Towards a Characterization of Difficult Instances of the Bin Packing Problem

A. Mexicano, J. Pérez, *Senior Member*, IEEE, D. Romero and L. Cruz

***Abstract*— We address the multidimensional characterization of difficult instances of the Bin Packing Problem, well known in the combinatorial optimization realm. In search for efficient procedures to solve hard combinatorial optimization problems previous investigations have attempted to identify the instances' characteristics with the greatest impact in their difficulty. In the same vein and focusing on the Bin Packing Problem, we used clustering techniques to determine not only one but a set of characteristics that altogether could be considered as the main source of the instances' difficulty. To this aim we selected 1,615 available instances of the Bin Packing Problem, and then we solved each instance with six well-known heuristic algorithms. From our experiment we identified relevant characteristics that correspond to 22 instances whose optimal solution was not obtained by any of the six heuristics. Finally, to validate our findings an adhoc heuristic based on these characteristics was developed. Our heuristic found two optimal solutions not previously reported, showing thus that this characterization can help to find improved solution algorithms.**

***Keywords*— bin packing, combinatorial optimization, heuristics, NP-hard, clustering.**

I. Introducción

En el ámbito de la programación matemática, una cuestión importante es el encontrar procedimientos eficientes para la resolución de problemas de optimización combinatoria clasificados como NP-duros. Estos problemas se consideran de gran dificultad ya que a la fecha no se conoce un algoritmo que pueda resolver alguno de ellos en tiempo polinomial. Razón por la cual, se han desarrollado heurísticas, es decir, procedimientos que, si bien no garantizan obtener un óptimo global, se espera que produzcan buenas soluciones en un tiempo razonable. El problema de Bin Packing (BPP) es un problema de optimización combinatoria NP-duro muy importante [1, 2], con aplicaciones prácticas como: la prepaginación y el formateo de la tabla de asignación de archivos [3], la programación de horarios y la multi-programación [4], la asignación de espacios publicitarios en televisión comercial [4, 5], la programación de multiprocesos [6, 7], etc. Este problema fue originalmente propuesto por Elion y Christofides en 1971 como "*The loading problem*" [8], sin embargo, fue acuñado como BPP por Garey y Johnson [9]. El BPP se describe de la siguiente forma. Para un conjunto dado de n objetos con tamaño s_i, para $i = 1,..., n$, y un número ilimitado de contenedores, cada uno con una capacidad entera $c>0$, encontrar el número mínimo de contenedores en el que todos los objetos se puedan acomodar [10]. Sin pérdida de generalidad, suponemos de aquí en adelante que una instancia está conformada por un conjunto de objetos $\{s_1, \ldots, s_n\}$, donde s_i es un entero positivo para $i = 1,..., n$ y $s_i \leq c$.

Varios algoritmos exactos han sido propuestos para resolver el BPP, incluyendo enfoques híbridos que incluyen métodos exactos como el branch and bound utilizando por Martello y Toth [11, 12], Sholl et al. [13], y Carvalho [14]. Por otra parte durante la última década, una gran cantidad de heurísticas para resolver este problema han sido desarrolladas, dichas heurísticas se puede encontrar en la literatura especializada [6, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24].

Motivados por la hipótesis de que "el desarrollo de algoritmos de solución eficientes para resolver el BPP puede ser guiado por una identificación a priori de las características que causan mayor impacto en la dificultad de las instancias", en esta investigación se experimentó con 1,615 instancias del BPP reportadas en la literatura. Todas las instancias fueron resueltas con seis algoritmos heurísticos bien conocidos, y después se utilizaron técnicas de Minería de Datos [25], la cual es una disciplina cuyo objetivo es identificar patrones que se encuentran de forma no explícita en los datos

El resto del artículo está organizado de la siguiente manera: en la Sección 2 se discuten algunos trabajos relacionados, la Sección 3 describe el método para determinar las principales características de las instancias difíciles de resolver del BPP y los resultados obtenidos. La Sección 4 contiene un ejemplo de cómo la caracterización de instancias puede, ser utilizada para mejorar los algoritmos de solución actuales del BPP. Finalmente, en la Sección 5 se presentan las conclusiones y algunos trabajos futuros.

II. Trabajos relacionados

En esta sección se discuten algunas investigaciones previas relacionadas con el tema de este trabajo, que ponen de relieve la importancia de conocer las características de los casos de un problema de optimización combinatoria, en general, como un factor en la eficiencia de los algoritmos de solución.

A. Caracterización de instancias de problemas NP-duros

De acuerdo con el teorema de optimización "*no free lunch*" [26] se sabe que, si un algoritmo es eficiente, en promedio, para una clase de problemas, entonces su desempeño debe empeorar con los problemas restantes, es decir, su alto rendimiento en una clase de problemas es, en general, pagado por el bajo rendimiento en otra clase.

A. Mexicano, Instituto Tecnológico de Cd. Victoria (ITCV), Cd. Victoria, México, mexicanoa@gmail.com

J. Pérez, Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), Cuernavaca, México, jpo_cenidet@yahoo.com.mx

D. Romero, Instituto de Matemáticas de la Universidad Autónoma de México (IMAT-UNAM), Cuernavaca, México, davidr@matcuer.unam.mx

L. Cruz, Instituto Tecnológico de Cd. Madero, Cd. Madero, México, lcruzr@prodigy.net.mx

Investigaciones como las desarrolladas en [27, 28] sugieren que el conocimiento de las características (métricas) de las instancias es esencial para desarrollar algoritmos de solución eficientes.

En los párrafos siguientes se discuten los artículos más destacados en esta área.

Tsang, Barrett y Kwan [27] llevaron a cabo una investigación para el problema de satisfacción de restricciones binario. Este trabajo fue desarrollado para mostrar que, en lugar de tratar de encontrar "el mejor" método general de solución, la investigación debe tratar de identificar el dominio de aplicación de cada método. Varias instancias fueron generadas así como su solución con diferentes algoritmos. Como resultado de la experimentación, para cada algoritmo se obtuvo una región de dominio donde se produjeron las mejores soluciones.

Merz y Freisleben [28] estudiaron el rendimiento de un algoritmo memético al ser aplicado sobre un conjunto de instancias del problema de bipartición de grafos. Varias métricas fueron utilizadas para caracterizar las instancias, entre ellas la rugosidad y la dependencia gráfica. Se encontró que el algoritmo memético tuvo un buen rendimiento en un subconjunto de casos con características claramente definidas, llegando a la conclusión de que sería de interés proponer características adicionales con el objetivo de desarrollar métodos heurísticos altamente eficientes.

