Aplicación de CMSA al problema de empaquetado con desbordamiento

XVII Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados

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

Christian Blum (christian.blum@iiia.csic.es)



Universidad Rey Juan Carlos



IIIA-CSIC INSTITUT D'INVESTIGACIÓ EN INTEL·LIGÈNCIA ARTIFICIAL





Introducción

- El problema de **Empaquetado con Desbordamiento** (**OBPP**, *Overflowing Bin Packing Problem*) es un problema $\mathcal{NP}$-**difícil**.

- **Generaliza** los problemas **clásicos de Bin Packing (BPP)** permitiendo que los contenedores se llenen **por encima o por debajo** de su capacidad, penalizando la desviación mediante la variable s_k.

- **Aplicaciones** del mundo real:
  - Planificación *justo a tiempo*.
  - Consolidación de servidores.
  - Gestión de recursos en computación en la nube.

Introducción

Definición del problema

Dado un conjunto de **tipos de ítems** con tamaños c_i, demandas d_i, y un conjunto de **contenedores** con capacidades C_k, se busca asignar todos los ítems a los contenedores.

A diferencia del **BPP clásico**, los contenedores pueden llenarse **por encima o por debajo** de su capacidad, y **todos** contribuyen a la función objetivo, incluso los vacíos.

Sea $\ell_k = \sum_i c_i \cdot x_{ik}$ la **carga** del contenedor k, donde $x_{ik} \in \mathbb{Z}_+$ es el número de ítems de tipo i asignados a k. El objetivo es:

$$\min \sum_k |C_k - \ell_k|$$

Introducción

Modelo

El modelo de **asignación lineal** introduce las variables auxiliares $s_k \geq 0$.

$$\min \sum_{k \in K} s_k$$

$$\text{s.t.} \quad \sum_{k \in K} x_{ik} = d_i, \quad i \in I,$$

$$\sum_{i \in I} c_i \cdot x_{ik} + s_k \geq C_k, \quad k \in K,$$

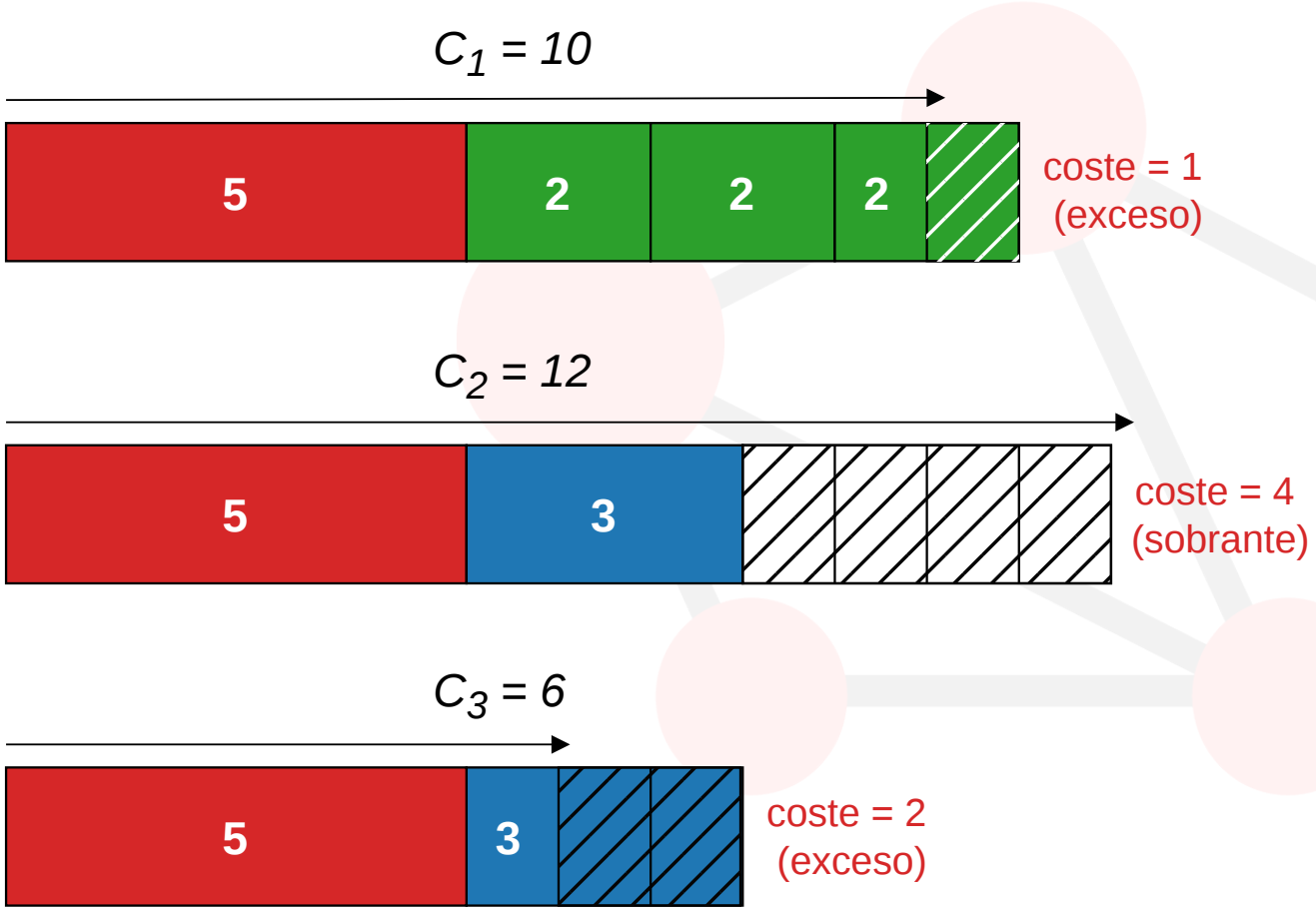
$$\sum_{i \in I} c_i \cdot x_{ik} - s_k \leq C_k, \quad k \in K,$$

