Tarea 1: Formulación de Problemas de Optimización

Juan Esteban Aguirre Olarte – 202210404 – je.aguirreo1

Formulación Matemática:

Para dar solución al problema de optimización de la mochila tridimensional (3DSKP) de dimensiones WxHxD[1] se identificaron los siguientes conjuntos, parámetros, variables de decisión, restricciones y función objetivo. Esto para definir el modelo de optimización a resolver.

A continuación, se presenta la formulación matemática para el problema 3DSKP pensando para una mochila de 5x5x5. No obstante, también está pensado para el caso general, solo que los conjuntos tienen ejemplos ilustrativos a un caso específico.

I. Conjuntos

I: Fichas -> $I = \{0,1,2,3, \dots, n\} \mid n = 11$

I_A: Fichas del tipo A -> $I_A = \{0,1,2,3,4,5\}$

I_B: Fichas del tipo B -> $I_B = \{6,7,8,9,10,11\}$

P: Posición de la ficha -> $P = \{(1,2,4), (1,4,2), \dots, (2,3,2), (3,2,2)\} \mid (w, h, d)$

P_A: Posición de la ficha A[2] -> $P_A = \{(1,2,4), (1,4,2), (2,1,4), (2,4,1), (4,1,2), (4,2,1)\}$

P_B: Posición de la ficha B[3] -> $P_B = \{(2,2,3), (2,3,2), (3,2,2)\}$

$PP_i: Posiciones\ posibles\ de\ la\ ficha\ i$
$= \{0: [(1,2,4), (1,4,2), (2,1,4), (2,4,1), (4,1,2), (4,2,1)], \dots, 11: [(2,2,3), (2,3,2), (3,2,2)]$

$M = 10$

$W = 5$

$H = 5$

$D = 5$

II. Parámetros

$w_p: Largo\ de\ la\ ficha\ en\ la\ posición\ p \in P$

$h_p: Alto\ de\ la\ ficha\ en\ la\ posición\ p \in P$

$d_p: Profundidad\ de\ la\ ficha\ en\ la\ posición\ p \in P$

III. Variables de Decisión

---

[1] Se pensó el problema de modo que el eje x sea el largo (de izquierda a derecha), el eje y el alto (de abajo a arriba) y el eje z sea la profundidad (de atrás hacia delante). En consecuencia, el origen y punto de referencia de las fichas está en la esquina inferior posterior izquierda.

[2] Dadas las dimensiones de la ficha tipo A (1x2x4) esta tiene 6 orientaciones posibles.

[3] Dadas las dimensiones de la ficha tipo B (2x2x3) esta tiene 3 orientaciones posibles ya que su largo y alto es el mismo.

Tarea 1: Formulación de Problemas de Optimización

Juan Esteban Aguirre Olarte – 202210404 – je.aguirreo1

s_i: $\begin{cases} 1, & si\ la\ ficha\ i \in I\ entra\ en\ la\ mochila \\ 0, & d.l.c \end{cases}$

x_i: $Coordenada\ de\ la\ ficha\ i \in I\ en\ el\ eje\ x$

y_i: $Coordenada\ de\ la\ ficha\ i \in I\ en\ el\ eje\ y$

z_i: $Coordenada\ de\ la\ ficha\ i \in I\ en\ el\ eje\ z$

o_{ip}: $\begin{cases} 1, & si\ la\ ficha\ i \in I\ lleva\ la\ posición\ p \in P \begin{cases} i < 6 & P = P_A \\ i \geq 6 & P = P_B \end{cases} \\ 0, & d.l.c \end{cases}$

b_{ij}^{d}: $\begin{cases} 1, & si\ la\ ficha\ i \in I\ está\ dispuesta\ de\ la\ forma\ d \in D\ respecto\ a\ j \in I \\ 0, & d.l.c \end{cases}$

IV. Función Objetivo

Maximizar el volumen ocupado por las fichas.

