Método Multi-Arranque aplicado al problema del Strip Packing Problem bidimensional

Alfonso Fernández Timón[1] y Abraham Duarte Muñoz[2]

Resumen--***Strip Packing Problem*** **en dos dimensiones pertenece a la categoría de problemas de empaquetado con grandes aplicaciones en la Industria actual. Trata la disposición de un conjunto de rectángulos de diferentes dimensiones sobre una cinta de anchura fija y altura indeterminada, de manera que la altura alcanzada por dichos objetos sea la menor posible. Estos objetos no pueden superponerse y, en determinados casos, pueden ser girados respecto a su orientación inicial. Debido a su gran aplicabilidad, se han desarrollado numerosos algoritmos para su resolución. En este trabajo hemos diseñado un algoritmo basado en una metaheurística Multi-Arranque. Este método ha sido aplicado a una conocida base de problemas *test* y los resultados obtenidos han sido comparados de manera satisfactoria con otros procedimientos encontrados en la literatura.**

Palabras clave—**Multi-Arranque, *Strip Packing Problem*.**

I. INTRODUCCIÓN

Los problemas de corte y empaquetado pertenecen a la categoría de problemas de optimización combinatoria. En ellos se dispone de un conjunto de piezas o ítems, de diferentes formas y tamaños, que han de ser colocadas sobre diferentes patrones sin superponerse unas sobre otras. El objetivo de tal disposición es minimizar el área desperdiciada o residuos, que correspondería con aquellas zonas de los patrones no ocupadas por ninguna pieza.

Existen diferentes criterios para clasificar los problemas de empaquetado, como son el tamaño y la cantidad de las piezas, las dimensiones del problema, etc. Uno de los métodos utilizados más frecuentemente para clasificar los problemas de empaquetados es aquel basado en las características del patrón sobre el cual se van a colocar los objetos. En base a este criterio, los problemas de empaquetado pueden ser divididos en dos categorías:

- *Bin Packing*: Los objetos han de ser colocados sobre un conjunto de patrones de dimensiones fijas, con el objetivo de minimizar el número de patrones utilizados.
- *Strip Packing*: Los objetos son dispuestos sobre un patrón donde todas sus dimensiones son fijas menos una, de tal manera que la longitud final alcanzada por los mismos en esta dimensión sea la menor posible.

Los problemas de empaquetado tienen una amplia aplicación en distintas facetas industriales: textil, cristal, piel, madera. Sin embargo, diferentes inconvenientes particulares a cada problema determinan la disposición de las piezas sobre el patrón. Entre estos inconvenientes podemos destacar las propiedades del material del patrón (inhomogeneidades en el color, calidad,…), la tecnología de corte (que obliga a que exista una determinada distancia entre las piezas), secuencia de empaquetado (los objetos frágiles no pueden estar situados al fondo del patrón), áreas vacías en el interior de objetos irregulares (que pueden ser aprovechadas), etc.

II. STRIP PACKING PROBLEM

El *Strip Packing Problem* (SPP) es un problema de optimización, que consiste en la colocación de una serie de objetos sobre un patrón donde todas sus dimensiones menos una son fijas. La longitud de esta dimensión es precisamente el parámetro a minimizar mediante la correcta colocación de los objetos.

En este trabajo nos restringiremos al caso de dos dimensiones, siendo los ítems a empaquetar rectángulos. Por lo tanto, los rectángulos $R(w_i,h_i)$ (para $i = 1, ..., n$, donde w_i representa la anchura e h_i la altura del rectángulo i) han de ser colocados sobre un objeto llamado banda, de anchura fija (w) y altura, en principio, infinita, teniendo como objetivo final minimizar la altura alcanzada por los mismos (Figura 1). Para ello, los rectángulos, que no pueden solaparse, tienen que cumplir una serie de requisitos:

- Como mínimo, uno de los lados de cada rectángulo ha de tener una longitud menor a w.
- Una vez colocados, los lados de los rectángulos han de estar paralelos a los ejes de la banda.
- Los rectángulos pueden ser o no rotados ± 90º.

