Escuela de Primavera en

IV Investigación de Operaciones

28 al 30 de abril de 2025 Modalidad Virtual

“Modelado Matemático Moderno: Caso Práctico con Julia”

Parte 2

Presentación por

Eduardo Salazar Treviño

Contenido

- Sesión 2: Modelado con JuMP y técnicas avanzadas
  - Ejercicios de Modelado:
    - Programación Lineal
    - Problema de Asignación
    - Problema de Bin Packing
  - Modelado avanzado
    - TSP con formulación MTZ
    - TSP con lazy constraints

Requisitos Previos

- Tener instalado Julia (https://julialang.org/install/)
- Tener instalado Visual Studio Code (NO Visual Studio) (https://code.visualstudio.com/)
- Solver de Optimización: Gurobi, CPLEX, HiGHS

Ejercicios de Modelado

Ejercicio 1: Programación Lineal Básica

Una empresa produce dos tipos de bebidas: Regular y Light. Cada bebida requiere agua y concentrado en las siguientes cantidades:

- Bebida Regular: 5 litros de agua y 2 litros de concentrado
- Bebida Light: 6 litros de agua y 1 litro de concentrado

La empresa dispone diariamente de 60 litros de agua y 15 litros de concentrado. El beneficio por cada litro de Bebida Regular es de $3, mientras que para la Bebida Light es de $2.

Respuesta

Ejercicio 2:
Problema de Asignación

Una empresa debe asignar 5 trabajadores a 5 tareas diferentes. Cada trabajador puede realizar cualquier tarea, pero con diferentes niveles de eficiencia.

Cada trabajador debe ser asignado exactamente a una tarea, y cada tarea debe ser realizada por exactamente un trabajador. El objetivo es minimizar el tiempo total para completar todas las tareas.

Respuesta

Ejercicio 3:
Problema de Bin Packing

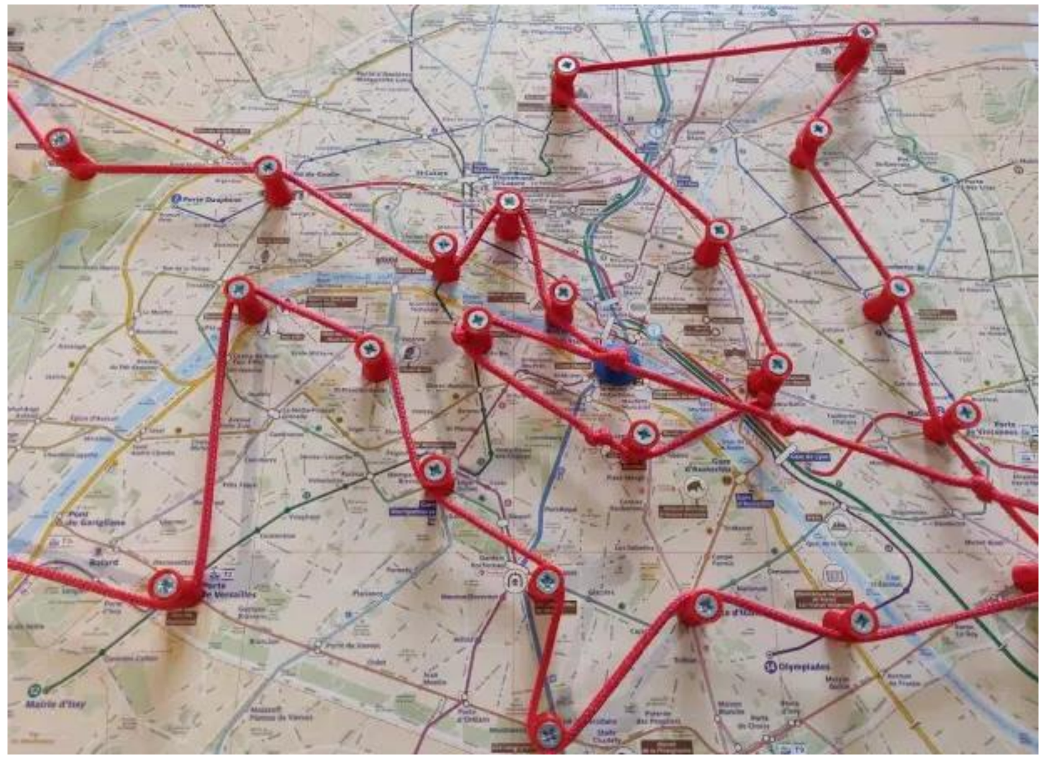
Se tienen n objetos con pesos $w_1, w_2, \ldots, w_n$ que deben ser empaquetados en contenedores de capacidad C. El objetivo es minimizar el número de contenedores utilizados.

