

GRASP para el problema de fijación de precios basado en preferencias

XVI Congreso Nacional sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB)

Raúl Fauste-Jiménez (r.fauste.2020@alumnos.urjc.es)

Sergio Salazar (sergio.salazar@urjc.es)

Isaac Lozano-Osorio (isaac.lozano@urjc.es)

Jesús Sánchez-Oro (jesus.sanchezoro@urjc.es)



Universidad
Rey Juan Carlos



grafo
RESEARCH

Índice

1 Introducción

- Definición formal del problema
- Revisión de la literatura

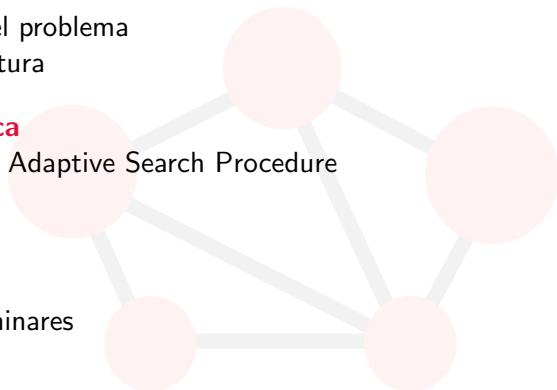
2 Propuesta algorítmica

- Greedy Randomized Adaptive Search Procedure
 - Fase constructiva
 - Fase de mejora

3 Resultados

- Experimentos preliminares
- Experimento final

4 Conclusiones



Introducción

- El *Rank Pricing Problem* (**RPP**) es un problema $\mathcal{NP}$ -**Completo** que busca **maximizar el beneficio** de una empresa, asignando el precio a sus productos, en función de unos presupuestos y preferencias.
- Cada **cliente** **comprará**, como mucho, **un solo producto** y se considera una **cantidad ilimitada** de cada **producto**.
- Aplicaciones del mundo real:
 - Retroalimentación empresarial.
 - Industria automovilística o electrónica.

Introducción

Definición formal del problema

Dado un conjunto de **productos** $I = \{i_1, \dots, i_n\}$ y un conjunto de **clientes** $K = \{k_1, \dots, k_t\}$, donde cada $k \in K$ tiene un subconjunto de productos preferidos $I^k \subseteq I$ tal que k no comprará ningún producto que no pertenezca a I^k . Cada cliente $k \in K$ tiene un valor de preferencia para cada producto $i \in I$, representado por p_i^k .

Cada cliente tiene un **presupuesto** fijo (b).

La **solución** consiste en obtener un **vector de precios** S que determine el precio de cada producto y que **maximice** el beneficio.

Introducción

Modelo

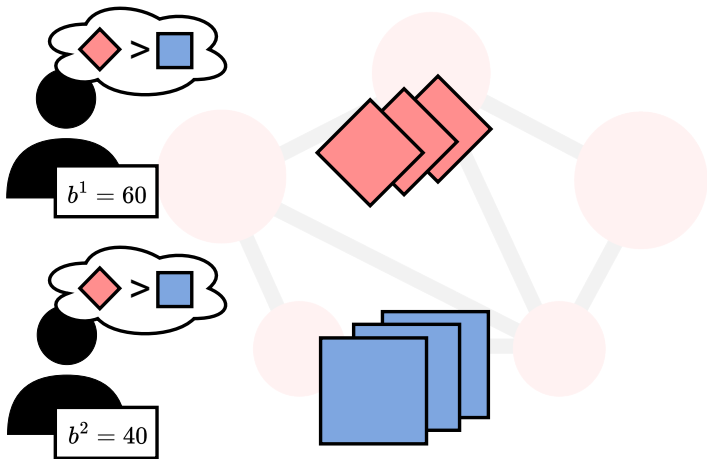
$$\begin{aligned}
 \max_S \quad & \sum_{k \in K} \sum_{i \in I^k} s_i x_i^k \\
 \text{s.t.} \quad & s_i \geq 0, \quad \forall i \in I,
 \end{aligned} \tag{1}$$

donde $\forall k \in K$, x_i^k es la solución óptima del problema mostrado en la Ecuación (2).

$$\begin{aligned}
 \max_{x^k} \quad & \sum_{i \in I^k} p_i^k x_i^k \\
 \text{s.t.} \quad & \sum_{i \in I^k} x_i^k \leq 1, \\
 & \sum_{i \in I^k} s_i x_i^k \leq b^k, \\
 & x_i^k \in \{0, 1\}, \quad \forall i \in I^k, \forall k \in K
 \end{aligned} \tag{2}$$