A estas restricciones se le suele unir una más, relativa a si las piezas colocadas han de seguir algún corte de tipo guillotina. Un corte es de tipo guillotina si cuando se aplica sobre un rectángulo produce dos rectángulos nuevos. Un patrón de ordenación (o solución final) es de tipo guillotina si

[1] ESCET, URJC, Avda Tulipan S/N, Móstoles. E-mail: a1fonso.fernandez@urjc.es
[2] ESCET, URJC, Avda Tulipan S/N, Móstoles. E-mail: abraham.duarte@urjc.es

se puede obtener por sucesivos cortes de tipo guillotina [1]. En otras palabras, la disposición final de los objetos en el patrón es de tipo guillotina, si la misma puede ser conseguida mediante sucesivos cortes de lado a lado de alguno de los rectángulos que se van obteniendo con anterioridad. En caso contrario, como es nuestro caso, se dice que el problema es de tipo no guillotinable.

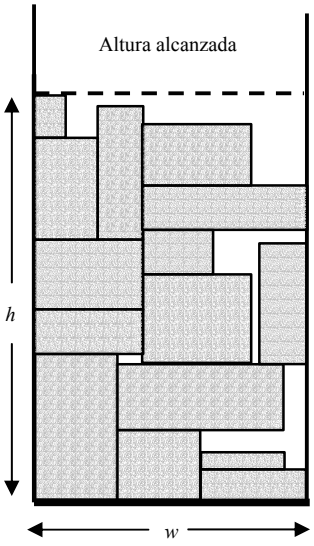

Fig. 1. Solución para un problema de SPP .

Cada solución al problema del SPP vendrá representada por un conjunto de permutaciones $\{(r_i,a_i,b_i)$ para $i = 1, ..., n\}$ de los rectángulos $\{R(w_i,h_i)\ i = 1, ..., n\}$, donde:

- r_i representa si $R(w_i,h_i)$ es rotado o no (r_i=0 representaría al objeto no girado).
- (a_i,b_i) son las coordenadas de la posición de la esquina inferior izquierda del rectángulo respecto al origen de coordenadas (Figura 2).

El problema del SPP está ampliamente documentado por un gran número de trabajos. En base a diversas heurísticas y metaheurísticas, se han desarrollado gran cantidad de algoritmos cuyo objetivo es encontrar, dentro del conjunto de posibles soluciones, aquella o aquellas cuya altura final esté más próxima al valor óptimo del problema. En el trabajo de Turton y Hopper [2], podemos encontrar una amplia referencia acerca de estos procedimientos heurísticos y metaheurísticos aplicados a la problemática del SPP bidimensional. Entre estas diversas heurísticas se encuentran las conocidas como *Bottom-Left* (BL), *Bottom Left Fill* (BLF), *Difference Process* (DP), las heurísticas de nivel (*First Fit Decrease Height* y *Next Fit Decrease Height*) y *hill-climbing*, entre otras.

Sin embargo, el mayor número de procedimientos desarrollados para encontrar soluciones adecuadas a los diferentes problemas de SPP, se basan en metaheuristicas muy variadas, como son el Recocido Simulado (SA) [3], Búsqueda Tabú (TS) [4], Algoritmos Genéticos [5] y GRASP [6,7].

En este trabajo hemos desarrollado un algoritmo basado en una metaheuristica Multi-Arranque. Lo hemos aplicado sobre una serie de problemas *tests* tomados del trabajo anteriormente referenciado de Hopper y Turton y los resultados obtenidos han sido comparados con los obtenidos en otros trabajos para los mismos problemas, con objeto de comprobar la bondad del procedimiento desarrollado.

