

Reglas de parada dependientes e independientes del problema. Estudio comparativo para el *Strip Packing Problem**

Jesús David Beltrán, Jose Eduardo Calderón, Rayco Jorge Cabrera, and J. Marcos Moreno Vega

Departamento de E.I.O. y Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de La Laguna. 38271 La Laguna
Santa Cruz de Tenerife
jmmoreno@ull.es

Resumen Probablemente, el criterio de parada sea el elemento de una Metaheurística que menos se ha estudiado. En general, las Metaheurísticas emplean criterios de parada independientes del problema que son ampliamente aplicables. En el presente trabajo se realiza un estudio comparativo entre reglas de parada independientes del problema y reglas dependientes del mismo para el *Strip Packing Problem*. La experiencia computacional muestra que estas últimas son más eficaces.

1. Introducción

Las Metaheurísticas de Búsqueda son procedimientos generales de resolución de problemas. Se caracterizan por ser aplicables a una gran variedad de problemas y por presentar, en la gran mayoría de los casos, un buen comportamiento. Es decir, suministran soluciones de alta calidad con un uso razonable de recursos (principalmente tiempo). Dentro de las metaheurísticas destacan las metaheurísticas de búsqueda por entornos. En estas, se considera, explícita o implícitamente, el concepto de estructura de entorno. Una estructura de entorno es una aplicación que, a cada solución del problema, asocia un conjunto de soluciones cercanas a la misma en algún sentido. Estas soluciones son las soluciones vecinas de la dada, y forman su entorno.

En una búsqueda por entorno, se realizan movimientos, desde la solución actual a alguna de su entorno, hasta que se cumpla el criterio de parada. El mecanismo que se usa para generar y aceptar una solución del entorno de la actual determina las diferentes metaheurísticas de búsqueda. Este mecanismo puede ser tan sencillo como generar una solución aleatoria y aceptarla siempre, o consistir en la aplicación de un proceso complejo, como aplicar un procedimiento

* Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia y Tecnología a través del proyecto TIC2002-04242-C03-01. Los fondos de este proyecto son, en un 70 %, fondos FEDER.

sofisticado de mejora. En todo caso, la búsqueda debe finalizar cuando se tenga cierta certeza de que se ha encontrado una solución de calidad. Esto puede conseguirse a través del criterio de parada usado en la búsqueda, ya que se pueden analizar las características de las soluciones encontradas y finalizar el procedimiento cuando estas características indiquen que se ha alcanzado una solución difícilmente mejorable.

En el presente trabajo, se realiza un estudio comparativo entre reglas de parada dependientes del problema y reglas independientes del mismo. El estudio pretende comparar la eficiencia y eficacia de varias heurísticas cuando se usan reglas de parada de los tipos anteriores. El trabajo se estructura como sigue. En la próxima sección se describen las Metaheurísticas de Búsqueda usadas en la experiencia computacional. En la sección 3, se enumeran propiedades deseables de una regla de parada y se definen las reglas dependientes independientes del problema. La sección 4 trata sobre el Strip Packing Problem y sobre la definición de una regla de parada para este problema. Por último, en la sección 5 se muestran los resultados computacionales obtenidos y se enumeran las conclusiones.

2. Metaheurísticas de Búsqueda

A continuación se describen las búsquedas por entornos usadas para comparar las reglas de parada dependientes e independientes del problema que definimos en la sección 3.

