Problema de optimización de compras por Internet

Integrantes:
Juan Andrés Romero (202013449) y Juan Sebastián Alegría (202011282)
Entrega 1: Primera aproximación del Modelo Matemático
Modelado, Simulación y Optimización

Departamento de Ingeniería de Sistemas y Computación
Universidad de Los Andes
Bogotá, Colombia

1 Descripción del Problema

1.1 Contexto del problema:

Dada la amplia cantidad de tiendas online que existen actualmente, surge la dificultad de los compradores de revisar manualmente todas las ofertas disponibles y seleccionar las mejores elecciones de compra manteniendo un costo bajo.
Para obtener una solución parcial al problema, existen los llamados comparadores de precio que generan un ranking de precios de dichas ofertas, sin embargo, estos están limitados a trabajar con la comparación de ofertas de un único producto al tiempo.

Ahora bien, la solución más óptima, que es la que se desea implementar, requiere comprobar todas las combinaciones de costos existentes para de este modo, obtener el menor costo de compra posible dado por la combinación entre costos de envío y de productos. Este problema es de naturaleza NP-hard.

1.2 Descripción formal del problema:

Dado un comprador que busca un conjunto de productos N para comprar en las tiendas de L, existe un conjunto de productos disponibles P, un costo C_{jl} de cada producto y un costo D_l de envío que están asociados a cada tienda.

1.3 Limitaciones del problema

Como restricciones se tiene que:

- Si el producto no está disponible, no se puede comprar (costo infinito) y por ende se descarta de N.
- De los productos disponibles que quiere el comprador, es necesario tener solamente uno de ellos seleccionado en alguna de las tiendas.

- El costo de envío de los productos de una tienda, es independiente de la cantidad de productos comprados.

Objetivo que se quiere lograr: Minimizar la sumatoria de costos de envío de tiendas seleccionadas sumado a la sumatoria del conjunto de productos seleccionados en una tienda. En resumen, se quiere minimizar el costo total de los productos seleccionados que a su vez tengan los mínimos costos de envío.

1.4 Ejemplo

A continuación se presenta una muestra de ejemplo, en la cual se evalúan 6 tiendas y se desea comprar 5 libros. Luego, en la segunda tabla se evidencia la solución limitada que usualmente ofrecen los comparadores de precio. Por otro lado, la tercera tabla muestra la solución óptima de menor costo.

Table 1. Prices of books and delivery costs offered by six internet shops.

Cost	Book a	Book b	Book c	Book d	Book e	Delivery	Total
Shop 1	18	39	29	48	59	10	203
Shop 2	24	45	23	54	44	15	205
Shop 3	22	45	23	53	53	15	211
Shop 4	28	47	17	57	47	10	206
Shop 5	24	42	24	47	59	10	206
Shop 6	27	48	20	55	53	15	218

Table 2. Price comparator solution—the result of the selection process.

	Book a	Book b	Book c	Book d	Book e	Delivery	Total
Price	18	39	17	47	44	45	210
Shop	Shop 1	Shop 1	Shop 4	Shop 5	Shop 2		

Table 3. Optimal purchase cost in selected shops.

	Book a	Book b	Book c	Book d	Book e	Delivery	Total
Cost	18	39	17	48	47	20	189
Shop	Shop 1	Shop 1	Shop 4	Shop 1	Shop 4		

Como se evidencia en el anterior ejemplo, dado un set de costos productos por cada tienda y un costo de entrega para esa tienda, un comparador de precios intenta elegir el libro de su tipo que tenga el menor costo y luego suma los costos de envío correspondientes, pero esto no garantiza obtener el menor costo total de compra.

De este modo, si probamos todas las posibles combinaciones de costos se elige

el libro d y e de una tienda con un costo más caro para estos, pero al final obtenemos un costo total menor. Por esto, este problema es NP-Hard.

2 Conjuntos, Parámetros y Variables de decisión

A continuación se presenta la simbología para los conjuntos, parámetros y variables a utilizar en el problema

Table 1. Conjuntos y parámetros.

Sets and Parameters	**Description**
P	Conjunto de Productos.
L	Conjunto de Tiendas.
N	Conjunto de Productos disponibles que quiere el comprador
C_{jl}	Costo de cada producto j en la tienda l
D_l	Costo de envío desde la tienda l

Table 2. Variables de decisión

Variables	**Description**
X_{jl}	Determina si un producto j de una tienda l es escogido o no para ser comprado (Variable Binaria)
Y_l	Determina si se realiza una compra en la tienda l. (Variable Binaria)

3 Función Objetivo y Restricciones

3.1 Función Objetivo

$$min\left(\sum_{l\in L}\left(\sum_{j\in P}(C_{jl}X_{jl})+D_lY_l\right)\right) \tag{1}$$

Explicación: La función objetivo busca minimizar la sumatoria de los costos de los productos elegidos para todas las tiendas. Si una tienda tiene al menos un producto elegido, la variable binaria Y_l nos permitirá incluir a la sumatoria final el costo de envío correspondiente a dicha tienda.