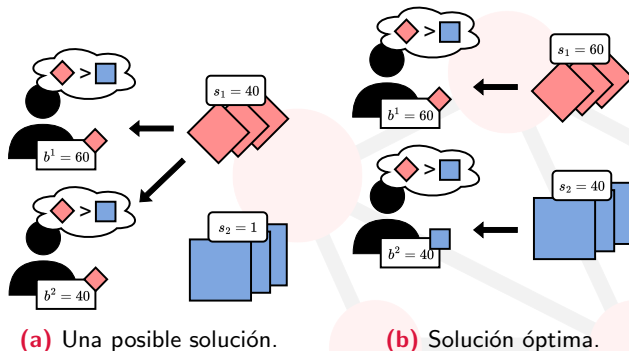
Introducción

Ejemplo



Introducción

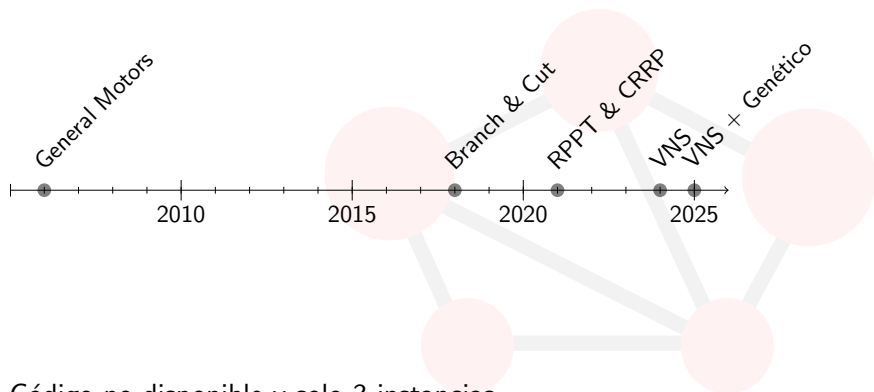
Ejemplo



Los **precios** de la **solución óptima** pertenecen al conjunto de presupuestos.

Introducción

Revisión de la literatura



Código no disponible y solo 3 instancias.

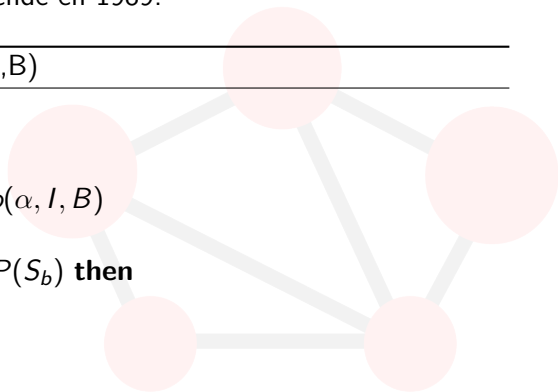
Propuesta algorítmica

Greedy Randomized Adaptive Search Procedure (GRASP)

Propuesto por Feo y Resende en 1989.

Algoritmo 1 GRASP(α, I, B)

```
1:  $S_b \leftarrow \emptyset$ 
2: for  $i \in 1 \dots it$  do
3:    $S \leftarrow \text{Constructivo}(\alpha, I, B)$ 
4:    $S \leftarrow \text{Mejora}(S, I)$ 
5:   if  $RPP(S) > RPP(S_b)$  then
6:      $S_b \leftarrow S$ 
7:   end if
8: end for
9: return  $S_b$ 
```



Propuesta algorítmica

Fase constructiva

- El criterio **voraz** elegido es el **basado en función objetivo** (pueden ocurrir empates).
- Se establece un **umbral** para determinar los candidatos entre los que elegir aleatoriamente.

$$U = v_{max} - \alpha \cdot (v_{max} - v_{min})$$

- Se utiliza un parámetro α para calcular el **umbral** que determina el espacio a la aleatoriedad.

Propuesta algorítmica

Fase de mejora

Se usa el operador *Intercambiar*(S, i, j) que **intercambiará los precios** asignados a los elementos $i, j \in I$. En concreto, dada una solución S compuesta por la asignación:

$$S = (s_1, \dots, s_{i-1}, s_i, s_{i+1}, \dots, s_{j-1}, s_j, s_{j+1}, \dots, s_n)$$

intercambiará los valores de los elementos i, j , obteniendo la solución:

$$S = (s_1, \dots, s_{i-1}, s_j, s_{i+1}, \dots, s_{j-1}, s_i, s_{j+1}, \dots, s_n)$$

Se analizan las dos conocidas **estrategias de exploración**: *first improvement* y *best improvement*.

