Combinación de Metaheurísticas con Solvers ILP en la Optimización Combinatoria

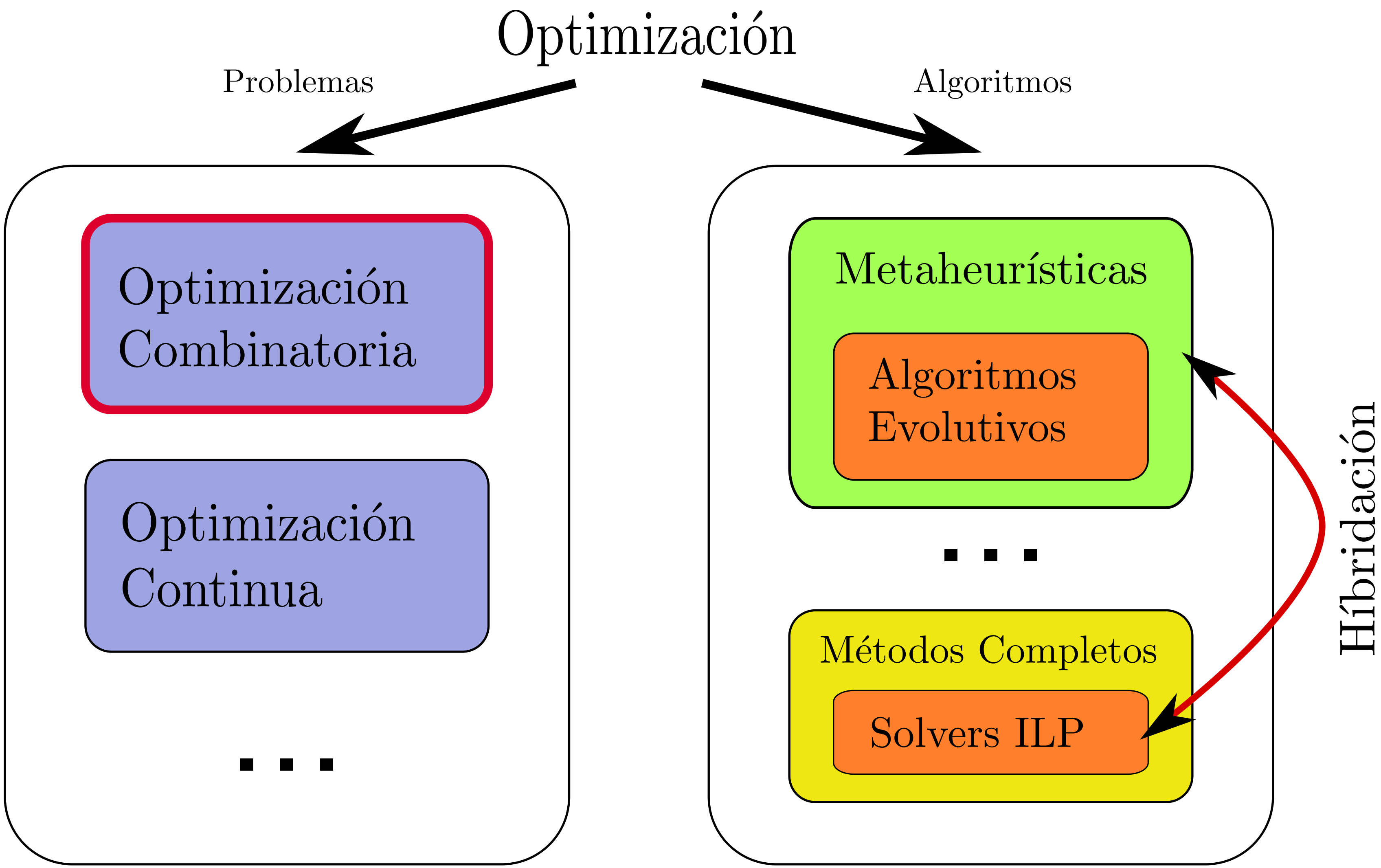
Christian Blum

Universidad Del Pais Vasco, España

Ikerbasque, Basque Foundation For Science

Disfrutando de la Comida

Preparar el Terreno



Motivación y Esquema de la Presentación (1)

Motivación:

- En el campo de las metaheurísticas tenemos algunas reglas de oro:
  1. Si, para el problema considerado, exisite una **buena heurística voraz** aplicamos **GRASP** o un **Método Greedy Iterado**
  2. Si, para el problema considerado, exisite un **vecindario eficiente** aplicamos **Búsqueda Local Iterada** or **Búsqueda Tabú**

- Al contrario, en cuanto a las metaheurísticas híbridas no tenemos reglas semejantes
  - ⋆ Solamente existen **muy pocas técnicas generales**
  - ⋆ No se sabe **para qué tipo de problema** funcionan realmente bien

Motivación y Esquema de la Presentación (2)

Esquema:

- Breve introducción: Metaheurísticas híbridas
- Cómo combinar metaheurísticas con solvers ILP
  - **Método híbrido estándar: Large neighborhood search (LNS)**
- Hipótesis sobre cuando LNS no funciona muy bien
- ¿Qué se puede utilizar en lugar de LNS?
  - **Híbrido reciente: Construct, Merge, Solve & Adapt (CMSA)**

Metaheurísticas Híbridas

Breve Introducción

Metaheurísticas Híbridas: Definición

Definición: ¿Qué es una metaheurística híbrida ?

- Problema: no se puede dar una definición precisa!

Posible caracterización:

Es un algoritmo para la optimización que resulta de la combinació de una metaheurística con una técnica diferente

A qué nos referimos con: una técnica diferente

- Metaheurística
- Ramificación y poda (*en inglés: branch & bound*)
- Programación dinámica
- Programación lineal entera (*en inglés: integer linear programming (ILP)*)

Metaheurísticas Híbridas: Historia

Historia:

- Durante muchos años diferentes comunidades coexistían de forma aislada
- Aunque metaheurísticas híbridas se empezaron a desarollar pronto, solo era de forma esporádica
- Desde hace unos 15 años la literatura sobre híbridos crece de forma signficativa:
  1. 1999: Congreso CP-AI-OR
  2. 2004: Workshop on Hybrid Metaheuristics (HM 200X)
  3. 2006: Matheuristics Workshops

Consecuencia: Hoy día el término metaheurística híbrida identifica una nueva línea de investigación

Tema Específico de esta Presentación

Tema específico: Combinación entre metaheurísticas y solvers ILP

Algunos solvers ILP generales:

- **IBM ILOG CPLEX:** libre para fines académicos
- **Gurobi:** libre para fines académicos
- **FICO Xpress:** 30 días de prueba. Versión gratuita para alumnos
- **MOSEK:** libre para fines académicos

¿Por qué Combinar Metaheurísticas con Solvers ILP?

Ventajas de metaheurísticas:

- Son muy buenas haciendo uso de información sobre el problema (heurísticas greedy)
- Son generalmente buenas en generar, en poco tiempo, soluciones de alta calidad

No obstante:

- Llegan a sus límites al tratar con instancias muy grandes
- Metaheurísticas fallan cuando la información sobre el problema es inadecuada

Objetivo: Aprovechar el valuoso conocimiento experto en el desarrollo de solvers ILP en el contexto de instancias grandes

Large Neighborhood Search basado en Solvers ILP

Idea: Uso de un solver ILP para encontrar la mejor solución en vecindarios extensos de la solución actual

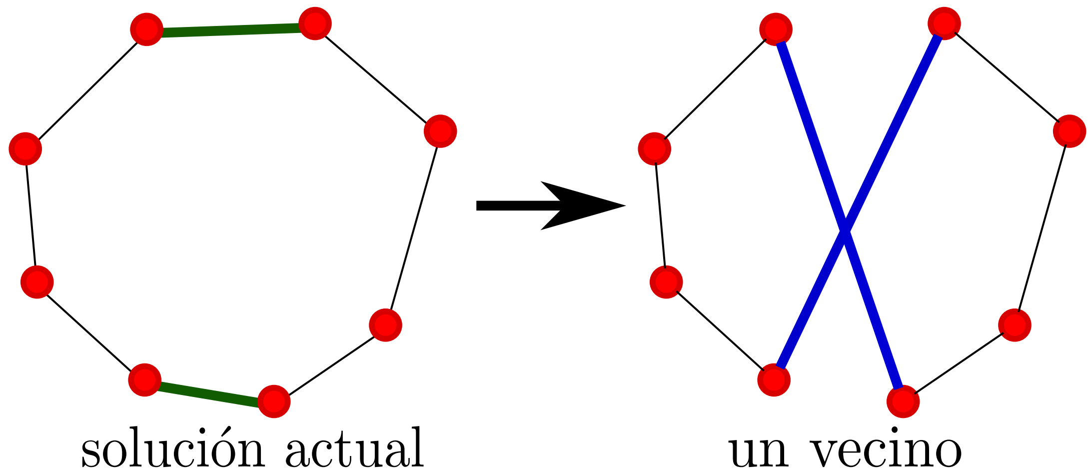
David Pisinger, Stefan Ropke. **Large Neighborhood Search**, *Handbook of Metaheuristics*, International Series in Operations Research & Management Science Volume 146, 2010, pp 399-419

Búsqueda Local (1)

- Ingrediente crucial de la búsqueda local: Elección de un vecindario
- Lo usual en metaheurísticas: vecindarios de tamaños preferiblemente pequeños

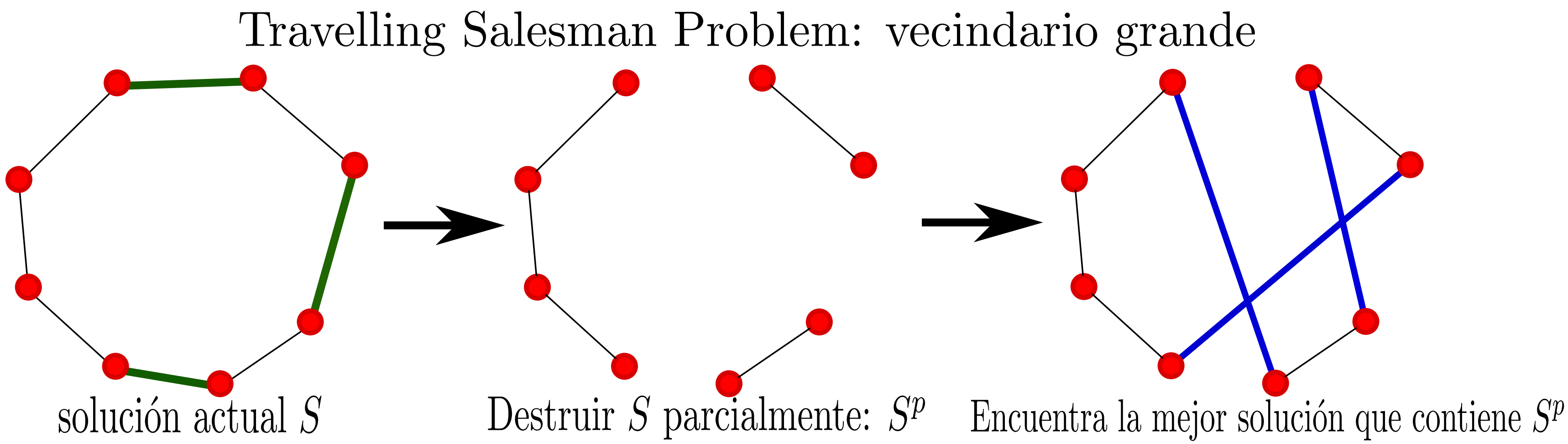
Ejemplo de un vecindario pequeño: vecindario 2-opt para el TSP (problema del viajante de comercio): $O(n^2)$ vecinos.