1. *Búsqueda Aleatoria Pura*: Generar aleatoriamente soluciones del problema hasta que se cumpla el criterio de parada. Devolver la mejor solución encontrada.
2. *Búsqueda Local*: Dada una solución inicial, determinar la mejor solución vecina de ésta. Si la nueva solución es mejor que la inicial, continuar el proceso con ella. En caso contrario, finalizar la búsqueda. Devolver la mejor solución encontrada.
3. *Búsqueda Multiarranque*: Aplicar búsquedas locales desde soluciones de inicio generadas aleatoriamente hasta que se cumpla el criterio de parada.
4. *Búsqueda por entorno variable (VNS) [7]*: Se parte de una solución inicial y de un conjunto de entornos $N_k, k = 1, \dots, k_{max}$ definidos sobre cualquier solución. Sea $k = 1$. Aplicar una búsqueda local con la estructura de entorno N_k hasta que se alcanza un óptimo local. Sea $k = k + 1$. Generar aleatoriamente una solución vecina del óptimo encontrado en la estructura N_k . Aplicar una búsqueda local con estructura N_1 desde la solución generada. Si se obtiene un óptimo local mejor que el actual, repetir el proceso con esta nueva solución. En caso contrario, hacer $k = k + 1$, generar una nueva solución vecina del óptimo actual en la estructura N_k y repetir. Cuando $k = k_{max}$, se genera una nueva solución de inicio y se repite el proceso hasta alcanzar un número máximo de iteraciones.
5. *Greedy Randomized Adaptive Search Procedure (GRASP) [6]*: Sea dada una función heurística h que mide la conveniencia de incluir un elemento como parte de la solución. En cada iteración, determinar el valor de la función h

sobre todos los elementos disponibles. Construir una lista restringida de candidatos (LRC) con los mejores elementos según el valor de h (en general, en LRC se incluyen los m (parámetro fijado por el usuario) mejores elementos). Seleccionar aleatoriamente un elemento de LRC e incluirlo en la solución parcial. Repetir el proceso anterior hasta que se obtenga una solución del problema. La anterior fase constructiva se repite hasta que se cumpla el criterio de parada. El GRASP para el problema del empaquetado rectangular bidimensional no guillotina usado en la experiencia computacional se describe en [2].

3. Reglas de Parada

En cualquier procedimiento de búsqueda de soluciones para un problema dado, uno de los elementos más importantes es el criterio de parada empleado. La regla de parada es responsable, en gran medida, del grado de eficiencia y eficacia del procedimiento de solución.

En ocasiones, el criterio de parada viene determinado por la búsqueda empleada. Así ocurre, por ejemplo, en una Búsqueda Local Descendente: el proceso de búsqueda finaliza cuando se encuentra una solución mejor que todas sus vecinas. Otras veces se emplean criterios de parada como finalizar después de un tiempo de CPU fijado a priori, o después de un número fijo de evaluaciones de la función objetivo. Estos criterios suelen ser poco eficaces, dado que no emplean ninguna información sobre la evolución de la búsqueda.

Se obtienen criterios de parada más eficaces al realizar un estudio del procedimiento, estudiar la función objetivo, estudiar la región factible, analizar la evolución de la búsqueda o evaluar las características de las soluciones obtenidas. En todo caso, cualquiera que sea la regla de parada empleada, ésta debe asegurar un equilibrio entre eficiencia y eficacia.

En [4] se enumeran las siguientes propiedades deseables de una regla de parada:

1. *Dependencia del problema:* si se conoce alguna propiedad de la región factible o de la función objetivo, ésta debe incluirse en el procedimiento para obtener reglas de parada. Así, en [9] se estudia la distribución del valor objetivo de los óptimos locales de la función objetivo y se obtienen reglas de parada empleando esta información.
2. *Dependencia muestral:* cuando se ejecuta una heurística para resolver un problema, se obtiene información de varias variables: valor objetivo, distancia entre óptimos locales, tamaño de la región de atracción de los óptimos locales, iteraciones necesarias para obtener el óptimo global, etc. El análisis de estos valores puede suministrar reglas de parada apropiadas. En [10] se aproxima la variable número de iteraciones necesarias para encontrar el óptimo global por medio de una distribución normal, que se emplea para obtener una regla de parada.

3. *Dependencia del método:* el estudio teórico de algunas heurísticas permite obtener reglas de parada que, al menos a nivel teórico, aseguran la convergencia al óptimo global del problema. Quizás el estudio más amplio en este sentido ha sido el realizado para el Recocido Simulado (ver [1]).
4. *Dependencia del costo y del recurso:* cuando se decide finalizar un procedimiento heurístico y se aporta como solución al problema la mejor solución encontrada, se incurre en dos pérdidas: una de finalización, que depende de la distancia entre el valor objetivo óptimo y aquel que suministra la heurística, y una de ejecución, que es función de la cantidad de recursos empleados. Estas pérdidas deben influir activamente en el criterio de parada. Obviamente, se pretende obtener una regla de parada que minimice ambas pérdidas. En [5] se realiza un amplio estudio de esta alternativa, y se proponen y analizan diversas reglas de parada.