En el caso del problema MAX-SAT, Hoos, Smyth y Stützle [29] analizaron cómo el rendimiento de los algoritmos estocásticos de búsqueda local (SLS) depende de las características del espacio de búsqueda subyacente. Ellos encontraron que tanto la longitud de autocorrelación de los saltos aleatorios (ACL) y la correlación de la distancia del desempeño (FDC), reflejan factores de dificultad en las instancias. Varios grados de dificultad fueron identificados para las instancias de prueba, algunos con la métrica ACL cuando la distribución de los pesos era variable, otros con la métrica FDC cuando la distribución del peso se mantuvo constante. Se llegó a la conclusión de que los conocimientos adquiridos a partir de las características del espacio de búsqueda de los algoritmos, su rendimiento y las características de las instancias, proporcionan una clave para mejorar la eficiencia de los algoritmos.

Los trabajos mencionados anteriormente sugieren dos factores importantes en la resolución de las instancias de un problema: uno es la clasificación de las instancias de acuerdo con algunas de las características, y la otra es la selección del mejor algoritmo que resuelve cada tipo de instancia. Por otra parte, a partir de una caracterización adecuada también existe la posibilidad de mejorar los algoritmos conocidos o la creación de otros nuevos.

B. Caracterización de instancias del BPP

En el caso del BPP, no hay consenso sobre qué tipo de instancias son las más difíciles de resolver por los algoritmos de solución actuales. Varias investigaciones [12, 13, 30, 31, 32] han sido desarrolladas en esta área, donde diversos conjuntos de instancias han sido propuestos y considerados como difíciles por diferentes autores; la Sección III.A describe estos casos en detalle.

La investigación más notable acerca de la dificultad de las instancias de BPP fue desarrollado por Schwerin y Wäscher [32], donde se identificaron tres indicadores relevantes para el BPP. En este trabajo se aplicó el algoritmo de primer ajuste decreciente (FFD) [31] en un conjunto de 40,000 instancias generadas aleatoriamente divididas en 400 grupos que contienen 100 casos cada uno. Como resultado se encontró que los tamaños de los objetos pueden afectar el rendimiento de los algoritmos y cada grupo se clasificó de acuerdo al porcentaje p de instancias que fueron resueltas de manera óptima por FFD como: "fácil" si $100 \geq p \geq 80$, "duro" si el $80 \geq p \geq 20$, y "extremadamente difícil" si $20 \geq p \geq 0$.

Otra investigación importante en el BPP se llevó a cabo por Cruz [33], donde siete algoritmos conocidos, un conjunto de instancias generadas aleatoriamente y un conjunto de cinco métricas fueron consideradas. En este trabajo las instancias se agruparon de acuerdo con el algoritmo que mejor los resolvió, por lo que para cualquier nueva instancia dada fue posible asignarla a alguno de los grupos formados de acuerdo con una medida de similitud; dicha medida es directamente dependiente con las métricas consideradas. Con ello fue posible identificar y aplicar el algoritmo que mejor resolvía cada nueva instancia. Los resultados experimentales muestran que para la mayoría de las instancias, el procedimiento de selección del mejor algoritmo para esa instancia fue el correcto.

La investigación más reciente es la presentada por Quiroz [34] en ésta, inicialmente se presenta un análisis exploratorio sobre el desempeño del algoritmo y las características de las instancias, posteriormente mediante relaciones causales se extrae el conocimiento necesario que da la pauta para hacer la mejora del algoritmo. En este trabajo se generaron 5 métricas para la caracterización de instancias de Bin Packing.

La Tabla I muestra las características de algunos trabajos relacionados, en comparación con los de esta investigación, donde el significado de las columnas es: 1. Propuesta de métricas para la caracterización de las instancias, 2. Consideración de más de un conjunto de instancias, 3. Uso de análisis multivariado, 4. Uso de métricas recopiladas de la literatura, 5. Uso de técnicas de minería de datos, 6. Aplicación al problema de Bin Packing de una dimensión.

TABLA I. TABLA COMPARATIVA DE TRABAJOS RELACIONADOS.

Autor	1	2	3	4	5	6
Tsang et al. [27]				✓		
Merz and Freisleben [28]	✓		✓	✓		
Hoos et al. [29]		✓		✓		
Schwerin and Wäscher [32]	✓					✓
Cruz [33]	✓		✓			✓
Quiroz[34]	✓	✓	✓	✓		✓
Este trabajo	✓	✓	✓	✓	✓	✓

III. CARACTERIZACIÓN DE LAS INSTANCIAS DIFÍCILES DEL BPP

En esta sección se describe el método para determinar las características principales de los casos difíciles de resolver del BPP.

Selección de conjuntos de instancias

Se seleccionó un conjunto I de 1,615 instancias del BPP que son consideradas por sus autores como difíciles de resolver y que pertenecen a repositorios que son ampliamente utilizados para evaluar los algoritmos de solución del BPP. Las instancias se encuentran entre las más citadas por los expertos en los últimos años y su solución óptima es conocida [6, 31, 35] dado que en algunos casos la solución ha sido alcanzada aplicando métodos exactos, los cuales son computacionalmente muy costosos.

En lo que resta del documento n será el número de elementos en una instancia, c la capacidad del contenedor, y s_i es el tamaño del elemento i, para $i = 1,..., n$. Además, se supone que todas las selecciones de los objetos que se mencionan como realizadas aleatoriamente se hacen sobre una distribución uniforme.

Falkenauer propuso en 1996 [30] dos conjuntos de instancias llamadas "uniform" y "triplets", respectivamente; estas instancias han sido citadas en [6, 13, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44], y se puede descargar de [45]. El conjunto "uniform" tiene 80 instancias (20 para cada valor de n), creadas aleatoriamente de la misma manera que lo hicieron Martello y Toth en 1990 [12]. Este conjunto tiene elementos de tamaños distribuidos de manera uniforme con los siguientes parámetros:

$n = 120, 250, 500, 1,000$

$c = 150$

$s_i \in [20, 100]$ para $i = 1,..., n$

El conjunto "triplets" contiene 80 instancias, con los siguientes parámetros:

$n = 60, 120, 249, 501$

$c = 1,000$

s_i se calcula de la siguiente manera: un objeto se genera primero con un tamaño distribuido aleatoriamente en el intervalo [380,490] que deja un espacio r entre 510 y 620 de c. El tamaño del segundo elemento se extrae al azar en el intervalo [250, r/2]. El tercer elemento completa la capacidad del contendor.

Lo que hace que estas instancias sean difíciles es el hecho de colocar dos elementos grandes o tres pequeños en un contenedor, lo cual conduce inevitablemente a una pérdida de espacio, e implica una solución subóptima. Hay cuatro clases de instancias con diferentes valores para n, y 20 casos para cada una.