Traveling Salesman Problem: vecindario 2-opt



Búsqueda Local (2)

Ejemplo de un vecindario grande: en el contexto del TSP



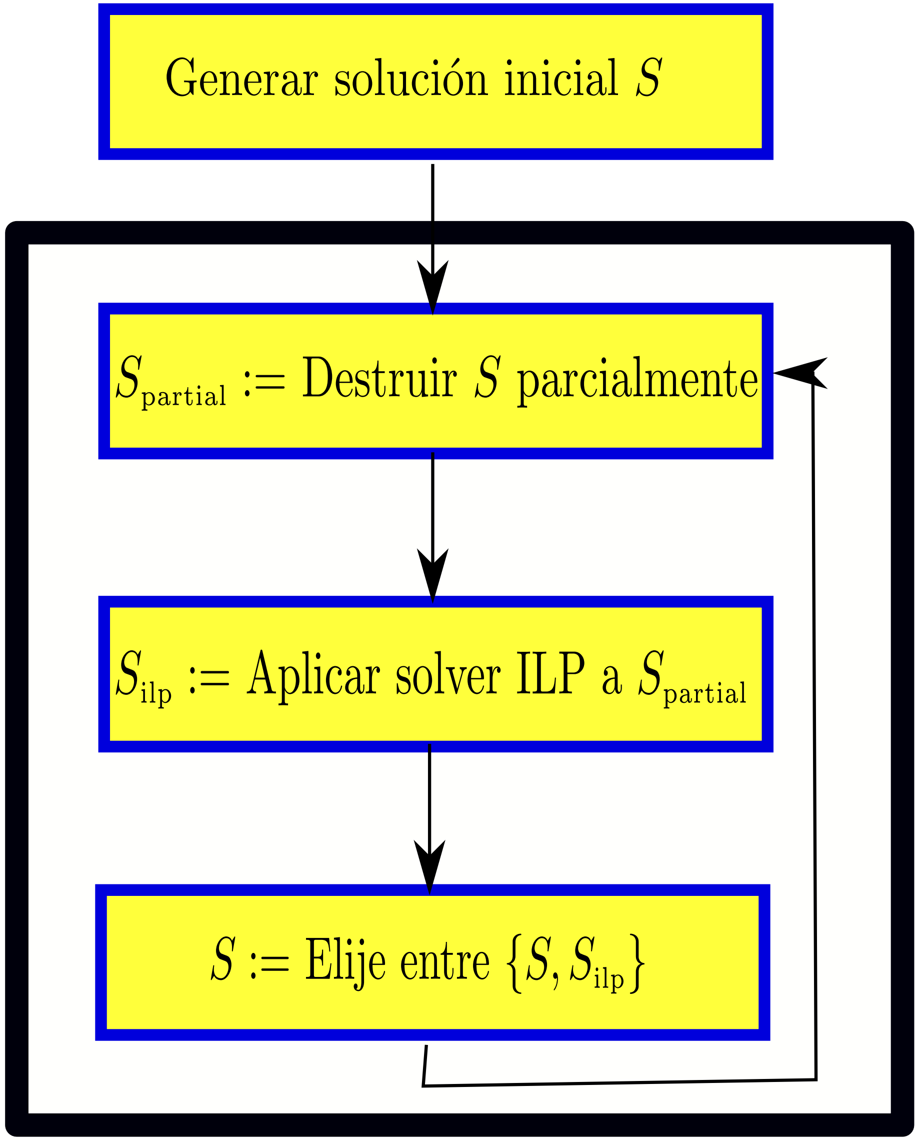
Compensación en la Búsqueda Local

- Vecindarios pequeños:
  1. **Ventaja:** son rápidos de explorar
  2. **Desventaja:** Calidad de los mínimos locales no es muy alta
- Vecindarios grandes:
  1. **Ventaja:** Calidad de los mínimos locales es más alta
  2. **Desventaja:** Encontrar el mejor vecino es, posiblemente, un problema NP-duro en sí mismo

Diferentes maneras de explorar un vecindario grande:

- De forma heurística
- Uso de métodos completos: por ejemplo, solvers ILP

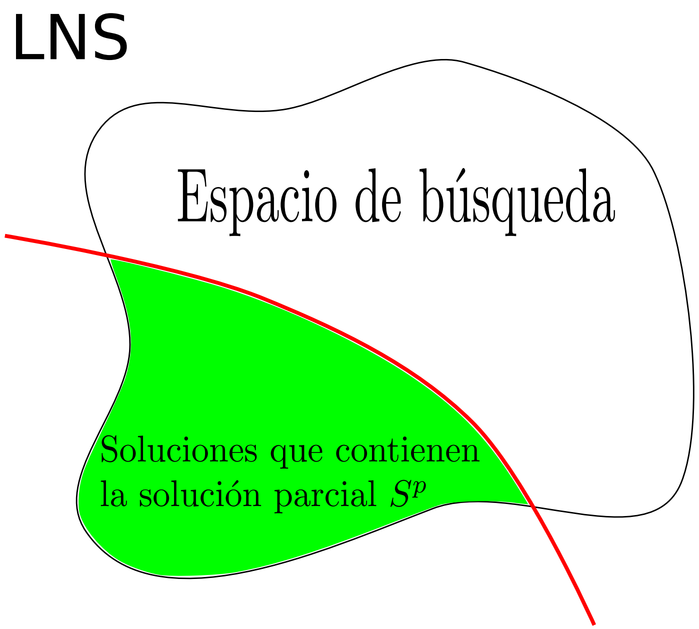
Large neighborhood search basado en solvers ILP: ILP-LNS



Aspecto crucial de ILP-LNS

- Importante: Aplicar un solver ILP para encontrar la mejor solución que contiene una solución parcial S_{parcial} significa aplicar el solver ILP a un espacio de búsqueda reducido.

- Consecuencia: ILP-LNS se puede aplicar a instancias más grandes que el solver ILP



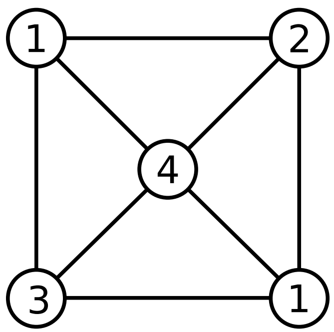
Ejemplo de aplicación: Minimum weight dominating set

- **Dado:** grafo $G = (V, E)$ no dirigido; cada $v_i \in V$ tiene un peso $w(v_i) \geq 0$
- **Soluciones válidas:** Cada $S \subseteq V$ es una solución válida, ssi

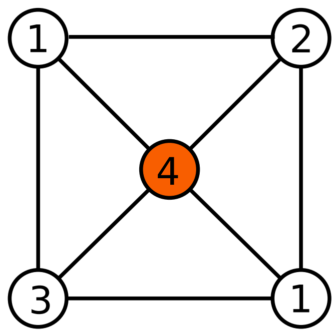
$$\forall\ v_i \in V\colon\ N[v_i] \cap S \neq \emptyset$$

- **Objetivo de la optimización:** Encontrar una solución S^* que minimice

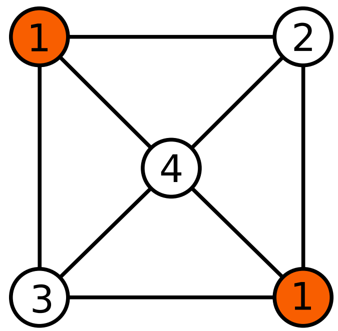
$$f(S^*) := \sum_{v_i \in S^*} w(v_i)$$



Grafo G



Una solución válida



La solución óptima

Modelo ILP

$$\mathbf{min} \sum_{v_i \in V} w(v_i) \cdot x_i \tag{1}$$

sujeto a:

$$\sum_{v_j \in N[v_i]} x_j \geq 1 \quad \text{para } v_i \in V \tag{2}$$

$$x_i \in \{0, 1\} \quad \text{para } v_i \in V$$

Toma nota:

- En este ILP: número de variables y restricciones es lineal
- Cómo encontrar la mejor solución que contiene S_{parcial}:

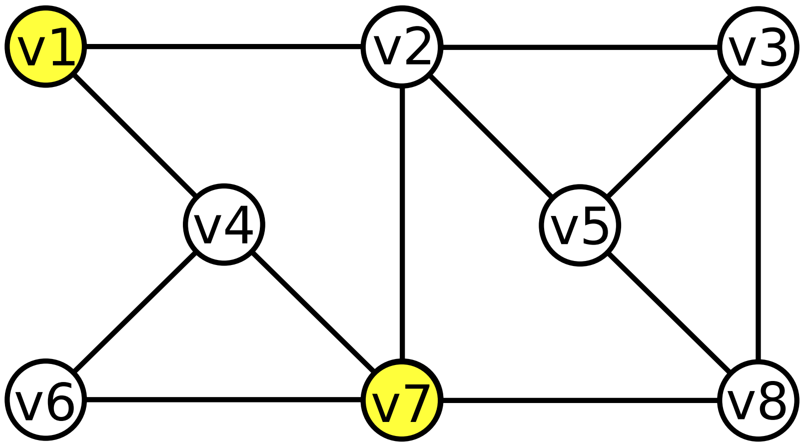
Añadir estas restricciones al ILP: $x_i = 1 \qquad \text{para } v_i \in S_{\text{parcial}}$