3.2 Restricciones:

Si se compra en l, se activa Y_l

$$\sum_{j \in P} X_{jl} - card(P) Y_l \leq 0 \qquad \forall l \in L \tag{2}$$

Explicación: Esta restricción indica que si se elige al menos un producto de una tienda l, se tiene que activar la variable Y, pues esta significa que se está realizando una compra en la tienda l.

Nota: Card(P) significa la cardinalidad del conjunto P, se eligió esta como constante para multiplicar la Y en la ecuación, debido a que por otras restricciones, la suma de las X_{ji} nunca será igual a ella.

Se deben de comprar los productos que quiere el comprador

$$\sum_{l \in L} X_{jl} = 1 \qquad \forall j \in N \tag{3}$$

Explicación: En esta restricción se indica que por cada producto disponible que quiere el comprador se tiene que comprar exactamente 1 de ellos. Es decir, no es posible comprar más de un producto específico por tienda.

Nota: Para descartar los productos no disponibles de N, se debe de realizar un preprocesamiento del problema a partir de los costos de cada producto. Esto se puede ver como un ciclo antes del cálculo del modelo que va a agregando a N los productos que están disponibles para comprar.

4 Implementación y resultados del Modelo Matemático

Para la comprobación de la implementación del modelo matemático se decidió implementar dos escenarios diferentes, uno sencillo y uno más complejo. Para el primero, se realizará un escenario simple con 3 tiendas y 3 productos, de los cuales, el cliente solo quiere 2 de ellos. De igual manera, se colocarán valores de precios similares de manera que no haya una solución evidente para este problema.

Por otro lado, en el segundo escenario se usarán los mismos datos enseñados en la prueba de ejemplo de la sección 1.4 de este documento y se intentará revisar si los resultados del modelo implementado en Pyomo coinciden.
En este caso hay un problema con 6 tiendas y 5 productos diferentes, donde el cliente quiere obtener por lo menos uno de cada uno. Esta selección de elementos, hace que el segundo escenario sea más complejo porque la cantidad de posibilidades a elegir aumenta con la cantidad de productos que quiere el cliente.

4.1 Escenario 1

Para el primer escenario, se utilizarán únicamente 3 tiendas y 3 productos, haciendo también que el cliente quiera solo dos productos. Para este caso, dichos productos serán: Producto A y Producto C.

La distribución de precios por tienda se puede evidenciar en la siguiente tabla

Table 3. Escenario 1. Precio de productos y costo de envío ofrecidos por 3 tiendas online

Costo	Producto A	Producto B	Producto C	Envío
Tienda 1	7	6	15	12
Tienda 2	13	4	10	10
Tienda 3	10	6	9	15

4.2 Resultados Escenario 1

En la siguiente imagen se puede ver que tras la ejecución del algoritmo de solución con Pyomo, se eligió únicamente la tienda 2 para obtener el menor costo de la compra de A y C.
Igualmente, en esta selección se llegó a un costo total de 33 unidades. Esto demuestra que la solución más factible no fue la tienda que tenía los precios más bajos sino otra que era más cara, pero su costo de envío era más bajo.

```
project on  main via 🐍 v3.10.8
➜ /opt/homebrew/bin/python3 /Users/zejiran/hack/uniandes/MSO/project/scenario1.py
Model unknown

  Variables:
    x : Size=9, Index=x_index
        Key    : Lower : Value : Upper : Fixed : Stale : Domain
        (1, 1) :     0 :   0.0 :     1 : False : False : Binary
        (1, 2) :     0 :   1.0 :     1 : False : False : Binary
        (1, 3) :     0 :   0.0 :     1 : False : False : Binary
        (2, 1) :     0 :   0.0 :     1 : False : False : Binary
        (2, 2) :     0 :   0.0 :     1 : False : False : Binary
        (2, 3) :     0 :   0.0 :     1 : False : False : Binary
        (3, 1) :     0 :   0.0 :     1 : False : False : Binary
        (3, 2) :     0 :   1.0 :     1 : False : False : Binary
        (3, 3) :     0 :   0.0 :     1 : False : False : Binary
    y : Size=3, Index=L
        Key : Lower : Value : Upper : Fixed : Stale : Domain
          1 :     0 :   0.0 :     1 : False : False : Binary
          2 :     0 :   1.0 :     1 : False : False : Binary
          3 :     0 :   0.0 :     1 : False : False : Binary

  Objectives:
    targetFunc : Size=1, Index=None, Active=True
        Key  : Active : Value
        None :   True :  33.0
```

4.3 Escenario 2

En este segundo escenario se implementará el ejemplo mostrado en la sección 1.4, el cual estaba conformado por 6 tiendas, 5 diferentes productos y el cliente quería obtener todos los anteriores productos, es decir del A al E. En este caso, los valores de los pares son los mismos de dicho ejemplo.