Las reglas de parada, al igual que los procedimientos de solución, pueden clasificarse en generales y específicas para un problema.

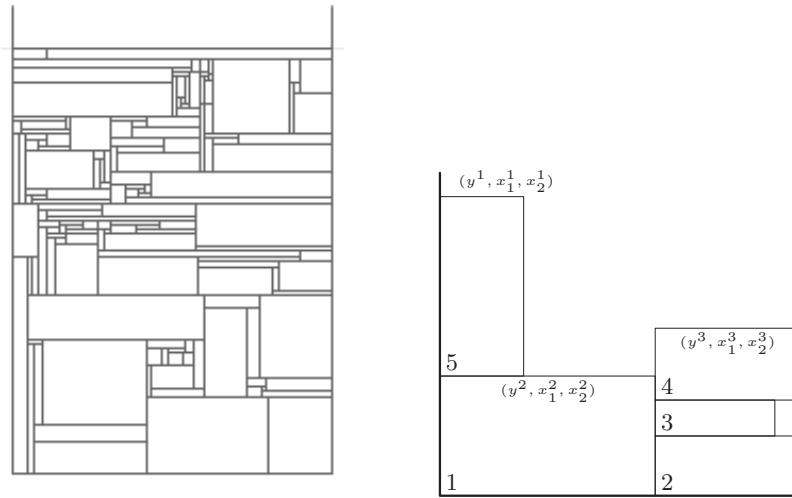
1. *Reglas de parada generales o independientes del problema.* Son reglas de parada aplicables a cualquier problema de optimización. Esto es así, dado que se basan en principios o estudios de uso general. Pertenecen a esta clase, los criterios poco eficaces de finalizar la búsqueda tras un número fijo de evaluaciones de la función objetivo, o al alcanzar un tiempo prefijado de CPU. También pertenecen a esta categoría los planes de enfriamiento que se obtienen para el Recocido Simulado [1] y las reglas de parada que se encuentran en [5],[9] y [10]
2. *Reglas de parada dependientes del problema.* En esta categoría se incluyen las reglas de parada que emplean propiedades o características específicas del problema (o soluciones del problema) que se aborda. En el presente trabajo se describen y analizan reglas de parada específicas para el problema del empaquetado rectangular bidimensional no guillotina.

4. Empaquetado rectangular bidimensional no guillotina

El problema del empaquetado rectangular bidimensional no guillotina (*Strip Packing Problem*) se formula como sigue. Dado un objeto rectangular de amplitud fija w y altura infinita, y un conjunto, $\mathcal{R} = \{R(w_1, h_1), \dots, R(w_n, h_n)\}$, de rectángulos con al menos uno de sus lados, w_i , h_i , menor que w , se desea empaquetar el conjunto \mathcal{R} en el objeto rectangular utilizando el menor espacio posible (o lo que es lo mismo, se pretende minimizar la altura del empaquetado). En este problema se pueden rotar los objetos y los cortes pueden ser de tipo no guillotina (ver figura 1(a)). Un corte es tipo guillotina si atraviesa el objeto desde un lado del mismo hasta el lado opuesto. En un corte no guillotina, lo anterior no es cierto.

4.1. Reglas de parada dependientes del problema

En general, la calidad de una solución se evalúa atendiendo únicamente al valor objetivo. Sin embargo, existen otros valores de la solución que indican la



(a) Ejemplo de empaquetado de rectángulos

(b) Contorno

Figura 1. Empaquetado y contorno

calidad de ésta y la posibilidad que existe de mejorarla. En el problema del empaquetado rectangular algunos de estos valores son: área de los desperdicios, forma del contorno superior o distribución de los items en el objeto.

En [3] se usan algunos de estos valores para evaluar las soluciones del problema del empaquetado rectangular bidimensional y para diseñar, por tanto, reglas de parada. Las reglas de parada propuestas emplean el valor de varios parámetros que caracterizan la bondad de una solución. El valor alcanzado en una solución del problema se compara con el valor que se considera deseable. Es decir, con el valor que satisface al decisor o que caracteriza una situación no mejorable eficientemente. El resultado de esta comparación indica si se debe continuar o no la búsqueda.