Generar la solución inicial: GREEDY (1)

Definiciones:

- V_{cov}: conjunto de nodos cubiertos (c.r.a. una solución parcial))
- $d(v|V_{\text{cov}})$: grado actual del nodo v sin considerar nodos cubiertos

Ejemplo:



$S = \{v_1, v_7\}, \quad V_{\text{cov}} = \{v_1, v_2, v_4, v_6, v_7, v_8\}, \quad d(v_3|V_{\text{cov}}) = 1$

Generar la solución inicial: GREEDY (2)

Pseudo-código de GREEDY:

1: **entrada:** un grafo $G = (V, E)$ con pesos en los nodos
2: $S := \emptyset$
3: $V_{\text{cov}} := \emptyset$
4: **while** $V_{\text{cov}} \neq V$ **do**
5: $\quad v^* := \text{argmax}_{v \in V \setminus V_{\text{cov}}} \left\{ \frac{d(v|V_{\text{cov}})}{w(v)} \right\}$
6: $\quad S := S \cup \{v^*\}$
7: $\quad V_{\text{cov}} := V_{\text{cov}} \cup N[v^*]$
8: **end while**
9: **salida:** S

Destrucción parcial de una solución

Método estándar: quitar un porcentaje $perc_{\text{dest}}$ de los nodos en S_{cur}

Cómo seleccionar estos nodos:

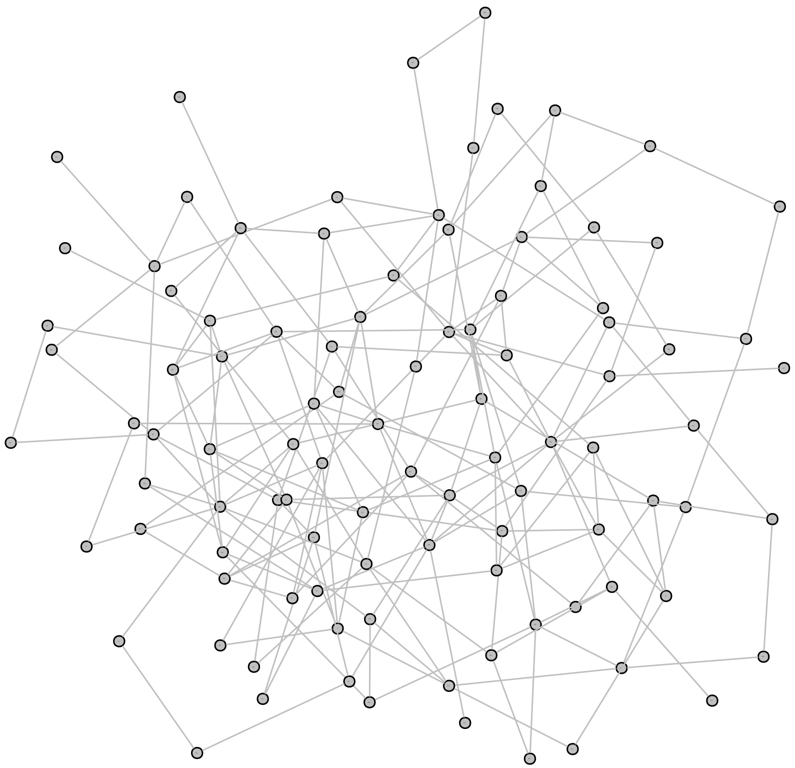
- Destrucción tipo $type_{\text{dest}}$ = aleatorio: nodos se seleccionan aleatoriamente
- Destrucción tipo $type_{\text{dest}}$ = sesgado: selección de nodos sesgada por función voraz

Elegir un valor para $perc_{\text{dest}}$:

- Elección dinámica de un valor de $[perc^{l}_{\text{dest}}, perc^{u}_{\text{dest}}]$
- Inicialmente $perc_{\text{dest}} := perc^{l}_{\text{dest}}$
- Cuando no se encuentra una solució mejor: $perc_{\text{dest}} := perc_{\text{dest}} + 5$
- Cuando se encuentra una solución mejor o se llega al límite superior: $perc_{\text{dest}} := perc^{l}_{\text{dest}}$

Instancias

- Grafos aleatorios con $|V| = \{100, 1000, 5000, 10000\}$ nodos
- Diferentes probabilidades para las aristas e_p (bajo, medio, alto):
  - ⋆ Para $|V| = 100$: $e_p \in \{0.03, 0.04, 0.05\}$
  - ⋆ Para $|V| > 100$: $e_p \in \{0.01, 0.03, 0.05\}$



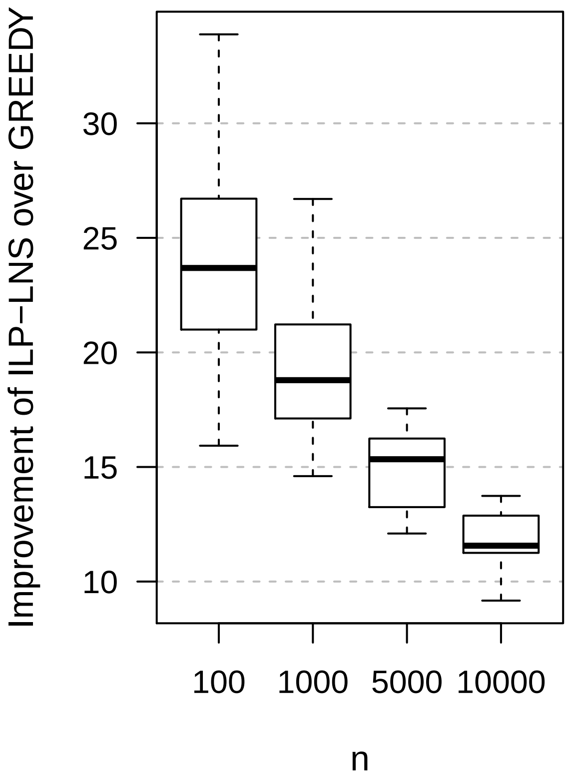
Tuneo de ILP-LNS

- $type_{\text{dest}}$ puede ser aleatorio o sesgado
- Los límites $(perc^{l}_{\text{dest}}, perc^{u}_{\text{dest}})$ para el porcentaje de destrucción:
  1. (X, X) donde $X \in \{10, 20, 30, 40, 50, 60, 70, 80, 90\}$
  2. $(X, Y) \in \{(10, 30), (10, 50), (30, 50), (30, 70)\}$
- $t_{\max}$: el tiempo máximo dado el solver ILP

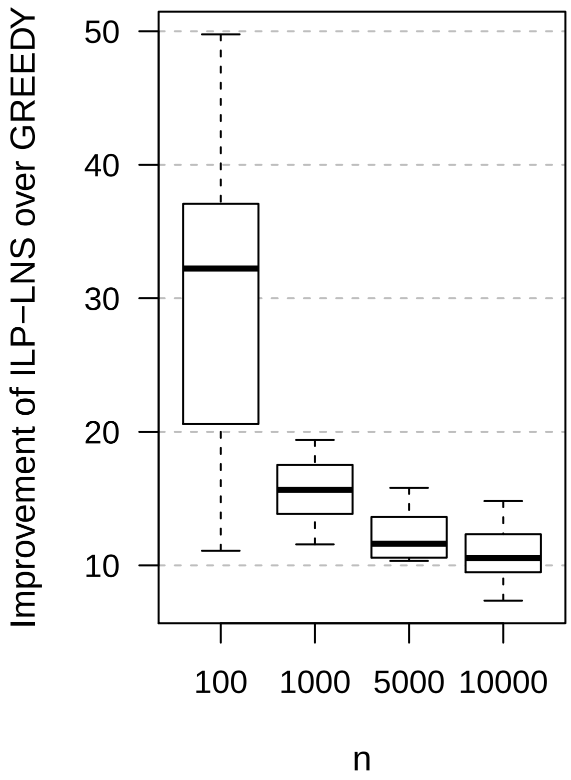
Valores seleccionados después de tunear con irace:

$\|V\|$	$type_{\text{dest}}$	$(perc^{l}_{\text{dest}}, perc^{u}_{\text{dest}})$	$t_{\max}$
100	1	$(60, 60)$	2.0
1000	0	$(90, 90)$	10.0
5000	1	$(50, 50)$	5.0
10000	1	$(40, 40)$	10.0

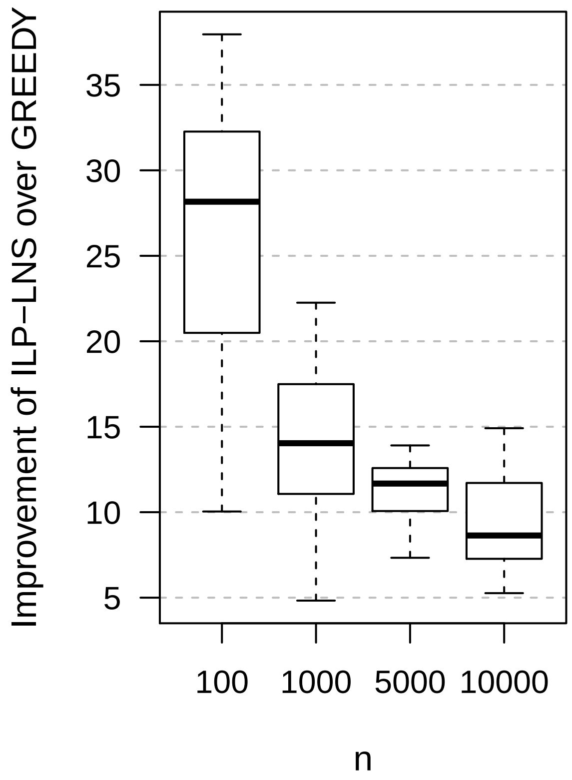
Resultados para el MWDS: mejora sobre GREEDY (en %)