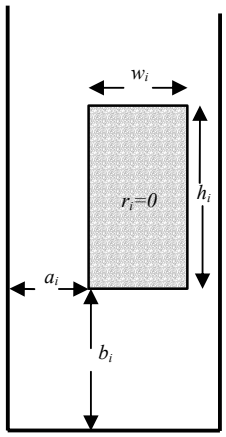

Fig. 2. Solución para un item *i*.

III. ALGORITMO MULTI-ARRANQUE PARA EL SPP

Un procedimiento multi-arranque es un algoritmo iterativo en el que cada iteración tiene dos fases [8]. En la primera fase se construye una solución, ya sea aleatoriamente o mediante un procedimiento heurístico. Una vez que se tiene una solución al problema, en la segunda fase, dicha solución generada se mejora (aunque no es necesario) mediante un algoritmo de búsqueda local o incluso otra metaheurística. El algoritmo desarrollado en este trabajo que permite resolver el SPP se basa en un procedimiento que tiene dos grandes bloques. A saber, un método constructivo aleatorizado y una búsqueda local. Este procedimiento se re-arranca un determinado número de veces, hasta que se satisface un criterio de parada. Cada ejecución del algoritmo multi-arranque es completamente independiente de las demás. En la Figura 3 se presenta un pseudo-código del algoritmo propuesto. En las siguientes sub-secciones, se describirá cada bloque por separado.

A. Métodos Constructivos para el SPP

La mayoría de las soluciones al problema del SPP se basan en un método constructivo, en el cual se añaden iterativamente elementos a una estructura inicialmente vacía (lo que se conoce como cinta,

banda, estructura o *strip*). El algoritmo desarrollado determina, en cada momento, qué pieza ha de ser incluida en cada iteración. Además, también se decide la estrategia de colocación de la misma en la estructura a rellenar.

```
{x: TipoSolucion} = MSM(M: integer; f : TipoFuncnObjetivo)
var
  i: integer;
  xtrial : TipoSolucion;
begin
  for i := 1 to M do
/*Construir una solución a través de cualquier heurístico o
proc. aleatorio*/
    fxtrialg :=ConstruirSolucion();
    fxtrialg := MejorarSolucion(xtrial ; f );
    fxg := ActualizarSolucion( f ;x;xtrial)
  end for
end
```

Fig. 3. Pseudo-código del algoritmo multi-arranque para el SPP.

Dentro de la fase constructiva, el primer paso de nuestro algoritmo consiste en ordenar los ítems a colocar en función de un criterio heurístico. Inicialmente escogemos cuatro criterios de ordenación:

- **Lado mayor (*L*):** ordena las piezas en función del lado mayor de cada rectángulo.
- **Perímetro (*P*):** ordena las piezas en función de la suma de los cuatro lados del rectángulo.
- **Área (*A*):** ordena las piezas en función del producto del lado mayor por el lado menor.
- ***Ratio* (*R*):** ordena las piezas en función del cociente entre el lado mayor y el lado menor.

Una vez ordenado el conjunto de piezas a colocar en función de cualquiera de estos criterios, hay que ir escogiendo sucesivamente a una de entre todas ellas. La pieza escogida desaparecerá de la lista de candidatos elegibles y pasará a formar parte de la solución.

Para seleccionar la pieza a colocar en cada iteración, creamos lo que en el contexto de GRASP [9] se conoce como una lista restringida de candidatos (LRC). De forma general y para problemas de minimización, esta lista viene definida por la siguiente expresión:

$$LRC = \{c_i \mid c_i \geq (c_{min} + \alpha(c_{max} - c_{min})),\ i = 1, ..,n\}$$

donde c_i es el coste asociado al elemento i (donde se ha asumido que hay n elementos), c_{min} es el menor coste y c_{max}, el mayor. El parámetro α: $0 \leq \alpha \leq 1$, determina el tamaño de la lista, de tal forma que si $\alpha = 1$, en la LRC sólo estaría en mejor candidato (función miope, voraz o *greedy* pura). En cambio, si $\alpha = 0$, estarían todos los elementos (función aleatoria pura).