Dada una solución arbitraria, X , podemos representar el contorno superior de la misma por un conjunto de segmentos (o niveles), tomados de izquierda a derecha (en la figura 1(b) el contorno lo determinan los items 1, 4 y 5), como el que sigue:

$$\mathcal{C} = \{(y^1, x_1^1, x_2^1), (y^2, x_1^2, x_2^2), \dots, (y^c, x_1^c, x_2^c)\},$$

donde:

$$\begin{aligned} y^i &\equiv \text{altura del } i\text{-ésimo segmento} \\ x_1^i &\equiv \text{punto inicial del } i\text{-ésimo segmento.} \\ x_2^i &\equiv \text{punto final del } i\text{-ésimo segmento} \end{aligned}$$

Además, $x_1^1 = 0$ y $x_2^c = w$.

En la figura 1(b) se observa también que, como consecuencia del proceso de búsqueda, pueden aparecer en la solución áreas no aprovechadas, llamadas *desperdicios*.

Dada una solución X , la posibilidad de obtener una solución mejor que ésta depende, en gran medida, de la forma del contorno superior y del área total de sus desperdicios. En general, las soluciones con contornos superiores suaves y desperdicios pequeños son difícilmente mejorables.

Sea $f(X)$ el valor objetivo de X y denotemos por $Desperdicio(X)$, al área total de los desperdicios de esta solución. Nótese que

$$f(X) = \max_{i=1,\dots,c} \{y^i\}.$$

Para medir la suavidad del contorno superior usamos la altura media de los niveles del contorno definida como:

$$AlturaMedia(X) = \frac{1}{c} \sum_{i=1}^c (f(X) - y^i)$$

Valores próximos a cero se corresponden con contornos suaves.

Así, la combinación de $Desperdicio(X)$ y $AlturaMedia(X)$ puede usarse para evaluar la calidad de una solución y, por tanto, como criterio de parada de cualquier procedimiento de resolución para este problema.

1. *Criterio de parada.* Finalizar la búsqueda cuando

$$Desperdicio(X) \leq \alpha_1 \wedge AlturaMedia(X) \leq \alpha_2 \quad (1)$$

En la experiencia computacional, los parámetros α_1 y α_2 se fijaron atendiendo al siguiente convenio. Sea $A = \sum_{i=1}^n w_i \cdot h_i$ el área total de los rectángulos. Los valores de α_1 indican el porcentaje de A que se considera en la regla de parada. Si, por ejemplo, se toma $\alpha_1 = 0,01$, esto quiere decir que el porcentaje de desperdicios debe ser menor o igual al 1% del área total de los rectángulos a empaquetar. Los valores de α_2 son absolutos: indican la altura media que se toma.

5. Experiencia Computacional, Conclusiones y Trabajos Futuros.

Para comparar la eficiencia y eficacia de la regla de parada dependiente del problema frente a las reglas de parada independientes del mismo, se resolvieron problemas de empaquetado con las diferentes metaheurísticas de búsqueda enumeradas en la sección 2. Para cada metaheurística se usó la regla de parada dependiente del problema descrita anteriormente y una regla de parada independiente del problema. Para evitar un número excesivo de iteraciones al usar la regla de parada dependiente del problema, se impuso que, en cualquier caso, no se alcanzase el valor de 1000 en el parámetro que determina la regla de

parada independiente del problema. Los casos en los que se alcanzó este límite máximo en alguna ejecución se muestran subrayados en las tablas de resultados. Indican ejecuciones en las que no se cumplió la regla de parada dependiente del problema. Las reglas de parada independientes del problema consideradas son: *Búsqueda Aleatoria*, número máximo de iteraciones (n_{iter}); *Búsqueda Multiarranque*, número máximo de búsquedas locales (n_{bl}); *GRASP*: número máximo de fases constructivas (n_{cons}); *Búsqueda por entornos variable*: número de máximo de iteraciones (n_{iter}). Los problemas tests usados son los correspondientes a las categorías C_1, C_2, \dots, C_6 de Hopper y Turton [8] (disponibles en la siguiente dirección: mscmga.ms.ic.ac.uk/jeb/orlib/stripinfo.html).

