Heurísticas de agrupación híbridas eficientes para el problema de empacado de objetos en contenedores

Laura Cruz-Reyes[1], Marcela Quiroz C.[1], Adriana C. F. Alvim[2], Héctor J. Fraire Huacuja[1], Claudia Gómez S.[1] y José Torres-Jiménez[3]

[1] Instituto Tecnológico de Ciudad Madero, México

[2] Universidad Federal do Estado do Rio de Janeiro, Brasil

[3] Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, México

lauracruzreyes@itcm.edu.mx, qc.marcela@gmail.com, adriana@uniriotec.br, automatas2002@yahoo.com.mx, cggs71@hotmail.com, jtj@tamps.cinvestav.mx

Resumen. *En este artículo se aborda un problema clásico muy conocido por su aplicabilidad y complejidad: el empacado de objetos en contenedores (Bin Packing Problem, BPP). Para la solución de BPP se propone un algoritmo genético híbrido de agrupación denominado HGGA-BP. El algoritmo propuesto está inspirado en el esquema de representación de grupos de Falkenauer, el cual aplica operadores evolutivos a nivel de contenedores. HGGA-BP incluye heurísticas eficientes para generar la población inicial y realizar mutación y cruzamiento de grupos; así como estrategias híbridas para el acomodo de objetos que quedaron libres al aplicar los operadores grupales. La efectividad del algoritmo es comparable con la de los mejores del estado del arte, superando los resultados publicados para el conjunto de instancias hard28, el cual ha mostrado el mayor grado de dificultad para los algoritmos de solución de BPP.*

Palabras clave. *Metodologías computacionales, inteligencia artificial, solución de problemas, problema de empacado de objeto en contenedores, algoritmo genético híbrido.*

Efficient Hybrid Grouping Heuristics for the Bin Packing Problem

Abstract. *This article addresses a classical problem known for its applicability and complexity: the Bin Packing Problem (BPP). A hybrid grouping genetic algorithm called HGGA-BP is proposed to solve BPP. The proposed algorithm is inspired by the Falkenauer grouping encoding scheme, which applies evolutionary operators at the bin level. HGGA-BP includes efficient heuristics to generate the initial population and performs mutation and crossover for groups as well as hybrid strategies for the arrangement of objects that were released by the group operators. The effectiveness of the algorithm is comparable with the best state-of-the-art algorithms, outperforming the published results for the class of instances hard28, which has shown the highest difficulty for algorithms that solve BPP.*

Keywords. *Computer methodologies, artificial intelligence, problem solving, bin packing problem, hybrid genetic algorithm.*

1 Introducción

El problema de empacado de objetos en contenedores en una dimensión (one-dimensional Bin Packing Problem BPP), consiste en almacenar un conjunto de objetos de diferentes tamaños $N = \{1,\ldots,n\}$, o pesos, en el menor número de contenedores de tamaño fijo sin violar la capacidad de ningún contenedor.

BPP es un problema de optimización combinatoria NP-duro, considerado intratable debido a que demanda una gran cantidad de recursos para su solución [2,16]. La importancia de BPP radica en que tiene un extenso número de aplicaciones industriales y logísticas, y gran cantidad de problemas prácticos pueden ser modelados como problemas de empacado [8, 9, 10].

A lo largo de los últimos veinte años, en la búsqueda de buenas y mejores soluciones para BPP se han diseñado una gran variedad de algoritmos. Los resultados más destacados han sido obtenidos mediante el uso de algoritmos híbridos y metaheurísticos.

Martello y Toth [24] proponen un procedimiento de ramificación y poda (MTP) que incluye un criterio de dominación [23] para reducir el espacio de búsqueda. El criterio de dominación establece que: si es posible intercambiar un conjunto de objetos de un contenedor por un solo objeto del mismo peso que el conjunto, la solución puede ser mejorada debido a que será más fácil distribuir el conjunto de objetos pequeños en el resto de los contenedores, a encontrar un lugar para el objeto grande, debido a que su dominio es mayor. E. Falkenauer [13] propone un algoritmo genético híbrido de agrupación (HGGA) que utiliza: un esquema de representación de la solución por grupos de objetos y un método de optimización local inspirado en el criterio de dominación de Martello y Toth.

Scholl *et al.* [28] desarrollan un procedimiento híbrido (BISON) que combina una búsqueda tabú con un método de ramificación y poda, usando una estrategia dual que consiste en minimizar el llenado de los contenedores dado un número fijo de éstos. BISON incluye un nuevo esquema de ramificación basado en límites inferiores del valor de la solución óptima. Coffman *et al.* [7] presentan un estudio exhaustivo de las principales heurísticas deterministas para Bin Packing, incluyendo los análisis del peor caso de los algoritmos FFD y BFD [23]. Schwerin y Wäscher [29] proponen un nuevo límite inferior para BPP basado en problemas de corte y lo integran al método MTP de Martello y Toth [24], obteniendo resultados de calidad con su nuevo algoritmo (MTPCS). Fleszar y Hindi [14] introducen un nuevo algoritmo (Pertubation-MBS´) que incorpora una versión modificada de la heurística MBS de Gupta y Ho [18], una búsqueda de vecindad variable y límites inferiores, obteniendo buenos resultados. Levine y Ducatelle [20] desarrollan un método híbrido (HACO-BP) que implementa la metaheurística de optimización basada en colonia de hormigas; este algoritmo incluye una estrategia de búsqueda local y se apoya en el criterio de dominación de Martello y Toth [23]. Bhatia y Basu [5] presentan un algoritmo genético de agrupación multi-cromosómico (MGGA) y una nueva heurística de empacado (better-fit).

Alvim *et al*. [1] proponen una heurística de mejora híbrida (HI_BP) en la que retoman la estrategia dual de Scholl *et al*. [28], así como las técnicas de reducción de Martello y Toth [23], obteniendo los mejores resultados del estado del arte, hasta ese momento.