Densidad baja

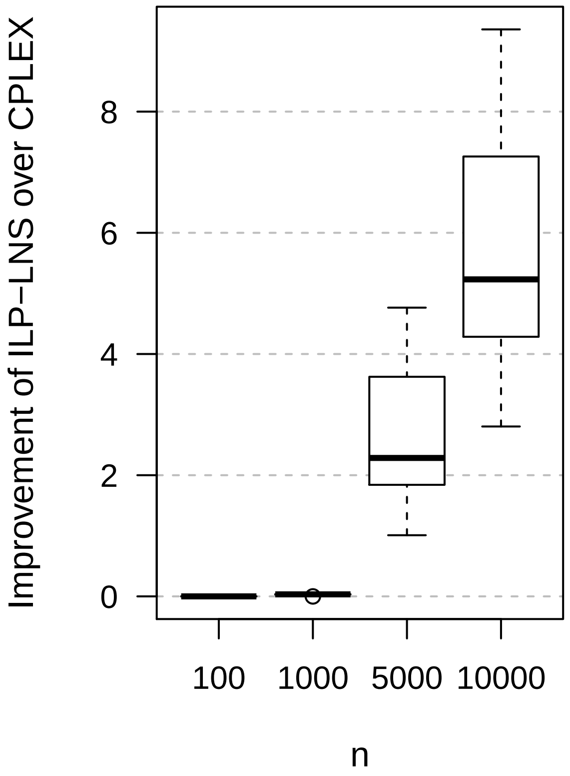


Densidad media

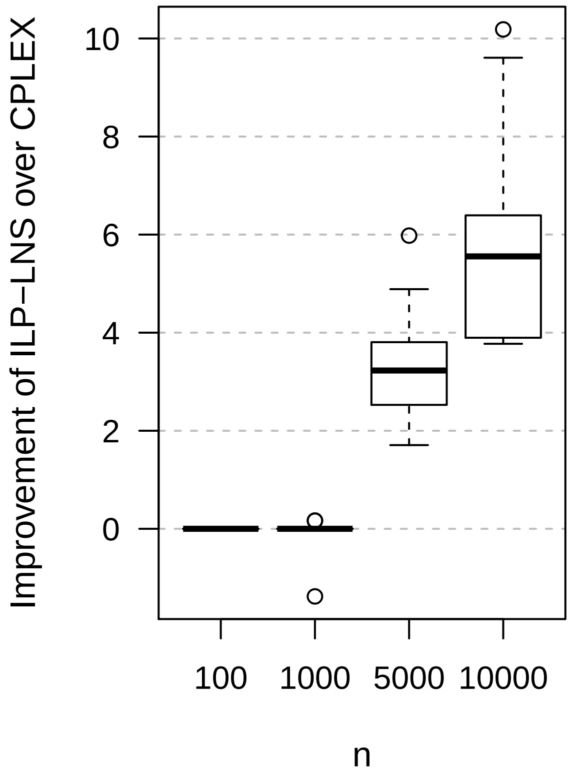


Densidad alta

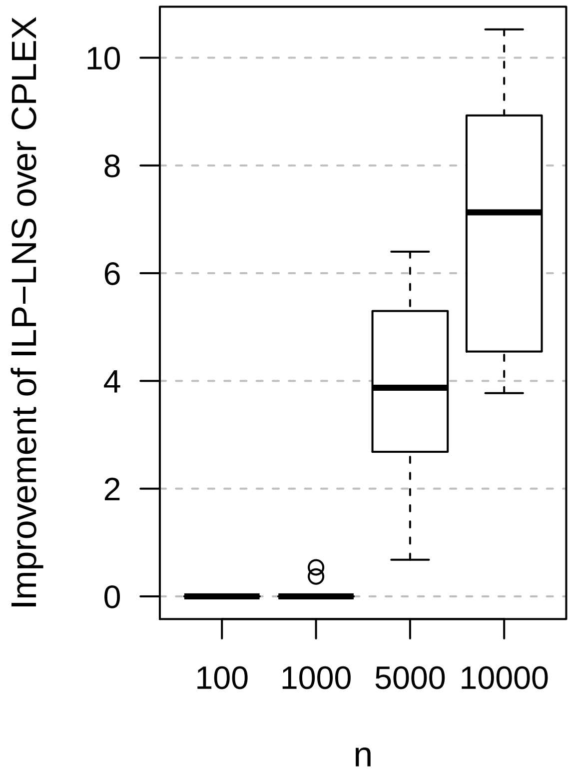
Resultados para el MWDS: mejora sobre CPLEX (en %)



Densidad baja



Densidad media



Densidad alta

Hipótesis y Pregunta Resultante

Hipótesis: basada en experiencia y un estudio de la literatura

LNS funciona bien siempre y cuando el número de componentes de solución (variables) sea lineal respecto a los parámetros del problema

Pregunta:

¿Qué hacer en caso que el ILP del problema considerado contenga un número grande de componentes de solución???

Construct, Merge, Solve & Adapt

Idea principal: La solución exacta de sub-instancias obtenidas tras juntar soluciones

Christian Blum, Borja Calvo. **A matheuristic for the minimum weight rooted arborescence problem**. *Journal of Heuristics*, 21(4): 479-499 (2015)

Idea Principal: Más en Detalle

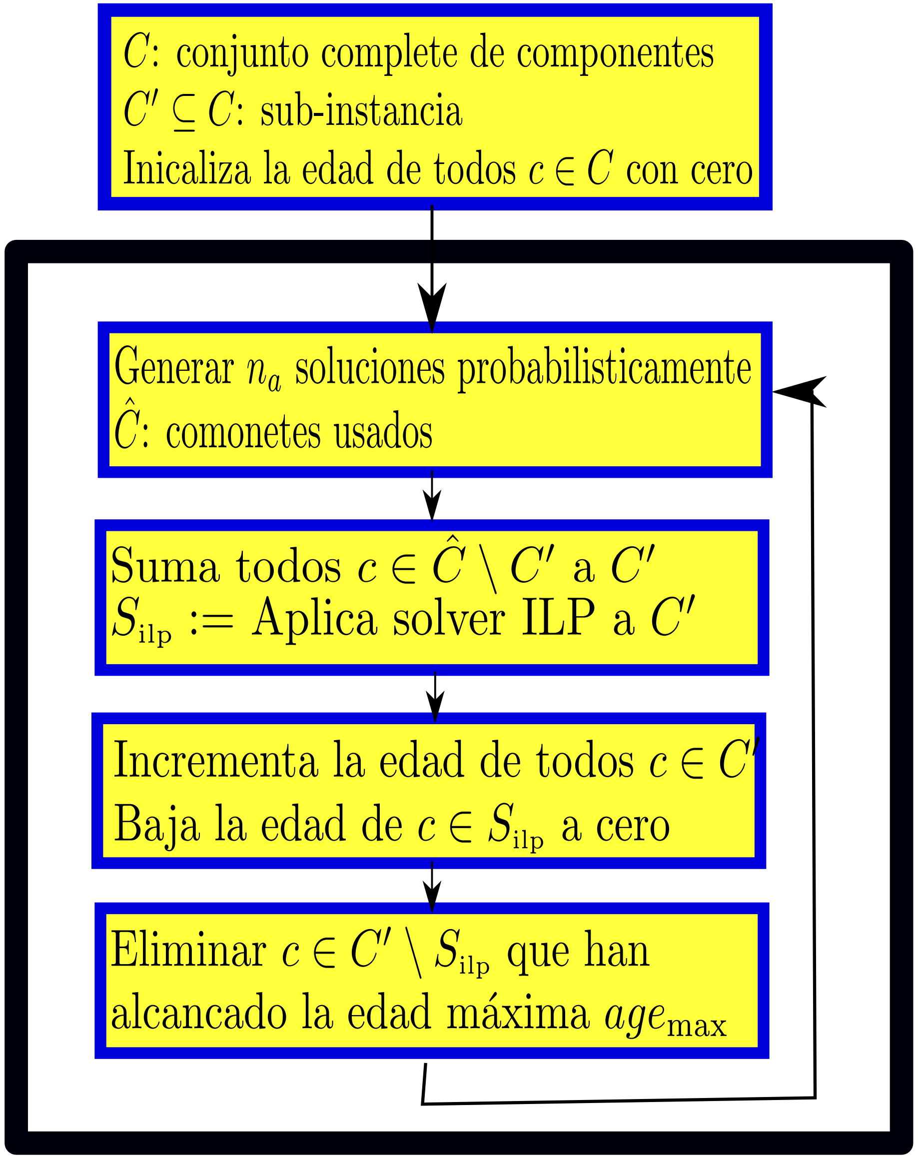
Observación: Cuando tenemos muchas componentes de solución, muchas de ellas solo aparecen en soluciones de baja calidad

Idea: Eliminar estos presumiblemente malos componentes de solución del ILP

Pasos del algoritmo propuesto:

- Generar presumiblemente buenas soluciones de manera probabilista
- Crear una sub-instancia juntando las componentes de las soluciones generadas
- Buscar el óptimo de la sub-instancia mediante un solver ILP
- Eliminar componentes presumiblemente inútiles de la sub-instancia

Construct, Merge, Solve & Adapt (Cmsa)



Aplicación: Minimum Common String Partition (1)

Entrada:

1. Dos secuencias relacionadas de longitud n sobre Σ
2. **Observa:** Secuencias s_1 y s_2 se llaman relacionadas si, y solo si, cada letra aparece las mismas veces en cada secuencia.

Soluciones válidas:

- Generar una partición P_1 de sub-secuencias de s_1 sin superposición
- Generar una partición P_2 de sub-secuencias de s_2 sin superposición
- La solución $S = (P_1, P_2)$ es válida si, y solo si, $P_1 = P_2$
- Función objetivo: $f(S) := |P_1| = |P_2|$

Objetivo: Minimización

Minimum Common String Partition (2)

Ejemplo:

- $s_1 :=$ **AGACTG**, $s_2 :=$ **ACTAGG**
- Solución trivial:
  - $P_1 = P_2 = \{\mathbf{A}, \mathbf{A}, \mathbf{C}, \mathbf{T}, \mathbf{G}, \mathbf{G}\}$
  - **Valor de la función objetivo:** 6
- Solución óptima S^*:
  - $P_1 = P_2 = \{\mathbf{ACT}, \mathbf{AG}, \mathbf{G}\}$
  - **Valor de la función objetivo:** 3

Trabajos en la Literatura

Hechos básicos:

- Introducido en 2005 en el contexto de la reordenación de genomas
- Complejidad: NP-duro

Trabajos relacionados:

- 2005: Un algoritmo voraz
- 2007: Algoritmo de aproximación
- 2008: Estudio relacionado con *fixed-parameter tractability (FPT)*
- 2013: Metaheurística (optimizació con colonias de hormigas)

Preliminares

Definiciones Dado s_1 y s_2 ...