Para el SPP, la LRC la conformarán, para cada iteración, aquellas piezas sin colocar cuyos parámetros de ordenación (lado mayor, perímetro, volumen o ratio) superen, en un porcentaje dado por α, un valor determinado. Este valor, después de unos experimentos previos, fue fijado, en todos los casos analizados en este estudio, a la semisuma de los parámetros de ordenación del primer y del último elemento de la lista que no hubieran sido previamente colocados. Este valor se actualiza para cada iteración, determinando una nueva lista restringida de candidatos, de la cual se escoge aleatoriamente un elemento de la misma. De esta manera, y para un mismo conjunto de piezas a colocar, se consiguen soluciones distintas, debido al carácter no determinista del algoritmo de elección de piezas. La Figura 4 muestra este método de elección para la primera iteración (primera pieza), cuando el parámetro de ordenación es el perímetro de los ítems. En la primera lista, aparecen todos los elementos seleccionables representados por las dimensiones de cada rectángulo. Nótese que por simplicidad, se ha omitido el perímetro asociado a cada pieza.

En la segunda lista, aparecen los elementos de la LRC; es decir, los elementos que todavía no han sido colocados cuyo perímetro supera la semisuma entre el mayor y el menor. En este caso en concreto:

- Perímetro de la pieza mayor: 36
- Perímetro de la pieza menor: 8
- Semisuma: (36 + 8) / 2 = 44 / 2 = 22

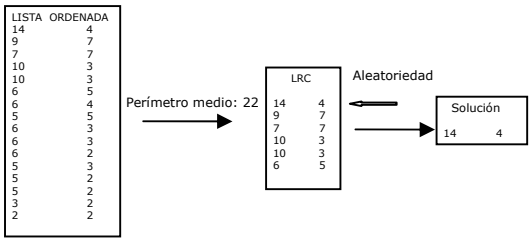

Fig. 4. LRC para la primera iteración siendo parámetro de ordenación el perímetro.

De entre ellos elegimos aleatoriamente a una pieza (en este caso la primera), representada en la Figura 4 mediante la última lista.

Indicar que el método descrito en esta sección no se ha considerado un método GRASP puro debido a que la LRC no se adapta en el sentido que se requiere en los algoritmos GRASP; es decir, la ordenación inicial de la lista no cambia durante la ejecución del algoritmo y sólo se actualiza porque la pieza seleccionada se elimina de la LRC, pero la ordenación permanece inalterada.

B. *Estrategia de colocación*

Una vez seleccionada la pieza, el siguiente paso consiste en colocarla en la posición más adecuada con respecto a un determinado criterio en la banda. En este trabajo se han estudiado e implementado dos métodos: BL y BLF [10,11]. La estrategia BL dicta que cada pieza a colocar se posiciona inicialmente en la esquina superior derecha de la estructura. A continuación, se desplaza hasta la posición más profunda posible. Una vez allí, la pieza es movida hacia la izquierda tanto como sea posible, repitiéndose esta rutina hasta que la pieza alcance una posición inamovible. El problema fundamental que presenta esta estrategia es que conduce a soluciones con gran número de desperdicios o residuos en la estructura a rellenar. En la Figura 5 se presenta gráficamente cómo actúa dicha estrategia.

Con el objetivo de minimizar el desperdicio de espacio, se propone el uso de la estrategia BLF, que tiene ciertas similitudes a la BL, aunque antes de colocar la pieza se comprueba que ésta no cabe en ninguno de los huecos generados hasta el momento a un nivel inferior. De nuevo, en la Figura 5 se muestra la solución a la que daría lugar esta estrategia para una pieza determinada.