$$x_{ik} \in \mathbb{Z}_+, \quad i \in I,\ k \in K.$$

Al minimizar s_k, en la solución óptima se cumple $s_k = |C_k - \ell_k|$.

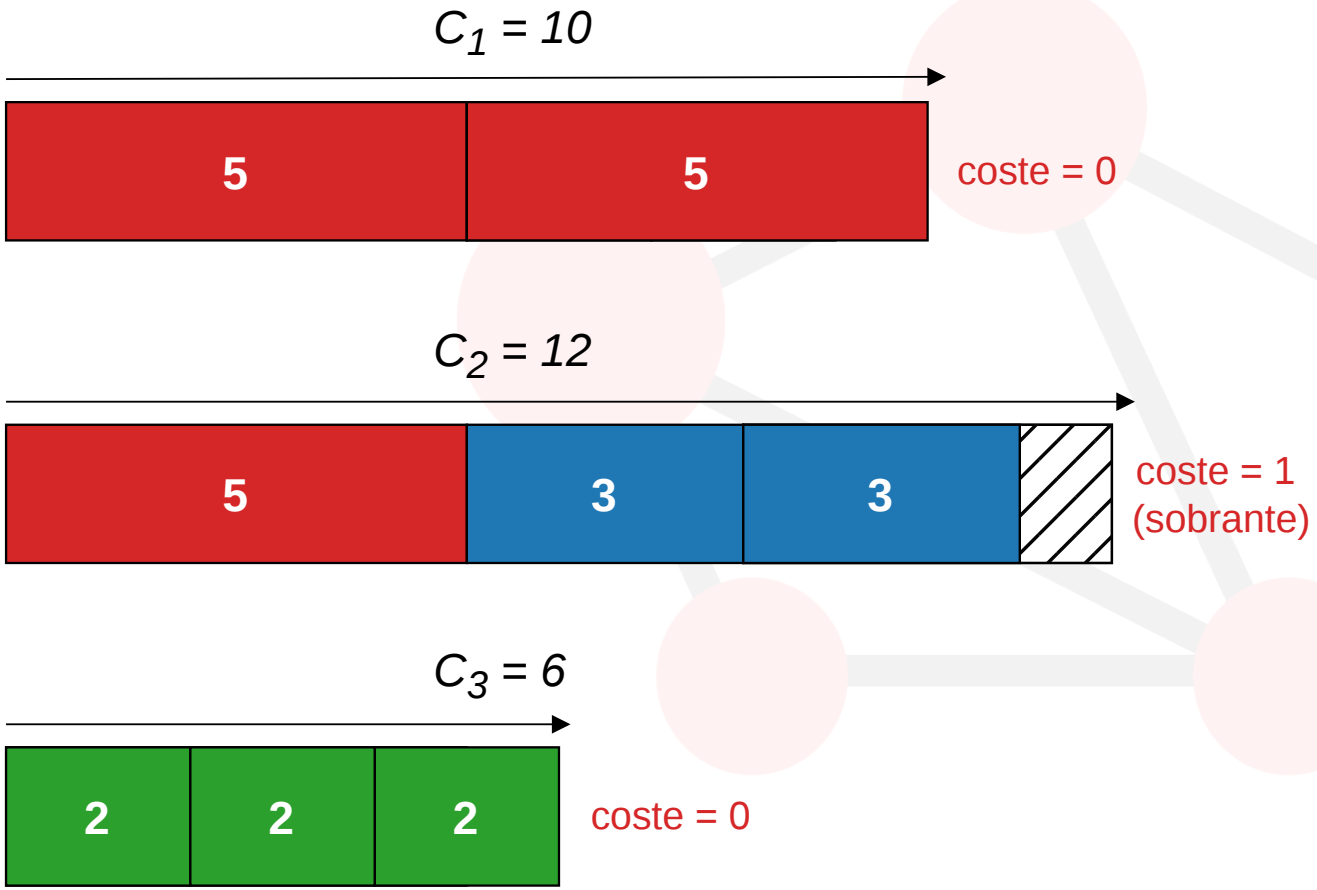
Introducción

Ejemplo



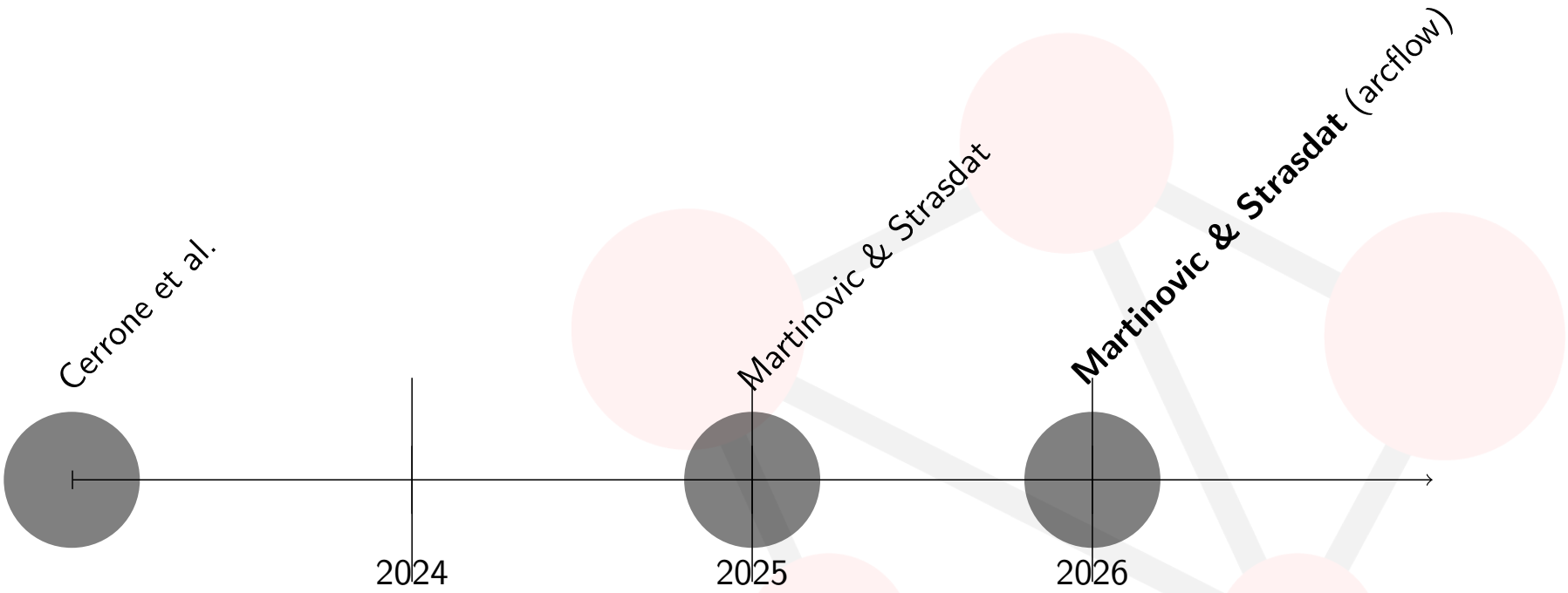
Introducción

Ejemplo



Introducción

Revisión de la literatura