- Un bloque común b_i es un tripleto $(t_i, k1_i, k2_i)$ donde
  1. t_i es una secuencia que empieza en la posición $1 \leq k1_i \leq n$ en s_1
  2. t_i es una secuencia que empieza en la posición $1 \leq k2_i \leq n$ en s_2
- B es el conjunto de todos los bloques comunes de s_1 y s_2
- Cualquier solución parcial S es un sub-conjunto de B tal que
  1. $\sum_{b_i \in \mathcal{S}} |t_i| = n$ (en caso de una **solución completa**)
  2. $\sum_{b_i \in \mathcal{S}} |t_i| < n$ (en caso de una **solución parcial**)
  3. Para cada par $b_i, b_j \in S$: t_i y t_j no tienen superposición en s_1 ni en s_2

Ejemplo: Conjunto B de Bloques Comunes

Secuencias de entrada: $s_1 = \mathbf{AGACTG}$ and $s_2 = \mathbf{ACTAGG}$

Conjunto B de todos los bloques comunes:

$$\left\{ \begin{array}{ll} b_1 = (\mathbf{ACT}, 3, 1) & b_8 = (\mathbf{A}, 3, 4) \\ b_2 = (\mathbf{AG}, 1, 4) & b_9 = (\mathbf{C}, 4, 2) \\ b_3 = (\mathbf{AC}, 3, 1) & b_{10} = (\mathbf{T}, 5, 3) \\ b_4 = (\mathbf{CT}, 4, 2) & b_{11} = (\mathbf{G}, 2, 5) \\ b_5 = (\mathbf{A}, 1, 1) & b_{12} = (\mathbf{G}, 2, 6) \\ b_6 = (\mathbf{A}, 1, 4) & b_{13} = (\mathbf{G}, 6, 5) \\ b_7 = (\mathbf{A}, 3, 1) & b_{14} = (\mathbf{G}, 6, 6) \end{array} \right\}$$

Solución $\{\mathbf{ACT}, \mathbf{AG}, \mathbf{G}\}$: $\mathcal{S} = \{b_1, b_2, b_{14}\}$

Observa: El conjunto B representa el conjunto de componentes de solución

Modelo ILP (1)

Secuencias de entrada: $s_1 =$ **AGACTG** and $s_2 =$ **ACTAGG**

$$B = \left\{ \begin{array}{l} b_1 = (\mathbf{ACT}, 3, 1) \\ b_2 = (\mathbf{AG}, 1, 4) \\ b_3 = (\mathbf{AC}, 3, 1) \\ b_4 = (\mathbf{CT}, 4, 2) \\ b_5 = (\mathbf{A}, 1, 1) \\ b_6 = (\mathbf{A}, 1, 4) \\ b_7 = (\mathbf{A}, 3, 1) \\ b_8 = (\mathbf{A}, 3, 4) \\ b_9 = (\mathbf{C}, 4, 2) \\ b_{10} = (\mathbf{T}, 5, 3) \\ b_{11} = (\mathbf{G}, 2, 5) \\ b_{12} = (\mathbf{G}, 2, 6) \\ b_{13} = (\mathbf{G}, 6, 5) \\ b_{14} = (\mathbf{G}, 6, 6) \end{array} \right\} \qquad M1 = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad M2 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Modelo ILP (2)

$$\mathbf{min} \sum_{i=1}^{m} x_i \tag{3}$$

sujeto a:

$$\sum_{i=1}^{m} |t_i| \cdot x_i = n \tag{4}$$

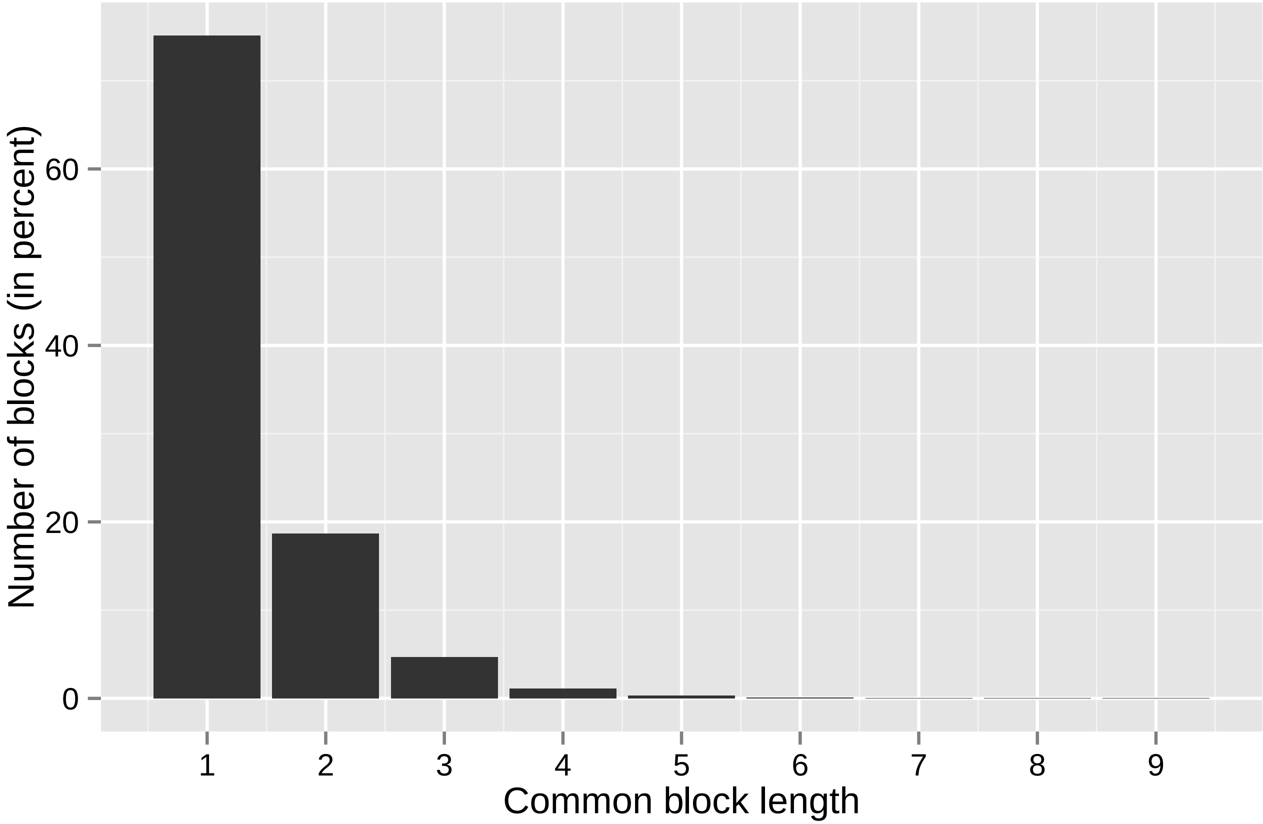
$$\sum_{i=1}^{m} M1_{i,j} \cdot x_i = 1 \quad \text{para } j = 1, \ldots, n \tag{5}$$

$$\sum_{i=1}^{m} M2_{i,j} \cdot x_i = 1 \quad \text{para } j = 1, \ldots, n \tag{6}$$

$$x_i \in \{0, 1\} \quad \text{para } i = 1, \ldots, m$$

Observa: Número muy grande de componentes de solución

Conjunto de componentes de solución: propiedades



$\boxed{\text{Observa:}}$ La inmensa mayoría de los bloques comunes de longitud 1 y 2 no formarán parte de buenas soluciones

Algoritmo Voraz Simple

Dada un solución parcial S_{parcial}: $B(S_{\text{parcial}}) \subset B$ son los bloques comunes para extender S_{parcial}

Pseudo-código:

1. $S_{\text{parcial}} := \emptyset$
2. **mientras** S_{parcial} no sea una solución completa
   - Selecciona el bloque común más largo b_i de $B(S_{\text{parcial}})\}$
   - $S_{\text{parcial}} := S_{\text{parcial}} \cup \{b_i\}$

