Non-guillotine 2D bin packing problem

- Eduardo Pérez Mederos
- Miguel Monterrey Varela
- Jaime González Valdés
- Óscar Mateos López

Universidad de la Laguna
ETS Ingeniería Informática

Índice

- Introducción del problema.
- Heurísticas utilizadas.
- Presentación de resultados.

Introducción del problema

- **Non-guillotine two-dimensional bin packing problem.**

  Dado un conjunto de *n items bidimensionales* (rectángulos), {i1, i2, ..., in}, de dimensiones respectivas (*h*(*i*1);*w*(*i*1)); (h(i_1), $w(i_1)$), ($h(i_2)$, $w(i_2)$), ..., ($h(i_n)$, $w(i_n)$), determinar el menor número de *bins* (rectágulos) de dimensiones (*H;W*) necesarios para empaquetar sin solapamientos todos los items. Se supondrá que $h(i_j) \leq H$, $w(i_j) \leq W$, $\forall j = 1, ..., n$ y que los items no pueden rotarse. El empaquetado resultante no necesariamente debe poder obtenerse con cortes tipo guillotina.

Introducción del problema

- **Función Objetivo.**

  Para el cálculo de la función objetivo debemos conocer:

- Las dimensiones de la caja (todas tienen la misma medida).
- Las dimensiones de los distintos rectángulos.
- Coordenadas de los distintos rectángulos a lo largo y ancho de la caja cuando estemos pasando las diferentes heurísticas.
- Espacio en una caja no vacía.

- ✓ Con estos datos tenemos suficiente información para minimizar el numero de cajas frente a un número de rectángulos.
- ✓ Usando el algoritmo **Finite First Fit**, colocamos cada rectángulo en las posiciones disponibles, según el órden establecido por la permutación, intentando minimizar el espacio desperdiciado.

Introducción del problema

- **Solución.**

  Representamos la solución de este problema mediante una permutación que indica el orden en que los ítems son considerados para su inclusión en el objeto, junto a un procedimiento que indica la posición que ocupan los ítems en el objeto.

Caja: 0 *Rectangulos:* *A -> [7, 4], Area = 28 en el punto -> (0,0)* *B -> [4, 3], Area = 12 en el punto -> (0,4)* *C -> [4, 2], Area = 8 en el punto -> (4,4)* *D -> [3, 3], Area = 9 en el punto -> (7,0)* *E -> [2, 2], Area = 4 en el punto -> (8,4)* *F -> [3, 1], Area = 3 en el punto -> (7,3)* *G -> [2, 1], Area = 2 en el punto -> (4,6)* *[...]*	*################* *#JJJJKKKK@@#* *#BBBBGGHHI@#* *#BBBBCCCCEE#* *#BBBBCCCCEE#* *#AAAAAAAFFF #* *#AAAAAAADDD#* *#AAAAAAADDD#* *#AAAAAAADDD#* *################* *La caja tiene ocupado un 96% de su area total.*

Heurísticas utilizadas

- **Métodos constructivos**
  - G.R.A.S.P.

Heurísticas utilizadas

- **G.R.A.S.P:**
  - GRASP (Greedy Randomize Adaptive Search Procedures - Procedimientos de Búsqueda basados en funciones Ávidas, Aleatorias y Adaptativas).
  - Es una técnica simple aleatoria e iterativa, en la que cada iteración provee una solución al problema y se guarda la mejor solución como resultado final. Hay dos fases en cada iteración GRASP: la primera construye una solución inicial por medio de una función ávida, en la segunda se fase aplica un procedimiento de búsqueda local a la solución construida, con la esperanza de encontrar una mejora.

Heurísticas utilizadas

- **Métodos de búsqueda**
  - Búsquedas Locales,
  - Búsquedas aleatorias puras,
  - Búsquedas por recorrido al azar,
  - Búsquedas Multiarranque,
  - Recocido Simulado,
  - Búsqueda Tabú,
  - VND,
  - BVNS,
  - Búsqueda Dispersa.

Heurísticas utilizadas

- **Búsquedas Locales:**
  - Dada una solución se obtienen sus vecinos por medio de una regla.
  - Se sustituye la solución por el mejor vecino si cumple el criterio de aceptación.
  - Se repiten los pasos anteriores mientras se satisfaga el criterio de continuación.