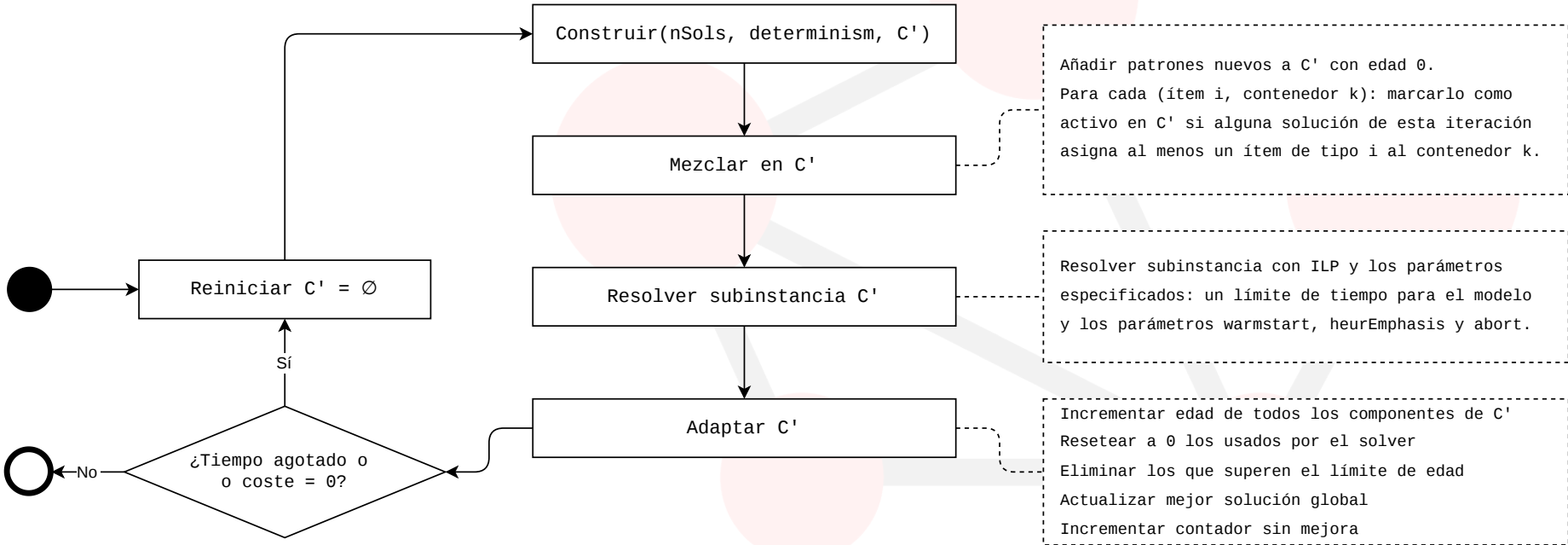
Propuesta algorítmica

Construct Merge Adapt & Solve (CMSA)

- Propuesto por Blum et al. en 2024.
- ¿Por qué CMSA?:
  - Combina la **calidad** de los métodos exactos con la **escalabilidad** de las heurísticas.
  - Resuelve **de forma óptima** subinstancias de tamaño reducido, evitando la explosión combinatoria del problema completo.
  - Requiere **poco diseño específico**: basta con definir los componentes de solución y un constructivo razonable.
  - Especialmente adecuado para problemas donde un **solver ILP** es eficiente en instancias pequeñas pero inviable en las grandes.