Wäscher y Gau [46] propusieron en 1996 un conjunto de instancias llamadas gau_1, citadas en [6, 38, 34, 41] que se pueden descargar de [47]. Los algoritmos propuestos en [21, 22, 39, 41] son los que han encontrado las mejores soluciones para este conjunto de instancias, en particular, 16 soluciones óptimas para este conjunto se describen en [41]. Este conjunto tiene 17 instancias, todos con un número distinto de elementos.

$n = 57, 60, 86, 91, 92, 96, 111, 114, 119, 141, 142, 144, 153, 163, 164, 228, 239.$

$c = 1,000$

$s_i \in [2, 7\,332]$ para $i = 1,..., n$.

Scholl et. al [13] en 1997 propusieron tres conjuntos de instancias: Set_1, Set_2 y Set_3, que se han citado en [6, 36, 20, 38, 39, 41, 44, 48, 49, 50], y se pueden descargar de [51]. El conjunto Set_1 tiene 720 instancias creadas como se muestra en [12]. Durante la creación se consideraron los siguientes parámetros:

$n = 50, 100, 200, 500$

$c = 100, 120, 150$

$s_i \in [1, 100], [20, 100], [30, 100]$ para $i = 1,..., n$.

Los tamaños de los elementos fueron seleccionados al azar de los intervalos dados. Veinte casos se generaron para cada combinación de n, c, y s_i.

El conjunto Set_2 tiene 480 casos, generados con un promedio de tres, cinco, siete o nueve elementos por contenedor, durante su creación se consideraron los siguientes parámetros:

$n = 50, 100, 200, 500$

$c = 1,000$

$w = c/3, c/5, c/7, c/9$ (w es el promedio deseado del tamaño de los elementos).

$\delta = 0.2, 0.5, 0.9$

s_i se elige aleatoriamente a partir del intervalo $[w - \delta w, w + \delta w]$, para $i = 1,..., n$.

Por ejemplo, si $w = c/5$ y $\delta = 0.2$, los tamaños de los elementos pertenecen al intervalo [160, 240]. Diez instancias se generaron para cada combinación posible de n, w y δ.

El conjunto Set_3 tiene 10 instancias considerados como difíciles de resolver debido a que, el número de valores posibles para los diferentes tamaños es demasiado grande. Los tamaños fueron seleccionados al azar a partir del rango [20,000, 35,000]. Esto asegura que los tamaños están ampliamente dispersos.

$n = 200$

$c = 100,000$

$s_i \in [20,000, 35,000]$ para $i = 1,..., n$.

En 1997 Schwerin y Wäscher [32] propusieron dos conjuntos de instancias llamadas was_1 y was_2, citadas en [6, 34, 36, 38, 39, 41], las cuales se pueden descargar de [47].

was_1 es un conjunto de 100 instancias encontradas como difíciles para el algoritmo FFD. Las instancias tienen los siguientes parámetros:

$n = 100$

$c = 1\,000$

$s_i \in [100, 150]$ para $i = 1,..., n$.

was_2 es un conjunto de 100 instancias cuyos parámetros son:

$n = 120$

$c = 1\,000$

$s_i \in [150, 200]$ para $i = 1,..., n$.

En 2002 Schoenfield [31] propuso un conjunto de 28 instancias difíciles llamadas Hard28; estas instancias se han citado en [35, 34, 39, 41], y puede ser descargadas en [52]. Actualmente, este conjunto es considerado como muy difícil en la literatura especializada. La heurística HGGA-BP [39] obtiene 11 soluciones óptimas, a la fecha es la heurística que obtiene mejores resultados. En este conjunto ocho casos

tienen un valor de n = 160, diez tienen n = 180, y diez tienen n = 200. Además,

c = 1,000

$s_i \in [1, 800]$ para i = 1,..., n.

Tomando en cuenta, que en la actualidad no existe un consenso en cuanto al grado de dificultad de las instancias, se optó por realizar una caracterización de las instancias más difíciles de resolver. Por ejemplo, Korf [53] y Schwerin y Wäscher [32] cuestionaron la dureza de las instancias creadas por Martello y Toth [12]. En cuanto a las instancias difíciles de Falkenauer [30], Gent [42] señaló que con el uso de métodos heurísticos sencillos es posible alcanzar valores satisfactorios, cuestionando así la dificultad del conjunto "uniforme". En lo que respecta a los casos de Scholl et. al [13], Kasap [20] mencionó que cada conjunto tiene un grado de dificultad (fácil, medio y difícil Set_1, Set_2 y Set_3, respectivamente). Por último, las instancias en el conjunto de gau_1 [46] fueron considerados en [20] como muy fácil debido a que una buena solución se puede lograr en la primera iteración del algoritmo FFD.

A. Cálculo de soluciones para instancias de prueba

Cada instancia en I fue resuelta con seis algoritmos bien conocidos: Primer Ajuste Decreciente (FFD) [5], Mejor Ajuste Decreciente (BFD) [5], Peor Ajuste Decreciente (WFD) [5], Mejor Triple Ajuste Decreciente (B3FD) [6], Algoritmo Híbrido genético de agrupamiento para Bin Packing (HGGA-BP) [39], y Algoritmo Híbrido Mejorado para Bin Packing (HI_BP) [6]. De este modo, se obtuvo un total de 9,690 soluciones, seis por cada instancia en I. El algoritmo HI_BP fue el mejor en cuanto al número de soluciones óptimas encontradas, ya que sólo 22 instancias no fueron resueltas por este algoritmo. Después de calcular las soluciones, se identificaron 22 instancias en las cuales ninguno de los algoritmos obtuvo una solución óptima. A este conjunto de instancias lo denominamos H.

La Tabla II muestra el total de instancias resueltas por cada uno de los algoritmos utilizados durante la experimentación.

TABLA II. NÚMERO DE SOLUCIONES NO RESUELTAS ÓPTIMAMENTE POR LOS ALGORITMOS DE PRUEBA.

