Formulación Matemática:

Para dar solución al problema de optimización de la mochila tridimensional (3DSKP) de dimensiones WxHxD[[1]](#footnote-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 tiene ejemplos ilustrativos a un caso especifico.

1. **Conjuntos**

*I: Fichas ->*

*IA: Fichas del tipo A ->*

*IB: Fichas del tipo B ->*

*P: Posición de la ficha -> P*

*PA: Posición de la ficha A[[2]](#footnote-2) ->*

*PB: Posición de la ficha B[[3]](#footnote-3)->*

1. **Parámetros**
2. **Variables de Decisión**

:

1. **Función Objetivo**

Maximizar el volumen ocupado por las fichas.

1. **Restricciones**

(1) Orientación única

(2) Contenencia

(3) No solapamiento

(4) Ubicación respecto a otra ficha

(5) Naturaleza

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)

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)

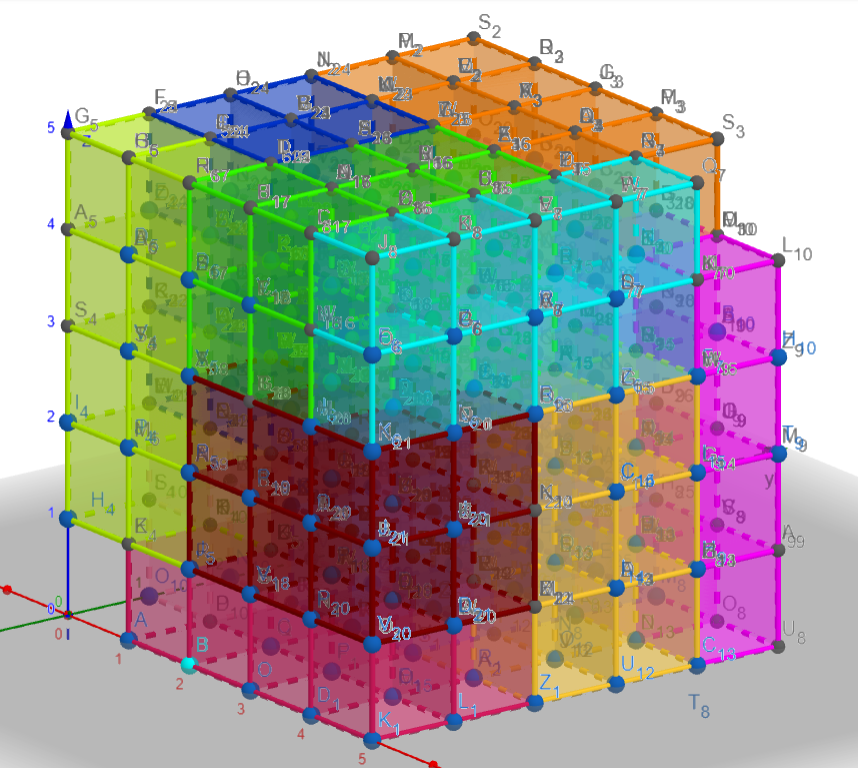
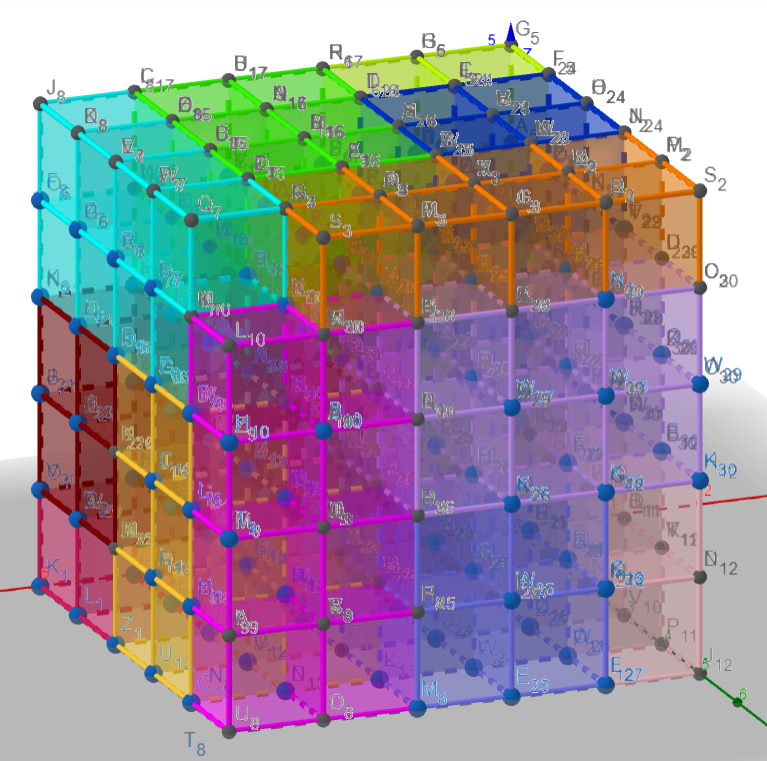
Orientación: (3, 2, 2)