Propuesta algorítmica

CMSA



Propuesta algorítmica

Construir

Constructivo voraz aleatorizado por contenedor (**Bin-Completion Greedy**):

- Por contenedor (mayor a menor capacidad), se añaden ítems mientras reduzcan la desviación actual, eligiendo el de **mayor reducción**.
- Los ítems no colocados se asignan con un **voraz de respaldo**, minimizando la desviación resultante.
- Con cierta probabilidad, se sustituye por un ítem **aleatorio** entre los candidatos válidos, introduciendo diversidad.

Propuesta algorítmica

Ejemplo

Contenedor $C_1 = 10$. Ítems disponibles: c_1=5, c_2=2, c_3=3.

- Desviación = 10. Añadir c_1=5 $\Rightarrow$ carga 5, desviación = 5.
- Candidatos: c_2 (desviación $\rightarrow$ 3) o c_3 (desviación $\rightarrow$ 2). Se elige c_3 por mayor reducción $\Rightarrow$ carga 8, desviación = 2.
- Candidatos: c_2 (desviación $\rightarrow$ 0) $\Rightarrow$ se añade $\Rightarrow$ carga 10, desviación = 0.
- No hay más ítems que reduzcan la desviación: el contenedor **se completa** con coste 0.

Propuesta algorítmica

Resolver

- Se construye una **subinstancia restringida** a los componentes activos en C'.
- Se resuelve mediante el modelo **arcflow**, usando **Gurobi** como solver exacto.
- Los ítems cuyo tipo no está activo, o cuya demanda no puede cubrirse con la red restringida, se asignan mediante una **heurística voraz** tras la resolución.

Propuesta algorítmica

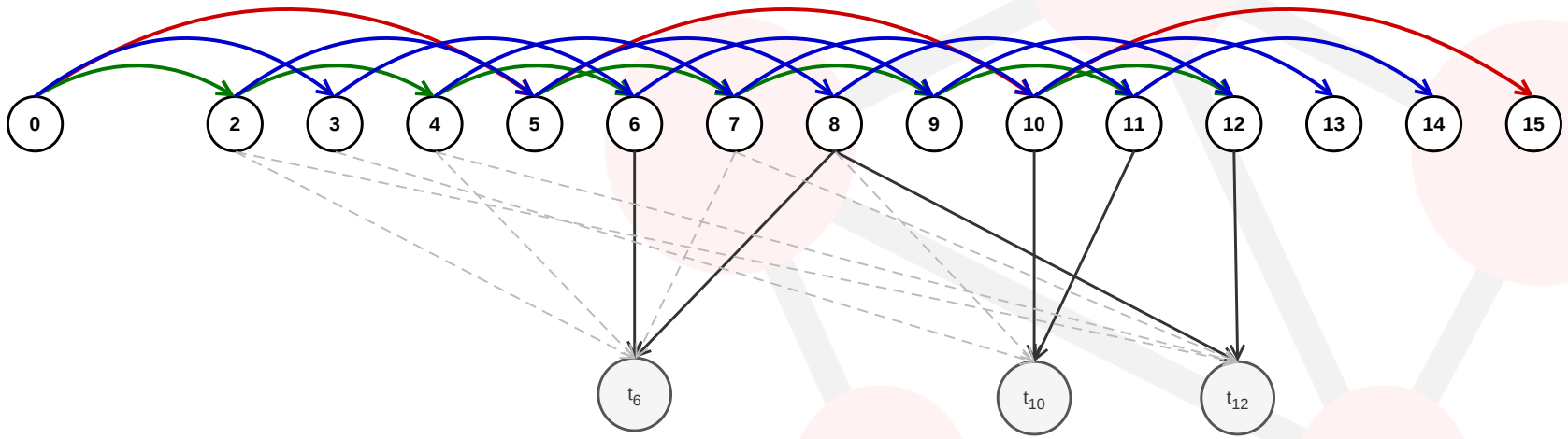
Modelo arcflow

- Cada **nodo** representa un nivel de **carga acumulada** en un contenedor.
- Cada **arista de ítem** $(u \rightarrow u + c_i)$ representa la colocación de un ítem de tipo i.
- Cada **arista de contenedor** $(v \rightarrow t_k)$ conecta con un nodo sumidero, con coste $|C_k - v|$.
- Minimizar el coste del flujo total $\equiv$ minimizar la desviación acumulada de todos los contenedores.