Algoritmo	Número de instancias no resueltas óptimamente
FFD	821
BFD	820
WFD	901
BFD	1,188
HGGA-BP	135
HI_BP	28

B. Métricas

Inicialmente se seleccionó un conjunto de 27 métricas procedentes de [33, 34, 36] para caracterizar las instancias en I. Después de aplicar análisis de correlación este conjunto se redujo a cinco métricas. La primera métrica seleccionada fue n, es decir, el número de elementos de la instancia; las cuatro métricas restantes se describen a continuación, donde s_i es el tamaño del objeto i, para i = 1,..., n, c es la capacidad del contenedor, $S = \sum_{i=1}^{n} s_i$ y φ es la frecuencia más alta en $\{s_1, \ldots, s_n\}$ de cualquier tamaño de objeto.

$$avg = S / n \tag{1}$$

$$m = \frac{\min_{i=1,\ldots,n}\{s_i\}}{c} \tag{2}$$

$$r = \frac{\max_{i=1,\ldots,n}\{s_i\}}{c} - m \tag{3}$$

$$\vartheta = \varphi / c \tag{4}$$

C. Modelado

Con la finalidad de conseguir un agrupamiento con las instancias más difíciles de resolver utilizando las 5 métricas relevantes, se aplicó un proceso iterativo de la siguiente manera: dado un entero $k \geq 2$, sea C(k) el k-ésimo agrupamiento en un espacio 5-dimensional, obtenido con el algoritmo de k-means. Se encontró que al utilizar C(45) fue la mejor opción en el sentido de que el conjunto H que contiene las 22 instancias difíciles de resolver, definidas en la Sección III.B se separa convenientemente del conjunto I en cuatro grupos A, B, C, y D. Finalmente, los parámetros utilizados para el modelado de los grupos con las instancias más difíciles de resolver fueron los siguientes: k = 45 y las métricas avg, m, ϑ, r y n.

Los grupos que contienen las instancias difíciles de resolver se muestran a continuación.

A {TEST0082} (una instancia).

B {TEST0014, TEST0030, TEST0058, TEST0005, TEST0022, TEST0065, TEST0084} (siete instancias).

C {BPP60, BPP832, BPP13, BPP709, BPP785, BPP144, BPP561, BPP781, BPP900, BPP195, BPP14, BPP119} (12 instancias).

D {BPP40, BPP645, BPP766, BPP181, BPP485, BPP178, BPP419, BPP360, BPP47, BPP640, BPP814, BPP742, BPP531, BPP359, BPP716, BPP175} (16 instancias).

Estos grupos incluyen algunas instancias (subrayadas) que han sido resueltas de forma óptima por las heurísticas HGGA-BP y HI_BP (ver Sección III.B) utilizadas en este trabajo.

La Tabla III muestra los valores del cálculo de las métricas sobre las instancias del grupo C.

TABLA III. VALORES DE LAS MÉTRICAS DEL GRUPO C

Instancia	avg	m	ϑ	r	n
BPP14	0.381	0.011	0.02	0.685	160
BPP832	0.375	0.006	0.02	0.690	160
BPP60	0.390	0.053	0.01	0.643	160
BPP13	0.372	0.001	0.02	0.697	180
BPP195	0.355	0.001	0.01	0.697	180

BPP709	0.372	0.009	0.02	0.688	180
BPP785	0.378	0.005	0.02	0.694	180
BPP119	0.380	0.002	0.02	0.695	200
BPP144	0.365	0.009	0.02	0.690	200
BPP561	0.360	0.001	0.02	0.699	200

La Tabla IV muestra los centroides obtenidos para los grupos A, B, C y D que contienen las instancias más difíciles de resolver.

TABLA IV. CENTROIDES DE LOS GRUPOS A, B, C Y D

Grupo	*avg*	*m*	ϑ	*r*	*n*
A	0.280	0.010	0.060	0.730	90
B	0.231	0.006	0.050	0.480	89
C	0.370	0.008	0.018	0.691	183
D	0.398	0.008	0.014	0.788	180

IV. GENERACIÓN DE SOLUCIONES ÓPTIMAS MEDIANTE LA INTERPRETACIÓN DE CARACTERÍSTICAS

En esta sección se muestra cómo los resultados obtenidos de la caracterización de instancias posibilitan el desarrollo de heurísticas destinadas a resolver casos difíciles. Para probar la utilidad del enfoque de solución se analizaron las características correspondientes al grupo C, lo cual permitió encontrar dos soluciones óptimas más que las previamente encontradas.

De acuerdo al análisis realizado sobre las características del grupo C, se encontró que en promedio: el rango del tamaño de los objetos es amplio (r = 0.691), existen pocas repeticiones del tamaño de los objetos (ϑ = 0.018) con un tamaño mínimo de objetos pequeño (m = 0.008), siendo el promedio casi del 40% de la capacidad del contenedor c (avg = 0.37). Estas características sugirieron que era posible reducir la dimensionalidad de una instancia buscando pares de objetos i, j con $s_i + s_j <= c$. De tal forma que se optó por:

- Hacer las combinaciones de dos objetos que llenen la capacidad del contenedor en su totalidad o en un porcentaje delimitado por la capacidad residual total de la instancia y extraer dichas combinaciones como parte de la solución (a este procedimiento lo denominamos de reducción R), y
- Aplicar un algoritmo W que resuelva la distribución de los elementos restantes en contendores.

El enfoque propuesto fue aplicado sobre los elementos del grupo C, utilizando como W un algoritmo metaheurístico genético inspirado en HGGA-BP [14], uno de los mejores algoritmos disponibles para el BPP. En las Secciones A y B se describe respectivamente, el enfoque de solución planteado y las soluciones óptimas encontradas producto del análisis de las características de los grupos.

A. Enfoque de solución planteado

El enfoque de solución (ver Algoritmo 1), está conformado de un procedimiento de reducción de la instancia denominado R (ver Algoritmo 2) y un algoritmo metaheurístico genético W, encargado de resolver la instancia reducida.

El Algoritmo 1 tiene como parámetros de entrada la instancia I y la capacidad del contenedor c; dichos elementos afectan de forma directa la complejidad ya que el número de objetos de la instancia, los pesos de los objetos y el tamaño de los contenedores determinan el comportamiento del algoritmo. Las variables *mejorSol*, S, I', S_1 y S_2 son inicializadas como conjuntos vacíos, ya que durante la ejecución del algoritmo, estas variables son las encargadas de almacenar las soluciones que se van generando (línea 1). La línea 2 muestra que la instancia es ordenada de forma decreciente, lo cual se logra con una complejidad O(n log n). En la línea 3 se muestra que el conjunto I'' es inicializado como I ya que este representa a la instancia actual en cada iteración y ésta es modificada después de aplicar el procedimiento de reducción R. En la línea 4 la variable *ResiduoTUtilizado* se inicializa en 0, esta variable tiene como función almacenar el valor de la capacidad residual que se va consumiendo de la instancia actual, después de hacer una reducción. La variable *BanderaRed* permite determinar cuándo ha sido consumida toda la capacidad residual teórica de la instancia (línea 5). La variable *umbral* permite administrar la capacidad residual permitida por contenedor en cada iteración (línea 6); la variable *umbral* es inicializada en 0 y es incrementada en una unidad en cada iteración, ver línea 18. La variable *nBin* almacena el número de contenedores en la solución parcial S_1 y es inicializada en O (ver línea 7). La variable n corresponde al número de objetos en la instancia (línea 8), L_1 corresponde al valor del límite inferior L_1 (línea 9) y L_2 corresponde al valor del límite inferior L_2 propuesto por Martello y Toth en [11] (línea 10), Ω corresponde al promedio de objetos por contenedor (línea 11), y ψ corresponde a la capacidad residual teórica total de la instancia (línea 12). Es importante mencionar que los cálculos L_1, L_2, Ω *y* ψ tienen una complejidad lineal O(n). Con la finalidad de evitar aplicar el procedimiento R a las instancias donde en promedio no se pueden asignar dos objetos por contenedor, se aplica la condición de la línea 13, lo cual permite que W se aplique de forma directa sobre la instancia (línea 32).