Singh y Gupta [31] desarrollan un enfoque evolutivo híbrido (C_BP) que combina un algoritmo genético de agrupación con una versión mejorada de la heurística Perturbation-MBS' de Fleszar y Hindi [14]. El desempeño obtenido es comparable con la estrategia HI_BP de Alvim *et al.* [1], sin embargo, el algoritmo C_BP es menos robusto.

A. Stawowy [32] propone una nueva estrategia evolutiva no especializada (ES) que incluye mutaciones inteligentes, una técnica de reducción del tamaño del problema y un esquema de representación basado en permutaciones con separadores de grupos. Los resultados obtenidos por el metaheurístico no hibridizado son comparables con los de algoritmos mucho más complicados. Rohlfshagen y Bullinaria [26] desarrollan un algoritmo genético inspirado en el proceso de formación de proteínas (ESGA) que obtiene resultados prometedores al ser comparado con otras estrategias.

Loh *et al.* [22] desarrollan un procedimiento (WA) que hace uso del concepto de recocido de pesos, el algoritmo propuesto es sencillo y fácil de implementar y obtiene un desempeño de alta calidad superando al mejor algoritmo del estado del arte, HI_BP [1].

Gómez-Meneses y Randall [17] presentan un procedimiento evolutivo híbrido de optimización extrema (HEO) que incorpora una búsqueda local para mejorar el empacado de los contenedores. R. Lewis [21] propone un algoritmo de escalado de colina (HC) que utiliza un esquema sencillo de mejora basado en el criterio de dominación y obtiene buenas soluciones superando, en algunos casos, el desempeño de algoritmos complejos.

El último trabajo en esta área fue presentado por Fleszar y Charalambous [15], los autores

proponen una modificación al procedimiento Perturbation-MBS' [14], que utiliza una estrategia para controlar el peso promedio de los objetos que son empacados en cada contenedor. La nueva heurística (Perturbation-SAWMBS) logra superar el desempeño reportado para los mejores algoritmos del estado del arte HI_BP, C_BP y WA [1, 22, 31]. Así mismo, los autores publican nuevos resultados para la heurística WA, los cuales contradicen el desempeño presentado inicialmente por sus autores.

En este artículo se presenta un algoritmo genético híbrido de agrupación para el problema de empacado de objetos en contenedores (*Hybrid Grouping Genetic Algorithm for Bin Packing*, *HGGA-BP*). Este algoritmo está inspirado en el esquema de representación de grupos propuesto por Falkenauer y Delchambre [12] para representar las soluciones como grupos de objetos asociados a contenedores y manipularlos con operadores grupales. HGGA-BP incorpora estrategias heurísticas eficientes, deterministas y aleatorias, para generar soluciones de calidad. Resultados experimentales muestran que HGGA-BP obtiene buenas soluciones en tiempos cortos, superando la efectividad de los mejores algoritmos del estado del arte en los casos de prueba más difíciles la clase hard28 [4].

2 Heurísticas híbridas de agrupación

De manera general, un algoritmo genético es una estrategia evolutiva poblacional donde un conjunto de soluciones son sometidas a un proceso de evolución que involucra: selección de individuos, cruzamiento y mutación, dando como resultado soluciones de mayor calidad. En esta sección se describen las heurísticas híbridas de agrupación propuestas para la creación y evolución de individuos. Las heurísticas propuestas incluyen diferentes algoritmos simples de empacado; su incorporación al algoritmo genético HGGA-BP se presenta en la siguiente sección.

2.1 Heurísticas de empacado

Las heurísticas de empacado son conocidos algoritmos deterministas, muy simples y rápidos, que han mostrado resultados satisfactorios en la solución de BPP [7, 19]. Se diferencian por la manera en que los objetos son tratados antes de ser acomodados y por la forma en que se elige el contenedor que almacenará cada objeto.

Primer Ajuste (First Fit, FF) [19]: Cada objeto en consideración es colocado en el primer contenedor que tenga suficiente capacidad disponible. En caso de que ningún contenedor parcialmente lleno pueda almacenarlo, el objeto es colocado dentro de un nuevo contenedor (vacío). Una variación a este método se establece cuando los objetos son tomados según el orden decreciente de sus pesos dicha variante es conocida como Primer Ajuste Decreciente (First Fit Decreasing, FFD).

Mejor Ajuste (Best Fit, BF) [19]: Cada objeto es acomodado en el contenedor más lleno que lo pueda almacenar, agregando nuevos contenedores cuando sea necesario. De igual manera que con FF, existe una variación, llamada Mejor Ajuste Decreciente (Best Fit Decreasing, BFD), que considera los objetos en orden decreciente de sus pesos.

Peor Ajuste (Worst Fit, WF) [19]: Contrario a Mejor Ajuste, cada objeto en consideración es almacenado en el contenedor menos lleno con capacidad residual suficiente para contenerlo. La variante que toma los objetos según el orden decreciente de sus pesos es conocida como Peor Ajuste Decreciente (Worst Fit Decreasing, WFD).

Best 3-Fit (B3F) [1]: Inicialmente se abre un número límite de contenedores. Luego, si existe un contenedor vacío, se selecciona y se coloca el objeto actual, de otro modo, se intenta llenar cada contenedor con objetos que no han sido seleccionados y que por pares suman la capacidad residual del contenedor. Para el resto de los objetos, el elemento actual es insertado en el contenedor más lleno en el que ajuste (como en BF). Si no existe un contenedor con capacidad suficiente un nuevo contenedor es agregado a la solución.

Heurísticas aleatorias: Las heurísticas FF, BF, WF y B3F son estrategias deterministas y son la base del proceso de generación de individuos en el algoritmo HGGA-BP pues las soluciones obtenidas con ellas son individuos de buena calidad [1, 7,19]. Para obtener un grupo de individuos diversos, soluciones distintas son

obtenidas al manejar órdenes aleatorios del conjunto de objetos a acomodar. Las versiones aleatorias de las heurísticas de empacado han sido denominadas: *FF_Aleatorio*, *BF_Aleatorio*, *WF_Aleatorio* y *B3F_Aleatorio*, respectivamente.