Heurísticas utilizadas

- **Búsquedas aleatorias puras:**
  - Para la búsqueda aleatoria pura, elegiremos una muestra de soluciones al azar y devolveremos la mejor.
  - El entorno de la solución es todo el espacio de búsqueda.
  - Utilizaremos dos criterios de parada:
    - No mejora la solución optima en n-veces.
    - Se ejecutara la función n-veces.

Heurísticas utilizadas

- **Búsquedas por recorrido al azar:**
  - Persigue una mayor capacidad de exploración o diversificación de la búsqueda.
  - El procedimiento consiste en realizar transformaciones de la solución actual hasta encontrar una mejora; cuando esto ocurre, se toma la nueva solución como solución actual y se continúa la búsqueda desde ella.
  - En la elección de estas transformaciones o movimientos es conveniente tener en cuenta también una serie de aspectos: en qué dirección realizar los movimientos, qué lejanía abarcar en las transformaciones. Si se repiten los fracasos en la búsqueda de una solución mejor, se puede probar otra forma distinta de realizar las transformaciones.

Heurísticas utilizadas

- **Búsquedas Multiarranque:**
  - Los procedimientos de búsqueda con Arranque Múltiple (Multi-Start) realizan varias búsquedas monótonas partiendo de diferentes soluciones iniciales.
  - La búsqueda monótona implicada puede ser cualquiera de las anteriormente descritas. Una de las formas más simples de llevar esto a cabo consiste en generar una muestra de soluciones iniciales o de arranque.
  - Esto es equivalente a generar al azar una nueva solución de partida cada vez que la búsqueda quede estancada en el entorno de una solución óptima local.

Heurísticas utilizadas

- **Recocido Simulado:**
  - Es un meta-algoritmo genérico para la optimización global del problema, es decir, encontrar una buena aproximación al óptimo global de una función en un espacio de búsqueda grande.
  - Para ciertos problemas, el recocido simulado puede ser muy eficaz a condición de que la meta sea simplemente encontrar una solución aceptablemente buena en una cantidad de tiempo fija, antes que la solución mejor.

Heurísticas utilizadas

- **Búsqueda Tabú:**
  - La búsqueda tabú es un algoritmo metaheurístico que puede utilizarse para resolver problemas de optimización combinatoria.
  - Utiliza un procedimiento de búsqueda local para moverse iterativamente desde una solución x hacia una otra x' en la vecindad de x, hasta satisfacer algún criterio de parada. La búsqueda entonces progresa moviéndose iterativamente de una solución x hacia una solución x'en N * (x).

Heurísticas utilizadas

- **VND (Variable neighbourhood descent):**
  - La Búsqueda de Entorno Variable es una metaheurística propuesta recientemente (1995) por Pierre Hansen y Nenad Mladenovi.
  - Está basada en el principio de cambiar sistemáticamente de estructura de entornos dentro de la búsqueda. Cuando la búsqueda queda atrapada, se cambia a un entorno más amplio.
  - Características:
    - Sencillez
    - Permite extensiones del esquema básico.

Heurísticas utilizadas

- **VND (Variable neighbourhood descent):**
- La VND está basada en tres hechos simples:
  - Un mínimo local con una estructura de entornos no lo es necesariamente con otra.
  - Un mínimo global es mínimo local con todas las posibles estructuras de entornos.
  - Para muchos problemas, los mínimos locales con la misma o distinta estructura de entorno están relativamente cerca.

Heurísticas utilizadas

- **BVNS:**
  - La búsqueda de entorno variable básica (Basic Variable Neighbourhood Search, BVNS) combina cambios determinísticos y aleatorios de estructura de entornos. No cambia en demasía con la heurística anterior.