Propuesta algorítmica

Subinstancia restringida a C'

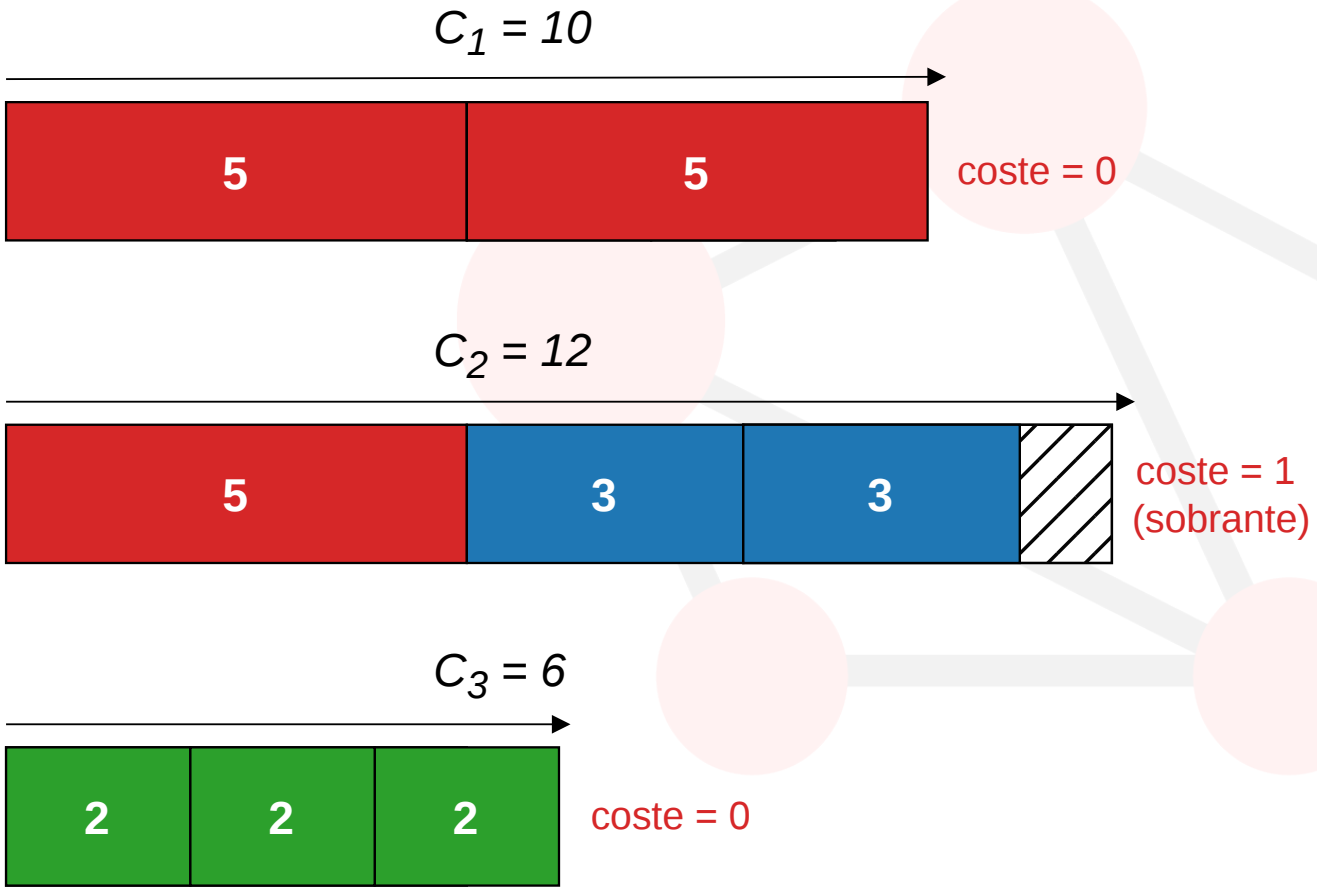
Instancia de ejemplo con tres tipos de ítems (c_1=5 en rojo, c_2=2 en verde, c_3=3 en azul) y tres contenedores (C_1=10, C_2=12, C_3=6).