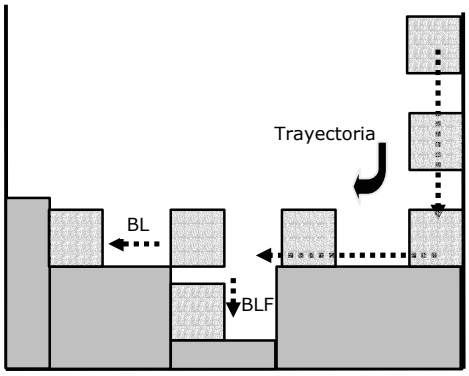

Fig. 5. Estrategias BL y BLF.

La estrategia BLF consiste en encontrar un hueco para la pieza comenzando desde la esquina inferior izquierda de la banda (origen de coordenadas). Esta comprobación se hace nivel a nivel desde el fondo de la banda. En todos los casos se probó inicialmente con la orientación de la pieza que hacía corresponder como base del rectángulo a su lado mayor. En caso de no encontrar ubicación para la pieza dentro de un determinado nivel, se probaba con la pieza girada 90 grados, antes de pasar al nivel inmediatamente superior. Todo el proceso descrito anteriormente se repite hasta la colocación completa de todas las piezas, dando como resultado del procedimiento constructivo una solución final del problema.

C. *Búsqueda local*

La estrategia constructiva y el método de colocación descritos en las sub-secciones previas conducen, en ocasiones, a resultados discretos, lo que induce la introducción de una fase de mejora que tome como partida la solución alcanzada en la fase constructiva. Por lo tanto, la segunda fase de nuestro algoritmo consiste en encontrar, a partir de la solución inicial constructiva, una disposición final de los ítems que mejore la de partida.

El procedimiento de búsqueda local se basa en la extracción de un conjunto de piezas situadas en los niveles superiores de la solución construida. El número de piezas que se extraen de la solución inicial es un parámetro del algoritmo, fijado mediante una experimentación previa. Nótese que se extraen las piezas del final, porque entre las que están colocadas en los niveles inferiores, prácticamente no existe espacio desperdiciado. Este hecho se ha verificado experimentalmente.

Una vez extraídas las piezas, éstas son ordenadas en función del criterio elegido inicialmente. A partir de aquí los pasos seguidos coinciden con los tomados en la fase constructiva; es decir, se crea la lista restringida de candidatos, se escoge de manera aleatoria a una de ellas y se coloca sobre la estructura que quedó después de la extracción de las piezas. Esta rutina se sigue para todas las piezas restantes, alcanzándose una solución final que puede mejorar a la de partida. El procedimiento que acabamos de presentar se repite para una misma solución constructiva una serie de veces (ver Tabla I). De nuevo, este es un parámetro del algoritmo que se fijó mediante una serie de experimentos preliminares.

D. *Multi-arranque*

Hasta el momento, se ha descrito cual es la estrategia de intensificación del algoritmo multi-arranque. En general, esta estrategia conduce a un óptimo local. Con el objetivo de evitar quedar atrapados en óptimos locales, los procedimientos multi-arranque re-arrancan el algoritmo, lo que le permite diversificar la búsqueda. Debido a la inclusión de componentes aleatorias en la obtención de cada una de las soluciones, el procedimiento desarrollado se repite un número de iteraciones, obteniéndose soluciones distintas, escogiéndose al final la mejor de las alcanzadas. Nótese que cada iteración es completamente independiente de las demás.

IV. EXPERIMENTACIÓN PREVIA

El algoritmo presentado en este trabajo tiene cinco parámetros, los cuales se han ajustado mediante unas pruebas preliminares que permitan el ajuste del algoritmo. A continuación, se describe el

criterio que se ha seguido para ajustar cada uno de ellos:

- **Parámetro α de la LRC:** fue fijado a la semisuma del mejor y peor valor de los elementos seleccionables.
- **Criterio de ordenación:** De los cuatro criterios de ordenación de las piezas presentados anteriormente (L = lado mayor, P = Perímetro, A = Área y R = Ratio), desechamos el del *ratio*, debido a que presentaba resultados no competitivos con los existentes en la literatura.
- **Elementos extraídos para la búsqueda local:** después de varias experimentaciones se ajustó al 30% de la altura, teniendo en cuenta los resultados obtenidos y los tiempos empleados.
- **Número de iteraciones de mejora**: para cada solución constructiva. Se fijó en función del tamaño del problema, teniendo en cuenta que los tiempos de cómputo no fuesen excesivos.
- **Número de re-arranques:** ejecuciones independientes del algoritmo. Al igual que el parámetro anterior, se fijó en función del tamaño de la instancia y el tiempo de ejecución.