Heurísticas utilizadas

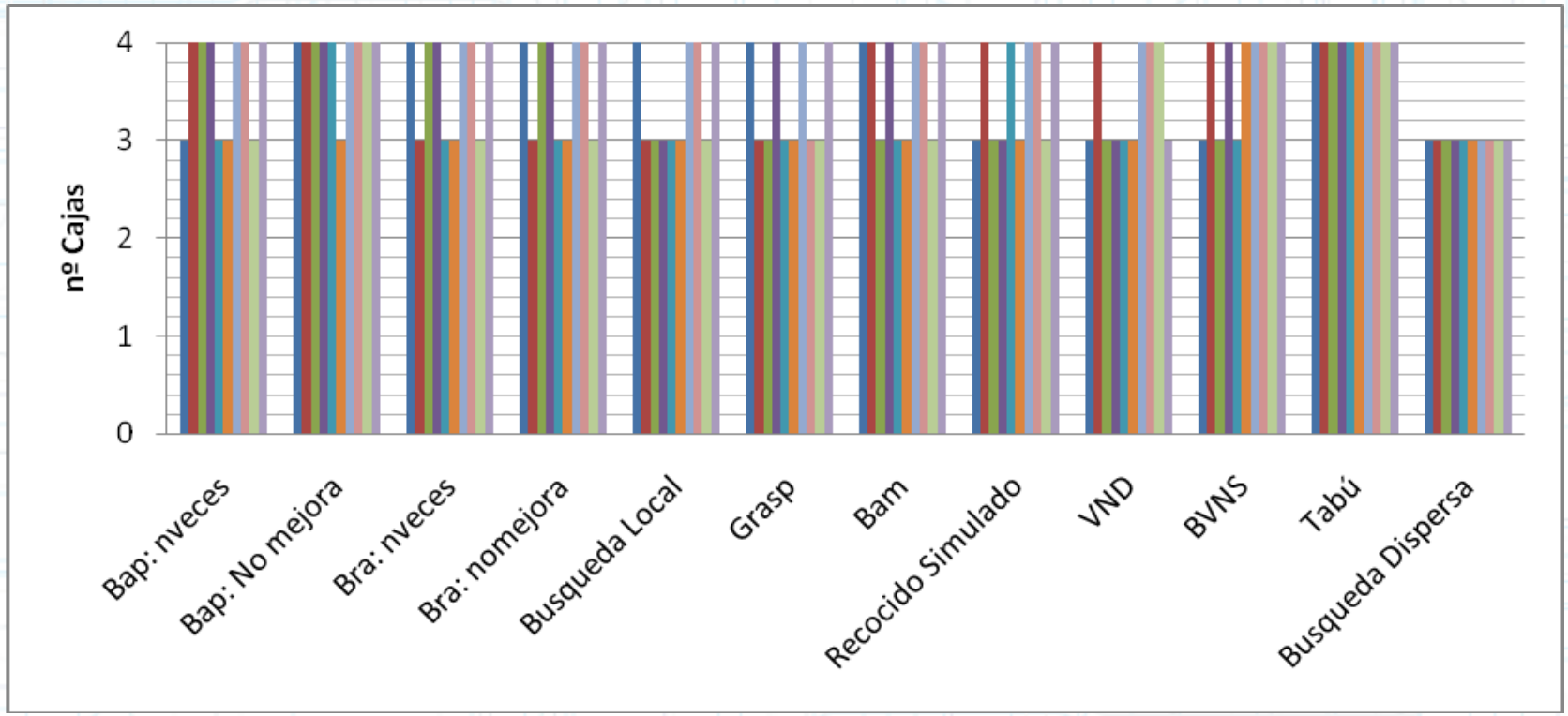
- **Búsqueda Dispersa:**
  - La Búsqueda Dispersa es un método evolutivo capaz de obtener soluciones de calidad a problemas difíciles. Algunos aspectos están claramente establecidos y otros requieren de más estudio. Su arquitectura permite el diseño de sistemas independientes del contexto. Actualmente está en plena expansión.
  - Esta heurística utiliza tres funciones auxiliares, CrearPoblacionInicial, Combinarsoluciones y SeleccionarSubconjunto.

Heurísticas utilizadas

- **Búsqueda Dispersa:**
  - Crear Poblacion Inicial: Conjunto de soluciones de calidad y diversad.
  - Combinar soluciones: Método que combina los subconjuntos de soluciones seleccionadas del conjunto de referencia para obtener una nueva solucion.
  - SeleccionarSubconjunto: Selecciona todos los subcojuntos de soluciones del conjunto de referencia para realizar buenas combinaciones.

