Workforce scheduling

2019-12-04

The mining industry in Chile is a big deal. It represents about 10% of the country GDP. Being a big industry, the possibility to know someone that works there is not difficult. My sister works in one of the plenty mining companies. She has different schedules, like 7 days on 7 days off, or 10 days on 10 days off. That is normal in this kind of industry, so I wanted to create an Integer Programming model for different type of those schedules.

First, I will define a **work cycle** as a pair (M, N), where M is the number of days on, and N is the total length of the work cycle (days on + days off).

Second, in a work cycle (M, N) there are N different patterns. Each pattern represents when the employees can start working: day 1, day 2 and so on until day N.

I am going to model (5, 7) work cycle, 5 days on 2 days off. I found an example in this book [1]. The principles are the same if you want to apply it to other kind of work cycle.

The problem is as follow:

1. The company wants to minimize the total number of employees required
2. The company requires a minimum number of employees each day
3. The employees must work 5 consecutive days, and after 2 days off

The key to model this problem, it is not to assign employees to each day, instead you have to decide how many employees assign to each pattern. With a (5, 7) cycle, you will have 7 patterns. For example, the first pattern is [1,1,1,1,1,0,0] meaning that employees start the first day of the cycle, work for 5 consecutive days, and then 2 days off. The second pattern is [0,1,1,1,1,1,0] meaning employees start working in the second day until complete 5 consecutive working days, and then 2 days off. You can notice that the pattern backwards in the second example with the second day off in the first day of the work cycle.

To create the patterns, I use a circulant matrix and I transpose it so every row represent different starting days. I am going to use Julia and the JuMP packages. It is really awesome programming language and package too!

```
using SpecialMatrices
design_matrix = SpecialMatrices.Circulant([1,1,1,1,1,0,0])'
```

```
## 7×7 Adjoint{Int64,Circulant{Int64}}:
##  1  1  1  1  1  0  0
##  0  1  1  1  1  1  0
##  0  0  1  1  1  1  1
##  1  0  0  1  1  1  1
##  1  1  0  0  1  1  1
##  1  1  1  0  0  1  1
##  1  1  1  1  0  0  1
```

You can also build a (7, 14) work cycle or any:

```
design_matrix = SpecialMatrices.Circulant([1,1,1,1,1,1,1,0,0,0,0,0,0,0])'
```

```
## 14×14 Adjoint{Int64,Circulant{Int64}}:
##  1  1  1  1  1  1  1  0  0  0  0  0  0  0
##  0  1  1  1  1  1  1  1  0  0  0  0  0  0
##  0  0  1  1  1  1  1  1  1  0  0  0  0  0
##  0  0  0  1  1  1  1  1  1  1  0  0  0  0
##  0  0  0  0  1  1  1  1  1  1  1  0  0  0
##  0  0  0  0  0  1  1  1  1  1  1  1  0  0
##  0  0  0  0  0  0  1  1  1  1  1  1  1  0
##  0  0  0  0  0  0  0  1  1  1  1  1  1  1
##  1  0  0  0  0  0  0  0  1  1  1  1  1  1
##  1  1  0  0  0  0  0  0  0  1  1  1  1  1
##  1  1  1  0  0  0  0  0  0  0  1  1  1  1
##  1  1  1  1  0  0  0  0  0  0  0  1  1  1
##  1  1  1  1  1  0  0  0  0  0  0  0  1  1
##  1  1  1  1  1  1  0  0  0  0  0  0  0  1
```

Optimization model:

$$\begin{aligned} \min_{x} \quad & \textstyle\sum_{i=1}^{7} x_i \\ \text{s.t.} \quad & \\ & \sum_{i=1}^{7} a_{ij} x_i \geq r_i \quad \forall j \in \{1..7\} \\ & x_i \geq 0 \quad \forall i \in \{1..7\} \\ & x_i \in \text{Integer} \end{aligned}$$

Where,

x_i number of employees assigned to the i cycle \ a_{ij} 1 if the j day is a working day for pattern i, 0 otherwise

```julia
# Loading packages
using JuMP
using SpecialMatrices
using GLPK

# Design matrix
design_matrix = SpecialMatrices.Circulant([1,1,1,1,1,0,0])';

# Employee requirement per day
employee_requirement = [17, 13, 15, 19, 14, 16, 11];

# size design_matrix
m, _ = size(design_matrix);

# First model
model = Model(with_optimizer(GLPK.Optimizer));
@variable(model, x[1:m], Int, lower_bound = 0);

for i in 1:m
    @constraint(model, sum(design_matrix[:, i] .* x) >= employee_requirement[i]);
end

@objective(model, Min, sum(x));

# Print the model
println(model)
```

```
## Min x[1] + x[2] + x[3] + x[4] + x[5] + x[6] + x[7]
## Subject to
##  x[1] + x[4] + x[5] + x[6] + x[7] >= 17.0
##  x[1] + x[2] + x[5] + x[6] + x[7] >= 13.0
##  x[1] + x[2] + x[3] + x[6] + x[7] >= 15.0
##  x[1] + x[2] + x[3] + x[4] + x[7] >= 19.0
##  x[1] + x[2] + x[3] + x[4] + x[5] >= 14.0
##  x[2] + x[3] + x[4] + x[5] + x[6] >= 16.0
##  x[3] + x[4] + x[5] + x[6] + x[7] >= 11.0
##  x[1] >= 0.0
##  x[2] >= 0.0
##  x[3] >= 0.0
##  x[4] >= 0.0
##  x[5] >= 0.0
##  x[6] >= 0.0
##  x[7] >= 0.0
##  x[1] integer
##  x[2] integer
##  x[3] integer
##  x[4] integer
##  x[5] integer
##  x[6] integer
##  x[7] integer
```

```
# Start the optimization
optimize!(model)

# Check outputs
termination_status(model)
```

```
## OPTIMAL::TerminationStatusCode = 1
```

```
println("Total number of employees required: ", sum(value.(x)))
```

```
## Total number of employees required: 23.0
```

```
design_matrix .* value.(x)
```

```
## 7×7 Array{Float64,2}:
##  6.0  6.0  6.0  6.0  6.0  0.0  0.0
##  0.0  3.0  3.0  3.0  3.0  3.0  0.0
##  0.0  0.0  3.0  3.0  3.0  3.0  3.0
##  7.0  0.0  0.0  7.0  7.0  7.0  7.0
##  1.0  1.0  0.0  0.0  1.0  1.0  1.0
##  2.0  2.0  2.0  0.0  0.0  2.0  2.0
##  1.0  1.0  1.0  1.0  0.0  0.0  1.0
```

