EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

"POR MI PATRIA Y POR MI BIEN"

TESIS

UN ESTUDIO FORMAL DE HEURÍSTICAS PARA EL PROBLEMA DE EMPACADO DE OBJETOS DE UNA DIMENSIÓN

Que para obtener el Grado de
Maestro en Ciencias de la Computación

Presenta
Ing. Jessica Elena González San Martín
G13070673

Director de Tesis
Dra. Laura Cruz Reyes

Co-director de Tesis
Dr. Bernabé Dorronsoro

Cd. Madero, Tamaulipas Mayo, 2021

Instituto Tecnológico de Ciudad Madero
Subdirección Académica
División de Estudios de Posgrado e Investigación

Cd. Madero, Tam. 18 de mayo de 2021

OFICIO No. : U.017/21
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN DE TESIS

C. JESSICA ELENA GÓNZALEZ SAN MARTÍN
No. DE CONTROL G13070673
P R E S E N T E

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su Examen de Grado de Maestría en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

"UN ESTUDIO FORMAL DE HEURISTICAS PARA EL PROBLEMA DE EMPACADO DE OBJETOS DE UNA DIMENSIÓN "

El Jurado está integrado por los siguientes catedráticos:

PRESIDENTE:	DR. HÉCTOR JOAQUÍN FRAIRE HUACUJA
SECRETARIO:	DR. NELSON RANGEL VALDEZ
VOCAL:	DRA. LAURA CRUZ REYES
SUPLENTE :	DRA. CLAUDIA GUADALUPE GÓMEZ SANTILLÁN
DIRECTOR DE TESIS:	DRA. LAURA CRUZ REYES
CO-DIRECTOR DE TESIS:	DR. BERNABE DORRONSORO

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con u sted el logrode esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experienciae inteligencia en beneficio de M éxico.

A T E N T A M E N T E
Excelencia en Educación Tecnológica ®
"Por mi patria y por mi bien"®

MARCO ANTONIO CORONEL GARCÍA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

EDUCACIÓN
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

c.c.p.- Archivo
MACG 'mdcoa'

Declaración de originalidad

Yo, Jessica Elena González San Martín, en mi calidad de autor manifiesto que este documento de tesis es producto original de mi trabajo y que no infringe derechos de terceros, tales como derechos de publicación, derechos de autor, patentes y similares. Por lo tanto, la obra es de mi exclusiva autoría y soy titular de los derechos que surgen de la misma.

Asimismo, declaro que en las citas textuales que he incluido y en los resúmenes que he realizado de publicaciones ajenas, indico explícitamente los datos de los autores y las publicaciones.

En caso de presentarse cualquier reclamación o acción por parte de un tercero en cuanto a los derechos de autor sobre la obra en cuestión, asumiré toda la responsabilidad y relevo de ésta a mi director de tesis, así como al Tecnológico Nacional de México, Instituto Tecnológico de Ciudad Madero y a sus respectivas autoridades.

Cd. Madero, Tamaulipas. Mayo 2021.

ISC. Jessica Elena González San Martín

Resumen

El problema de Bin Packing de una dimensión (1D-BPP) es un problema clásico de optimización que es conocido por su aplicabilidad y complejidad, el cual pertenece a una clase especial de problemas denominada NP-duro. A través de los años el esfuerzo de muchos investigadores ha concretado en una variedad de algoritmos que han mostrado un desempeño satisfactorio, sin embargo, en la actualidad no existe un algoritmo heurístico capaz de encontrar la solución óptima para todas las posibles instancias de un problema de este tipo a pesar de los esfuerzos de la comunidad científica.

En este trabajo se presenta un estudio de los algoritmos más relevantes y un análisis comparativo de los principales trabajos relacionados con 1D-BPP con el fin de identificar componentes y/o estrategias que muestran un impacto positivo en el desempeño de estos.

Se propone una nueva versión de la metahuerística Grouping Genetic Algorithm with Controlled Gene-Transmission (GGA-CGT) la cuál denominamos GGA-CGT/D. Para este algoritmo se diseñaron tres estrategias: un método de selección del límite inferior más adecuado al problema que se resuelve, un método de reducción del problema y un método de diversificación de soluciones los cuales ayudaron a mejorar el desempeño del algoritmo original.

Los resultados obtenidos de un extenso estudio computacional confirman que GGA-CGT/D logra superar el desempeño de los mejores algoritmos del estado del arte. Como caso de estudio se seleccionó el conjunto de instancias más aceptado para comparar algoritmos competitivos. Este conjunto es parte de BPPLIB e incluye a la clase Hard28, que parece tener el mayor grado de dificultad para los algoritmos BPP. También se aborda un nuevo conjunto de instancias retadoras de grandes dimensiones llamado BPP$v_{u_}c$. El algoritmo propuesto puede resolver óptimamente todas las instancias seleccionadas de BBPLIB, y un gran número de BPP$v_{u_}c$, resolviendo en total 2309 instancias difíciles, abiertas hasta el momento, de las cuales 49 no habían sido resueltas por ninguno de los algoritmos seleccionados como caso de estudio.