V. Experiencia computacional y resultados

El algoritmo presentado en este trabajo ha sido implementado en lenguaje Pascal por el mismo programador, no ejerciéndose opción alguna para optimizar el proceso de compilación del código. El compilador utilizado es Borland Delphi 5.0 y todas las pruebas se ejecutaron en un ordenador Pentium IV a 2.5 GHz con 1 MB de RAM.

Para poder comprobar el grado de bondad de nuestro procedimiento, tomamos como problemas *test* los que aparecen en el trabajo de Hopper y Turton [2], disponibles en:

http://people.brunel.ac.uk/~mastjjb/jeb/orlib/stripinfo.html

Se trata de una batería de 21 problemas agrupados en 7 categorías (C1-C7) de 3 problemas cada una, abarcando el rango de 16 a 197 objetos. Cada categoría corresponde a un número fijo de objetos a colocar sobre una banda también de anchura fija y donde la altura óptima a alcanzar es idéntica para los tres problemas (h_{opt}) de cada categoría.

Cada uno de los problemas estudiados fue resuelto un número distinto de iteraciones, dependiendo de la categoría y del número de objetos a colocar. Los resultados obtenidos en nuestro trabajo fueron comparados con los obtenidos en otros trabajos realizados sobre la misma base de problemas. Aunque el número de trabajos que recurren a esta base de datos es grande, sólo algunos de ellos tratan la problemática de permitir que las piezas giren 90º. De entre estos trabajos destacan los de Hopper y Turton (creadores de la base de datos), que obtenían las soluciones a partir de la aplicación de metodologías basadas en distintas metaheurísticas, hibridadas con las heurísticas de colocación BL y BLF [2]. Los mejores resultados fueron obtenidos para el algoritmo basado en el Recocido Simulado, aunque el tiempo empleado para la obtención de las soluciones era superior en todos los casos a los empleados por las otras metaheurísticas. El otro trabajo destacable es el de Beltrán y colaboradores [6], que hibridaban un metaheurística GRASP para la fase constructiva con una metaheuristica de vecindad variable (VNS) para la fase de mejora. En este caso, los tiempos empleados eran bastante inferiores a los del Recocido Simulado, siendo los resultados similares en calidad.

La Tabla I presenta los resultados obtenidos en este trabajo junto con los parámetros particulares para cada instancia analizada. En la primera columna se indica el nombre de la clase y la instancia (respectivamente, primera y segunda subcolumna), en la segunda columna, el tiempo aproximado empleado por el algoritmo, en la tercera se indica la cantidad de objetos que tenía cada instancia, en la cuarta la anchura de la cinta (w) y en la quinta, la altura óptima para cada instancia (h_{opt}). En las columnas sexta, séptima y octava se muestran respectivamente el número de construcciones que hace el algoritmo, el número de veces que se le aplica el procedimiento de mejora a cada solución y el porcentaje de piezas que saca el procedimiento. Finalmente, en las columnas nueve, diez y once, se muestra los resultados obtenidos por el algoritmo presentado en este trabajo para tres criterios de ordenación distintos. A saber: lado (L), perímetro (P) y área (A).