2.2 Construcción de individuos

Todas las estrategias de construcción de individuos inician con el cálculo de un *límite inferior* del número de contenedores LM, que es igual al número de objetos de tamaño mayor que la mitad de la capacidad del contenedor. LM representa el número de objetos "grandes" que no podrían combinarse entre sí. Una vez calculado dicho límite, los LM objetos más pesados son almacenados cada uno en un contenedor y el resto de los objetos son acomodados con alguna de las técnicas para empacado de objetos descritas en la sección anterior.

2.3 Generación de la población

Con el método *PI_ D-A* la población inicial se forma con dos tipos de individuos: deterministas y aleatorios. La parte determinista se conforma de cuatro individuos creados con FFD, BFD, WFD, y B3FD. La población aleatoria se compone de individuos creados con las estrategias FF_Aleatorio, BF_Aleatorio, WF_Aleatorio y B3F_Aleatorio.

El cálculo de la *Aptitud* de cada individuo de la población es realizado utilizando la función de costo introducida por Falkenauer y Delchambre, que evalúa el promedio de llenado de los contendores que conforman una solución [12]. Dado que, según esta función, las soluciones que poseen un mayor promedio de llenado son mejores, el objetivo del algoritmo HGGA-BP es maximizar el valor de las aptitudes de los individuos que conforman la población. La Ecuación 1 define el cálculo de la aptitud, donde m es el número de contenedores utilizado en la solución, S_i es la suma de los tamaños de los objetos en el contenedor i y c es la capacidad del contenedor.

$$Aptitud = \frac{\sum_{i=1}^{m} \left(S_i / c\right)^2}{m} \quad (1)$$

2.4 Operadores genéticos para grupos

Un esquema de codificación por agrupación para BPP fue propuesto por Falkenauer y Delchambre [12]. En este tipo de estructura cromosómica los genes del individuo están representados por contenedores y no por objetos. Para manipular grupos de objetos en el algoritmo HGGA-BP se propone utilizar operadores genéticos especiales y heurísticas de reacomodo de objetos que quedan libres al aplicar dichos operadores.

2.4.1 Heurísticas de reacomodo de objetos

En el proceso de solución, los operadores genéticos generan y modifican individuos, producto de este proceso algunos contenedores son vaciados y existen objetos libres que necesitan ser reinsertados en la solución. En este trabajo se proponen dos nuevos procedimientos de reacomodo.

Reacomodo_Voraz: Este procedimiento recorre cada contenedor intentando intercambiar uno de sus objetos por un objeto libre de mayor tamaño, sin violar su capacidad. Después de recorrer todos los contenedores, los objetos libres se reacomodan utilizando la heurística de empacado BFD.

Reacomodo_por_Pares: Esta estrategia consta de dos etapas: Generar una permutación aleatoria del orden de los contenedores y recorrer por pares todos los objetos de cada contenedor intentando intercambiar el par de objetos por una mejor opción. Las alternativas de sustitución son dos: a) Sustituir el par de objetos del contenedor por un objeto libre de peso igual o mayor que no sobrepase la capacidad del contenedor y b) Sustituir el par de objetos del contenedor por un par de objetos diferentes que sumen lo mismo o que llenen más el contenedor. Finalmente, si existen objetos libres, la heurística BF_Aleatorio es aplicada para completar la solución.

2.4.2 Cruzamiento, mutación y eliminación

Para el algoritmo HGGA-BP, se implementa una modificación del operador de cruzamiento propuesto por Falkenauer y Delchambre [12] y se proponen un operador de cruzamiento y dos operadores de mutación que tratan de mejorar la aptitud de los individuos. Adicionalmente, se propone un operador de sustitución para diversificar la población al sustituir individuos con aptitudes repetidas.

Cruzamiento_BFD: En HGGA-BP se implementa el cruzamiento para grupos propuesto por Falkenauer y Delchambre [12], con la diferencia de que los objetos libres son acomodados con la heurística BFD en lugar de FFD. Cruzar individuos implica determinar los contenedores que deben ser heredados de padres a hijos. Aquellos contenedores que sobreviven el proceso evolutivo se caracterizan por ser dominantes sobre otros. En esta técnica, se crean, de manera aleatoria, dos puntos de corte en cada individuo a cruzar, que dividen cada solución en tres segmentos. El nuevo individuo (denominado hijo) está formado por el primer segmento de contenedores del primer padre, el segundo segmento de contenedores del segundo padre y el resto de los contenedores del primer padre, eliminándose contenedores con objetos duplicados y reacomodando los objetos libres con la heurística BFD.

Mutación_RV: Este operador de mutación elimina un porcentaje de los contenedores menos llenos e intenta reacomodar los objetos libres utilizando la heurística Reacomodo_Voraz. Para cada individuo a mutar, el número de contenedores a vaciar se calcula en relación al tamaño de la solución, dependiendo de qué tan grande o pequeño sea el individuo.

Cruzamiento_RpP: Este nuevo método de cruzamiento incorpora explotación de buenas soluciones. En esta estrategia, los individuos a cruzar son seleccionados de manera aleatoria tomando en cuenta sólo soluciones de buena calidad (la mejor mitad de la población). A diferencia del Cruzamiento_BFD, en ambos padres, cada contenedor es considerado como un punto de corte. En cada punto de corte se procesan un contenedor del primer padre y un contendor del segundo padre. El contenedor que se encuentra lleno es el primero en ser heredado a la nueva solución, para luego heredar el otro contenedor; si ambos o ningún contenedor se encuentra lleno, se da preferencia al primer padre. Los contenedores que incluyen objetos duplicados son eliminados y los objetos libres son reinsertados aplicando Reacomodo_por_Pares.

Mutación_RpP: Un segundo método de mutación es incluido para individuos con buenas aptitudes y consiste principalmente en: calcular el número de contenedores a vaciar de acuerdo al tamaño de la solución y al número de contenedores incompletos y reacomodar los objetos libres utilizando Reacomodo_por_Pares.