Abstract

The one-dimensional Bin Packing problem (1D-BPP) is a classical optimization problem that is known for its applicability and complexity, which belongs to a special class of problems called NP-hard. Through the years the efforts of many researchers have resulted in a variety of algorithms that have shown satisfactory performance, however, at present there is no heuristic algorithm capable of finding the optimal solution for all possible instances of a problem of this type despite the efforts of the scientific community.

This paper presents a study of the most relevant algorithms and a comparative analysis of the main works related to the 1D-BPP to identify components and/or strategies that show a positive impact on their performance.

A new version of the Grouping Genetic Algorithm with Controlled Gene-Transmission (GGA-CGT) heuristic is proposed, which we call GGA-CGT / D. For this algorithm, three strategies were designed: a method of selecting the lower limit more appropriate to the problem being solved, a method of reducing the problem and a method of diversifying solutions which helped to improve the performance of the original algorithm.

The results obtained from an extensive computational study confirm that GGA-CGT / D manages to outperform the best algorithms in the state of the art. As a case study, the most accepted set of instances was selected to compare competitive algorithms. This set is part of BPPLIB and includes the Hard28 class, which appears to have the highest degree of difficulty for BPP algorithms. A new set of large challenging instances called BPP$v_{u_}c$ is also addressed. The proposed algorithm can optimally solve all the selected instances of BBPLIB, and a large number of BPP$v_{u_}c$, solving a total of 2309 difficult instances, open so far, of which 49 had not been solved by any of the algorithms selected as a case study.

Agradecimientos

Quiero agradecer primeramente a Dios por este logro, por permitirme llegar hasta donde estoy, por el camino recorrido y los momentos vividos. Por nunca soltarme de su mano en los momentos más difíciles.

Agradezco a mis padres por sus sabios consejos y su confianza, por brindarme su apoyo incondicional en todo momento y ante toda situación.

A mi directora de tesis, Dra. Laura Cruz Reyes por darme la oportunidad de trabajar en su equipo, ya que con sus conocimientos, experiencia, paciencia y motivación ha logrado en mí que pueda terminar mis estudios con éxito**.**

Doy gracias también a todos los doctores que formaron parte de mi comité ya que con sus recomendaciones se lograron alcanzar buenos resultados, los cuales también son suyos.

Tabla de contenido

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El problema de empacado de objetos de una dimensión, mejor conocido como One-dimensional Bin Packing (1D-BPP) es definido como un problema que consiste en almacenar objetos de diferentes tamaños, o pesos, en el menor número de contenedores de tamaño fijo (Quiroz, 2014; Falkenauer, 1996). Es un problema clásico de optimización combinatoria NP-duro (Garey&Johnson, 1979) y los métodos más comúnmente utilizados para su solución son a través de metaheurísticas que incluyen estrategias inteligentes ya que la complejidad del problema hace computacionalmente difícil su solución a través de algoritmos exactos, por este motivo es considerado intratable ya que demanda una gran cantidad de recursos para su solución exacta.

El 1D-BPP, que por simplicidad en lo sucesimo se referirá como BPP o sólo Bin Packing, es muy utilizado para tratar problemas reales en la industria, algunos ejemplos son la planificación de carga de camiones, consolidación de carga en centros de datos, administración de servicios en la nube, entre otros.

Se ha conservado como un problema de estudio vigente debido a las diversas aplicaciones que ofrece, por este motivo, hoy en día en el estado del arte existen diversos algoritmos heurísticos para la resolución del problema, para este trabajo se seleccionarán aquellos que muestren los mejores resultados como caso de estudio.

1.1 Antecedentes

Este proyecto se deriva de trabajos anteriormente realizados que están relacionados con la investigación y desarrollo de algoritmos para resolver el problema de Bin Packing. Tales trabajos son:

a) Bin Packing and Related Problems General Arc-Flow Formulation with Graph Compression (Brandao&Pedroso, 2013).

b) A Grouping Genetic Algorithm with Controlled Gene Transmission for the Bin Packing Problem (Quiroz, 2014).
c) Consistent Neighborhood Search for One-Dimensional Bin Packing and Two-Dimensional Vector Packing (Buljubašić&Vasquez, 2016).
d) Average-weight-controlled bin-oriented heuristics for the one-dimensional bin-packing problem (Fleszar&Charalambous, 2011).
e) Hybrid Improvement Heuristic for the Bin Packing Problem (Alvim, et. al, 2004).

Estos algoritmos muestran resultados aceptables, sin embargo, existen ciertos tipos de instancias de Bin Packing para las que se ha demostrado que las soluciones obtenidas por estos algoritmos están muy alejadas de las soluciones óptimas.