Table 4. Escenario 2. Precio de productos y costo de envío ofrecidos por 6 tiendas online

Costo	Producto A	Producto B	Producto C	Producto D	Producto E	Envío	Total
Tienda 1	18	39	29	48	59	10	203
Tienda 2	24	45	23	54	44	15	205
Tienda 3	22	45	23	53	53	15	211
Tienda 4	28	47	17	57	47	10	206
Tienda 5	24	42	24	47	59	10	206
Tienda 6	27	48	20	55	53	15	218

4.4 Resultados Escenario 2

Tras ejecutar el modelo, se identificó que los resultados coincidían satisfactoriamente con lo expuesto en el ejemplo explicado anteriormente.

Se puede ver que los pares elegidos fueron: Tienda 1 para los productos 1, 2 y 4, y tienda 4 para los productos 3 y 5. Resultando con un costo de compra total de 189, el cual es mucho menor que si se hubiera comprado en la tienda con el costo total más barato.

Ejecución del segundo escenario

```
project on  main via 🐍 v3.10.8
➜ /opt/homebrew/bin/python3 /Users/zejiran/hack/uniandes/MSO/project/scenario2.py
Model unknown

  Variables:
    x : Size=30, Index=x_index
        Key    : Lower : Value : Upper : Fixed : Stale : Domain
        (1, 1) :     0 :   1.0 :     1 : False : False : Binary
        (1, 2) :     0 :   0.0 :     1 : False : False : Binary
        (1, 3) :     0 :   0.0 :     1 : False : False : Binary
        (1, 4) :     0 :   0.0 :     1 : False : False : Binary
        (1, 5) :     0 :   0.0 :     1 : False : False : Binary
        (1, 6) :     0 :   0.0 :     1 : False : False : Binary
        (2, 1) :     0 :   1.0 :     1 : False : False : Binary
        (2, 2) :     0 :   0.0 :     1 : False : False : Binary
        (2, 3) :     0 :   0.0 :     1 : False : False : Binary
        (2, 4) :     0 :   0.0 :     1 : False : False : Binary
        (2, 5) :     0 :   0.0 :     1 : False : False : Binary
        (2, 6) :     0 :   0.0 :     1 : False : False : Binary
        (3, 1) :     0 :   0.0 :     1 : False : False : Binary
        (3, 2) :     0 :   0.0 :     1 : False : False : Binary
        (3, 3) :     0 :   0.0 :     1 : False : False : Binary
        (3, 4) :     0 :   1.0 :     1 : False : False : Binary
        (3, 5) :     0 :   0.0 :     1 : False : False : Binary
        (3, 6) :     0 :   0.0 :     1 : False : False : Binary
        (4, 1) :     0 :   1.0 :     1 : False : False : Binary
        (4, 2) :     0 :   0.0 :     1 : False : False : Binary
        (4, 3) :     0 :   0.0 :     1 : False : False : Binary
        (4, 4) :     0 :   0.0 :     1 : False : False : Binary
        (4, 5) :     0 :   0.0 :     1 : False : False : Binary
        (4, 6) :     0 :   0.0 :     1 : False : False : Binary
        (5, 1) :     0 :   0.0 :     1 : False : False : Binary
        (5, 2) :     0 :   0.0 :     1 : False : False : Binary
        (5, 3) :     0 :   0.0 :     1 : False : False : Binary
        (5, 4) :     0 :   1.0 :     1 : False : False : Binary
        (5, 5) :     0 :   0.0 :     1 : False : False : Binary
        (5, 6) :     0 :   0.0 :     1 : False : False : Binary
    y : Size=6, Index=L
        Key : Lower : Value : Upper : Fixed : Stale : Domain
          1 :     0 :   1.0 :     1 : False : False : Binary
          2 :     0 :   0.0 :     1 : False : False : Binary
          3 :     0 :   0.0 :     1 : False : False : Binary
          4 :     0 :   1.0 :     1 : False : False : Binary
          5 :     0 :   0.0 :     1 : False : False : Binary
          6 :     0 :   0.0 :     1 : False : False : Binary

  Objectives:
    targetFunc : Size=1, Index=None, Active=True
        Key  : Active : Value
        None :   True : 189.0
```