42114: Integer Programming

Solutions

Week 1

Updated: August 30, 2022

Week 1: Integer Programming

Wolsey Exercise 1.1

We start by defining the decision variables x_1,...,$x_7 \in \{0, 1\}$ corresponding to choosing each of the 7 investments.

(i) We cannot invest in all of the investments, which is equivalent to investing in at most 6 of the 7 investments:

$$\sum_{i=1}^{7} x_i \leq 6$$

(ii) We have to choose at least one of the investments:

$$\sum_{i=1}^{7} x_i \geq 1$$

(iii) Investment 1 cannot be chosen if investment 3 is chosen. This also implies the opposite. Therefore we can only choose at most one of them:

$$x_1 + x_3 \leq 1$$

(iv) Investment 4 can be chosen only if investment 2 is also chosen. This can be ensured by an inequality forcing us also to choose investment 2 when choosing investment 4, while also allowing us to choose investment 2 but not investment 4:

$$x_4 \leq x_2$$

(v) Since we must either choose both investments 1 and 5 or choose neither, their corresponding x'es must always have the same value (whether it is 0 or 1):

$$x_1 = x_5$$

(vi) It is important to note that the "or" is interpreted as a logical OR and not an "exclusive or".

The question asks for two different constraints and using the tricks from the modelling part of the lecture we can write it as two constraints and let the solver choose which one is enforced and which one is not. This requires that we define two extra binary variables $y_1, y_2 \in \{0, 1\}$, then:

$$x_1 + x_2 + x_3 \geq y_1$$
$$x_2 + x_4 + x_5 + x_6 \geq 2y_2$$
$$y_1 + y_2 = 1$$

An alternative and smarter version is:

$$2x_1 + 2x_2 + 2x_3 + x_2 + x_4 + x_5 + x_6 \geq 2$$

This eliminates the two binary variables and makes it fit into one single constraint.

An equivalent formulation of the constraint, In the unlikely event that we are looking for an "XOR" formulation, that is, either the first constraint must be fulfilled and the second **not** or vice versa, we get a slightly more complicated formulation:

$$\begin{aligned} x_1 + x_2 + x_3 &\geq y_1 - My_2 \\ x_2 + x_4 + x_5 + x_6 &\leq y_1 + My_2 \\ x_1 + x_2 + x_3 &\leq 0y_2 + My_1 \\ x_2 + x_4 + x_5 + x_6 &\geq 2y_2 - My_1 \end{aligned}$$

Wolsey Exercise 1.2

This exercise can be solved in two ways. Assuming that x_1 and x_2 are *constants* or assuming that they're decision variables. Both are acceptable solutions.
Let us initially assume that they're **constants**, which makes the exercise slightly easier:

(i) We can use the idea from "Functions with N possible solutions" from the lecture. In this case we introduce binary variable y_1 for choosing x_1 and binary variable y_2 for choosing x_2. We then formulate:

$$\begin{aligned} u = \min \quad & x_1y_1 + x_2y_2 \\ \text{s.t.} \quad & y_1 + y_2 = 1 \\ & y_1, y_2 \in \{0,1\} \end{aligned}$$

Alternatively, since there is only two values to choose between, we save one variable and complete the task with only a single binary variable, since if we choose x_1 it is equivalent to **not** choosing x_2 and vice versa. Only a single binary variable y is introduced that chooses between x_1 and x_2. We then get:

$$\begin{aligned} u = \min \quad & x_1y + x_2(1-y) \\ \text{s.t.} \quad & y \in \{0,1\} \end{aligned}$$

(ii) Once again, we introduce a binary variable y that chooses between $x_1 - x_2$ and $x_2 - x_1$. One of them will be negative while the other is positive (or both zero). We then get:

$$\begin{aligned} v = \max \quad & (x_1 - x_2)y + (x_2 - x_1)(1-y) \\ \text{s.t.} \quad & y \in \{0,1\} \end{aligned}$$

We now assume that x_1 and x_2 are **decision variables**. This makes the problems more difficult as e.g. x_1y from **(i)** is a non-linear term if both x_2 and y are variables.