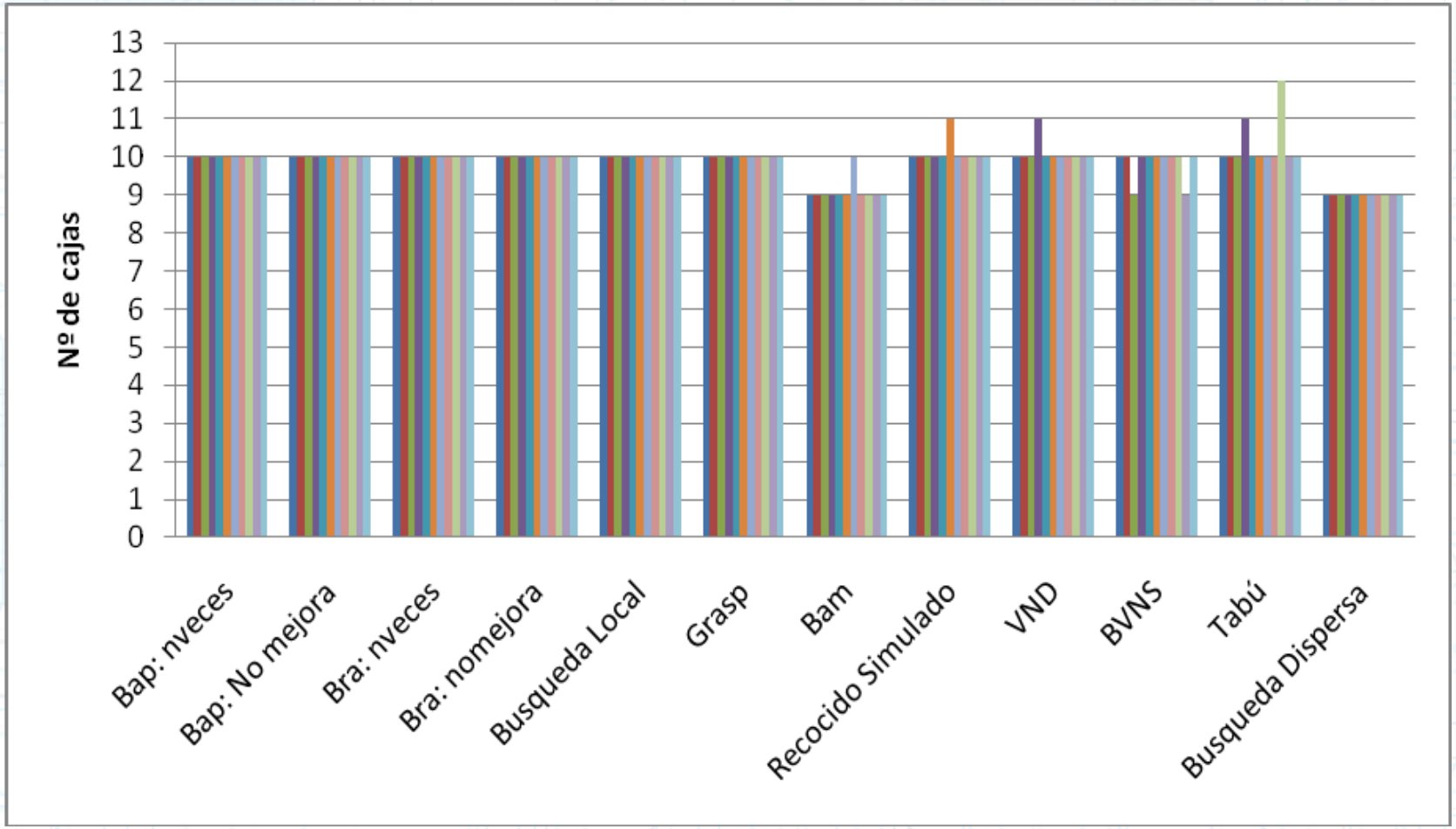
Presentación de resultados

- A continuación veremos resultados para las distintas heurísticas durante 10 ejecuciones distintas.
- Las heurísticas que además requieran iteraciones quedarán fijadas en 7.
  - Primer módulo de valores (caja 12x12 y 31 elementos):