Nota: Este método se usa dentro de CMSA de forma probabilista

Instancias y Tuneo

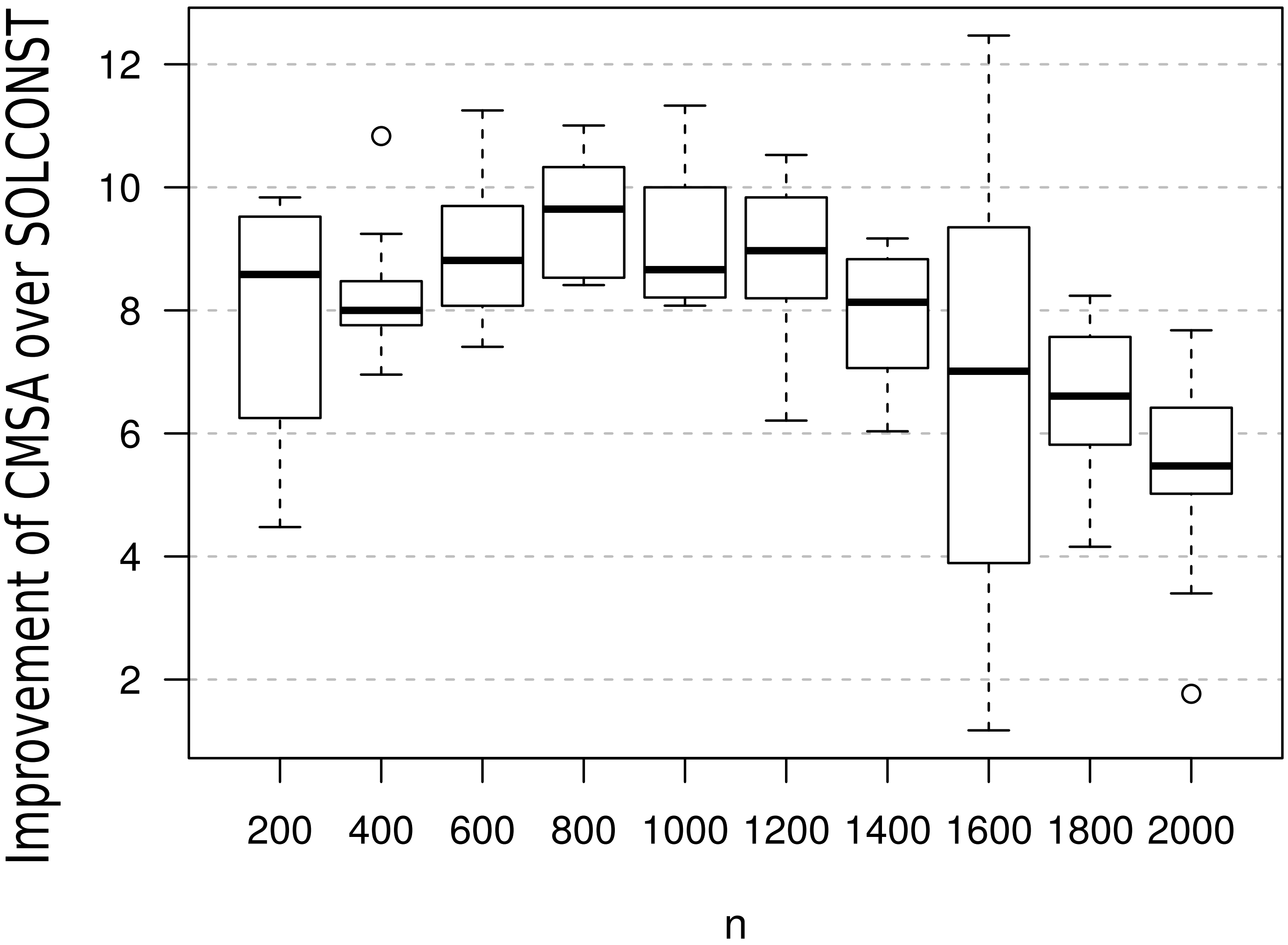
Instancias: 300

- Longitud de las secuencias de entrada: $n \in \{200, 400, \ldots, 1800, 2000\}$
- Tamaño del alfabeto: $|\Sigma| \in \{4, 12, 20\}$

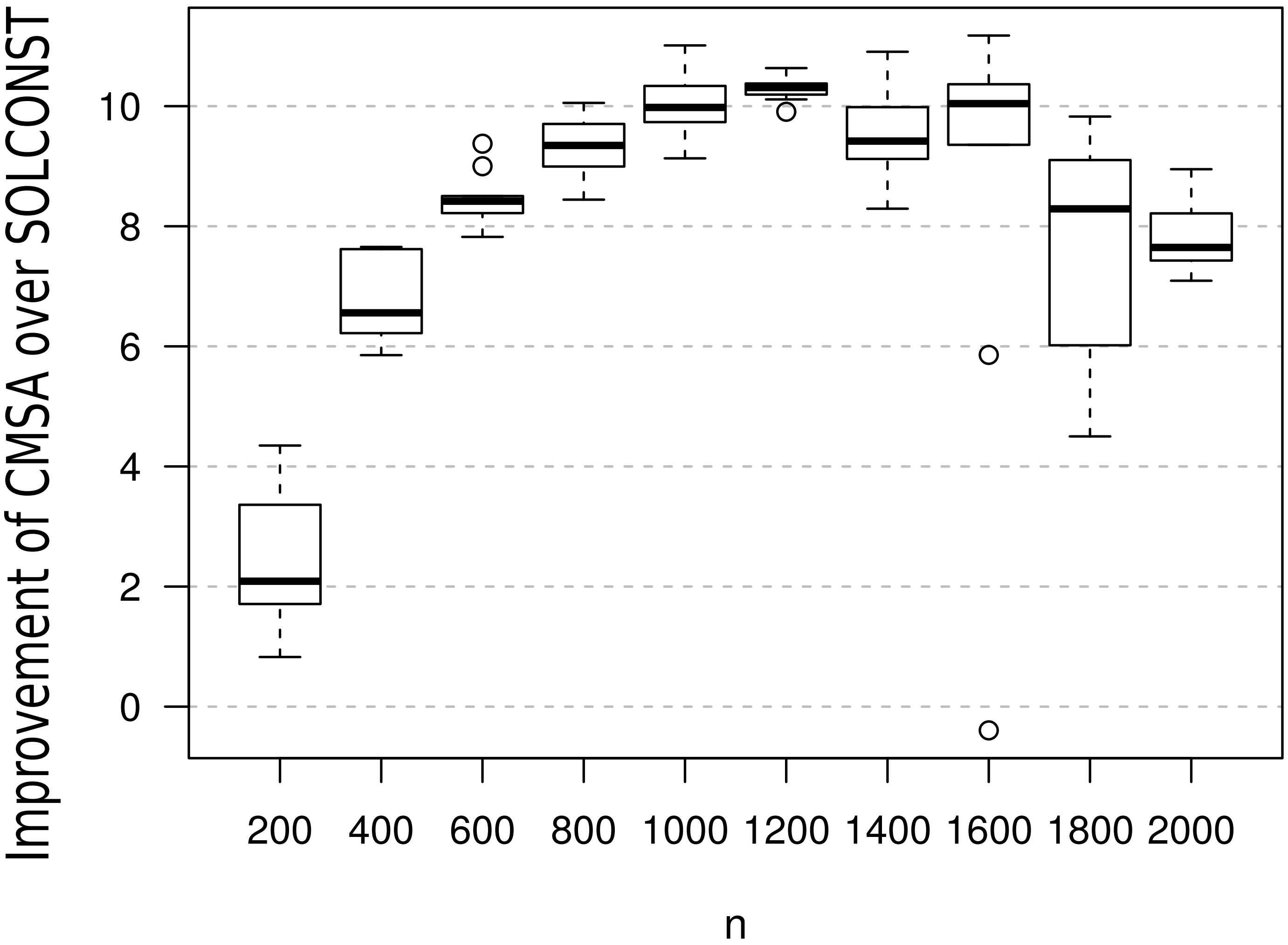
Tuneo con la herramienta irace:

n	n_a	age_{max}	d_{rate}	l_{size}	t_{max}
400	50	10	0.0	10	60
800	50	10	0.5	10	240
1200	50	10	0.9	10	480
1600	50	5	0.9	10	480
2000	50	10	0.9	10	480

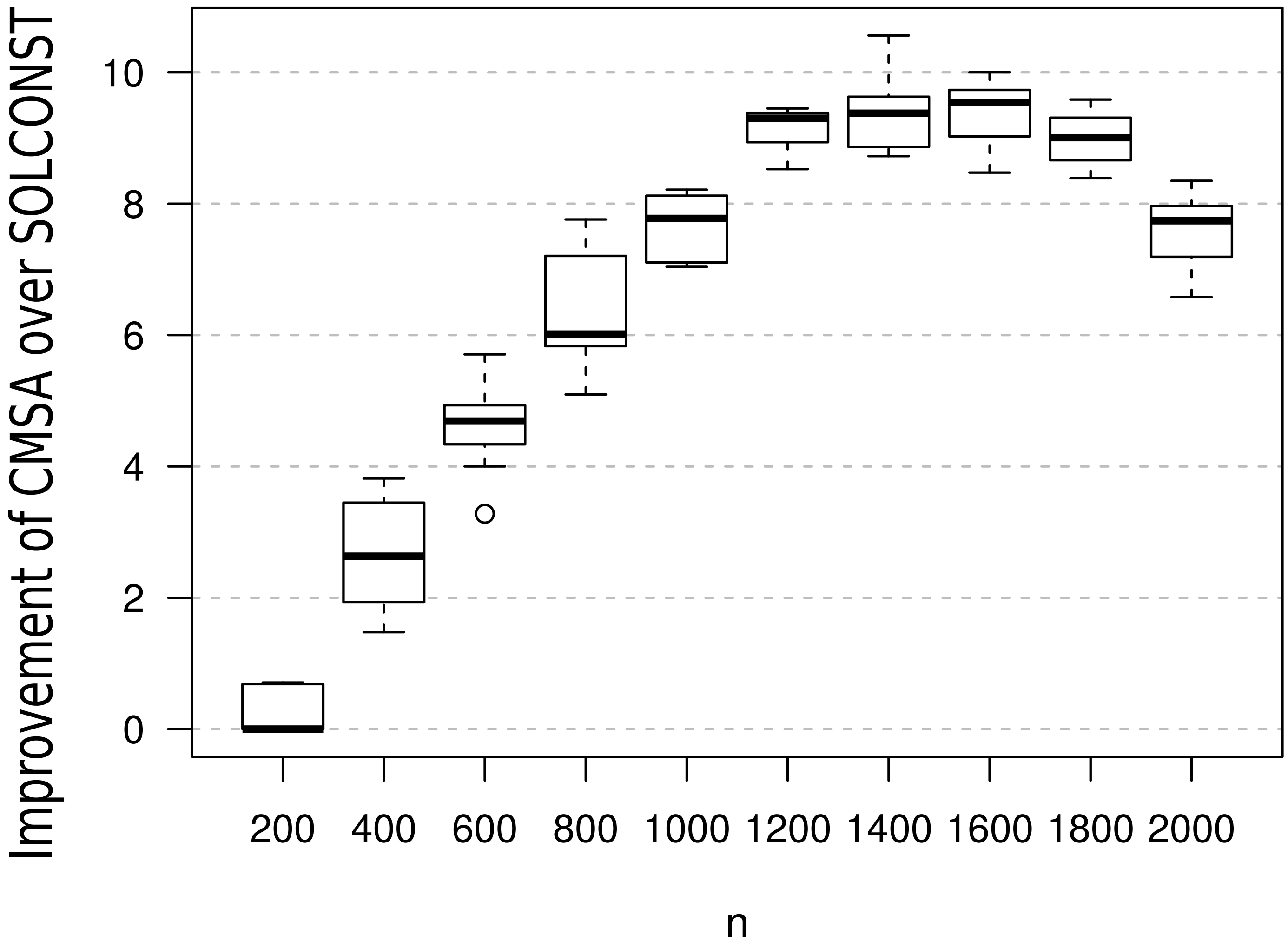
Resultados: mejora sobre GREEDY ($|\Sigma| = 4$)