Eliminación_por_Sustitución: Este nuevo operador es utilizado para eliminar individuos con aptitudes duplicadas en la población, siguiendo el enfoque de substitución. El operador previene la convergencia prematura del algoritmo a regiones sub-óptimas, al mismo tiempo que conserva la variabilidad en la población. En cada generación, los individuos que presentan un valor de aptitud repetido son sustituidos por nuevos individuos: n_{b3f} generados con la heurística de empacado B3F_Aleatorio y el resto con FF_Aleatorio.

3 Algoritmo genético HGGA-BP

La Tabla 1 presenta el algoritmo HGGA-BP propuesto en este trabajo. El proceso comienza generando una población inicial con el método *PI_D-A*, el cual crea un conjunto de individuos (soluciones) con heurísticas deterministas y aleatorias (Línea 3). En cada iteración, dado un porcentaje de cruzamiento y con base en la aptitud, se selecciona un conjunto de individuos para aplicarles el operador *Cruzamiento_BFD* (Líneas 5-7). Posteriormente, según el porcentaje de mutación, los peores individuos de la población sufren pequeñas alteraciones genéticas usando *Mutación_RV*; el objetivo es mejorar su aptitud (Línea 8).

En una segunda etapa se procede a intensificar la búsqueda, con el objetivo de perfeccionar el empacado de las mejores soluciones. Primero, los individuos más aptos son clonados para obtener nuevos individuos mediante el operador *Mutación_RpP* (Líneas 10-12). Luego, los individuos con aptitud repetida

Tabla 1. Algoritmo genético híbrido de agrupación para BPP: HGGA-BP

```
1   Inicio
2      Inicializar parámetros: max_gen, L2, gen1, gen2
3      Generar la población inicial con PI_D-A
4      mientras generación < max_gen y mejor_solución > L2
5         Seleccionar individuos a cruzar en proporción a su aptitud
6         Aplicar Cruzamiento_BFD y generar nuevos_individuos
7         Sustituir los peores individuos por los nuevos_individuos
8         Aplicar Mutación_RV a los peores individuos
9         si generación > gen1
10           Clonar los mejores individuos de la población
11           Aplicar Mutación_RpP a los individuos clonados
12           Sustituir los peores individuos por los individuos clonados
13        fin si
14        Aplicar Eliminación_por_Sustitución a individuos repetidos
15        si generación > gen2
16           Seleccionar los mejores individuos para cruzar
17           Aplicar Cruzamiento_RpP para generar nuevos_individuos
18           Sustituir los peores individuos por los nuevos_individuos
19        fin si
20        Registrar la mejor_solución
21     fin mientras
22  fin procedimiento
```

son reemplazados por nuevos individuos al aplicar el operador *Eliminación_por_Sustitución* (Línea 14). Finalmente, los mejores individuos de la población son apareados para generar individuos de alta calidad producto del operador *Cruzamiento_RpP* (Líneas 16-18).

Al término de cada generación la mejor solución de la población es registrada (Línea 20). El resultado final del algoritmo es el individuo más apto de todo el proceso evolutivo. El algoritmo iterará un máximo número generaciones *max_gen*, y se detendrá antes en caso de encontrar una solución cuyo tamaño coincidacon el límite L_2 de Martello y Toth [23].

En el cálculo de L_2, la idea principal es encontrar un valor ε, entre cero y la mitad de la capacidad del contenedor, que maximice el resultado obtenido al dividir el conjunto de pesos en objetos grandes (con peso mayor que $1 - \varepsilon$) y pequeños (con peso menor que ε). Para cada valor de ε, se asume que los objetos grandes son almacenados en diferentes contenedores y los objetos pequeños son utilizados para terminar de llenar los contenedores donde se almacenaron los objetos grandes.

4 Experimentos computacionales

HGGA-BP fue desarrollado en lenguaje C++ y las experimentaciones se ejecutaron en una computadora personal con procesador Intel Xenon a 1.866 Ghz y 4GB de RAM, sobre el sistema operativo Windows XP profesional SP2.

Con el fin de demostrar la robustez del algoritmo, para cada caso de prueba se realizaron 30 ejecuciones del algoritmo con diferentes semillas de números aleatorios. Los valores reportados como soluciones finales de cada instancia representan los promedios de las 30 soluciones generadas.

El algoritmo se configuró mediante un estudio experimental sobre el desempeño de las heurísticas propuestas [25]. El tamaño de la población es de 150 individuos. La población inicial se crea con diferentes heurísticas (1 con *FFD*, 1 con *BFD*, 1 con *WFD*, 1 con *B3FD*, 1 con *B3F_Aleatorio*, 56 con *FF_Aleatorio*, 60 con *BF_Aleatorio* y 30 con *WF_Aleatorio*). El máximo

número de generaciones max_gen es 100. En cada generación se aplican los operadores *Cruzamiento_BFD* y *Mutación_RV* al 30% de los individuos de la población. En *Mutación_RV*, el intervalo de contenedores a vaciar es 10-35%. Después de la décima generación (gen_1=10), el operador *Mutación_RpP* es aplicado, en cada generación, para generar 15 nuevos individuos que sustituyen a las 15 peores soluciones y para introducir pequeñas modificaciones en 15 buenos individuos. *Cruzamiento_RpP* se utiliza para generar 35 nuevos individuos que sustituyen a las 35 peores soluciones y es aplicado después de la generación número 20 (gen_2=20). El operador *Eliminación_por_Sustitución* es aplicado en cada generación para generar n_{b3f}=3 individuos con la heurística de empacado *B3F_Aleatorio* y el resto con la estrategia *FF_Aleatorio*.

4.1 Instancias de prueba

El desempeño de los algoritmos para BPP ha sido evaluado con diferentes clases de instancias de referencia consideradas retadoras. Los casos de prueba, elegidos por diferentes investigadores para comparar las cualidades de sus estrategias con las de otros algoritmos del estado del arte, conforman un conjunto de 1615 casos estándar reconocidos por la comunidad científica. Las 1615 instancias de prueba se encuentran en sitios de Internet reconocidos [3, 6, 11, 27], sus soluciones óptimas son conocidas [1, 4, 27]. Las instancias consideradas se han dividido en cuatro grupos, tomando en cuenta su origen y los sitios de los que fueron obtenidas.