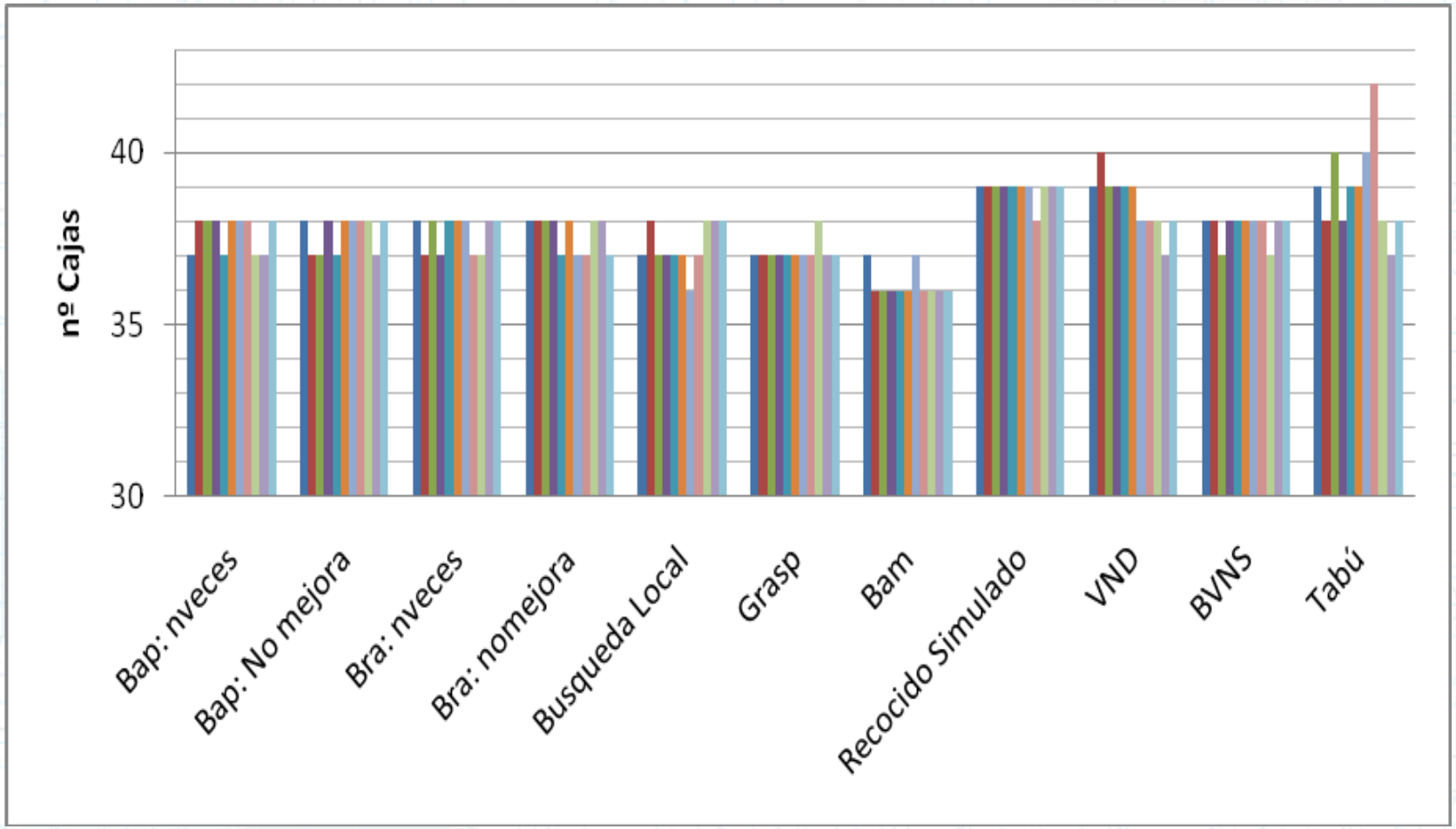
Presentación de resultados

- Segundo módulo de valores (caja 10x8 y 48 elementos):