Cada categoría está formada por 3 problemas con las siguientes características: categoría C_1 , $n = 16$ o 17 rectángulos, amplitud $w = 20$, altura óptima $h_{opt} = 20$; categoría C_2 , $n = 25$ rectángulos, amplitud $w = 40$, altura óptima $h_{opt} = 15$; categoría C_3 , $n = 28$ o 29 rectángulos, amplitud $w = 60$, altura óptima $h_{opt} = 30$; categoría C_4 , $n = 49$ rectángulos, amplitud $w = 60$, altura óptima $h_{opt} = 60$; categoría C_5 , $n = 72$ o 73 rectángulos, amplitud $w = 60$, altura óptima $h_{opt} = 90$; categoría C_6 , $n = 97$ rectángulos, amplitud $w = 80$, altura óptima $h_{opt} = 120$.

Las tablas 1, 2, 3 y 4 recogen, respectivamente, los resultados obtenidos con la Búsqueda Aleatoria Pura, la Búsqueda Multiarranque, VNS y GRASP. La primera columna de estas tablas indica la categoría de problemas de Hopper y Turton considerados. A continuación aparecen los valores objetivos y el tiempo de CPU consumido por las correspondientes heurísticas para los dos tipos de reglas de parada. Se trata de valores promedios (mínimo, medio y máximo) sobre los 3 problemas de cada categoría. Cada uno de estos problemas fue resuelto 10 veces con cada heurística. Para las reglas independientes del problema se consideraron varios valores para el parámetro que determina dichas reglas. Esto se refleja en las correspondientes tablas de resultados. Para las reglas de parada dependientes del problema se fijó experimentalmente los valores de α_1 y α_2 (ver expresión 1, página 6) para cada categoría. Se fijaron varios valores para los parámetros y, en base a la calidad de las soluciones obtenidas y del tiempo empleado por las correspondientes heurísticas, tomaron los que presentaron un mejor comportamiento.

De los resultados obtenidos se concluye que:

1. *La regla de parada dependiente del problema es más eficaz que la regla de parada independiente del mismo.* En casi todos los casos, se obtuvieron mejores soluciones con el primer tipo de regla de parada. La única heurística para la que este hecho no fue cierto es la Búsqueda Aleatoria Pura. Esto se explica por el mayor número de iteraciones desarrolladas con la regla independiente del problema, y por la baja calidad de las soluciones obtenidas con la Búsqueda Aleatoria Pura.
2. *Cuanto mejor es la heurística mas recomendable es usar reglas de parada dependientes del problema.* En heurísticas que suministran soluciones de baja calidad, como la Búsqueda Aleatoria Pura, es improbable que se encuentre una solución que cumpla el criterio de parada dependiente del problema.

El caso contrario se presenta para el GRASP, que es capaz de encontrar soluciones que satisfacen el criterio de parada en pocas iteraciones.

3. *Las reglas de parada independientes del problema son, en general, mas eficientes.* El tiempo requerido por la regla de parada que hemos propuesto es sensiblemente mayor que el empleado por las reglas independientes del problema. Las diferencias disminuyen cuando las reglas se usan en heurísticas mas elaboradas (VNS y GRASP).
4. *Combinar buenas heurísticas con criterios de parada dependientes del problema suministra buenos procedimientos para resolver el problema.* A la eficiencia de las heurísticas se une la eficacia de las reglas de parada por lo que el usuario dispondrá de un procedimiento que garantiza la obtención de buenas soluciones en un tiempo razonable.

Algunos de las líneas futuras de investigación que nos planteamos son:

1. Comparar las reglas de parada dependientes del problema con otras independientes del mismo no contempladas hasta ahora. En particular con las propuestas que pueden encontrarse en [9] y [11].
2. Diseñar y evaluar nuevas reglas de parada dependientes del problema, tanto para el *Strip Packing Problem*, como para otros problemas.

Cuadro 1. Búsqueda Aleatoria Pura. Valores objetivos y tiempos de CPU (promedios por categoría) obtenidos con las reglas dependientes e independientes del problema.