En la Tabla II se presentan los resultados obtenidos por el algoritmo multi-arranque descrito en este trabajo comparado con los procedimientos más representativos encontrados en la literatura para estos mismos problemas. En concreto, la comparativa es con el algoritmo GRASP + VNS descrito en [6] (tercera columna) y el algoritmo de recocido simulado (SA) hibridado con heurísticos descrito en [2] (cuarta columna). Para cada categoría de problemas, los resultados mostrados representan los valores medios de los mejores valores obtenidos para los tres problemas que conforman cada categoría. Finalmente, la Tabla III muestra las desviaciones relativas de todos los resultados respecto al valor óptimo de cada categoría del problema. La última fila de la tabla muestra el valor medio de desviación para todas las categorías.

TABLA I
RESULTADOS OBTENIDOS EN ESTE TRABAJO

PROBLEMA		T (S)	OBJETOS	w	h_{opt}	IT. CONSTRUCTIVO	IT. MEJORA	NIVEL (%)	L	P	A
C1	1	Inst.	16	20	20	60	20		21	21	21
	2		17						21	21	21
	3		16						21	21	21
C2	1	2	25	40	15	200	50		16	16	16
	2		25						16	16	16
	3		25						16	16	16
C3	1	4	28	60	30	200	50		32	32	32
	2		29						31	31	31
	3		28						32	32	31
C4	1	120	49	60	60	1000	200	30	62	62	62
	2		49						61	61	62
	3		49						61	61	61
C5	1	250	73	60	90	1000	200		92	92	92
	2		73						91	92	92
	3		73						92	92	91
C6	1	600	97	80	120	1000	200		123	123	123
	2		97						122	122	121
	3		97						122	122	122
C7	1	5400	196	160	240	1000	200		247	246	247
	2		197						244	244	244
	3		196						245	245	246

Los resultados mostrados en la Tabla II y la Tabla III muestran que, para instancias pequeñas, el algoritmo SA presenta un comportamiento ligeramente superior a los otros dos algoritmos. En cambio, cuando el tamaño de las instancias se incrementa, este algoritmo empieza a obtener peores resultados.

Por el contrario el algoritmo GRASP + VNS, tiene justo el comportamiento contrario; es decir, para instancias pequeñas el resultado es el peor de los tres algoritmos comparados; en cambio, para instancias grandes es el que muestra un mejor comportamiento.

El algoritmo multi-arranque descrito en este trabajo tiene un comportamiento más estable, de tal forma que tanto para instancias pequeñas como para instancias grandes, el comportamiento es bueno. Este hecho lo confirma el dato de que el error medio para todos los experimentos en menor en el algoritmo propuesto. Con respecto al criterio de ordenación de las piezas se puede concluir que para estas instancias, el criterio que ha mostrado el mejor comportamiento es el del ordenar por el área de las piezas.

Los resultados obtenidos en esta experimentación confirman que el procedimiento desarrollado en este trabajo produce soluciones comparables con las obtenidas en otros trabajos anteriores, para todas las categorías, indistintamente del criterio de ordenación elegido. Además tiene la ventaja de que los tiempos de ejecución son relativamente bajos (comparados con el procedimiento del recocido simulado) y similares a los empleados en los otros algoritmos.

TABLA II
RESULTADOS OBTENIDOS EN ÉSTE Y OTROS TRABAJOS PARA LOS PROBLEMAS TESTS DE HOPPER.

PR.	h_{opt}	GRAPS + VNS	SA	MULTI - START		
				L	P	A
C1	20	21.3	20.8	21.0	21.0	21.0
C2	15	16.0	15.9	16.0	16.0	16.0
C3	30	32.3	31.5	31.7	31.7	31.3
C4	60	62.0	61.8	61.3	61.3	61.7
C5	90	92.3	92.7	91.7	92.0	91.7
C6	120	122.7	123.6	122.3	122.3	122.0
C7	240	244.3	249.6	245.3	245.0	245.3

TABLA III
DESVIACIÓN (%) DE LOS RESULTADOS PRESENTADOS EN LA TABLA I RESPECTO A LOS VALORES ÓPTIMOS RESPECTIVOS.