Solo se conservan aristas cuyo camino usa **tipos activos** en C' para esa capacidad; el resto se **descarta** (gris discontinuo), reduciendo el tamaño de la red.

Propuesta algorítmica

Solución óptima



Propuesta algorítmica

Preprocesamiento de la instancia

Martinovic y Strasdat proponen una técnica de **preprocesamiento** para reducir el tamaño de la instancia antes de resolverla.

Si existe un par (i, k) tal que $c_i = C_k$ (el tamaño de un ítem coincide **exactamente** con la capacidad de un contenedor) siempre existe una solución **óptima** en la que ese contenedor recibe únicamente una copia de ese ítem, con **coste cero**.

Por tanto, tanto el contenedor k como una unidad del ítem i pueden **eliminarse** de la instancia sin alterar el valor óptimo.

Esta reducción se aplica de forma **iterativa** hasta que no existan más pares de este tipo.

Resultados

Entorno experimental

- Lenguaje de programación: **Java 25** sobre **Oracle GraalVM**.
- Características del servidor: **AMD EPYC 9655P**, **32 cores** y **32 GB** de RAM, ejecución monohilo. Solver exacto: **Gurobi 11.0.3**.
- Límite de tiempo: **600 segundos** por instancia.
- Métricas:
  - **F.O:** valor medio de la función objetivo.
  - **Tiempo (s):** tiempo medio en encontrar la mejor solución.
  - **Desv.:** desviación absoluta media respecto al óptimo conocido.
  - **#Óptimos:** número de coincidencias con la solución exacta.

Resultados

Conjuntos de instancias

Conjunto	Instancias	Dificultad
MS1 / MS2	500 + 500	Difícil
Belov_1 - Belov_4	200	Difícil
Crainic_1 - Crainic_3	840	Fácil
Dragone F1-F3	720	Fácil
Hemmelmayr_1 / Hemmelmayr_2	150 + 50	Fácil

Tabla 1: Conjuntos de instancias utilizados en la evaluación.

Difícil: mayor escala o mayor coste computacional.
Fácil: resueltas a optimalidad de forma inmediata.

Resultados

Experimentos preliminares

Se utilizó `irace` sobre un **subconjunto de 39 instancias** de mayor dificultad (Belov, `MS1`, `MS2`) para obtener la mejor configuración.

Los **mejores parámetros** encontrados fueron:

- `nSols` $= 850$ (soluciones construidas por iteración)
- `nRestarts` $= 5$ (reinicios de C')
- `age` $= 4$ (edad máxima de un componente)
- `determinism` $= 0.8$ (determinismo del constructivo)
- `heurEmphasis` $=$ true (prioriza buscar soluciones factibles)
- `abort` $=$ false (sin parada anticipada en la subinstancia)
- `gurobiSeconds` $= 187$ (tiempo máximo por subinstancia)

Resultados

Comparativa ILP (arcflow) vs CMSA (arcflow)

Tabla 2: Resultados computacionales de CMSA comparados con el ILP.

