of them have to. Also, constraints (G), (H), and (I) are incompatible, but at least two of them must be satisfied. Finally, a subset of the constraints (I) to (P) have to be satisfied in such a way that, if a constraint holds, then the next must not hold. For instance, if (M) holds, then (N) must not. If (P) holds, (I) must not.

Re-write the model as an MILP model that takes into account the above "constraints on the constraints". (Hint: for each constraint, use one binary variable which is equal to one **if and only if** the constraint holds.)

Solution. Define a binary variable y for each constraint: y_a, y_b, \dots, y_p . Each is true **if and only if** the corresponding constraint is satisfied:

$$\begin{array}{lll}
 y_a = 1 \Leftrightarrow ax \geq a_0, & y_a = 1 \Rightarrow ax \geq a_0, & y_a = 0 \Rightarrow ax < a_0, \\
 y_b = 1 \Leftrightarrow bx \geq b_0, & y_b = 1 \Rightarrow bx \geq b_0, & y_b = 0 \Rightarrow bx < b_0, \\
 \vdots & \text{and hence } \vdots & \text{and } \vdots \\
 y_p = 1 \Leftrightarrow px \geq p_0, & y_p = 1 \Rightarrow px \geq p_0, & y_p = 0 \Rightarrow px < p_0.
 \end{array}$$

Therefore, a solution $(x; y_a, y_b, \dots, y_p)$ is feasible if and only if

$$\begin{array}{ll}
 ax \geq a_0 - M_a(1 - y_a), & ax \leq a_0 - \epsilon + M'_a y_a, \\
 bx \geq b_0 - M_b(1 - y_b), & bx \leq b_0 - \epsilon + M'_b y_b, \\
 \vdots & \vdots \\
 px \geq p_0 - M_p(1 - y_p), & px \leq p_0 - \epsilon + M'_p y_p
 \end{array}$$

and if the “constraints on the constraints” hold:

$$\begin{aligned}
 y_a + y_b + y_c + y_d + y_e + y_f &= 4 \\
 y_g + y_h + y_i &\geq 2 \\
 y_i &\leq 1 - y_j \\
 y_j &\leq 1 - y_k \\
 y_k &\leq 1 - y_l \\
 y_l &\leq 1 - y_m \\
 y_m &\leq 1 - y_n \\
 y_n &\leq 1 - y_o \\
 y_o &\leq 1 - y_p \\
 y_p &\leq 1 - y_i
 \end{aligned}$$

5 Branch&Bound (4 pts.)

Solve the following MILP problem by Branch&Bound:

$$\begin{aligned}
 \min \quad & x_1 + x_2 \\
 \text{s.t.} \quad & 2x_1 + 5x_2 \geq 10 \\
 & 2x_1 - 2x_2 \geq -3 \\
 & x_2 \leq 3 \\
 & 2x_1 - x_2 \leq 6 \\
 & x_1, x_2 \in \mathbb{Z}
 \end{aligned}$$

Solve the LP relaxation of every node through the graphical method, and every time you have to branch, branch on a variable of your choice. Report the branch&bound tree and, for each node, the graph of the resulting LP, the lower bound at that node and the upper bound, if available.

Solution.

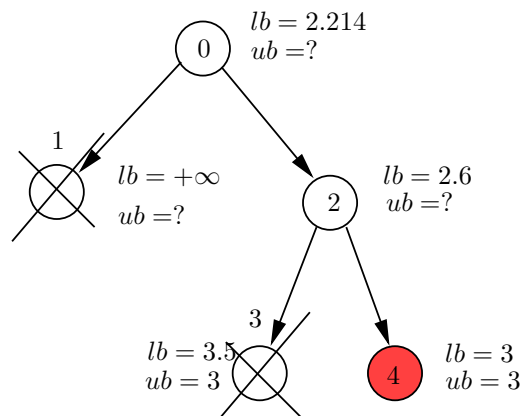
Node 0: The LP relaxation is $(x_1, x_2) = (\frac{5}{14}, \frac{26}{14})$, fractional, giving a lower bound of $\frac{31}{14} = 2.214$. Both variables are fractional, and we branch on x_1 .

Node 1: created with $x_1 \leq \lfloor \frac{5}{14} \rfloor = 0$, is infeasible and we can eliminate it.

Node 2: its LP relaxation, where $x_1 \geq \lceil \frac{5}{14} \rceil = 1$, has solution $(1, \frac{8}{5})$ and gives a lower bound of $1 + \frac{8}{5} = 2.6$. We need to branch on x_2 .

Node 3: its LP relaxation, with $x_2 \leq \lfloor \frac{8}{5} \rfloor = 1$, has solution $(\frac{5}{2}, 1)$ and lower bound 3.5. We should branch here, but let's first take a look at...

Node 4: its LP relaxation gives an integer feasible solution $(1, 2)$ with lower bound 3. It is also an upper bound, which means we are done with this node and we can also discard node 3 as its lower bound is above 3.



6 Branch&Bound (4 pts.)

Solve the following Knapsack problem with Branch&Bound, using the method found on the lecture notes to solve the LP relaxation at each node:

$$\begin{aligned} \min & 2x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 \\ & 7x_1 + 6x_2 + 6x_3 + 5x_4 + 3x_5 \geq 17 \\ & x_1, x_2, x_3, x_4, x_5 \in \{0, 1\} \end{aligned}$$

Clearly represent the Branch&Bound tree, reporting, for each node, the lower and the upper bound, and the reason why you pruned a node.

Solution.

Node 0: The LP relaxation has an optimal solution $(1, 1, \frac{2}{3}, 0, 0)$, a corresponding lower bound of 6, and an upper bound of 7 that can be found by simply rounding x_3 up to 1: $(1, 1, 1, 0, 0)$. We need to branch on x_3 .

Node 1: Here $x_3 = 0$. The LP relaxation has an optimal solution of $(1, 1, \boxed{0}, \frac{4}{5}, 0)$, with the boxed number being imposed by the branching rule. The corresponding lower bound is 7.2, so we can safely eliminate this node as there is already an upper bound of 7.

Node 2: Here $x_3 = 1$. The LP relaxation has an optimal solution of $(1, \frac{2}{3}, \boxed{1}, 0, 0)$, with lower bound 6.333 and upper bound as in node 0. **As the objective function is integer**, a lower bound of 6.333 means that the objective function will never be less than 7, which means that 7 is an optimal solution to the problem and we can stop at this node.