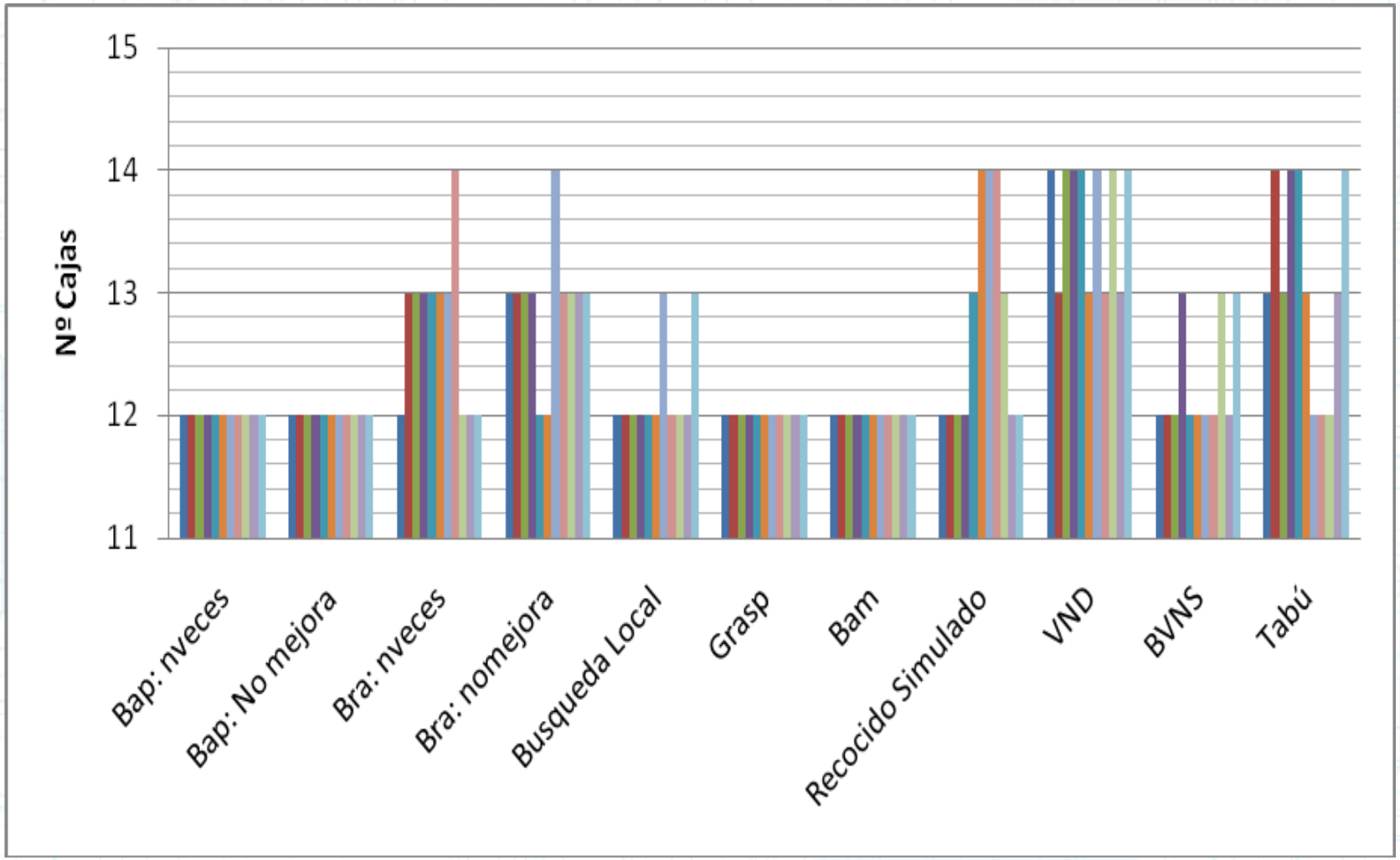
Presentación de resultados

- Tercer módulo de valores (caja 10x8 y 161 elementos):