El proceso de reducción R se aplica sólo a instancias cuyo promedio de objetos por contenedor es menor o igual a 2 ($\Omega \leq 2$) y mientras la capacidad residual de la instancia no se rebase (línea 14).

En la línea 15 se observa que a la variable I' se le asigna el resultado de aplicar el procedimiento de reducción R, posteriormente la variable *ResiduoConsumido* se actualiza con el total de la capacidad residual utilizada en la asignación de objetos a los contenedores durante el procedimiento de reducción (línea 16), además la variable *ResiduoTUtilizado* se actualiza sumando a ésta la capacidad residual consumida en la reducción realizada (línea 17) y la variable *umbral* es aumentada en una unidad para realizar las siguientes reducciones con mayor capacidad residual (línea 18). La línea 19 muestra la condición que evalúa si la reducción realizada no rebasó el total de la capacidad residual de la instancia y en caso de no ser así, se actualiza la instancia I'' eliminando los objetos de I' (línea 20), y los objetos en I' son asignados a la solución parcial S_1 (línea 21). Para calcular la segunda solución parcial S_2 se aplica el algoritmo W sobre I'' (línea 22). La unión de las soluciones

S_1 y S_2 es asignada a S (línea 26). La línea 27 es una condición que permite almacenar la mejor solución que se obtiene mientras se ejecuta Algoritmo 1 el cual continúa iterando, si el número de contenedores en S no es igual al límite inferior L_2, lo que ocasiona reducir la instancia actual I'' (línea 15). Al terminar el algoritmo obtiene en la variable *mejorSol* la mejor solución encontrada y termina (ver parámetros de salida).

Algoritmo 1 Enfoque de solución
Parámetros de entrada: I, c
Parámetro de salida: *mejorSol*
1: $mejorSol \leftarrow \{\}$, $S \leftarrow \{\}$, $I' \leftarrow \{\}$, $S_1 \leftarrow \{\}$, $S_2 \leftarrow \{\}$
2: $I = \{s_1, s_2, \ldots, s_n\}; s_1 \geq s_2 \geq \ldots \geq s_n$
3: $I'' \leftarrow I$
4: $ResiduoTUtilizado \leftarrow 0$
5: $BanderaRed \leftarrow 0$
6: $umbral \leftarrow 0$
7: $nBin \leftarrow 0$
8: calcular $n \leftarrow |I|$
9: calcular $L_1 \leftarrow \sum_{i=1}^{n} s_i/c$
10: calcular L_2
11: calcular $\Omega \leftarrow \lfloor n/L_1 \rfloor$
12: calcular $\psi \leftarrow c * L_1 - \sum_{i=1}^{n} s_i$
13: **si** ($\Omega \leq 2$) **entonces**
14: **mientras** ($|mejorSol| \neq L_2 \wedge BanderaRed = 0$) **hacer**
15: $I' \leftarrow R(I'', umbral, c, nBin)$
16: $ResiduoConsumido \leftarrow (nBin * umbral)$
17: $ResiduoTUtilizado \leftarrow ResiduoTUtilizado + ResiduoConsumido$
18: $umbral \leftarrow umbral + 1$
19: **si** ($\psi - ResiduoTUtilizado \geq 0$) **entonces**
20: $I'' \leftarrow I'' \backslash I'$
21: $S_1 \leftarrow I'$
22: $S_2 \leftarrow W(I'')$
23: **si no**
24: $BanderaRed \leftarrow 1$
25: **fin si**
26: $S \leftarrow S_1 \cup S_2$
27: **si** ($|mejorSol| = 0 \vee |mejorSol| > |S|$) **entonces**
28: $mejorSol \leftarrow S$
29: **fin si**
30: **fin mientras**
31: **si no**
32: $mejorSol \leftarrow W(I)$
33: **fin si**
34: **fin algoritmo**

El procedimiento de reducción R cuya complejidad es lineal O(n) ya que solo se requiere recorrer una vez cada elemento de la instancia para determinar los elementos a eliminar. Se puede observar que R (Algoritmo 2) recibe como parámetros de entrada: la instancia I con los objetos ordenados de forma decreciente, el valor de la variable *umbral*, la capacidad del contenedor c y el número de contenedores a los que se les asignan objetos en la iteración actual *nBin*. Para encontrar los pares de objetos, se colocan dos apuntadores, uno apunta al objeto con el peso mayor de la instancia (*max*) y el otro apunta al objeto con el peso menor de la instancia (*min*) (líneas 1 y 2). Mientras existan objetos que no han sido evaluados en la instancia y los índices de los apuntadores sean diferentes, el proceso de combinar objetos se repite (línea 3). Si la suma de los pesos de los objetos del par actual es igual a $c - umbral$, el par de objetos es extraído de I y los objetos son asignados al contenedor *nBin* en la solución (líneas 4-7). En caso contrario, si la suma es menor que $c - umbral$, el apuntador *min* apunta al siguiente objeto menor más cercano (líneas 9 y 10). Para el caso del apuntador mayor (*max*), si la suma es mayor que $c - umbral$ el apuntador *max* cambia de posición y apunta el siguiente objeto menor (líneas 11-13). Al terminar el procedimiento de reducción, la variable *nBin* contiene el número de contenedores a los cuales se les asignaron objetos y regresa el subconjunto I' que corresponde a la solución parcial S_1 de la instancia I (ver parámetro de salida).