El primer grupo consiste en dos conjuntos de instancias propuestas por E. Falkenauer [13].

***Uniform* (*u*)** [3]: Conjunto de 80 instancias identificadas con la letra *u* debido a que su principal característica es que los pesos de los objetos están uniformemente distribuidos entre 20 y 100. La capacidad del contenedor *c* es de 150 y existen cuatro clases de casos cada uno con *n* = 120, 250, 500 y 1000 objetos. Cada clase posee 20 instancias identificadas respectivamente por *u*_120, *u*_250, *u*_500 y *u*_1000. El valor de la solución óptima para cada una de estas instancias es conocido [1].

***Triplets* (*t*)** [3]: Conjunto de 80 instancias difíciles, identificadas con la letra *t*. Su nombre se debe a que las instancias fueron construidas con una solución óptima conocida de *n*/3 contenedores, de tal forma que cada contenedor de la solución óptima debe almacenar exactamente tres objetos que lo llenan completamente. El tamaño de las instancias es de *n* = 60, 120, 249 y 501, definiéndose así cuatro clases, el tamaño del contenedor *c* es de 100, mientras que los pesos están distribuidos entre 25 y 50. Cada clase posee 20 instancias identificadas respectivamente por *t*_60, *t*_120, *t*_249 y *t*_501.

El segundo grupo está formado por tres conjuntos de casos de prueba introducidos por Scholl *et al.* [28]. En cada conjunto, las diferentes clases de problemas fueron creados variándose el número de objetos *n*, la capacidad del contenedor *c* y los posibles pesos de los objetos.

***Data Set 1* (*set_1*)** [27]: Construidas de forma similar que algunas instancias propuestas por Martello y Toth [24] que resultaron difíciles. Son un conjunto de 720 instancias denotadas con *n*-*c*-*w* por los datos que manejan: *n* = 50, 100, 200 y 500, capacidad del contenedor *c* = 100, 120, 150 y los pesos *w* generados uniformemente en intervalos de [1,100], [20,100] y [30,100]. La combinación de los diferentes parámetros resulta en 36 clases, cada clase contiene 20 instancias.

***Data Set 2* (*set_2*)** [27]: Incluye 480 instancias con pesos generados con una distribución uniforme, denotadas por *n*-*w*-*b*. Cada sigla representa la configuración de un parámetro de entrada: número de objetos *n* con valores de 50, 100, 200 y 500, la capacidad de los contenedores *c* es 1000. Con el objetivo de generar instancias cuyo número medio de objetos por contenedor variara entre tres y nueve, se consideraron otros dos parámetros: el peso medio deseado *w* con valores c/3, c/5, c/7, c/9, y una desviación máxima de dicho peso *b* = 20%, 50%, 90%. Por ejemplo cuando *w* = c/5 y *b*= 50%, los pesos de los objetos fueron generados de manera aleatoria con una distribución uniforme en el intervalo discreto [100, 300]. Al Combinar las características anteriores se cuenta con 48 clases, cada una con 10 instancias.

***Data Set 3* (*set_3*)** [27]: Conjunto formado por una única clase con diez instancias consideradas difíciles. Cada instancia posee 200 objetos, capacidad de contenedor 100000 y los pesos

están distribuidos uniformemente de manera dispersa entre 20000 y 35000.

El tercer grupo incluye tres clases de instancias sugeridas por Schwerin, Wäscher y Gau. Las primeras dos clases se han definido como ffd-hard (difíciles de resolver por la heurística FFD) [29,30]. La tercera clase también ha sido considerada difícil [33].

Was_1 [11]: Formado por 100 instancias de capacidad *c* = 1000. Cada instancia posee *n* = 100 objetos. Los pesos de los objetos varían entre 150 y 200.

Was_2 [11]: Incluye 100 instancias con capacidad del contenedor *c* = 1000, cada instancia contiene 120 objetos con pesos entre 150 y 200.

Gau_1 [11]: Son 17 instancias con características variadas. La capacidad de los contenedores es igual a 10000, el número de objetos *n* varía de 57 a 239 y los pesos se encuentran distribuidos entre 2 y 7332.

El grupo cuatro contiene 28 instancias difíciles que han sido consideradas en problemas de corte de una dimensión (one-dimensional Cutting Stock Problem).

Hard28 [6]: Incluye las 28 instancias más difíciles usadas por G. Belov [4], dichas instancias no pudieron ser resueltas por algoritmos de corte ni por métodos de reducción. El número de objetos *n* varía entre 160 y 200, los pesos de los objetos varían entre 1 y 800 y la capacidad del contenedor *c* es 1000.

Los últimos dos conjuntos de instancias, gau_1 y hard28, han mostrado poseer casos de prueba de alto grado de dificultad [4, 15]. Destacándose el conjunto hard28, para el que existe un mayor número de instancias que los algoritmos no logran solucionar de manera óptima.

4.2 Resultados experimentales

Para investigar la efectividad del algoritmo HGGA-BP, se realizó un estudio comparativo de los resultados obtenidos por este algoritmo con aquellos obtenidos por los mejores algoritmos reportados en la literatura: HI_BP [1], WA [22] y Perturbation-SAWMBS [15].

La Tabla 2 detalla, para cada clase de instancias, el número de casos de prueba y, para cada procedimiento, el número de instancias para las que el algoritmo alcanza la solución óptima conocida (columna ópt.), el radio teórico promedio de las soluciones obtenidas (columna

Tabla 2. Resultados obtenidos por los mejores algoritmos heurísticos aplicados a BPP (tiempo en seg.)

