

Integer Programming

Mehmet Hakan Satman (Ph.D.)

December 3, 2025

Contents

1	The Knapsack Problem	1
2	Project Selection Problem	2
3	The Assignment Problem	3
4	Fixed Charge Problem	4
5	Set Covering Problem	6
6	The Shortest Path Problem with Integer Programming	8
7	Switching Constraints	9

1 The Knapsack Problem

Items	1	2	3	4	5	6
Weight	2	3	4	5	9	7
Value	3	4	5	6	10	8

Table 1: Knapsack problem data

The capacity of the knapsack is 15 kg. We want to maximize the total value of the items in the knapsack without exceeding its capacity.

$$\text{Maximize } Z = 3x_1 + 4x_2 + 5x_3 + 6x_4 + 10x_5 + 8x_6$$

Subject to

$$2x_1 + 3x_2 + 4x_3 + 5x_4 + 9x_5 + 7x_6 \leq 15$$

$$x_i \in \{0, 1\} \quad \text{for } i = 1, 2, \dots, 6$$

Now, suppose that the item 3 and item 4 are related, meaning that if we include item 3 in the knapsack, we must also include item 4 (If we include item 4, we must also include item 3). This can be modeled with the following constraint:

$$\text{Maximize } Z = 3x_1 + 4x_2 + 5x_3 + 6x_4 + 10x_5 + 8x_6$$

Subject to

$$2x_1 + 3x_2 + 4x_3 + 5x_4 + 9x_5 + 7x_6 \leq 15$$

$$x_3 = x_4$$

$$x_i \in \{0, 1\} \quad \text{for } i = 1, 2, \dots, 6$$

Here is the Julia solution using Operations Research package:

```

using OperationsResearchModels
values = [3, 4, 5, 6, 10, 8]
weights = [2, 3, 4, 5, 9, 7]
capacity = 15
problem = KnapsackProblem(values, weights, capacity)
result = solve(problem)

```

The output:

```

julia> result.objective
18.0

```

```

julia> result.selected
6-element Vector{Bool}:
 1
 1
 1
 1
 0
 0

```

2 Project Selection Problem

Project	Year 1	Year 2	Year 3	Return
1	50	20	30	200
2	60	30	20	250
3	40	10	50	150
4	30	40	30	180
5	20	50	40	160
Budget	150	100	120	

Table 2: Project selection data

Project 1 requires 50 dollars (in thousands) of resource in year 1, 20 dollars in year 2, and 30 dollars in year 3. When the project is completed, it returns 200 dollars. First year budget is 150 dollars, second year budget is 100 dollars, and third year budget is 120 dollars. We want to select projects to maximize the total return without exceeding the yearly budgets.

$$\begin{aligned}
 &\text{Maximize } Z = 200x_1 + 250x_2 + 150x_3 + 180x_4 + 160x_5 \\
 &\text{Subject to} \\
 &\quad 50x_1 + 60x_2 + 40x_3 + 30x_4 + 20x_5 \leq 150 \quad (\text{Year 1 Budget}) \\
 &\quad 20x_1 + 30x_2 + 10x_3 + 40x_4 + 50x_5 \leq 100 \quad (\text{Year 2 Budget}) \\
 &\quad 30x_1 + 20x_2 + 50x_3 + 30x_4 + 40x_5 \leq 120 \quad (\text{Year 3 Budget}) \\
 &\quad x_i \in \{0, 1\} \quad \text{for } i = 1, 2, \dots, 5
 \end{aligned}$$

Here is the Julia solution using Operations Research package:

```

using JuMP, HiGHS
model = Model(HiGHS.Optimizer)
@variable(model, x[1:5], Bin)
@objective(model, Max, 200x[1] + 250x[2] + 150x[3] + 180x[4] + 160x[5])
@constraint(model, 50x[1] + 60x[2] + 40x[3] + 30x[4] + 20x[5] <= 150)

```

```
@constraint(model, 20x[1] + 30x[2] + 10x[3] + 40x[4] + 50x[5] <= 100)
@constraint(model, 30x[1] + 20x[2] + 50x[3] + 30x[4] + 40x[5] <= 120)
optimize!(model)
println(value.(x))
```

The output:

```
[1.0, 1.0, 0.0, 1.0, 0.0]
```

3 The Assignment Problem

	Task 1	Task 2	Task 3	Task 4
Worker 1	9	2	7	8
Worker 2	6	4	3	7
Worker 3	5	8	1	8
Worker 4	7	6	9	4

Table 3: Assignment problem cost matrix

We have 4 workers and 4 tasks. The cost of assigning each worker to each task is given in the table above. We want to assign each worker to exactly one task such that the total cost is minimized.