Categoría	$n_{iter} = 1000$		$n_{iter} = 3000$		Dependiente	
	Obj	CPU	Obj	CPU	Obj	CPU
C_1	22.00	0.0533	21.67	0.2161	22.00	0.0000
	22.30	0.0897	22.53	0.2492	22.80	0.0484
	23.00	0.1113	23.00	0.2779	23.67	0.1106
C_2	15.67	0.1093	15.67	0.3638	16.67	0.0000
	16.33	0.1204	16.06	0.3859	16.80	0.0360
	17.00	0.1438	16.33	0.4238	18.00	0.0904
C_3	32.67	0.1601	32.67	0.5097	33.00	0.0000
	33.86	0.1700	33.53	0.5343	33.70	0.0568
	34.67	0.2031	34.67	0.5702	34.30	0.1503
C_4	65.33	0.3228	64.67	1.0058	66.00	0.0000
	66.80	0.3457	66.20	1.0419	67.13	0.1119
	69.00	0.3833	67.33	1.0670	69.00	0.3098
C_5	97.67	0.4934	98.00	1.5394	98.33	0.1600
	99.80	0.5280	99.40	1.5629	99.86	0.2154
	101.00	0.5507	101.00	1.5963	100.67	0.5103
C_6	133.00	0.8391	131.20	2.5598	132.67	0.0729
	133.76	0.8606	132.80	2.6115	134.23	0.4910
	135.30	0.8801	133.67	2.6601	136.00	0.8430

Referencias

1. Aarts, E, Korst, J.: Simulated Annealing and Boltzmann Machines. John Wiley and Sons (1989)

Cuadro 2. Multiarranque. Valores objetivos y tiempos de CPU (promedios por categoría) obtenidos con las reglas dependientes e independientes del problema.

Categoría	$n_{bl} = 5$		$n_{bl} = 10$		Dependiente	
	Obj	CPU	Obj	CPU	Obj	CPU
C_1	21.00	0.0494	20.67	0.0728	20.34	0.0000
	21.07	0.0678	20.97	0.1276	20.70	4.1755
	21.67	0.1106	21.34	0.1696	21.00	<u>8.6399</u>
C_2	16.00	0.2193	16.00	0.4394	15.67	0.1432
	16.03	0.2577	16.00	0.5181	16.06	5.8133
	16.34	0.3300	16.00	0.5833	16.34	<u>50.9374</u>
C_3	32.00	0.4933	31.67	1.0396	31.00	0.1800
	32.23	0.5730	32.13	1.1599	31.77	19.5333
	33.00	0.6601	32.67	1.3163	32.67	<u>67.9766</u>
C_4	62.00	3.3496	63.00	6.8105	63.34	0.6564
	64.50	4.1564	63.93	8.2131	64.43	7.5621
	65.00	4.9414	64.00	8.4609	65.34	46.1565
C_5	94.67	12.5468	95.00	25.6503	94.34	14.2535
	96.50	15.3992	96.03	31.1182	95.17	113.7660
	98.00	18.8735	97.00	35.5898	96.00	488.5232
C_6	127.67	40.1327	128.00	88.6536	126.34	18.9444
	129.63	50.8121	128.83	104.6234	128.03	448.2769
	131.00	61.5333	130.00	119.4967	129.34	612.7662

Cuadro 3. VNS. Valores objetivos y tiempos de CPU (promedios por categoría) obtenidos con las reglas dependientes e independientes del problema.