Clase	no. inst.	HI_BP [1]			WA [22]			Pert.-SAWMBS [15]			HGGA-BP [Este trabajo]		
		ópt.	radio	tiempo	ópt.	radio	tiempo	ópt.	radio	tiempo	ópt.	radio	tiempo
u	80	80	1	0.03	71	1.0011	0.28	79	1.0001	0.00	79	1.0001	1.38
t	80	80	1	0.98	0	1.0232	0.15	80	1	0.00	80	1	3.42
set_1	720	720	1	0.19	703	1.0003	0.25	720	1	0.01	718	1.0000	2.08
set_2	480	480	1	0.01	468	1.0006	0.04	480	1	0.00	480	1	0.67
set_3	10	10	1	4.60	9	1.0017	0.08	10	1	0.16	9	1.0017	6.56
was_1	100	100	1	0.02	100	1	0.00	100	1	0.00	100	1	0.02
was_2	100	100	1	0.02	100	1	0.02	100	1	0.01	100	1	0.52
gau_1	17	12	1.0122	0.60	13	1.0122	0.02	16	1.0025	0.04	15	1.0047	1.10
hard28	28	5	1.0119	**≥** 0.48	5	1.0119	**≥** 0.59	5	1.0119	**≥** 0.24	**8***	1.0106	4.31
Total	**1615**	**1587**	**1.0026**	**0.77**	**1469**	**1.0057**	**0.16**	**1590**	**1.0016**	**0.05**	**1589**	**1.0019**	**2.22**

HGGA-BP	1.866 GHz Intel Xenon
HI_BP	1.7 GHz Pentium IV
WA	2.33 GHz Core2
Pert.-SAWMBS	2.33 GHz Core2

≥ Un incremento significativo en el tiempo de ejecución no mejoró los resultados

*** HGGA-BP supera a los mejores algoritmos en el conjunto Hard28**

radio), así como el tiempo promedio de ejecución medido en segundos (columna tiempo). El *radio teórico* es una medida de calidad, y representa la razón entre el número de contenedores de la solución encontrada por el algoritmo y el número de contenedores utilizados en la solución óptima, cuando la solución obtenida es óptima el radio teórico será igual a 1 y será mayor en otro caso.

Los resultados del algoritmo HI_BP fueron obtenidos al ejecutar el código en lenguaje C proporcionados por la Dra. Adriana C. F. Alvim, los cuales coinciden con lo publicado en su artículo [1]. Los resultados de los procedimientos WA y Perturbation-SAWMBS fueron obtenidos de la publicación de Fleszar y Charalambous [15], donde se destaca que los resultados anunciados por Loh *et al.* [22] para la heurística WA son incorrectos. Con la finalidad de verificar los resultados de la heurística WA se realizó la codificación de dicho algoritmo, obteniendo resultados similares a los reportados por Fleszar y Charalambous [15], por lo que coincidimos con dichos autores en que los resultados publicados por Loh *et al.* [22] son incorrectos.

En todos los casos, cuando los algoritmos no logran encontrar la solución óptima, la solución encontrada posee sólo un contenedor extra respecto a la solución óptima.

En la misma Tabla 2 se han resaltado los resultados obtenidos para el conjunto de instancias hard28, el cual contiene las instancias hasta ahora más difíciles, para los algoritmos conocidos. Puede observarse que la estrategia propuesta en este trabajo, HGGA-BP, supera la efectividad de los mejores algoritmos del estado del arte, al resolver un mayor número de instancias. Para este conjunto de instancias, Fleszar y Charalambous [15] señalan que incluso incrementando el número de iteraciones de su algoritmo Perturbation-SAWMBS de 2,000 a 100,000 no es posible mejorar las soluciones encontradas, destacando la dificultad de estas instancias y recomendando su uso para futuros estudios de heurísticas de BPP. Resultados similares fueron obtenidos al incrementar el tiempo de búsqueda del procedimiento HI_BP.

La Tabla 3 presenta los resultados detallados del algoritmo HGGA-BP sobre el conjunto hard28, en esta tabla se incluye, para cada instancia, el valor del límite inferior de contenedores L_2, la solución óptima y las soluciones mínima, máxima y promedio obtenida por el algoritmo HGGA-BP en 30 corridas.

Los resultados más relevantes de la Tabla 3 se resaltaron con color de fondo en escala de grises y letras en negritas.

Color gris obscuro: resalta las cinco instancias de prueba (hBPP814, hBPP359, hBPP716, hBPP119 y hBPP175) que son solucionadas de manera óptima por los mejores algoritmos del estado del arte.

Color gris claro: resalta los tres casos de prueba (hBPP640, hBPP531 y hBPP814) en los que HGGA-BP supera a dichos algoritmos.

Sólo letras negritas: resalta tres casos (hBPP360, hBPP742 y hBPP47) para los cuales HGGA-BP obtiene la solución óptima en alguna de las 30 corridas.

Las 17 instancias restantes son consideradas muy difíciles pues el algoritmo HGGA-BP no logra obtener la solución óptima en ninguna corrida. Sería interesante conocer las características que marcan la diferencia en el grado de dificultad dentro de este conjunto de instancias para poder definir estrategias adecuadas que permitan obtener su solución óptima.

Para fundamentar las conclusiones obtenidas al comparar los resultados de los mejores algoritmos del estado del arte con HGGA-BP se efectuó un estudio estadístico comparativo de resultados usando la prueba no paramétrica *T* de Wilcoxon (Wilcoxon signed rank test) con un nivel de significancia de 0.05. El estudio fue realizado con el objetivo de comparar el desempeño de HGGA-BP con los mejores algoritmos: HI_BP [1] y Perturbation-SAWMBS [15]. Las pruebas fueron realizadas con el procedimiento UNIVARIATE de la herramienta SAS 9.2 para Windows.