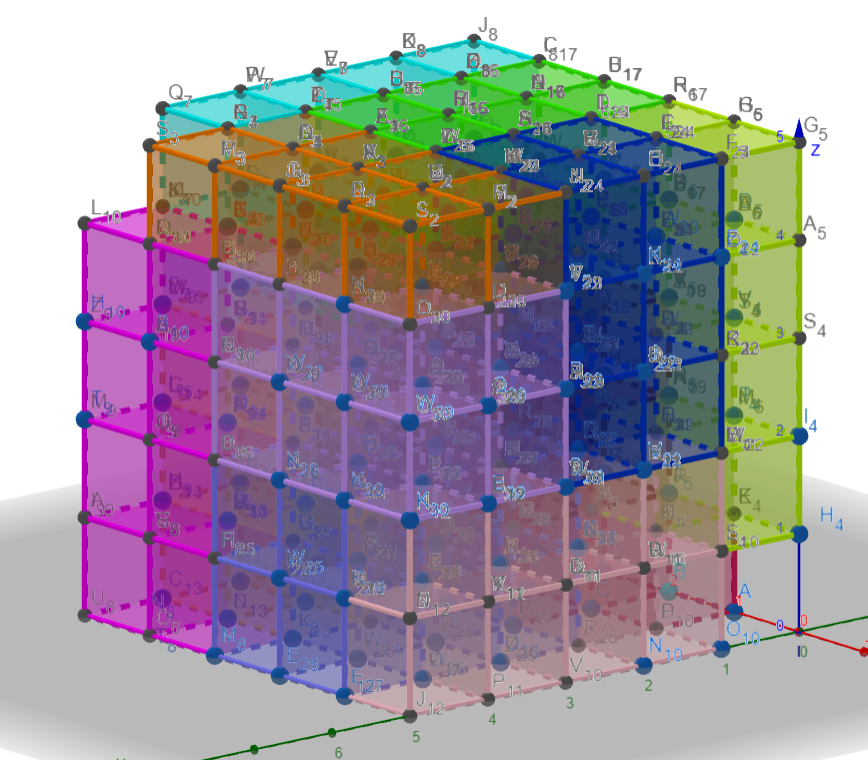
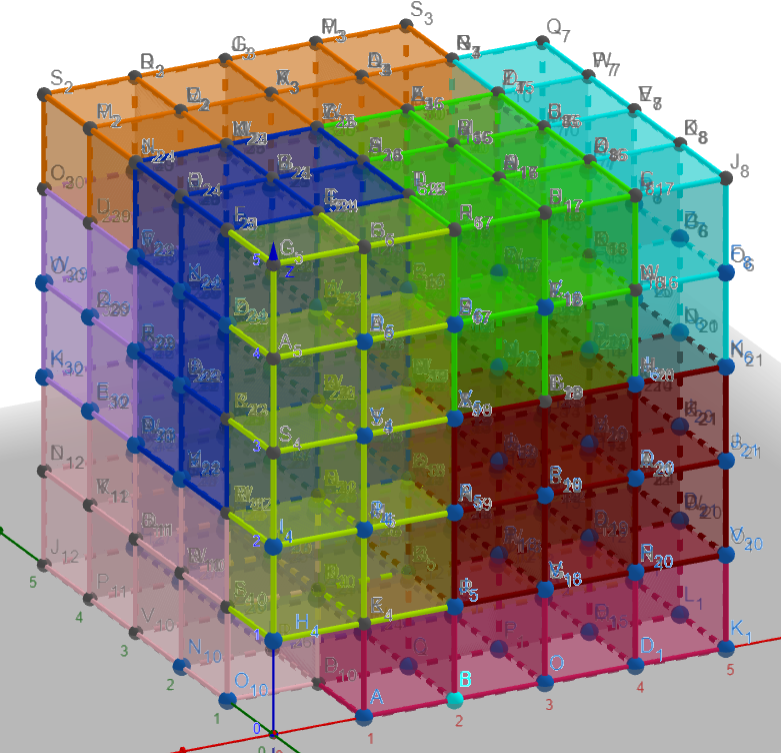
Ficha 11:

Posición: (3, 2, 0)

Orientación: (2, 2, 3)

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

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

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

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()

# 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")

1. Se pensó el problema de modo que el eje sea el largo (de izquierda a derecha), el eje el alto (de abajo a arriba) y el eje 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. [↑](#footnote-ref-1)
2. Dadas las dimensiones de la ficha tipo A (1x2x4) esta tiene 6 orientaciones posibles. [↑](#footnote-ref-2)
3. Dadas las dimensiones de la ficha tipo B (2x2x3) esta tiene 3 orientaciones posibles ya que su largo y alto es el mismo. [↑](#footnote-ref-3)