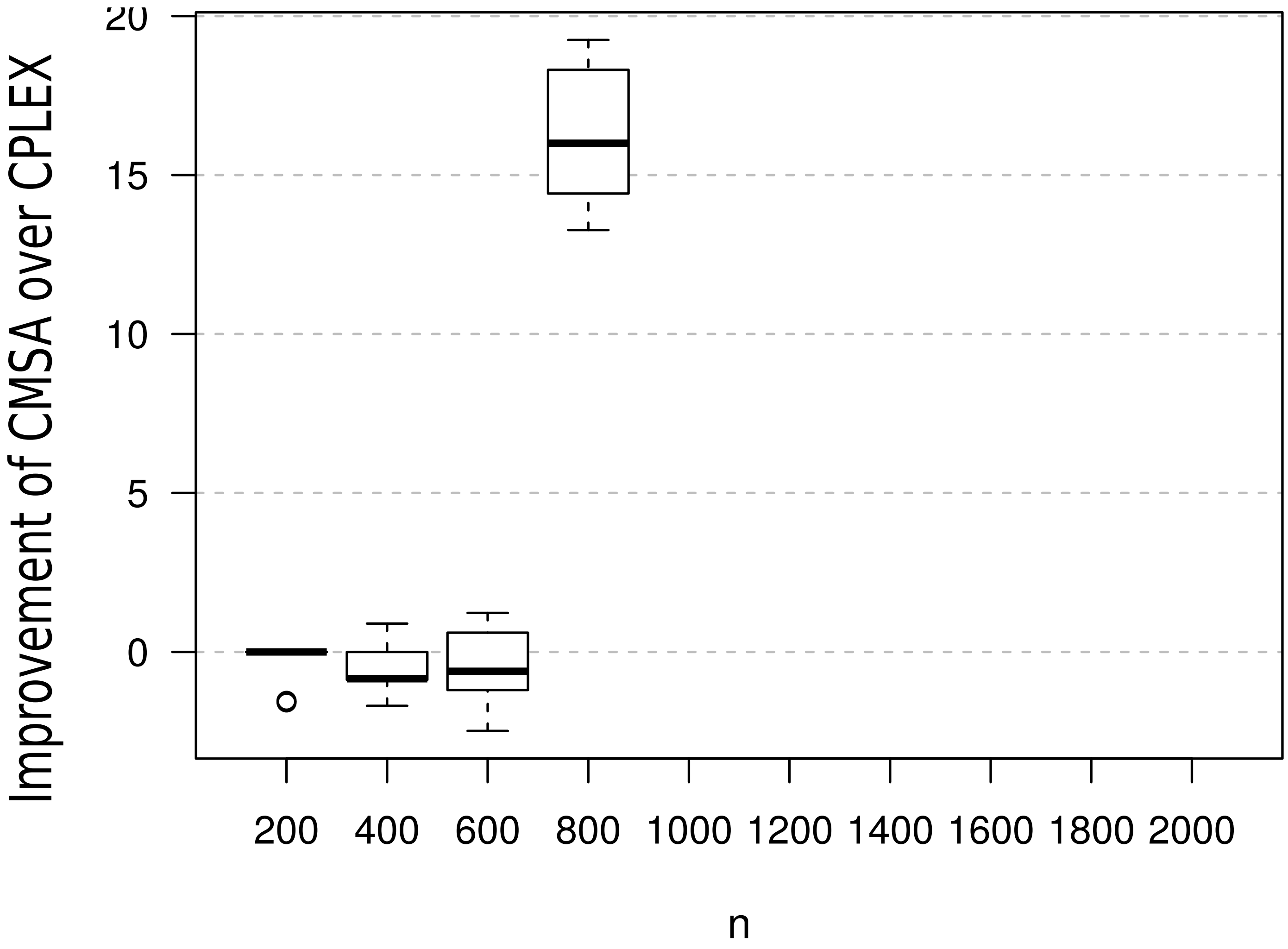
Resultados: mejora sobre GREEDY ($|\Sigma| = 12$)



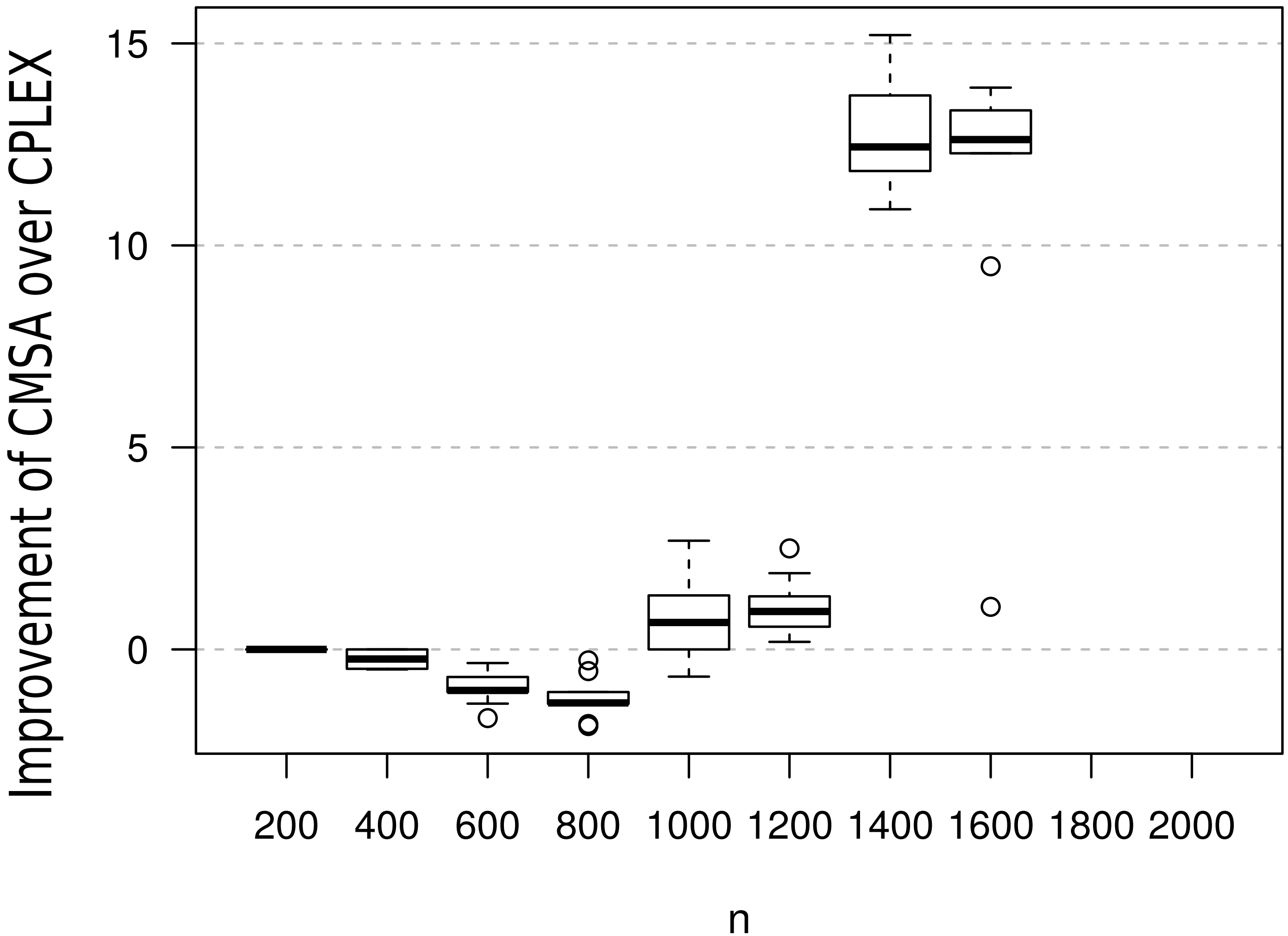
Resultados: mejora sobre Greedy ($|\Sigma| = 20$)



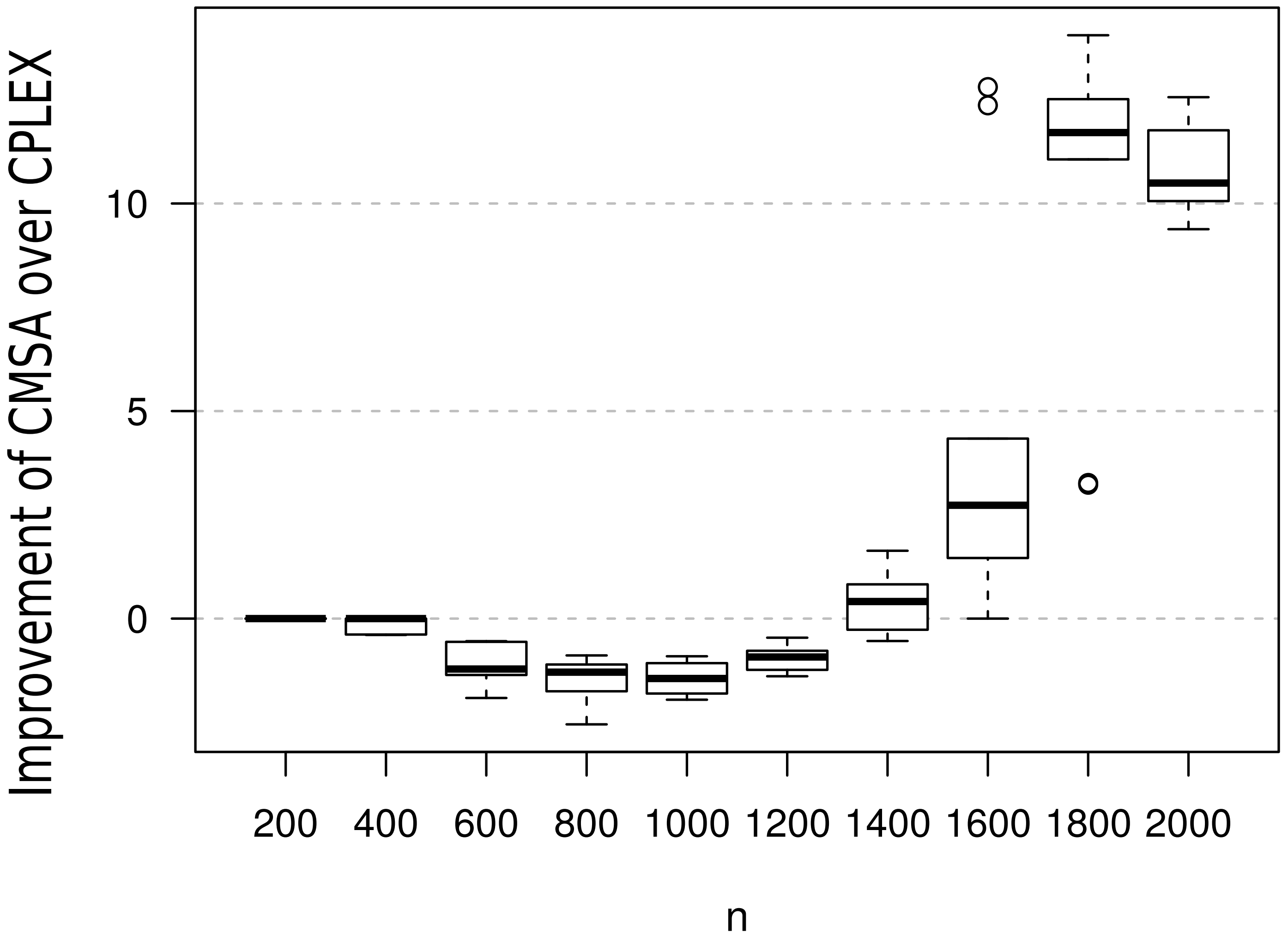
Resultados: mejora sobre CPLEX ($|\Sigma| = 4$)