$$\mathbf{max}\left(\sum_{i\in I}\sum_{p\in PP_i} w_p h_p d_p o_{ip}\right)$$

V. Restricciones

(1) Orientación única

$$\sum_{p\in P} o_{ip} = s_i, \quad \forall i \in I \mid \begin{cases} i < 6 & P = P_A \\ i \geq 6 & P = P_B \end{cases}$$

(2) Contenencia

$$x_i + \sum_{p\in PP_i} w_p o_{ip} \leq W + M(1 - s_i), \forall i \in I$$

$$y_i + \sum_{p\in PP_i} h_p o_{ip} \leq H + M(1 - s_i), \forall i \in I$$

$$z_i + \sum_{p\in PP_i} d_p o_{ip} \leq D + M(1 - s_i), \forall i \in I$$

(3) No solapamiento

$$x_i + \sum_{p\in P} w_p o_{ip} \leq x_j + M\left(1 - b_{ij}^{0}\right) + M(2 - s_i - s_j), \qquad \forall i, j \in I \mid i < j$$

$$x_j + \sum_{p\in P} w_p o_{jp} \leq x_i + M\left(1 - b_{ij}^{1}\right) + M(2 - s_i - s_j), \qquad \forall i, j \in I \mid i < j$$

$$y_i + \sum_{p\in P} h_p o_{ip} \leq y_j + M\left(1 - b_{ij}^{2}\right) + M(2 - s_i - s_j), \qquad \forall i, j \in I \mid i < j$$

Tarea 1: Formulación de Problemas de Optimización

Juan Esteban Aguirre Olarte – 202210404 – je.aguirreo1

$$y_j + \sum_{p \in P} h_p o_{jp} \leq y_i + M\left(1 - b_{ij}^3\right) + M(2 - s_i - s_j), \qquad \forall i, j \in I \mid i < j$$

$$z_i + \sum_{p \in P} d_p o_{ip} \leq z_j + M\left(1 - b_{ij}^4\right) + M(2 - s_i - s_j), \qquad \forall i, j \in I \mid i < j$$

$$z_j + \sum_{p \in P} d_p o_{jp} \leq z_i + M\left(1 - b_{ij}^5\right) + M(2 - s_i - s_j), \qquad \forall i, j \in I \mid i < j$$

(4) Ubicación respecto a otra ficha

$$\sum_{d \in D} b_{ij}^d \geq 1 \;\; \forall i, j \in I \mid i < j$$

(5) Naturaleza

$$x_i \in \mathbb{N}^+, \forall\, i \in I$$

$$y_i \in \mathbb{N}^+, \forall\, i \in I$$

$$z_i \in \mathbb{N}^+, \forall\, i \in I$$

$$s_i \in \{0,1\};\; \forall\, i \in I$$

$$o_{ip} \in \{0,1\};\; \forall\, i \in I, p \in P \begin{cases} i < 6 & P = P_A \\ i \geq 6 & P = P_B \end{cases}$$

$$b_{ij}^d \in \{0,1\};\; \forall\, i, j \in I, d \in D$$

Implementación de Código:

Se implementó el modelo de optimización en el lenguaje de programación Python usando la librería Gurobi. El código se encuentra en la sección de Anexos o en el repositorio. (https://github.com/Asguirrent/IIND-4109.git)

Resultados Obtenidos:

En lo que respecta a la solución del problema del 3DSKP (para el caso 5x5x5 de 12 fichas), se encontró una solución óptima que logra maximizar la cantidad de fichas en la mochila, es decir, acomodar las 12 fichas.