Algoritmo 2 Procedimiento de reducción R
Parámetros de entrada: $I=\{s_1, \ldots, s_n\}$, *umbral*, c, *nBin*
Parámetro de salida: I'
1: $max \leftarrow 1$
2: $min \leftarrow n$
3: **mientras** (($s_{max} <>$ NULL) $\wedge$ ($max <> min$)) **hacer**
4: **si** ($s_{max} + s_{min} = c - umbral$) **entonces**
5: $nBin \leftarrow nBin + 1$
6: $B_{nBin} \leftarrow \{s_{max}, s_{min}\}$
7: $I' \leftarrow I' \cup B_{nBin}$
8: **si no**
9: **si** ($s_{max} + s_{min} < c - umbral$) **entonces**
10: $min \leftarrow min - 1$
11: **si no**
12; **si** ($s_{max} + s_{min} > c - umbral$) **entonces**
13: $max \leftarrow max + 1$
14: **fin si**
15: **fin si**
16: **fin mientras**
17: **fin algoritmo**

Considerando que la complejidad más alta para realizar los diferentes cálculos en el Algoritmo 1 sin considerar W es O(n log n) y tomando en cuenta que W es un algoritmo metaheurístico genético de naturaleza aleatoria y cuya ejecución no sólo depende de la cantidad de elementos de entrada sino que también de la capacidad del contenedor y el tamaño de los elementos se puede afirmar que la complejidad del **Algoritmo 1** no es mayor a W dado que la reducción de la instancia reduce el espacio de búsqueda de soluciones óptimas y por tanto el Algoritmo 1 tiende a reducir el tiempo de ejecución y logra mejorar los resultados obtenidos.

B. Soluciones óptimas obtenidas

Los valores óptimos de los casos en el grupo C se han encontrado previamente mediante el uso de programación lineal [5], pero sólo dos de los valores, BPP14 y BPP119, se obtuvieron mediante el uso de metaheurísticas y a la fecha ninguna metaheurística ha reportado su valor óptimo. Al aplicar el **Algoritmo 1** se encontraron no sólo las soluciones óptimas de las instancias BPP14 y BPP119 previamente reportadas, sino también dos soluciones más, BPP561 y

BPP781, que no han sido resueltas por ningún enfoque metaheurístico conocido. El tiempo de cálculo requerido por **Algoritmo 1** para las instancias BPP561 y BPP781, fue de 2.18 y 4.82 segundos, respectivamente. Los experimentos se llevaron a cabo en un equipo con la siguiente configuración: Intel Corei7, procesador de 2 GHz, 4 GB de RAM, 32 bits, y el sistema operativo Windows 7.

A continuación se muestran las soluciones óptimas encontradas utilizando **Algoritmo 1**:

a) Solución de la instancia BPP561

La solución se obtuvo con 72 contenedores, el procedimiento de reducción *R*, redujo 28 objetos de la instancia en 14 contenedores: (697 303) (681 319) (669 331) (666 334) (656 344) (652 348) (647 353) (625 375) (588 412) (581 419) (552 448) (537 463) (537 463) (519 481)

La metaheurística *W* acomodó el resto de objetos en 58 contenedores: (63 429 508) (89 398 513) (672 186 142) (366 313 321) (494 421 85) (47 454 499) (211 168 621) (292 540 168) (450 472 78) (672 10 318) (231 553 216) (624 260 116) (427 449 124) (472 302 226) (304 176 520) (169 139 692) (521 253 226) (260 592 148) (225 580 195) (539 276 185) (133 179 688) (89 598 313) (106 266 628) (700 198 102) (130 451 419) (665 200 135) (654 258 88) (190 111 699) (259 241 500) (388 170 442) (456 138 406) (617 282 101) (614 185 201) (131 527 342) (542 413 45) (670 329) (631 229 140) (1 638 361) (22 364 614) (352 45 603) (3 666 331) (468 280 252) (135 584 281) (413 453 134) (439 347 214) (104 603 293) (607 304 89) (135 529 336) (693 114 193) (317 158 525) (626 351 23) (650 197 153) (661 338) (508 491) (684 314) (86 452 462) (17 595 388) (543 256 99 69 32)

b) Solución de la instancia BPP781

La solución se obtuvo con 71 contenedores, R redujo 30 objetos de la instancia utilizando 15 contendores: (700 300) (679 321) (667 333) (653 347) (652 348) (646 354) (610 390) (595 405) (568 432) (553 447) (543 457) (535 465) (531 469) (527 473) (508 492)

La metaheurística *W* acomodó el resto de los objetos en 56 contenedores: (157 698 145) (33 388 579) (55 677 268) (223 210 567) (83 510 407) (256 556 188) (611 86 303) (147 645 208) (519 62 419) (89 308 603) (24 403 73) (6 463 531) (270 100 630) (663 209 128) (339 69 592) (253 148 599) (330 498 172) (667 281 52) (8 691 301) (20 517 463) (627 167 206) (608 118 274) (96 654 250) (85 271 644) (554 9 437) (94 515 391) (624 357 19) (639 218 143) (598 258 144) (350 631 19) (687 193 120) (185 149 666) (125 256 619) (631 277 92) (69 303 628) (99 653 248) (218 196 586) (534 15 451) (160 186 654) (653 325 22) (400 185 415) (626 130 244) (390 69 541) (682 98 220) (509 206 285) (144 553 303) (3 479 518) (10 342 648) (221 229 550) (562 418 20) (520 3 477) (532 388 80) (535 363 63 39) (533 185 177 105) (557 400 42) (554 393 53)

V. Conclusiones

En este trabajo se muestra que es posible identificar características comunes en las instancias difíciles de resolver del BPP, que pueden guiar hacia mejoras de los algoritmos metaheurísticos actuales o contribuir a desarrollar nuevos.

Para validar nuestra propuesta, se realizaron experimentos utilizando seis algoritmos bien conocidos, 1,615 instancias y 27 métricas del BPP. Cada instancia fue evaluada con cada algoritmo con el propósito de identificar las de mayor dificultad. Para cada instancia se seleccionó el mejor resultado de entre los 66 calculados. Con el fin de caracterizar las instancias difíciles, las 27 métricas fueron aplicadas sobre las 1,615 instancias, posteriormente, se aplicaron algunas técnicas de reducción de características tales como la matriz de correlaciones para reducir la dimensionalidad de las métricas a cinco. Para la identificación de los patrones que describen las instancias más difíciles de resolver se aplicaron técnicas de agrupamiento, en particular el algoritmo k-means. Como resultado, se obtuvo un centroide asociado a cada grupo. Estos centroides expresan un patrón de valores promedio de las métricas seleccionadas. A partir de los experimentos realizados se encontró:

a) Un conjunto de cinco métricas que permiten formar cuatro grupos que contienen las 22 instancias que no fueron resueltas de manera óptima por ninguno de los seis algoritmos considerados.

b) Se confirmó la utilidad de la extracción de conocimiento con el fin de mejorar los algoritmos de solución actuales.