Para el conjunto total de instancias la prueba mostró que no existen diferencias significativas en el desempeño de los tres algoritmos. Así mismo, al analizar los conjuntos de instancias u, t, set_1, set_2, set_3, was_1, was_2 y gau_1 tampoco se observaron diferencias significativas en el desempeño de los tres algoritmos. Sin embargo, para el conjunto hard28 la prueba *T* de Wilcoxon mostró que HGGA-BP es superior a los

Tabla 3. Resultados de HGGA-BP con hard28

Instancias	L_2	Óptimo	Min	Max	Prom
hBPP14	61	62	62	62	62
hBPP832	60	60	61	61	61
hBPP40	59	59	60	60	60
hBPP360	**62**	**62**	**62**	**63**	**63**
hBPP645	58	58	59	59	59
hBPP742	**64**	**64**	**64**	**65**	**65**
hBPP766	62	62	63	63	63
hBPP60	63	63	64	64	64
hBPP13	67	67	68	68	68
hBPP195	64	64	65	65	65
hBPP709	67	67	68	68	68
hBPP785	68	68	69	69	69
hBPP47	**71**	**71**	**71**	**72**	**72**
hBPP181	72	72	73	73	73
hBPP359	75	76	76	76	76
hBPP485	71	71	72	72	72
hBPP640	74	74	74	75	74
hBPP716	75	76	76	76	76
hBPP119	76	77	77	77	77
hBPP144	73	73	74	74	74
hBPP561	72	72	73	73	73
hBPP781	71	71	72	72	72
hBPP900	75	75	76	76	76
hBPP175	83	84	84	84	84
hBPP178	80	80	81	81	81
hBPP419	80	80	81	81	81
hBPP531	83	83	83	83	83
hBPP814	81	81	81	82	81

otros procedimientos, al mejorar la solución en seis instancias: hBPP640, hBPP531, hBPP814, hBPP360, hBPP742 y hBPP47 (ver Tabla 3).

5 Conclusiones y trabajo futuro

Desarrollamos un nuevo algoritmo genético híbrido, denominado HGGA-BP, para resolver el problema clásico de empacado de objetos en contenedores. HGGA-BP está inspirado en el esquema de representación propuesto por Falkenauer [13] para problemas de agrupación como BPP.

El metaheurístico híbrido HGGA-BP conjunta diferentes heurísticas de solución de BPP y crea un balance entre exploración y explotación del espacio de búsqueda con el fin de obtener las mejores soluciones. El tiempo de ejecución del algoritmo propuesto es muy corto al ser comparado con el requerido por otras estrategias poblacionales [13, 20, 32]. La calidad de las soluciones encontradas por HGGA-BP es similar a la obtenida por los mejores algoritmos del estado del arte, superando los resultados mostrados por las mejores estrategias de BPP sobre el conjunto de instancias hasta ahora más difíciles (hard28).

La revisión de los resultados obtenidos por los mejores algoritmos de solución de BPP reveló que aún existen instancias de la literatura que presentan un alto grado de dificultad para las cuales las estrategias incluidas en los algoritmos no parecen conducir a mejores soluciones. Al realizar el análisis de la literatura, se observó que ninguno de los algoritmos del estado del arte ha sido analizado para explicar el porqué de su buen o mal desempeño. Por otro lado, tampoco existe un análisis general sobre la estructura y dificultad de las instancias de BPP. Es importante identificar cuáles son las características que distinguen a las instancias de BPP y que pueden ser causa de su grado de dificultad. Así mismo, es necesario entender el comportamiento de los algoritmos y evaluar las estrategias que les permiten alcanzar su desempeño.

Referencias

1. **Alvim, A.C.F., Ribeiro, C.C., Glover, F., & Aloise, D.J. (2004).** A hybrid improvement heuristic for the one-dimensional bin packing problem. *Journal of Heuristics*, 10(2), 205–229.
2. **Baase, S. & Gelder, A.V. (2000).** *Computer Algorithms, Introduction to Design and Analysis* (3rd ed.). Reading, Mass.: Addison-Wesley Longman.
3. **Beasley, J.E. (1990).** OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11), 1069–1072. Retrieved from http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.html.
4. **Belov, G. (1992).** Problems, Models and Algorithms in One and Two Dimensional Cutting. Ph.D. Thesis, Technischen Universitat Dresden, St. Petersburg, Russland.
5. **Bhatia, A.K. &Basu, S.K. (2004).** Packing Bins Using Multi-chromosomal Genetic Representation and Better-Fit Heuristic. *Neural Information*

Processing, Lecture Notes in Computer Science, 3316, 181–186.