Now the good part. Can you see something “not optimal”? There are two days that the optimization problem assign 1 employee (row 5 and 7) and 2 employees (row 6). The mining companies normally outsources the transporation from the closest city to the mining. They are going to pay for a full bus to only transport 1 employee or 2 in this case. Waste of money.

I am going to include in the problem a new decision variable. This decision variable will take value 1 if x is positive and 0 otherwise. This variable will help us to minimize the number of patterns used.

$$\begin{aligned}
\min_{x} \quad & \textstyle\sum_{i=1}^{7} x_i + \sigma \cdot \sum_{i=1}^{7} v_i \\
\text{s.t.} \quad & \\
& \sum_{i=1}^{7} a_{ij} x_i \geq r_i \quad \forall j \in \{1..7\} \\
& x_i \geq 0 \quad \forall i \in \{1..7\} \\
& M \cdot v_i \geq x_i \quad \forall i \in \{1..7\} \\
& x_i \in \text{Integer} \\
& v_i \in \{0,1\}
\end{aligned}$$

```julia
# size design matrix
m, _ = size(design_matrix);

# Our friend the big M
M = sum(employee_requirement);

# Small constant - our regularization parameter
σ = 0.5;

# The model and first set of constraints as before
model = Model(with_optimizer(GLPK.Optimizer));
@variable(model, x[1:m], Int, lower_bound = 0);

for i in 1:m
    @constraint(model, sum(design_matrix[:, i] .* x) >= employee_requirement[i]);
end


# new decision variable
@variable(model, v[1:m], Bin);
for i in 1:m
    @constraint(model, M * v[i] >= x[i]);
end

@objective(model, Min, sum(x) + σ * sum(v));

println(model)
```

```
## Min x[1] + x[2] + x[3] + x[4] + x[5] + x[6] + x[7] + 0.5 v[1] + 0.5 v[2] + 0.5 v[3] + 0.5 v
[4] + 0.5 v[5] + 0.5 v[6] + 0.5 v[7]
## Subject to
##  x[1] + x[4] + x[5] + x[6] + x[7] >= 17.0
##  x[1] + x[2] + x[5] + x[6] + x[7] >= 13.0
##  x[1] + x[2] + x[3] + x[6] + x[7] >= 15.0
##  x[1] + x[2] + x[3] + x[4] + x[7] >= 19.0
##  x[1] + x[2] + x[3] + x[4] + x[5] >= 14.0
##  x[2] + x[3] + x[4] + x[5] + x[6] >= 16.0
##  x[3] + x[4] + x[5] + x[6] + x[7] >= 11.0
##  105 v[1] - x[1] >= 0.0
##  105 v[2] - x[2] >= 0.0
##  105 v[3] - x[3] >= 0.0
##  105 v[4] - x[4] >= 0.0
##  105 v[5] - x[5] >= 0.0
##  105 v[6] - x[6] >= 0.0
##  105 v[7] - x[7] >= 0.0
##  x[1] >= 0.0
##  x[2] >= 0.0
##  x[3] >= 0.0
##  x[4] >= 0.0
##  x[5] >= 0.0
##  x[6] >= 0.0
##  x[7] >= 0.0
##  x[1] integer
##  x[2] integer
##  x[3] integer
##  x[4] integer
##  x[5] integer
##  x[6] integer
##  x[7] integer
##  v[1] binary
##  v[2] binary
##  v[3] binary
##  v[4] binary
##  v[5] binary
##  v[6] binary
##  v[7] binary
```

```
optimize!(model)

termination_status(model)
```

```
## OPTIMAL::TerminationStatusCode = 1
```

```
println("Total number of employees required: ", sum(value.(x)))
```

```
## Total number of employees required: 23.0
```

```
design_matrix .* value.(x)
```

```
## 7×7 Array{Float64,2}:
##  0.0  0.0  0.0  0.0  0.0  0.0  0.0
##  0.0  6.0  6.0  6.0  6.0  6.0  0.0
##  0.0  0.0  0.0  0.0  0.0  0.0  0.0
##  8.0  0.0  0.0  8.0  8.0  8.0  8.0
##  0.0  0.0  0.0  0.0  0.0  0.0  0.0
##  2.0  2.0  2.0  0.0  0.0  2.0  2.0
##  7.0  7.0  7.0  7.0  0.0  0.0  7.0
```

I include a small constant σ to reduce the influence of the second objective because our primary goal is to reduce the total number of employees. You can think of this, like the regularization parameter when doing regularized linear or logistic regression.

With this new model, there are 3 patterns without employees assigned while we keep the number of total employees required to the minimum level as in the first model.

[1] Practical Management Science, 6th Edition, W.L. Winston, S.C. Albright