(i) We introduce a new variable z. Then we get:

$$\begin{array}{rlll} u = \max & z & & \\ \text{s.t.} & z & \leq & x_1 \\ & z & \leq & x_2 \\ & x_1, x_2 & \geq & 0 \\ & x_1, x_2 & \leq & C \\ & z & \geq & 0 \end{array}$$

(ii) We use z again and we then get:

$$\begin{array}{rlll} v = \min & z & & \\ \text{s.t.} & x_1 - x_2 & \leq & z \\ & x_2 - x_1 & \leq & z \\ & x_1, x_2 & \geq & 0 \\ & x_1, x_2 & \leq & C \\ & z & \geq & 0 \end{array}$$

Modelling Knapsack Problems

1. **The Subset Sum Problem:** Just as in the 0-1-Knapsack Problem we need to decide which items are in the knapsack and which are not. The definition of x_i therefore remains the same. Furthermore, we can still not exceed capacity, hence the capacity constraint is still part of the problem:

   $$\sum_{i=1}^{n} a_i x_i \leq b$$

   Instead of maximizing profit, we want to maximize the combined weight of the items in the knapsack. Hence, the objective function is very similar to the above constraint:

   $$\max \sum_{i=1}^{n} a_i x_i$$

   This gives the full mathematical model:

   $$\begin{array}{rlll} \max & \displaystyle\sum_{i=1}^{n} a_i x_i & & \\ \text{s.t.} & \displaystyle\sum_{i=1}^{n} a_i x_i & \leq & b \\ & x_i & \in & \{0,1\} \quad \forall i = 1...n \end{array}$$

2. **Precedence Constrained Knapsack Problem:** For this problem, the entire 0-1-Knapsack Problem is reused. We then need to add constraints to ensure that the precedence constraints

are obeyed. This is done by adding the constraint $x_{i_2} \leq x_{i_1}$ for each pair $(i_1, i_2) \in A$. Then we have the model:

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n} c_i x_i \\
\text{s.t.} \quad & \sum_{i=1}^{n} a_i x_i \leq b \\
& x_{i_2} \leq x_{i_1} \qquad \forall (i_1, i_2) \in A \\
& x_i \in \{0,1\} \qquad \forall i = 1...n
\end{aligned}$$

3. **Multiple Knapsack Problem:** Now we do not only have to decide whether each item is in a knapsack or not, but if it is in a knapsack, we have to decide which of the m knapsacks it will be put into. Therefore the decision variable x_i will be replaced by a new decision variable x_{ij}, which is 1 is item i is put into knapsack j and 0 otherwise.
The "original" capacity constraint is changed to apply to each knapsack separately:

$$\sum_{i=1}^{n} a_i x_{ij} \leq b_j \qquad \forall j = 1, ..., m$$

The objective function is also changed to include item put into all the knapsacks:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{m} c_i x_{ij}$$

Finally we have to make sure that an item is at most put into one knapsack:

$$\sum_{j=1}^{m} x_{ij} \leq 1 \qquad \forall i = 1, ..., n$$

The full model is then:

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} c_i x_{ij} \\
\text{s.t.} \quad & \sum_{i=1}^{n} a_i x_{ij} \leq b_j \qquad \forall j = 1...m \\
& \sum_{j=1}^{m} x_{ij} \leq 1 \qquad \forall i = 1...n \\
& x_{ij} \in \{0,1\} \qquad \forall i = 1...n, \forall j = 1...m
\end{aligned}$$

4. **Multiple Choice Knapsack Problem:** Let a_{il} be the weight of variant l of item i and let c_{il} be the corresponding profit. We redefine the decision variable x_i to be x_{il}, where x_{il} is 1 if variant l of item i is put into the knapsack and 0 otherwise.
The objective function is then changed to accommodate the different variants:

$$\max \sum_{i=1}^{n} \sum_{l \in N_i} c_{il} x_{il}$$

Likewise the capacity constraint is modified:

$$\sum_{i=1}^{n} \sum_{l \in N_i} a_{il} x_{il} \leq b$$

Finally, a new constraint is added, ensuring that exactly one variant of each item is put into the knapsack:

$$\sum_{l \in N_i} x_{il} = 1$$

Hence, the model becomes:

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n} \sum_{l \in N_i} c_{il} x_{il} & & & \\
\text{s.t.} \quad & \sum_{i=1}^{n} \sum_{l \in N_i} a_{il} x_{il} & \leq & b & \\
& \sum_{l \in N_i} x_{il} & = & 1 & \\
& x_{il} & \in & \{0,1\} & \forall i = 1...n, \forall l \in N_i
\end{aligned}$$

Manpower Planning Problem

1. Let x_i be the number of employees that start working on day i (Monday being the first day of the week), then works for five days consecutively and then has two days of rest. If we get past Sunday we "wrap around" and start back on the Monday. So e.g. x_1 will be the number of employees that start on a Monday, works until Friday and then have Saturday and Sunday off, while x_4 is the number of employees that start work on a Thursday, works until Monday in the following week and have Tuesday and Wednesday off etc.
   This is in fact the only variables we need. Now the demands can be written as constraints. We will have one constraint for each day. They reflect that the sum of employees that work on a specific day have to be at least the number of employees needed.
   So e.g. the constraint for Monday would contain all the x_i variables that implies that employees are working on Mondays. We therefore get:

   $$x_1 + x_4 + x_5 + x_6 + x_7 \geq b_1$$

   Note that we cannot write $=$ instead of $\geq$ as there might not exist a solution that fulfills the constraints to equality. Hence, we might need to use more employees than what is strictly necessary.

The full model is then:

$$
\begin{aligned}
z = \min \quad & \sum_{i=1}^{7} x_i \\
\text{s.t.} \quad & x_1 + x_4 + x_5 + x_6 + x_7 \geq b_1 \\
& x_1 + x_2 + x_5 + x_6 + x_7 \geq b_2 \\
& x_1 + x_2 + x_3 + x_6 + x_7 \geq b_3 \\
& x_1 + x_2 + x_3 + x_4 + x_7 \geq b_4 \\
& x_1 + x_2 + x_3 + x_4 + x_5 \geq b_5 \\
& x_2 + x_3 + x_4 + x_5 + x_6 \geq b_6 \\
& x_3 + x_4 + x_5 + x_6 + x_7 \geq b_7 \\
& x_i \geq 0 \text{ and integer} \qquad \forall i = 1...7
\end{aligned}
$$

2. Inputting the model into Julia (see at the end of the document) with
   $b_1 = 10, b_2 = 5, b_3 = 10, b_4 = 5, b_5 = 10, b_6 = 5, b_7 = 10$
   gives the solution:
   $z = 14, x_1 = 3, x_3 = 4, x_4 = 4, x_6 = 3$, with all other variables equal to 0.
   This is not necessarily a unique best solution as there might be other equally good solutions.

3. Inputting the model into Julia (see at the end of the document) with
   $b_1 = 8, b_2 = 8, b_3 = 8, b_4 = 8, b_5 = 8, b_6 = 8, b_7 = 7$
   gives the solution:
   $z = 11, x_1 = 2, x_2 = 2, x_3 = 1, x_4 = 2, x_5 = 1, x_6 = 2, x_7 = 1$.
   This is not necessarily a unique best solution as there might be other equally good solutions.

4. The number of unused man days can on an overall level be calculated as $5z - \sum_{i=1}^{7} b_i$. Every hired employee will need to work for five days, therefore the total contribution of staff members can be calculated as $5z$. The actual number of man days needed is only $\sum_{i=1}^{7} b_i$, so the difference between these numbers is the "surplus"/unused man days.
   For problem 2. and 3. the number of unused man days are 15 and 0.
   If you want the number on a daily basis, this can be established by taking the difference between the right hand sides and the left hand sides of the constraints of the problem.

5. The coefficient of the variables in the objective function give us the opportunity to penalize "expensive days off" or "expensive work days". So if e.g. it is expensive to have people working on a Saturday or Sunday, we might give a higher coefficient to $x_2, x_3, x_4, x_5, x_6, x_7$, which contains these days (it might be even higher for x_3, x_4, x_5, x_6 where the employees are working on both Saturdays and Sundays).

Wolsey Exercise 1.5

Let us define K to be the set of courses and I_k to be the set of sections or repetitions for course $k \in K$. It is necessary with a specific I_k for each course, in case we want to model that e.g. course 2 is repeated 5 times a day while course 4 is only repeated twice. If all courses have the same number of repetitions, I_k could be replaced with a single I.
In order to simplify the description of some of the constraints we define the parameter a_{ikt} based on t_{ik}. Let $a_{ikt} = 1$ if $t_{ik} = t$ and $a_{ikt} = 0$ otherwise.