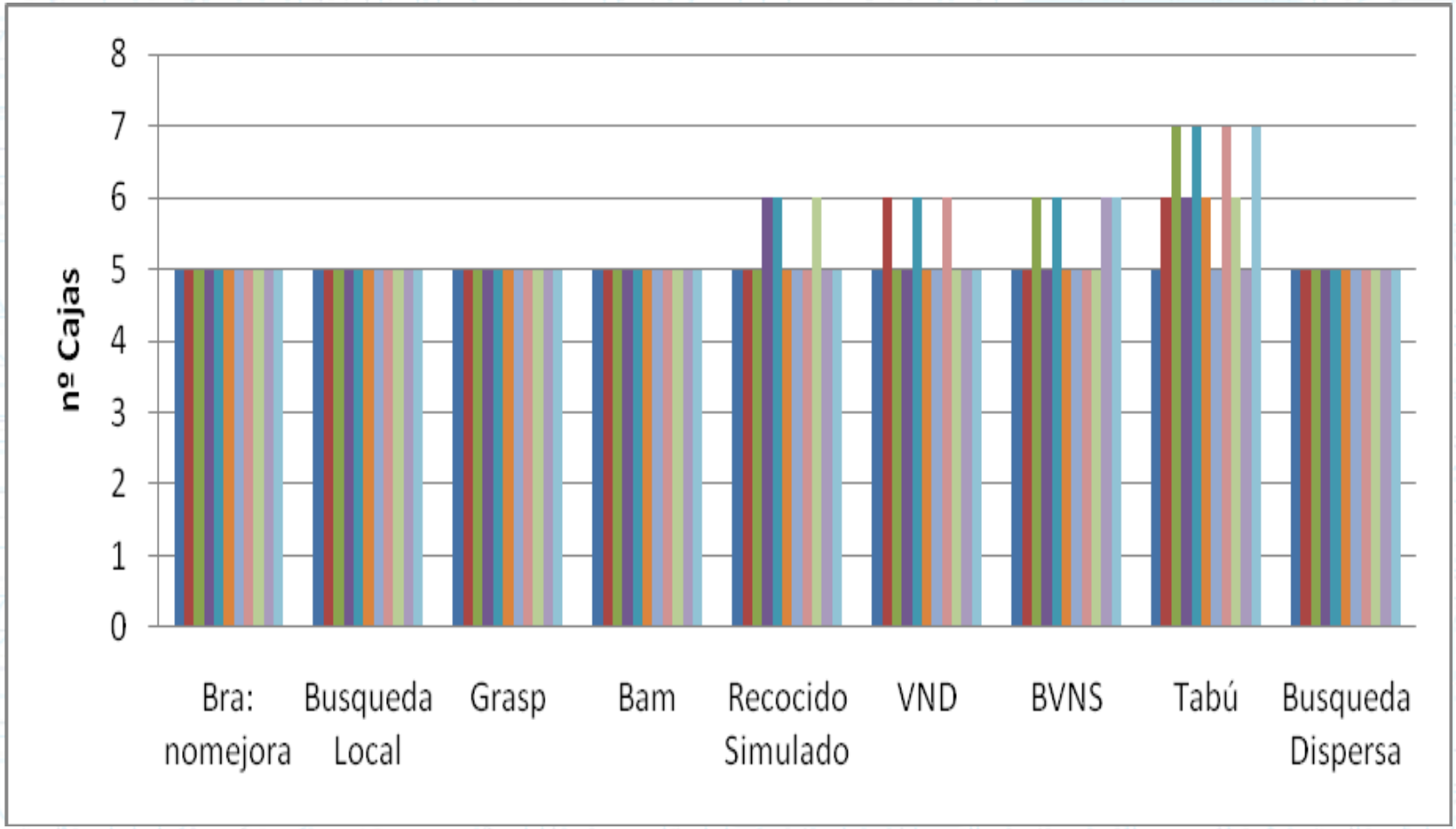
Presentación de resultados

- Cuarto módulo de valores (caja 20x20 y 225 elementos):



Presentación de resultados

- Quinto módulo de valores (caja 10x8 y 33 elementos):



Presentación de resultados

- Tiempos de ejecución de las diferentes heurísticas para un caso representativo.

Bap: No mejora	Bra: nveces	Bra: nveces	Bra: nomejora	Busqueda Local	Grasp
1998	1935	2820	3039	19	1645
Bam	Recocido Simulado	VND	BVNS	Tabú	Busqueda Dispersa
9167	3014	14	14	2695	120824

  - Observamos que las mas rápidas son el VND y el BVNS con apenas 14 milisegundos en su ejecución. Por el otro lado vemos que la búsqueda multiarranque tarda un poco más que las demás, y ya por último, vemos como la búsqueda dispersa dura más de 2 minutos en ejecutarse.