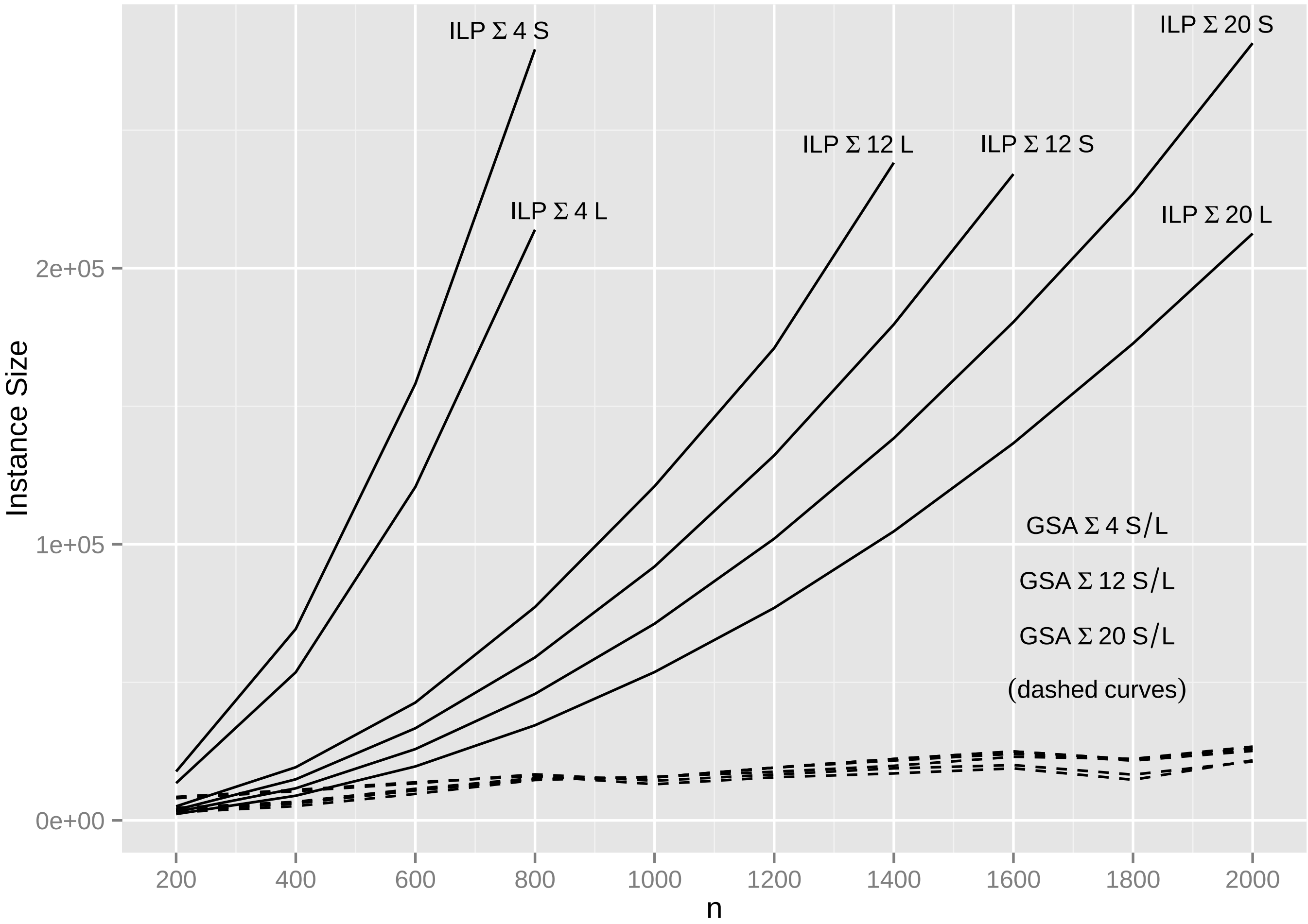
Resultados: mejora sobre CPLEX ($|\Sigma| = 12$)



Resultados: mejora sobre CPLEX ($|\Sigma| = 20$)

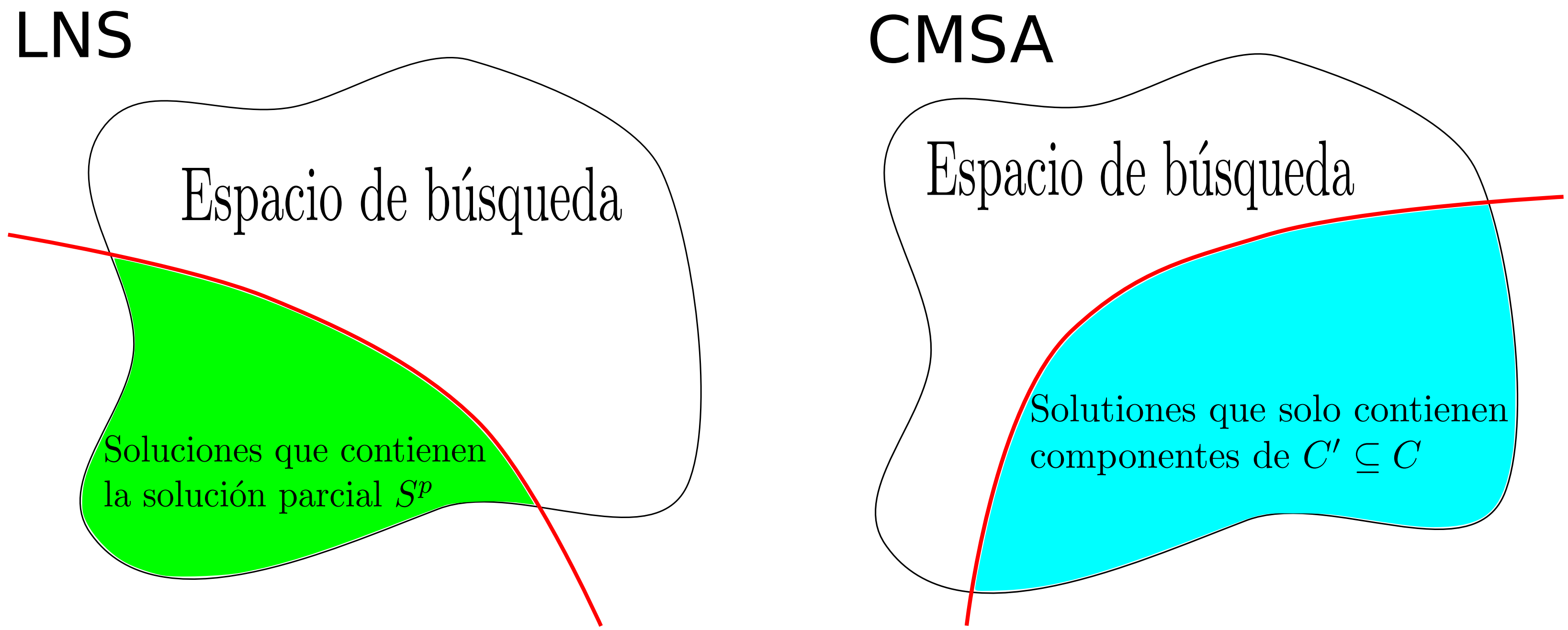


Evolución de los Tamaños de las Sub-Instancias



Diferencias entre LNS y CMSA: Resumen

Cómo se reduce el tamaño de la instancia original:



¿Cómo se genera la sub-instancia de la siguente generación?

- **LNS:** Destrucción parcial de la solución actual
- **CMSA:** Generar nuevos componentes de solución y quitar antiguas

Resumen y Posibles Líneas de Investigación

Resumen:

- **CMSA:** Un nuevo algoritmo híbrido para la optimizació combinatoria
- **Hipótesis:**
  - **LNS** funciona mejor para problemas con un número lineal de componentes de solución
  - **CMSA** funciona mejor para problemas con un número super-lineal de componentes de solución

Líneas de investigación:

- **Construcción de soluciones:** probabilidades no estáticas
- Una manera más inteligente para **eliminar componentes** de sub-instancias
- Estudios teóricos que confirman (o descartan) la hipótesis sobre LNS y CMSA

Investigadores Implicados

Christian Blum

Borja Calvo

Pedro Pinacho

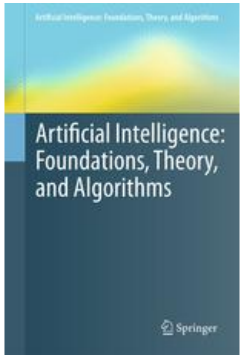
Jóse Antonio Lozano

Manuel López-Ibáñez

¿Preguntas?

Literatura:

- C. Blum, B. Calvo. **A matheuristic for the minimum weight rooted arborescence problem**. *Journal of Heuristics*, 21(4): 479-499 (2015)
- C. Blum, J. Puchinger, G. R. Raidl, A. Roli. **Hybrid metaheuristics in combinatorial optimization: A survey**. *Applied Soft Computing*, 11(6): 4135–4151 (2011)



Libro (pub. en preve): C. Blum, G. R. Raidl. Hybrid Metaheuristics – Powerful Tools for Optimization, Springer Series on Artificial Intelligence, 2015