Respuesta

Modelado Avanzado

TSP

Dada una lista de ciudades y las distancias entre cada par de ellas, ¿cuál es la ruta más corta posible que visita cada ciudad exactamente una vez y al finalizar regresa a la ciudad origen?



Formulación MTZ

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & & (1)\\
\text{s.t.} \quad & \sum_{j=1, j\neq i}^{n} x_{ij} = 1 & \forall i \in \{1,2,\ldots,n\} \quad & (2)\\
& \sum_{i=1, i\neq j}^{n} x_{ij} = 1 & \forall j \in \{1,2,\ldots,n\} \quad & (3)\\
& u_i - u_j + n x_{ij} \leq n-1 & \forall i,j \in \{2,\ldots,n\}, i \neq j \quad & (4)\\
& x_{ij} \in \{0,1\} & \forall i,j \in \{1,2,\ldots,n\} \quad & (5)\\
& u_i \geq 0 & \forall i \in \{1,2,\ldots,n\} \quad & (6)
\end{aligned}
$$

Enlista las restricciones de eliminación de subtours de manera cuadrática, es fácil de programar pero lento.

Formulación MTZ

```
model = Model(Gurobi.Optimizer)

@variable(model, x[1:n, 1:n], Bin) # x[i,j] = 1 si vamos de ciudad i a j
@variable(model, u[1:n] >= 0) # Variables auxiliares para eliminar subcircuitos

# Función objetivo: minimizar la distancia total
@objective(model, Min, sum(dist_matrix[i, j] * x[i, j] for i in 1:n, j in 1:n))

# Restricciones
for i in 1:n
  # Cada ciudad tiene exactamente una salida
  @constraint(model, sum(x[i, j] for j in 1:n if i != j) == 1)
  # Cada ciudad tiene exactamente una entrada
  @constraint(model, sum(x[j, i] for j in 1:n if i != j) == 1)
end
# Restricciones MTZ para eliminar subcircuitos
for i in 2:n, j in 2:n
  if i != j
    @constraint(model, u[i] - u[j] + n * x[i, j] <= n - 1)
  end
end

optimize!(model)
```

Formulación Lazy Constraints

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1, j\neq i}^{n} x_{ij} = 1 \qquad \forall i \in 1,2,\ldots,n \tag{2}$$

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1 \qquad \forall j \in 1,2,\ldots,n \tag{3}$$

$$\sum_{i\in S}\sum_{j\in S, j\neq i} x_{ij} \leq |S| - 1 \qquad \forall S \subset 1,2,\ldots,n, 2 \leq |S| < n \tag{4}$$

$$x_{ij} \in 0,1 \qquad \forall i,j \in 1,2,\ldots,n \tag{5}$$

S representa cualquier subconjunto de ciudades, la restricción 4 elimina los subtours al manejar que para cualquier *S*, el número de aristas usados en él sea como máximo |*S*| - 1, se agregan de manera “lazy” al modelo en lugar de enumerar todas al inicio.

Formulación Lazy Constraints

Nos vamos al código acompañante.

Sesión de Preguntas

Gracias…