PR.	h_{opt}	GRAPS + VNS	SA	MULTI - START		
				L	P	A
C1	20	6.7	4	5.0	5.0	5.0
C2	15	6.7	6	6.7	6.7	6.7
C3	30	7.8	5	5.6	5.6	4.4
C4	60	3.3	3	2.2	2.2	2.8
C5	90	2.6	3	1.9	2.2	1.9
C6	120	2.2	3	1.9	1.9	1.7
C7	240	1.8	4	2.2	2.1	2.4
DESVIACIÓN MEDIA TOTAL		4.4	4	**3.8**	**3.7**	**3.6**

VI. CONCLUSIONES Y FUTUROS TRABAJOS

En el presente trabajo se ha desarrollado un procedimiento basado en una metaheuristica multi-arranque para aplicarlo a la resolución de problemas del tipo *Strip Packing Problem* en dos dimensiones.

En concreto, se ha diseñado un algoritmo constructivo basado en cuatro métodos de ordenación y un procedimiento de mejora. El algoritmo desarrollado se ha aplicado sobre un conjunto de problemas *test*, y se ha comparado con los procedimientos que se han aplicado previamente a este problema, produciendo resultados comparables a los obtenidos en trabajos anteriores. Como trabajo futuro, se propone la aplicación de este algoritmo a otras bases de problemas, incluyendo instancias con mayor número de objetos. Por último, queda por analizar la evolución del algoritmo presentado para otros tipos de Listas Restringidas de Candidatos, diferentes a la aquí presentada.

AGRADECIMIENTOS

Los autores quieren agradecer la ayuda prestada en este trabaja a Rayco Jorge Cabrera. Este trabajo ha sido financiado parcialmente por el Ministerio de Educación y Ciencia (proyecto TIN2005-08943-C02-02).

REFERENCIAS

[1] Parreño, Francisco. Algoritmos heurísticos y exactos para problemas de corte no guillotina en dos dimensiones. (Tesis doctoral, Universidad de Valencia, 2004).

[2] Hopper, E. and Turton, B. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. European Journal of Operational Research, 128 (2001) 34.

[3] Hopper, E. and Turton, B. Application of genetic algorithm to packing problems – a review. In: Chawdry, P.K., Roy, R., Kant, R.K. (Eds.). Proceedings of the Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing, (Springer, 1997), 279.

[4] Iori, M., Martello, S., and Monaci, M. Metaheuristic algorithms for the strip packing problem, in: P.M. Paradolos and V. Korotkith (Eds.) Optimization and Industry: New Frontiers (Kluwer, 2003), 159.

[5] Bortfeldt, A. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. European Journal of Operational Research, 172 (2006) 814.

[6] Beltrán, J.C., Calderón, J.E., Cabrera, R.J. and Moreno, J.M. Procedimientos constructivos adaptativos (GRASP) para el problema del empaquetado bidimensional, Revista Iberoamericana de Inteligencia Artificial, 15 (2002), 26.

[7] Alvarez-Valdes, R., Parreño, F. and Tamarit, J.M. Reactive GRASP for the Strip Packing Problem, pendiente de publicación .

[8] Marti, Rafael. Multi-Start Methods, in: F. Glover and G. Kochenberger, Handbook on MetaHeuristics (Kluwer, 2003), 355.

[9] Resende, M.G.C. and Ribeiro, C.C. Greedy Randomized Adaptative Search Procedures. Handbook of Metaheuristics (ed. F. glover and G. Kochenberger) Kluwer Academic Publishers (2003) 219.

[10] Baker, B.S., Coffman Jr., E.G. and Rivest, R.L. Orthogonal packing in two dimensions. SIAM Journal of Computing, 9 (1980) 846.

[11] Chazalle, B. The bottom-left bin-packing heuristic algorithms: An efficient implementation. IEEE Transactions on Computers c32/8 (1983), 697.