Las posiciones de cada ficha se presentan a continuación:

Ficha 0:

Posición: (0, 0, 1)

Orientación: (2, 1, 4)

Ficha 1:

Posición: (3, 4, 0)

Orientación: (2, 1, 4)

Tarea 1: Formulación de Problemas de Optimización

Juan Esteban Aguirre Olarte – 202210404 – je.aguirreo1

Ficha 2:

Posición: (0, 1, 0)

Orientación: (1, 4, 2)

Ficha 3:

Posición: (0, 3, 4)

Orientación: (4, 2, 1)

Ficha 4:

Posición: (4, 0, 3)

Orientación: (1, 4, 2)

Ficha 5:

Posición: (1, 0, 0)

Orientación: (4, 2, 1)

Ficha 6:

Posición: (2, 0, 3)

Orientación: (2, 3, 2)

Ficha 7:

Posición: (0, 3, 2)

Orientación: (3, 2, 2)

Ficha 8:

Posición: (0, 1, 2)

Orientación: (2, 2, 3)

Ficha 9:

Posición: (1, 2, 0)

Orientación: (2, 3, 2)

Ficha 10:

Posición: (2, 0, 1)

Orientación: (3, 2, 2)

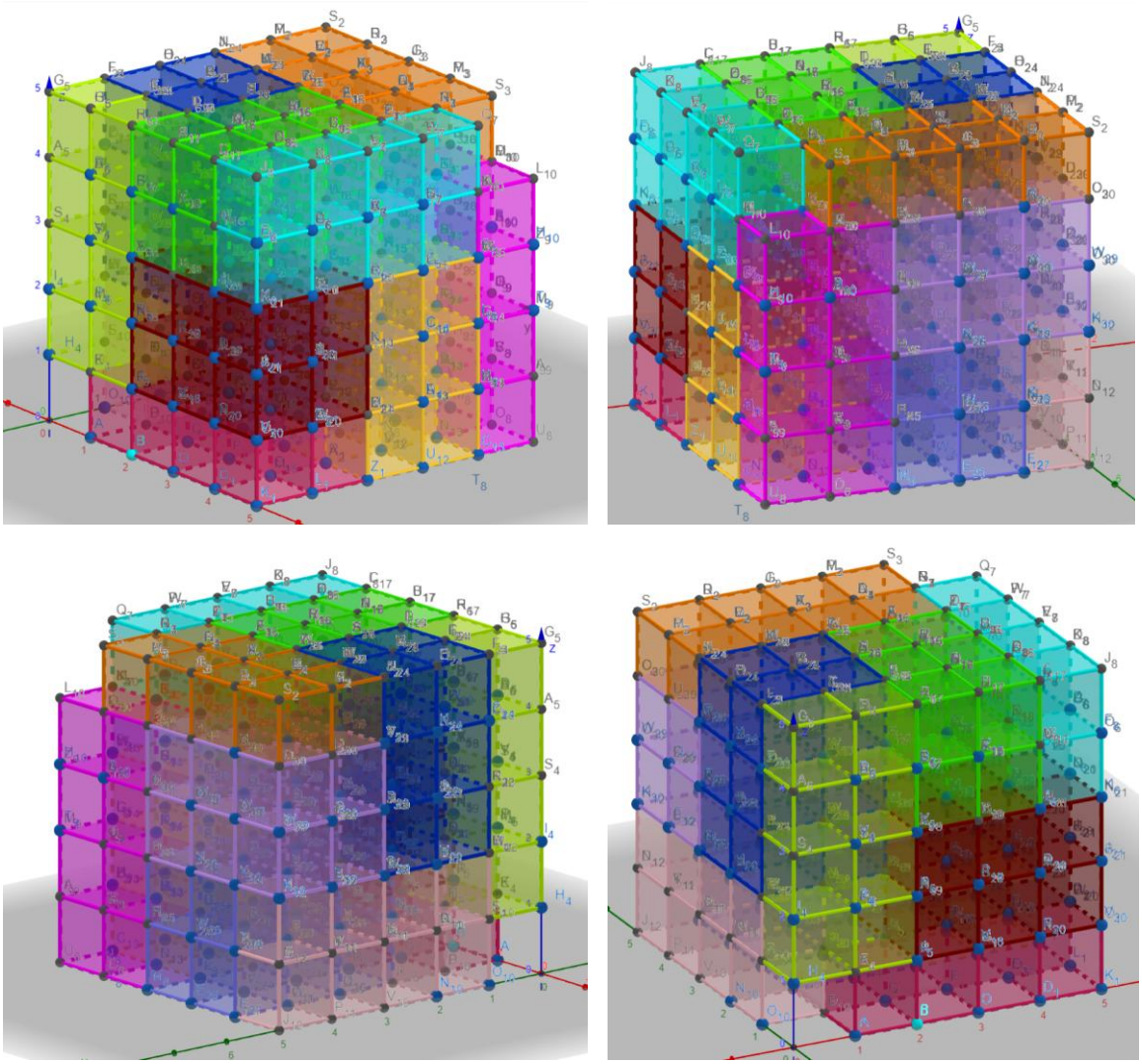
Ficha 11:

Posición: (3, 2, 0)

Orientación: (2, 2, 3)

Tarea 1: Formulación de Problemas de Optimización
Juan Esteban Aguirre Olarte – 202210404 – je.aguirreo1

Usando la herramienta Geogebra 3D se representó la solución mostrada anteriormente, nótese como ninguna ficha se solapa y las 12 fichas están contenidas en la mochila de 5x5x5.



Referencias:

Egeblad, J., & Pisinger, D. (2008). Heuristic approaches for the two- and three-dimensional knapsack packing problem. Computers & Operations Research, 36(4), 1026-1049. https://doi.org/10.1016/j.cor.2007.12.004

Mamedov, K. S., Mamedov, K. K., & Yolchuyeva, S. K. (2016). Erratum to: "Solving the Mixed-Integer Knapsack Problem by Decrease of Dimension and Use of Dynamic Programming". Automatic Control And Computer Sciences, 50(1), 63. https://doi.org/10.3103/s0146411616010107

Pisinger, D. (1999). Linear Time Algorithms for Knapsack Problems with Bounded Weights. Journal Of Algorithms, 33(1), 1-14. https://doi.org/10.1006/jagm.1999.1034

Tarea 1: Formulación de Problemas de Optimización

Juan Esteban Aguirre Olarte – 202210404 – je.aguirreo1

Soto, D., Soto, W., & Pinzón, Y. (2013). A Parallel Nash Genetic Algorithm for the 3D Orthogonal Knapsack Problem. International Journal of Combinatorial Optimization Problems and Informatics, 4(3), 2-10.

Anexos

```python
import gurobipy as gp
from gurobipy import GRB
import itertools

model = gp.Model("3D_Knapsack_MaxVol")

# Parámetros
W= 5  # Tamaño de la mochila
D=5
H=5
I = list(range(12))  # 12 fichas
P_a = [(1,2,4), (1,4,2), (2,1,4), (2,4,1), (4,1,2), (4,2,1)]
P_b = [(2,2,3),(2,3,2),(3,2,2),(2,2,3),(2,3,2),(3,2,2)]
PP={}
for i in I:
    if i<4:
        PP[i]=P_a
    else:
        PP[i]=P_b
M = 10  # Big-M mayor que S

# Variables
s = model.addVars(I, vtype=GRB.BINARY, name="s")  # Selección de fichas
x = model.addVars(I, lb=0, ub=W, vtype=GRB.INTEGER, name="x")
y = model.addVars(I, lb=0, ub=H, vtype=GRB.INTEGER, name="y")
z = model.addVars(I, lb=0, ub=D, vtype=GRB.INTEGER, name="z")
o = {}  # Orientaciones
for i in I:
    orientaciones = P_a if i < 6 else P_b
    for k in range(len(orientaciones)):
        o[(i, k)] = model.addVar(vtype=GRB.BINARY, name=f"o_{i}_{k}")

# Función objetivo: Volumen total
volume = gp.quicksum(
   (PP[i][k][0] * PP[i][k][1] * PP[i][k][2]) * o[i, k]
    for i in I for k in range(6)
)
model.setObjective(volume, GRB.MAXIMIZE)

# Restricciones
```