		ILP			CMSA			
Conjunto	Instancias	FO	Óptimos	Tiempo (s)	FO	Óptimos	Tiempo (s)	Desv
MS1	500	**0,00**	**439**	**148,09**	18,94	359	229,24	18,94
MS2	500	**2,31**	**395**	**194,09**	17,68	306	337,20	15,37
Belov_1	50	**43759,88**	**50**	**149,64**	43759,96	46	599,48	0,08
Belov_2	50	**36828,18**	**50**	**136,82**	64328,54	45	601,80	27500,36
Belov_3	50	**27845,38**	**50**	**120,71**	38893,14	46	605,36	11047,76
Belov_4	50	**23833,60**	**50**	**110,28**	71013,24	42	600,78	47179,64
Crainic_1	300	**129,81**	**300**	0,14	**129,81**	**300**	**<0,01**	0,00
Crainic_2	60	**102,30**	**60**	0,26	**102,30**	**60**	**<0,01**	0,00
Crainic_3	480	**59,09**	**480**	0,05	**59,09**	**480**	**<0,01**	0,00
Dragone_F1	240	**0,00**	**240**	1,54	**0,00**	**240**	**<0,01**	0,00
Dragone_F2	240	**0,00**	**240**	2,91	**0,00**	**240**	**<0,01**	0,00
Dragone_F3	240	**730,62**	**240**	0,75	**730,63**	**240**	**<0,01**	0,00
Hemmelmayr_1	150	**3048,30**	**150**	0,96	**3048,30**	**150**	**<0,01**	0,00
Hemmelmayr_2	50	**64,52**	**50**	0,26	**64,52**	**50**	**<0,01**	0,00
Total	**2960**	**2474,25**	**2794**	**67,04**	3928,15	2604	137,82	1453,90

Conclusiones

✔ CMSA alcanza el **óptimo** en los conjuntos: Crainic, Dragone y Hemmelmayr.

✔ Los conjuntos **Belov**, **MS1** y **MS2** suponen un **reto abierto**, con desviaciones considerables.

🕘 **Trabajos futuros:**

- Sustituir el modelo **arcflow** por una formulación basada en **cobertura de conjuntos**.
- Diseñar **operadores de búsqueda local** específicos para el OBPP.
- Explorar estrategias de construcción adaptadas a instancias de gran tamaño.

Resultados

Comparativa ILP (arcflow) vs CMSA (cobertura de conjuntos)

Tabla 3: Resultados computacionales de CMSA comparados con el ILP.

		ILP			CMSA			
Conjunto	Instancias	FO	Óptimos	Tiempo (s)	FO	Óptimos	Tiempo (s)	Desv
MS1	500	**0,00**	439	148,09	0,22	**481**	**19,46**	0,22
MS2	500	**2,31**	395	194,09	2,40	**471**	**31,43**	0,09
Belov_1	50	**43759,88**	**50**	149,64	43763,96	45	**58,03**	4,08
Belov_2	50	**36828,18**	**50**	136,82	36834,50	45	**62,58**	6,32
Belov_3	50	**27845,38**	**50**	120,71	27845,62	49	**75,41**	0,24
Belov_4	50	**23833,60**	**50**	110,28	23834,12	47	**88,43**	0,52
Crainic_1	300	**129,81**	**300**	0,14	**129,81**	**300**	**<0,01**	0,00
Crainic_2	60	**102,30**	**60**	0,26	**102,30**	**60**	**<0,01**	0,00
Crainic_3	480	**59,09**	**480**	0,05	**59,09**	**480**	**<0,01**	0,00
Dragone_F1	240	**0,00**	**240**	1,54	**0,00**	**240**	**<0,01**	0,00
Dragone_F2	240	**0,00**	**240**	2,91	**0,00**	**240**	**<0,01**	0,00
Dragone_F3	240	**730,62**	**240**	0,75	**730,62**	**240**	**<0,01**	0,00
Hemmelmayr_1	150	**3048,30**	**150**	0,96	**3048,30**	**150**	**<0,01**	0,00
Hemmelmayr_2	50	**64,52**	**50**	0,26	**64,52**	**50**	**<0,01**	0,00
Total	**2960**	**2474,25**	2794	67,04	2474,49	**2898**	**13,76**	0,24

Agradecimientos

Este trabajo ha sido financiado gracias a:

- **Ministerio de Ciencia e Innovación** con ref. PID2024-156045NB-I00 financiado por MCIN /AEI /10.13039/501100011033 and by ERDF A way of making Europe.
- **CIRMA**: "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).





Aplicación de CMSA al problema de empaquetado con desbordamiento

XVII Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados

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

Christian Blum (christian.blum@iiia.csic.es)



Universidad Rey Juan Carlos



IIIA-CSIC INSTITUT D'INVESTIGACIÓ EN INTEL·LIGÈNCIA ARTIFICIAL