Consideramos que los valores de las métricas pueden ayudar en la obtención de conocimiento útil y adicional acerca de los conjuntos de instancias difíciles. Tal conocimiento puede ser incorporado en los algoritmos de solución para hacerlos más eficaces y eficientes.

Por último, creemos que la metodología seguida en este trabajo también podría ser utilizada para caracterizar las instancias difíciles de otros problemas NP-duros.

Referencias

[1] M. R. Garey, R. L. Graham, J. D. Ullman, "Worst-case analysis of memory allocation algorithms", En: Proceedings of the UK Workshop on Computational Intelligence. New York, USA, pp. 143–150, 1972.

[2] R. M. Karp, R. E. Miller, J. W. Thatcher, "Reducibility among combinatorial problems", JSymLog, vol. 40, pp. 618–619, 1975. 2

[3] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, R. Graham, "Worst-case performance bounds for simple one-dimensional packing algorithms", SIAM J on Computing, vol. 3, pp. 299–325, 1974.

[4] A. R. Brown, M. B. C. S. Jeffreys, H. Limited, "Optimum Packing and Depletion: The Computer in Space- and Resource-usage Problems", Macdonald and American Elsevier Inc., 1971.

[5] D. S. Johnson, Near-optimal bin packing algorithms, tesis de doctorado; Massachusetts Institute of Technology; Massachusetts; 1973.

[6] A. C. F. Alvim, F. Glover, C. Ribeiro, D. Aloise. "A hybrid improvement heuristic for the onedimensional bin packing problem", J of Heuristics vol.10, pp. 205–229, 2004.

[7] J. E. Coffman, M. Garey, D. Johnson, "An application of bin-packing to multiprocessor scheduling", SIAM J on Computing, vol.7, pp.1–17, 1987.

[8] S. Eilon, N. Christofides, "The loading problems", Management Science, Theory Series, vol. 17, pp. 259–268, 1971.

[9] M. R. Garey, D. Johnson, Computers and Intractability, a Guide to the Theory of NP-Completeness, New York: W. H. Freeman and Company, 1979.

[10] J. E. Coffman, C. Courboubetis, M. R. Garey, D. S. Johnson, P. W. Shor, R. R. Weber. "Perfect packing theorems and the average case behavior of optimal and online bin packing", SIAM Rev, vol. 44, pp.95–108, 2002.