Tarea 1: Formulación de Problemas de Optimización
Juan Esteban Aguirre Olarte – 202210404 – je.aguirreo1

```
for i in I:
    orient = P_a if i <6 else P_b
    # Selección-orientación
    model.addConstr(gp.quicksum(o[i, k] for k in range(len(orient))) ==
s[i])
    # Contención con Big-M
    w_i = gp.quicksum(orient[k][0] * o[i, k] for k in range(len(orient)))
    h_i = gp.quicksum(orient[k][1] * o[i, k] for k in range(len(orient)))
    d_i = gp.quicksum(orient[k][2] * o[i, k] for k in range(len(orient)))
    model.addConstr(x[i] + w_i <= W + M * (1 - s[i]))
    model.addConstr(y[i] + h_i <= H + M * (1 - s[i]))
    model.addConstr(z[i] + d_i <= D + M * (1 - s[i]))

# No solapamiento (solo entre fichas seleccionadas)
for i, j in itertools.combinations(I, 2):
    orient_i = P_a if i < 4 else P_b
    w_i = gp.quicksum(orient_i[k][0] * o[i, k] for k in
range(len(orient_i)))
    h_i = gp.quicksum(orient_i[k][1] * o[i, k] for k in
range(len(orient_i)))
    d_i = gp.quicksum(orient_i[k][2] * o[i, k] for k in
range(len(orient_i)))

    orient_j = P_a if j < 4 else P_b
    w_j = gp.quicksum(orient_j[k][0] * o[j, k] for k in
range(len(orient_j)))
    h_j = gp.quicksum(orient_j[k][1] * o[j, k] for k in
range(len(orient_j)))
    d_j = gp.quicksum(orient_j[k][2] * o[j, k] for k in
range(len(orient_j)))

    # Variables de dirección (b[i,j,m])
    b = model.addVars(6, vtype=GRB.BINARY, name=f"b_{i}_{j}")
    model.addConstr(gp.quicksum(b[m] for m in range(6)) >= 1)

    # Restricciones con Big-M ajustado para s_i y s_j
    model.addConstr(x[i] + w_i <= x[j] + M*(1 - b[0]) + M*(2 - s[i] - s[j]))
    model.addConstr(x[j] + w_j <= x[i] + M*(1 - b[1]) + M*(2 - s[i] - s[j]))
    model.addConstr(y[i] + h_i <= y[j] + M*(1 - b[2]) + M*(2 - s[i] - s[j]))
    model.addConstr(y[j] + h_j <= y[i] + M*(1 - b[3]) + M*(2 - s[i] - s[j]))
    model.addConstr(z[i] + d_i <= z[j] + M*(1 - b[4]) + M*(2 - s[i] - s[j]))
    model.addConstr(z[j] + d_j <= z[i] + M*(1 - b[5]) + M*(2 - s[i] - s[j]))

model.optimize()
```

Tarea 1: Formulación de Problemas de Optimización
Juan Esteban Aguirre Olarte – 202210404 – je.aguirreo1

```
# Resultados
if model.status == GRB.OPTIMAL:
    print(f"Volumen total: {model.objVal}")
    for i in I:
        if s[i].X > 0:
            orient = P_a if i < 6 else P_b
            #print('3')
            for p in range(len(orient)):
                #print('4')
                if o[i,p].X >0:
                    #print('5')
                    print(f"Ficha {i}:")
                    print(f"  Posición: ({x[i].X:.0f}, {y[i].X:.0f},
{z[i].X:.0f})")
                    print(f"  Orientación: {orient[p]}")
                    break
else:
    print("No solución óptima")
```