Minimize $Z = 9x_{11} + 2x_{12} + 7x_{13} + 8x_{14} + 6x_{21} + \dots + 9x_{43} + 4x_{44}$
 Subject to

$$x_{11} + x_{12} + x_{13} + x_{14} = 1 \quad (\text{Worker 1 assigned to one task})$$

$$x_{21} + x_{22} + x_{23} + x_{24} = 1 \quad (\text{Worker 2 assigned to one task})$$

$$x_{31} + x_{32} + x_{33} + x_{34} = 1 \quad (\text{Worker 3 assigned to one task})$$

$$x_{41} + x_{42} + x_{43} + x_{44} = 1 \quad (\text{Worker 4 assigned to one task})$$

$$x_{11} + x_{21} + x_{31} + x_{41} = 1 \quad (\text{Task 1 assigned to one worker})$$

$$x_{12} + x_{22} + x_{32} + x_{42} = 1 \quad (\text{Task 2 assigned to one worker})$$

$$x_{13} + x_{23} + x_{33} + x_{43} = 1 \quad (\text{Task 3 assigned to one worker})$$

$$x_{14} + x_{24} + x_{34} + x_{44} = 1 \quad (\text{Task 4 assigned to one worker})$$

$$x_{ij} \in \{0, 1\} \quad \text{for } i, j = 1, 2, 3, 4$$

Here is the Julia solution using Operations Research package:

```
using OperationsResearchModels
costs = [9 2 7 8; 6 4 3 7; 5 8 1 8; 7 6 9 4]
problem = AssignmentProblem(costs)
result = solve(problem)
```

The output:

```
julia> result.cost
13.0
```

```
julia> result.solution
4×4 Matrix{Float64}:
 0.0  1.0  0.0  0.0
 1.0  0.0 -0.0  0.0
 0.0  0.0  1.0  0.0
-0.0  0.0  0.0  1.0
```

This means:

- Worker 1 is assigned to Task 2
- Worker 2 is assigned to Task 1
- Worker 3 is assigned to Task 3
- Worker 4 is assigned to Task 4

4 Fixed Charge Problem

Package	Fixed Cost	Variable Cost per Unit
A	20	1
B	30	0.8
C	25	0.9
D	19	1.2

Table 4: Fixed charge problem data

- If a package is selected then the corresponding fixed cost is paid, variable cost is paid per minutes used
- 250 minutes will be used in total
- Any package can be used and switched to anytime
- What is the optimal combination of the use of these packages?

Let's define the decision variables:

$$y_i = \begin{cases} 1 & \text{if package } i \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i = A, B, C, D$$

$$x_i = \text{minutes used from package } i \quad \text{for } i = A, B, C, D$$

$$\text{Minimize } Z = 20y_A + 30y_B + 25y_C + 19y_D + 1x_A + 0.8x_B + 0.9x_C + 1.2x_D$$