(i) Let $x_{ik} \in \{0,1\}$ be the binary decision variable, indicating if John choose to take section $i \in I_k$ of course $k \in K$.

John wants to maximize his preferences. The objective function then becomes:

$$\max \sum_{k \in K, i \in I_k} p_{ik} \cdot x_{ik}$$

He has to take exactly 4 courses:

$$\sum_{k \in K, i \in I_k} x_{ik} = 4$$

He can only take one section of each course:

$$\sum_{i \in I_k} x_{ik} \leq 1 \qquad \forall k \in K$$

He cannot have more than one course in each time slot:

$$\sum_{k \in K, i \in I_k} a_{ikt} x_{ik} \leq 1 \qquad \forall t \in \{10, ..., 19\}$$

From this we have the full integer program:

$$\begin{aligned}
\max \quad & \sum_{\substack{k \in K \\ i \in I_k}} p_{ik} \cdot x_{ik} & & & \\
\text{s.t.} \quad & \sum_{\substack{k \in K \\ i \in I_k}} x_{ik} & = & 4 & \\
& \sum_{i \in I_k} x_{ik} & \leq & 1 & \forall k \in K \\
& \sum_{\substack{k \in K \\ i \in I_k}} a_{ikt} x_{ik} & \leq & 1 & \forall t \in \{10, ..., 19\} \\
& x_{ik} & \in & \{0,1\} & \forall k \in K, \forall i \in I_k
\end{aligned}$$

(ii) To make sure that no more than two consecutive hours of class is taken at a time, we add the constraint:

$$\sum_{k \in K, i \in I_k} (a_{ikt} x_{ik} + a_{ik(t+1)} x_{ik} + a_{ik(t+2)} x_{ik}) \leq 2 \qquad \forall t \in \{10, ..., 17\}$$

Within three time consecutive time slots, at most two courses can be chosen.

(iii) John wants to start his day as late as possible. Let T be a variable, denoting the starting time of John's first class of the day. Then we add the constraint:

$$T \leq t_{ik} x_{ik} + 19(1 - x_{ik}) \qquad \forall k \in K, i \in I_k$$

which ensures that T is at most the starting time of each of John's courses. The objective function is then changed to:

$$\max T$$

maximizing T and therefore also making sure that John starts his day as late as possible.

Julia Code

Week 1: Manpower Planning Problem

```julia
## Packages needed for solving MIPs
using JuMP, GLPK

m = Model(GLPK.Optimizer) # Defining model

@variable(m,x[1:7] >= 0, Int) # Defining integer variables

n = length(x)                          # Defining the number of variables
b = [10 5 10 5 10 5 10]                # Used for solving Problem 2
#b = [8 8 8 8 8 8 7]                    # Used for solving Problem 3

# Setting up the objective function
@objective(m, Min, sum( x[i] for i=1:n))

# Defining the constraints
@constraint(m, x[1] + x[4] + x[5] + x[6] + x[7] >= b[1]) # Monday
@constraint(m, x[1] + x[2] + x[5] + x[6] + x[7] >= b[2]) # Tuesday
@constraint(m, x[1] + x[2] + x[3] + x[6] + x[7] >= b[3]) # Wednesday
@constraint(m, x[1] + x[2] + x[3] + x[4] + x[7] >= b[4]) # Thursday
@constraint(m, x[1] + x[2] + x[3] + x[4] + x[5] >= b[5]) # Friday
@constraint(m, x[2] + x[3] + x[4] + x[5] + x[6] >= b[6]) # Saturday
@constraint(m, x[3] + x[4] + x[5] + x[6] + x[7] >= b[7]) # Sunday

JuMP.optimize!(m) # Solving the model

# Printing the objective value
println("Objective: ", JuMP.objective_value(m))

for i in 1:n
    # Printing solution as individual items
    println("x$i = ",JuMP.value(x[i]), " ")
end

println("X = ", JuMP.value.(x)) # Printing solution as a vector

z = objective_value(m) # Saving the objective value
# Computing the number unused man days for Problem 4
println(" Number of unused man day = ", 5*z- sum(b))
```

Contents