Categoría	$n_{iter} = 5$							
	$k_{max} = 2$		$k_{max} = 3$		$k_{max} = 2$		$k_{max} = 3$	
	Obj	CPU	Obj	CPU	Obj	CPU	Obj	CPU
C_1	21.00	0.0499	21.00	0.0500	21.00	0.0162	20.67	0.0000
	21.47	0.0703	21.23	0.1043	21.10	5.7957	20.87	5.9228
C_2	22.00	0.1101	22.00	0.1301	21.67	<u>14.2806</u>	21.00	<u>19.9732</u>
	16.00	0.2534	16.00	0.3298	16.00	0.0707	16.00	0.0689
	16.33	0.3243	16.23	0.4446	16.03	6.7811	16.16	8.1192
C_3	17.00	0.4066	17.00	0.5298	16.34	<u>33.4999</u>	16.67	<u>53.5325</u>
	31.67	0.5262	31.34	0.7332	31.00	0.1793	31.00	0.2367
	32.70	0.6713	32.23	1.0033	32.13	43.7602	31.96	32.6896
C_4	34.34	0.8067	33.00	1.3000	33.34	<u>98.0533</u>	32.67	<u>166.0765</u>
	63.67	4.1732	63.00	6.5165	63.00	1.6466	62.67	1.8499
	64.40	5.4056	63.83	7.8573	63.70	19.4276	63.46	34.4796
C_5	65.67	6.1132	64.67	8.8033	65.00	425.4091	64.34	247.3433
	94.67	18.3432	94.67	25.5599	94.67	5.9132	94.67	9.0267
	95.83	21.7066	95.53	31.3406	95.46	45.0673	95.66	34.0690
C_6	97.67	25.8699	96.67	63.3432	96.34	192.6066	97.34	114.6866
	126.67	55.8966	126.00	88.4876	127.34	22.7571	125.67	23.4500
	128.30	74.4796	128.10	108.0381	128.56	75.7498	128.36	56.3257
	130.00	98.8332	129.34	134.0592	130.00	282.9195	129.67	164.3001

Cuadro 4. GRASP. Valores objetivos y tiempos de CPU (promedios por categoría) obtenidos con las reglas dependientes e independientes del problema.

Categoría	Independiente		Dependiente	
	Obj	CPU	Obj	CPU
C_1	21.33	0.0000	21.67	0.0000
	22.17	0.0000	21.67	0.0019
	23.67	0.0000	21.67	0.0198
C_2	16.34	0.0000	16.00	0.0000
	16.90	0.0017	16.00	0.0037
	17.67	0.0169	16.00	0.0201
C_3	31.67	0.0000	31.67	0.0000
	33.00	0.0000	31.93	0.0127
	33.66	0.0000	32.00	0.0400
C_4	62.00	0.0000	61.67	0.0000
	63.10	0.0016	62.00	0.0087
	64.67	0.0162	62.33	0.0539
C_5	92.00	0.0000	92.00	0.0000
	93.30	0.0020	92.67	0.0133
	94.34	0.0000	93.00	0.0569
C_6	122.67	0.0000	122.00	0.0000
	123.87	0.0106	122.93	0.0470
	125.00	0.0195	123.67	0.1100

2. Beltrán, Jesús David; Eduardo Calderón, Jose; Jorge Cabrera, Rayco; Moreno Vega, J. Marcos: Procedimientos constructivos adaptativos (GRASP) para el problema del empaquetado de rectángulos Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial **15** (2002) pp. 26–3
3. Beltrán, Jesús David; Eduardo Calderón, Jose; Jorge Cabrera, Rayco; Moreno Vega, J. Marcos: Reglas de Parada para el Problema del Empaquetado Rectangular Bidimensional No Guillotina. Actas de la VIII Conferencia Iberoamericana de Inteligencia Artificial (IBERAMIA) (2002)
4. Boender, C.G.E., Rinnooy Kan, A.H.G., Vercellis, C.: Stochastic Optimization Methods. Stochastics in Combinatorial Optimization (1986) 94–112
5. Boender, C.G.E., Rinnooy Kan, A.H.G.: Bayesian Stopping Rules for Multistart Global Optimization Methods. Mathematical Programming **37** (1987) 59–80
6. Feo, T. A.; Resende, M.G.C.: Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization **6** (1995) 109–133
7. Hansen, P.; Mladenovic, N. An Introduction to Variable Neighborhood Search, en Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization, Kluwer, 1999
8. Hopper, E., Turton, B.C.H.: A Review of the Application of Meta-Heuristics Algorithms to 2D Strip Packing Problems. Artificial Intelligence Review **16** (2001) 257–300
9. Los, M., Lardinois, C.: Combinatorial Programming, Statistical Optimization and the Optimal Transportation Network Problem. Transportation Research **2** (1982) 89–124
10. Moreno-Vega, J. M., Moreno, J. A.: Una Regla de Parada para la Búsqueda con Arranque Múltiple. Actas de las I Jornadas de Informática. (1995) 271–280
11. Zhang, Weixiong: Iterative state-space reduction for flexible computation. Artificial Intelligence **126** (2001) 109–138