Resultados

Entorno experimental

- Lenguaje de programación: **Java 21**.
- Características del servidor: Ubuntu Server 20.04 en **AMD EPYC 7282, 16 cores y 8 GB** de RAM.
- Métricas:
 - **F.O**: media de la función objetivo.
 - **Tiempo (s)**: tiempo medio de ejecución en segundos.
 - **GAP(%)**: desviación media de la mejor solución conocida.
 - **#Óptimos**: veces que el algoritmo es capaz de alcanzar la solución óptima del experimento.
- Instancias: 3 + 33 **nuevas** con un **presupuesto** de 1 a 1000, para los **clientes** de 30 a 200, y de 5 a 150 **productos**.

Resultados

Experimentos preliminares

Se utilizó irace con un **conjunto preliminar** del 20% de instancias para obtener la mejor configuración.

Los **mejores parámetros** encontrados fueron:

- Búsqueda local: *Best Improvement*.
- Aleatoriedad: $\alpha = 0.01$.
- Condición de parada: 87 iteraciones.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., y Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives, 3, 43–58.

Resultados

Experimento final: GRASP vs Estado del Arte

Propuesta más reciente del estado del arte: *Variable Neighborhood Search* (VNS).

Instancias	GRASP				VNS			
	F.O	Tiempo (s)	GAP (%)	#Óptimo	F.O	Tiempo (s)	GAP (%)	#Óptimo
30x5	807	0,18	0,00	1	807	600,00	0,00	1
30x25	1018	1,93	2,30	0	960	600,00	7,87	0
60x50	1962	39,30	2,73	0	1842	600,00	8,68	0
Total	1262,33	13,74	1,68	1	1203,00	600,00	5,52	1

Tabla 1: Comparativa GRASP frente a VNS.

Las 3 instancias del estado del arte van de 30 a 60 **clientes**, y de 5 a 50 **productos**.

Resultados

Experimento final: GRASP vs Gurobi

Las **nuevas instancias** generadas van de 100 a 200 **clientes** y de 25 a 150 **productos**.

Algoritmo	F.O.	Tiempo (s)	GAP. (%)	#Óptimos
GRASP	3762,89	201,59	2,85%	1
Gurobi	38833,72	12465,24	0,00%	18

Tabla 2: Comparativa GRASP frente a Gurobi.

Gurobi **no** es capaz de **generar una solución** para instancias ≥ 200 **clientes** y ≥ 50 **productos**.

Resultados

Experimento final: GRASP vs Gurobi

Instancias	F.O	Tiempo (s)	Instancias	F.O	Tiempo (s)
200x50_0	85146	678,62	200x100_0	90803	4929,35
200x50_1	85269	666,88	200x100_1	95345	4278,20
200x50_2	84620	675,62	200x100_2	98065	4493,50
200x50_3	88054	629,00	200x100_3	94881	4338,71
200x50_4	88629	624,93	200x100_4	85678	4672,00
200x150_0	93015	17039,00	200x150_3	97056	16426,00
200x150_1	96212	15124,42	200x150_4	99797	14056,74
200x150_2	102741	15087,95			

Tabla 3: Resultados GRASP en instancias $K \geq 200$ e $I \geq 50$.

Conclusiones

✓ Los resultados son **prometedores**.

🕒 Trabajos futuros:

- Obtener el **código del estado del arte** para poder evaluar frente al resto de instancias.
- Añadir alguna componente que fomente la **diversidad** ($\alpha = 0.01$).
- **Estudiar los parámetros** más detalladamente.
- Incorporar técnicas de **aprendizaje automático** para guiar la exploración del espacio de soluciones.
- Trabajar en **problemas relacionados** como **RPPT** y **CRPP**.

Agradecimientos

Este trabajo ha sido financiado gracias a:

- **Ministerio de Ciencia e Innovación** con ref. PID2021-126605NB-I00 y PID2021-125709OA-C22 financiado por MCIN /AEI /10.13039/501100011033 and by ERDF A way of making Europe.
- **CIRMA**: “This work has been supported by Comunidad Autónoma de Madrid, CIRMA-CM Project (TEC-2024/COM-404)”.
- **Cátedra ENIA**: Ministerio para la Transformación Digital y de la Función Pública (Ref. TSI-100930-2023-3).



GRASP para el problema de fijación de precios basado en preferencias

XVI Congreso Nacional sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB)

Raúl Fauste-Jiménez (r.fauste.2020@alumnos.urjc.es)

Sergio Salazar (sergio.salazar@urjc.es)

Isaac Lozano-Osorio (isaac.lozano@urjc.es)

Jesús Sánchez-Oro (jesus.sanchezoro@urjc.es)



Universidad
Rey Juan Carlos



grafo
RESEARCH