Subject to

$$x_A + x_B + x_C + x_D = 250 \quad (\text{Total minutes used})$$

$$x_A \leq M \times y_A \quad (\text{Linking constraint for package A})$$

$$x_B \leq M \times y_B \quad (\text{Linking constraint for package B})$$

$$x_C \leq M \times y_C \quad (\text{Linking constraint for package C})$$

$$x_D \leq M \times y_D \quad (\text{Linking constraint for package D})$$

$$x_i \geq 0 \quad \text{for } i = A, B, C, D$$

using JuMP, HiGHS

```
const BigM = 10^6
```

```
m = Model(HiGHS.Optimizer)
```

```
@variable(m, ya, Bin)
```

```
@variable(m, yb, Bin)
```

```
@variable(m, yc, Bin)
```

```

@variable(m, yd, Bin)
@variable(m, xa >= 0)
@variable(m, xb >= 0)
@variable(m, xc >= 0)
@variable(m, xd >= 0)

@objective(m, Min, 20*ya + 30*yb + 25*yc + 19*yd + 1*xa + 0.8*xb + 0.9*xc + 1.2*xd)

@constraint(m, xa + xb + xc + xd >= 250)

@constraint(m, xa <= BigM*ya)
@constraint(m, xb <= BigM*yb)
@constraint(m, xc <= BigM*yc)
@constraint(m, xd <= BigM*yd)

optimize!(m)

println("Optimal value: ", objective_value(m))
println("x: ", value(xa), ", ", value(xb), ", ", value(xc), ", ", value(xd))
println("y: ", value(ya), ", ", value(yb), ", ", value(yc), ", ", value(yd))

```

Output:

```

Optimal value: 230.0
x: 0.0, 250.0, 0.0, 0.0
y: -0.0, 1.0, -0.0, -0.0

```

As we see from the output, only package B is used for 250 minutes, resulting in a total cost of 230:

$$30 + 0.8 \times 250 = 230$$

5 Set Covering Problem

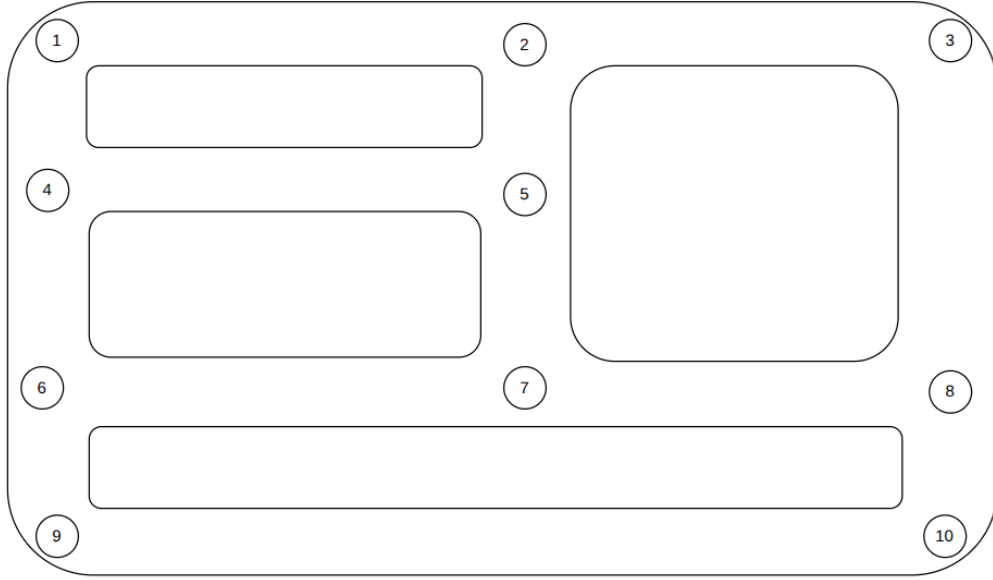


Figure 1: Set covering problem example

In the above figure, there are 10 street lamps (numbered from 1 to 10) located in a map. At least two street lamps must be active (turned on) in a street segment to ensure safety. Find the minimum number of street lamps to be activated such that all street segments are covered.