6. **CaPaD.** Cutting and Packing at Dresden University: Test instances & results. Retrieved fromhttp://www.math.tu-dresden.de/~capad/cpd-ti.html#pmp%2024.
7. **Coffman, E.G., Garey, M.R., & Johnson D.S. (1997).** Approximation algorithms for bin packing: a survey. In Hochbaum D.S. (Ed.), *Approximation algorithms for NP-hard problems* (46–93). Boston: PWS Publishing.
8. **Crainic, T.G., Perboli, G., Rei, W., & Tadei, R. (2011).** Efficient Lower Bounds and Heuristics for the Variable Cost and Size Bin Packing Problem. *Computers and Operations Research*, 38(11), 1474–1482.
9. **Cruz, L., Nieto-Yáñez, D.M., Rangel-Valdez, N., Herrera, J.A., González, J., Castilla, G., & Delgado-Orta, J.F. (2007).** DiPro: An Algorithm for the Packing in Product Transportation Problems with Multiple Loading and Routing Variants. *6th Mexican International Conference on artificial Intelligence (MICAI2007:Advances in Artificial Intelligence)*, Lecture Notes in Artificial Intelligence, 4827, 1078–1088.
10. **Di Natale, M. & Bini, E. (2007).** Optimizing the FPGA implementation of HRT systems. *13th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS'07),* Bellevue, Washington, USA, 22–31.
11. **ESICUP.** Euro Especial Interest Group on Cutting and Packing. One Dimensional Cutting and Packing Data Sets. Retrieved from http://paginas.fe.up.pt/~esicup/tiki-list_file_gallery.php?galleryId=1%2023
12. **Falkenauer, E. & Delchambre, A. (1992).** A Genetic Algorithm for Bin Packing and Line Balancing. *1992 IEEE International Conference on Robotics and Automation*, Nice, France, 2, 1186–1192.
13. **Falkenauer, E. (1996).** A Hybrid Grouping Genetic Algorithm for Bin Packing. *Journal of Heuristics*, 2(1), 5–30.
14. **Fleszar, K. & Hindi, K. (2002).** New heuristics for one-dimensional bin-packing. *Computers & Operations Research*, 29(7), 821–839.
15. **Fleszar, K. & Charalambous, C. (2011).** Average-weight-controlled bin-oriented heuristics for the one-dimensional bin-packing problem. *European Journal of Operational Research*. 210(2), 176–184.
16. **Garey, M.R. & Johnson, D.S. (1979).** *Computers and Intractability: A Guide to the Theory of NP Completeness*. San Francisco: W. H. Freeman
17. **Gómez-Meneses, P. & Randall, M.A. (2009).** Hybrid Extremal Optimisation Approach for the Bin Packing Problem. *Artificial Life: Borrowing from Biology, 4th Australian Conference, ACAL 2009, Lecture Notes in Computer Science*, 5865, 242–251.
18. **Gupta, J.N.D. & Ho, J.C. (1999).** A new heuristic algorithm for the one-dimensional bin-packing problem. *Production Planning & Control: The Management of Operations*, 10(6), 598–603. Retrieved from http://www.math.tu-dresden.de/~capad/cpd-ti.html#pmp%2024
19. **Johnson, D.S. (1974).** Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8(3), 272–314.
20. **Levine, J. & Ducatelle, F. (2004).** Ant colony optimization and local search for bin packing and cutting stock problems. *Journal of the Operational Research Society*, 55(7), 705–716.
21. **Lewis, R. (2009).** A general-purpose hill-climbing method for order independent minimum grouping problems: A case study in graph colouring and bin packing. *Computers & Operations Research*, 36(7), 2295–2310.
22. **Loh, K.H., Golden, B., &Wasil, E. (2008).** Solving the one-dimensional bin packing problem with a weight annealing heuristic. *Computers & Operations Research*, 35(7), 2283–2291.
23. **Martello, S. &Toth, P. (1990).** *Knapsack Problems: Algorithms and Computer Implementations*. New York: J. Wiley & Sons.
24. **Martello, S. & Toth, P. (1990).** Lower Bounds and Reduction Procedures for the Bin Packing Problem. *Discrete Applied Mathematics,* 28(1), 59–70.
25. **Quiroz, M. (2009).** Caracterización de Factores de Desempeño de Algoritmos de Solución de BPP. Tesis de maestría, Instituto Tecnológico de Cd. Madero, Tamaulipas, México.
26. **Rohlfshagen, P. & Bullinaria, J.A. (2010).**Nature inspired genetic algorithms for hard packing problems. *Annals of Operations Research*, 179(1), 393–419.
27. **Scholl, A. & Klein, R. (s.f.).** Bin Packing. Retrieved from http://www.wiwi.uni-jena.de/Entscheidung/binpp/
28. **Scholl, A., Klein, R., & Jürgens, C. (1997).** BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers and Operations Research*, 24(7), 627–645.

29. **Schwerin, P. & Wäscher, G. (1999).** A new lower bound for the bin-packing problem and its integration to MTP. *Pesquisa Operacional*, 19(2), 111–130.
30. **Schwerin, P. & Wäscher, G. (1997).**The bin packing problem: A problem generator and some numerical experiments with FFD packing and MTP. *International Transactions in Operational Research*, 4(5-6), 337–389.
31. **Singh, A. & Gupta, A.K. (2007).** Two heuristics for the one-dimensional bin-packing problem. *OR Spectrum*, 29(4), 765–781.
32. **Stawowy, A. (2008).** Evolutionary based heuristic for bin packing problem. *Computers & Industrial Engineering*, 55(2), 465–474.
33. **Wäscher, G. & Gau, T. (1996).** Heuristics for the integer one-dimensional cutting stock problem: A computational study. *OR Spektrum*, 18(3), 131–144.

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 una 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.

Marcela Quiroz C. Nació en México en 1984. En el 2009 recibió el grado de maestra en ciencias de la computación del Instituto Tecnológico de Ciudad Madero, México. Actualmente es una estudiante de doctorado en el Instituto Tecnológico de Ciudad Madero, México. Sus áreas de interés incluyen técnicas de optimización, aprendizaje automático, algoritmia experimental y explicaciones del desempeño algorítmico.

Adriana C. F. Alvim. Profesora de tiempo completo en la Universidade Federal do Estado do Rio de Janeiro (UNIRIO), Brasil. Recibió el grado de doctora en ciencias de la computación de la Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) en el 2003. Sus intereses de investigación incluyen heurísticas y metaheurísticas para problema de optimización combinatoria duros.

Héctor J. Fraire Huacuja. En 1988 recibió el grado de maestro en ciencias de la Información de la Universidad Autónoma de Nuevo León, México, y el grado de doctor en ciencias de la computación del Centro Nacional de Investigación y Desarrollo Tecnológico, México en el 2005. Actualmente es un profesor de tiempo completo en el Instituto Tecnológico de Ciudad Madero, México. Sus áreas de interés incluyen optimización heurística y aprendizaje automático.

Claudia Gómez S. En el 2000 recibió el grado de maestra en ciencias de la computación del Instituto Tecnológico de León, México y en el 2010 recibió el grado de doctora en ciencias de la computación del Instituto Politécnico Nacional, México. Actualmente es una profesora de tiempo completo en el Instituto Tecnológico de Ciudad Madero, México. Sus áreas de interés incluyen redes complejas, optimización y agentes inteligentes.

José Torres-Jiménez. Estudió ingeniería electrónica en el Instituto Tecnológico de Nuevo Laredo y recibió los grados de maestro en ciencias de la computación y doctor en ciencias de la computación del Instituto Tecnológico y de Estudios Superiores de Monterrey, México. Actualmente es un profesor de tiempo completo en el Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, México. Sus áreas de interés incluyen optimización combinatoria, sistemas de bases de datos, ingeniería de software e inteligencia artificial.

Artículo recibido el 03/01/2011; aceptado el 08/11/2011.