[11] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations. Chichester: Wiley, 1990.
[12] S. Martello, P. Toth, "Lower bounds and reduction procedures for the bin packing problem", Discrete Appl Mathematics, vol. 28, pp. 59–70, 1990.
[13] A. Scholl, R. Klein, C. Jürgens, "Bison: a fast hybrid procedure for exactly solving the onedimensional bin packing problem", Computers & Operations Research, vol. 24(7), pp. 627–645, 1997.
[14] V. Carvalho. "Exact solution of bin-packing problems using column generation and branch-and-bound", Annals of Operations Res, 86, pp.629–659, 1999.
[15] A. K. Bhatia, S. K. Basu,. "Packing bins using multi-chromosomal genetic representation and better-fit heuristic", En R. P. Nikhil, K. Nikola, K. M. Rajani, P. Srimanta, K- P. Swapan, editores. Neural Information in Processing, Lecture Notes in Computer Science. Springer, pp. 181–186 2004.
[16] F. Ducatelle, J. Levine, "Ant colony optimization and local search for bin packing and cutting stock problems", J of the Operational Research Society, vol. 55, pp.705–716, 2004.
[17] K. Fleszar, K. S. Hindi, "New heuristics for one-dimensional bin-packing", Computers & Operations Research, vol. 29, pp. 821–839, 2002.
[18] M. P. Gómez, M. Randall, "A hybrid extremal optimisation approach for the bin packing problem", En: Goebel, R., Siekmann, J., Wahlster, W., editores. Lecture Notes in Artificial Intelligence. Springer, pp. 242–251, 2009.
[19] X. Jing, X. Zhou, Y. Xu, "A hybrid genetic algorithm for bin packing problem based on item sequencing", J of Information and Computing Science, vol. 1, pp.61–64, 2006.
[20] N. Kasap, A. Agarwal, "Augmented-neural networks approach for de Bin Packing problem", En: Proceedings of the 4th International Symposium on Intelligent Manufacturing Systems. Sakarya, Turkey, pp. 348–358, 2004.
[21] K. Loh, B. Golden, E. Wasil, "Solving the one-dimensional bin packing problem with a weight annealing heuristic", Computers & Operations Research, vol. 35(7), pp. 2283–2291, 2008.
[22] A. Singh, A. K. Gupta, "Two heuristics for the one-dimensional bin-packing problem", OR Spectrum, vol. 29(4), pp. 765–781, 2007.
[23] A. Stawowy, "Evolutionary based heuristic for bin packing problem," Computers & Industrial Engineering, vol. 55(2), pp. 465–474, 2008.
[24] O. Ülker, E. Korkmaz, E. Özcan, "A grouping genetic algorithm using linear linkage encoding for bin packing", En: Rudolph G et al., editores. Parallel Problem Solving from Nature. Springer, p. 1140–1149, 2008.
[25] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth. Cross industry standard process for datamining version 1.0 step by step datamining guide. ftp://ftp.software.ibm.com/software/analytics/spss/support/Modeler/Documentation/14/UserManual/CRISP-DM.pdf, 2000. Última visita: noviembre 2015.
[26] D. H. Wolpert, W. G. Macready, "No free lunch theorems for optimizations", IEEE Transactions on Evolutionary Computation, vol.1, pp. 67–82, 1996.
[27] E. P. K. Tsang, J. E. Borrett, A. C. M. Kwan, "An attempt to map the performance of a range of algorithm and heuristic combinations", En: Proceedings of the Artificial Intelligence and Simulated Behavior Conference. IOS Press, pp. 203–216, 1995.
[28] P. Merz, B. Freisleben, "Fitness landscapes, memetic algorithms and greedy operators for graph bi-partitioning", Evolutionary Computation, vol. 8, pp. 61–91, 2000.
[29] H. Hoos, K. Smyth, T. Stutzle, "Search space features underlying the performance of stochastic local search algorithms for max-sat", En: Yao, X., Burke, E., Lozano, J., Smith, J., editores. Parallel Problem Solving from Nature. Springer, pp. 51–60, 2004.
[30] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing", J of Heuristics, vol. 2, pp. 5–30, 1996.
[31] J. E. Schoenfield, Fast, exact solution of open bin packing problems without linear programming. Reporte técnico, US Army Space & Missile Defense Command, 2002.
[32] P. Schwerin, G. Wäscher, "The bin-packing problem: a problem generator and some numerical experiments with ffd packing and mtp", International Transactions in Operational Research, vol. 4(5-6), pp. 337–389, 1997.
[33] L. Cruz, Clasificación de algoritmos heurísticos para la solución de problemas de bin packing, tesis de doctorado; Centro Nacional de Investigación y Desarrollo Tecnológico; Cuernavaca, Morelos, México; 2004.
[34] M. Quiroz, L. Cruz, J. Torres, C. Gómez, M. López, J. Carrillo, G. Castilla, "Improving the performance of heuristic algorithms based on causal inference", En: Batyrshin, I., Sidorov, G., editores. Advances in Artificial Intelligence. Springer, pp. 137–148, 2011.
[35] G. Belov. Problems, models and algorithms in one and two dimensional cutting, tesis de doctorado, Fakultät Mathematik und Naturwissenschaften der Technischen Universität Dresden, 2004.
[36] V. H. Álvarez. Modelo para representar la complejidad del problema y el desempeño de algoritmos, tesis de maestría, Instituto Tecnológico de Ciudad Madero; México, 2006.
[37] E. K. Burke, M. R. Hyde, G. Kendall. "Evolving bin packing heuristics with genetic programming", En T. P. Runarsson et al., editores. Parallel Problem Solving from Nature. Springer. pp. 860–869, 2006.
[38] L. Cruz, D. Nieto, P. Tomás, G. Castilla, "Solving bin packing problem with a hybridization of hard computing and soft computing,". En: E. Corchado, J. M. Corchado, A. Abraham, editores. Lecture Notes in Innovations in Hybrid Intelligent Systems. Springer, pp. 223–230, 2007.
[39] L. Cruz, M. Quiroz, A. Alvim, H. Fraire, S. Gómez, J. Torres, "Heurísticas de agrupación híbridas eficientes para el problema de empacado de objetos en contenedores", Computación y Sistemas, vol. 16(3), pp.349–360, 2012.
[40] F. Ducatelle, J. Levine, "Ant colony optimization for Bin Packing and Cutting Stock problems," En: Proceedings of the UK Workshop on Computational Intelligence", Edinburgh, Scotland, pp. 1–8, 2001.
[41] K. Fleszar, C. Charalambous, "Average-weight-controlled bin-oriented heuristics for the onedimensional bin-packing problem", Discrete Optimization, vol. 210, pp. 176–184, 2011.
[42] L. P. Gent, "Heuristic solution of open bin packing problem", J of Heuristics vol. 3, pp. 299–304, 1998.
[43] S. Khuri, M. Schütz, J. Heitkötter, "Evolutionary heuristics for the bin packing problem", En: Proceedings of the 2nd International Conference on Artificial Neural Networks and Genetic Algorithms. Krakow, Poland: Springer, pp. 285–288, 1995.
[44] P. Ross, S. Schulenburg, B. J. Marin, E. Hart, "Hyper-heuristics: learning to combine simple heuristics in Bin-Packing problems", En: Proceedings of the Genetic and Evolutionary Computation, New York, USA,. p. 942–948, 2002.
[45] Beasley, J.B.. The operational research library. http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.html. Última visita: 2015.
[46] G. Wäscher, T. Gau, "Heuristics for the one-dimensional cutting stock problem: a computational study", OR Spektrum, vol. 18, pp. 131–144, 1996.
[47] ESICUP. Euro especial interest group on cutting and packing (esicup). http://paginas.fe.up.pt/~esicup/tiki-list_file_gallery.php?galleryId=1. Última visita: 2015.
[48] A. C. F. Alvim. Uma heurística híbrida de melhoria para o problema de bin packing e sua aplicação ao problema de escalonamento de tarefas, Tesis de doctorado; Catholic University of Rio de Janeiro, Brasil, 2003.
[49] P. Rohifshagen, J. A. Bullinaria, "Genetic algorithm with exon shuffling crossover for hard bin packing problems", En: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, London, England, pp. 1365–1371, 2007.
[50] P. Ross, B. J. Marin, S. Schulenburg, E, Hart, "Genetic algorithm with exon shuffling crossover for hard bin packing problems", En: Proceedings of the Genetic and Evolutionary Computation. Chicago, USA; vol. 2724, pp. 1295–1306, 2003.
[51] R. Klein, A. Scholl, Bin packing. http://www.wiwi.uni-jena.de/Entscheidung/binpp/. Última visita: 2015.
[52] G. Scheithauer, J. Riehme, J. Rietz, G. Belov, Cutting & Packing at Dresden University. http://www.math.tu-dresden.de/~capad/. Última visita: 2015.
[53] R. E. Korf, "A new algorithm for optimal bin packing", En: Proceedings of the Eighteenth National Conference on Artificial Intelligence. Edmonton, Canada, pp. 731–736, 2002.

Adriana Mexicano Santoyo es profesora en el Instituto Tecnológico de Cd. Victoria. Obtuvo el grado de Doctora en el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) en el año 2012. Sus áreas de interés incluyen el análisis de algoritmos, optimización combinatoria, descubrimiento de conocimiento, entre otras.

Joaquín Pérez Ortega es profesor investigador en el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET). Obtuvo el grado de Doctor en Ciencias de la Computación por el Instituto Tecnológico de Monterrey (ITEMS) en 1999. Es miembro del Sistema Nacional de Investigadores. Es miembro Senior de la IEEE. Sus áreas de interés incluyen la optimización combinatoria, minería de datos, ingeniería de software, bases de datos, metaheurísticas, entre otras.

David Romero Vargas es Director del Instituto de Matemáticas de la Universidad Nacional Autónoma de México (IMA-UNAM). Recibió el grado de Doctor en matemáticas aplicadas por L'Université Scientifique et Medicale de Grenoble, Fracia en 1978. Es miembro del Sistema Nacional de Investigadores. Es miembro regular de la Academia Mexicana de Ciencias. Sus áreas de interés incluyen sistemas complejos, modelado matemático y simulación.

Laura Cruz Reyes recibió el grado de maestra en ciencias de la computación del Instituto Tecnológico y de Estudios Superiores de Monterrey, México, en 1999 y el grado de doctora en ciencias de la computación del Centro Nacional de Investigación y Desarrollo Tecnológico, México, en el 2004. Es profesora de tiempo completo en el Instituto Tecnológico de Ciudad Madero, México. Sus áreas de interés incluyen redes complejas, aprendizaje automático y técnicas de optimización.