Let's define the binary decision variables:

$$x_i = \begin{cases} 1 & \text{if street lamp } i \text{ is activated} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } i = 1, 2, \dots, 10$$

Here is the mathematical formulation of the set covering problem:

Minimize $Z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10}$
 Subject to

$$\begin{aligned} x_1 + x_2 + x_3 &\geq 2 && \text{(Street segment on the top)} \\ x_4 + x_5 &\geq 2 && \text{(Street segment on line 2)} \\ x_6 + x_7 + x_8 &\geq 2 && \text{(Street segment on line 3)} \\ x_9 + x_{10} &\geq 2 && \text{(Street segment on the bottom)} \\ x_1 + x_4 + x_6 + x_9 &\geq 2 && \text{(Left vertical street segment)} \\ x_2 + x_5 + x_7 &\geq 2 && \text{(Middle vertical street segment)} \\ x_3 + x_8 + x_{10} &\geq 2 && \text{(Right vertical street segment)} \\ x_i &\in \{0, 1\} && \text{for } i = 1, 2, \dots, 10 \end{aligned}$$

Here is the Julia solution:

```
using JuMP, HiGHS
```

```
const minLampsRequired = 2
```

```

m = Model(HiGHS.Optimizer)

@variable(m, x[1:10], Bin)

@objective(m, Min, sum(x))

@constraint(m, x[1] + x[2] + x[3] >= minLampsRequired)
@constraint(m, x[4] + x[5] >= minLampsRequired)
@constraint(m, x[6] + x[7] + x[8] >= minLampsRequired)
@constraint(m, x[9] + x[10] >= minLampsRequired)
@constraint(m, x[1] + x[4] + x[6] + x[9] >= minLampsRequired)
@constraint(m, x[2] + x[5] + x[7] >= minLampsRequired)
@constraint(m, x[3] + x[8] + x[10] >= minLampsRequired)

optimize!(m)

println("Minimum number of lamps required: ", objective_value(m))

for i in 1:10
    println("Lamp ", i, ": ", value(x[i]))
end

```

Here is the output:

```

Minimum number of lamps required: 8.0
Lamp 1: 0.0
Lamp 2: 1.0
Lamp 3: 1.0
Lamp 4: 1.0
Lamp 5: 1.0
Lamp 6: 0.0
Lamp 7: 1.0
Lamp 8: 1.0
Lamp 9: 1.0
Lamp 10: 1.0

```

6 The Shortest Path Problem with Integer Programming

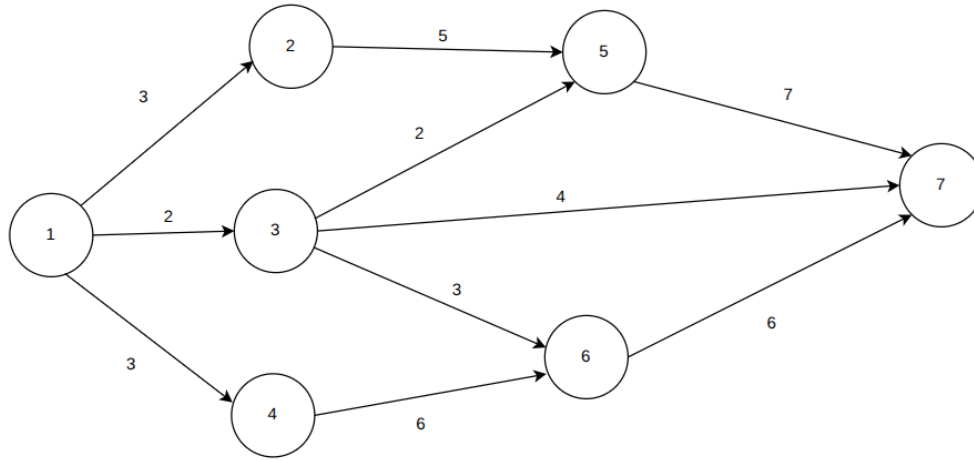


Figure 2: Shortest path problem graph

- Node 1 is the starting node
- Node 7 is the destination (final, sink) node
- Each edge has a cost associated with it (shown on the edges)
- Find the shortest (minimum cost) path from node 1 to node 7

Let's define the binary decision variables:

$$x_{ij} = \begin{cases} 1 & \text{if edge } (i, j) \text{ is included in the path} \\ 0 & \text{otherwise} \end{cases} \quad \text{for each edge } (i, j)$$

Here is the mathematical formulation of the shortest path problem:

Minimize $Z = 3x_{12} + 2x_{13} + 3x_{14} + 5x_{25} + 2x_{35} + 4x_{37} + 3x_{36} + 6x_{46} + 7x_{57} + 6x_{67}$
 Subject to

$$\begin{aligned} x_{12} + x_{13} + x_{14} &= 1 && \text{(Node 1)} \\ x_{12} &= x_{25} && \text{(Node 2)} \\ x_{13} &= x_{35} + x_{36} + x_{37} && \text{(Node 3)} \\ x_{14} &= x_{46} && \text{(Node 4)} \\ x_{25} + x_{35} &= x_{57} && \text{(Node 5)} \\ x_{36} + x_{46} &= x_{67} && \text{(Node 6)} \\ x_{37} + x_{57} + x_{67} &= 1 && \text{(Node 7)} \\ x_{ij} &\in \{0, 1\} && \text{for each edge } (i, j) \end{aligned}$$

Here is the Julia solution:

```
using OperationsResearchModels
```

```
connections = Connection[
    Connection(1, 2, 3),
```



```

Connection(1, 3, 2),
Connection(1, 4, 3),
Connection(2, 5, 5),
Connection(3, 5, 2),
Connection(3, 7, 4),
Connection(3, 6, 3),
Connection(4, 6, 6),
Connection(5, 7, 7),
Connection(6, 7, 6)]

```

```
problem = ShortestPathProblem(connections)
```

```
result = solve(problem)
```

```
println("Cost is ", result.cost)
println("Path is ", result.path)
```

Here is the output:

```
Cost is 6.0
```

```
Path is Connection[Connection(1, 3, 2), Connection(3, 7, 4)]
```

So, the shortest path is

$$1 \rightarrow 3 \rightarrow 7$$

7 Switching Constraints

The firm produces 2 types of products, Product A and Product B. The production requires machines (in hours) and labor (in hours) as resources. Two different machines are used for packaging the products. Only one packaging machine should be selected. The production data is given in the table below:

	Product A	Product B	Capacity
Labor	7	8	5600
Machine 1	2	3	2000
Machine 2	4	2	2400
Profit	30	40	

Table 5: Production data

The firm wants to determine the production quantities of Product A and Product B to maximize profit while adhering to resource constraints and selecting only one packaging machine.

Let's define the decision variables:

x_A = production quantity of Product A

x_B = production quantity of Product B

$$y = \begin{cases} 1 & \text{if Machine 1 is selected} \\ 0 & \text{otherwise} \end{cases}$$

The mathematical formulation of the problem is as follows:

Maximize $Z = 30x_A + 40x_B$
 Subject to
 $7x_A + 8x_B \leq 5600$ (Labor constraint)
 $2x_A + 3x_B \leq 2000 + M \times (1 - y)$
 $4x_A + 2x_B \leq 2400 + M \times y$
 $x_A, x_B \geq 0$
 $y \in \{0, 1\}$

When $y = 1$, Machine 1 is selected, and the constraint for Machine 2 becomes non-restrictive due to the large value of M . When $y = 0$, Machine 2 is selected, and the constraint for Machine 1 becomes non-restrictive. Here is the Julia solution:

```

using JuMP, HiGHS

bigM = 1e6

model = Model(HiGHS.Optimizer)

@variable(model, xa >= 0)
@variable(model, xb >= 0)
@variable(model, y, Bin)

@objective(model, Max, 30*xa + 40*xb)

@constraint(model, 7*xa + 8*xb <= 5600)
@constraint(model, 2*xa + 3*xb <= 2000 + bigM * (1-y))
@constraint(model, 4*xa + 2*xb <= 2400 + bigM * y)

optimize!(model)

println("Optimal value of xa: ", value(xa))
println("Optimal value of xb: ", value(xb))
println("Optimal value of y: ", value(y))
println("Optimal objective value: ", objective_value(model))
println("Solver status: ", termination_status(model))
  
```

The output:

```

Optimal value of xa: 0.0
Optimal value of xb: 700.0
Optimal value of y: -0.0
Optimal objective value: 28000.0
Solver status: OPTIMAL
  
```

As we see from the output, Machine 2 is selected ($y = 0$), and the optimal production quantities are 0 for Product A and 700 for Product B, resulting in a maximum profit of 28000.