A la fecha no existe un algoritmo que sea superior a todos los demás ante los diferentes tipos de instancias disponibles para este problema, por lo tanto, el objetivo de este trabajo es realizar un estudio teórico y experimental de los mejores algoritmos propuestos en el estado del arte y desarrollar métodos basados en las estrategias que cada uno de ellos utiliza.

1.2 Descripción del problema

En esta tesis se aborda el problema de empacado de objetos en contenedores de una dimensión. La estrategia de solución se soporta en el estudio del comportamiento algorítmico al resolver un conjunto instancias del problema. Este análisis se puede describir de la siguiente manera:

Dados:

1) un algoritmo metahurístico A para resolver un problema P,
2) un conjunto de instancias $I = \{i_1, i_2, \ldots, i_n\}$ del problema P.
3) un conjunto de indicadores $F = \{ f_1, f_2, \ldots, f_m \}$ que caractericen factores del problema y del algoritmo que definan el proceso de optimización algorítmico.

Se busca:

1) un conjunto de relaciones $R = \{r_1, r_2, \ldots, r_p\}$, para $1 \leq p \leq m$, tal que cada r_i establece una relación causal entre los elementos de G_i, $G_i \subseteq F$ que incluye factores que impactan el desempeño algorítmico,
2) aplicar el conocimiento adquirido a través de relaciones causales *R*, a explicar el desempeño de algoritmos, con la posibilidad de mejorar su diseño con mejores estrategias de solución, la búsqueda de relaciones se puede complementar con una revisión profunda del estado del arte.

En otras palabras, se busca caracterizar el proceso de optimización de un algoritmo que resuelve 1-DBPP, de manera que permita explicar por qué el algoritmo objeto de estudio sigue cierto comportamiento al resolver un problema de optimización y aplicar el conocimiento adquirido en: 1) modificar al menos una de las estrategias que los componen, 2) comparar, con soporte estadístico, los algoritmos originales y actualizados y 3) sentar las bases para elaborar un artículo panorámico (survey) que muestre una visión global de la investigación sobre algoritmos metaheurísticos aplicados a 1D-BPP y destaque futuras líneas de investigación.

1.3 objetivo de la tesis

Objetivo general

El objetivo principal de este trabajo es revisar a profundidad, de manera teórica y experimental, el estado del arte sobre la solución del problema de empacado de objetos en contenedores de una dimensión con algoritmos metaheurísticos.

Objetivos específicos

1) Analizar el estado del arte de algoritmos que resuelven el empacado mediante el método de revisión sistemática.
2) Clasificar, seleccionar e implementar algoritmos metaheurísticos con base en el desempeño reportado y en su representatividad.
3) Desarrollar nuevas versiones de los algoritmos seleccionados, modificando al menos una de las estrategias que los componen.
4) Comparar, con soporte estadístico, los algoritmos originales y actualizados.
5) Sentar las bases para elaborar una publicación (survey) que muestre una visión global de la investigación sobre algoritmos meaheurísticos aplicados al empacado de una dimensión y destacar futuras líneas de investigación.

1.4 Justificación

Con el transcurso del tiempo los investigadores han ido desarrollando algoritmos más sofisticados para solucionar problemas de optimización, dentro de ellos podemos encontrar el problema de Bin Packing que a través de los años se ha conservado como un problema de estudio vigente debido a las diversas aplicaciones que ofrece, ya que un problema con *n* dimensiones puede ser modelado como un problema de una sola dimensión.

Tras diversos estudios experimentales se han determinado que algunos algoritmos funcionan mejor que otros, sin embargo, existen instancias difíciles que no han sido resueltas. Hay algoritmos que resuelven instancias sencillas y algunos otros se enfocan en la resolución de instancias difíciles. Por este motivo se han combinado las estrategias de estos algoritmos en uno sólo para intentar resolver todas las instancias obteniendo resultados no favorables o peores que los algoritmos que resuelven una parte de las instancias existentes.

Debido a que los retos de alta dimensión demandan algoritmos con estas características (resuelvan la mayor parte de las instancias incluyendo las difíciles), este trabajo propone estudiar los algoritmos más destacados para la solución de 1D-BPP que se encuentran en el estado del arte, así como las diversas estrategias que cada uno de ellos utiliza y realizar una comparación con soporte estadístico de sus desempeños para destacar futuras líneas de investigación.

1.5 Alcances y Limitaciones

1) El estudio se aplica sólo al problema de empacado de objetos de contendores de una dimensión y algoritmos de solución del estado del arte relevantes para este problema.
2) Sólo se trabajan con instancias de BPP de una dimensión reconocidas como estándar en la literatura y las propuestas en (Carmona-Arroyo, et. al., 2021) por el grupo de investigación al que se pertenece.
3) Sólo se usan los factores de desempeño descritos en las tesis de maestría y doctorado de la Dra. Marcela Quiroz (Quiroz, 2014).

1.6 Organización del documento

El documento está organizado de la siguiente manera: En esta sección (capítulo 1) se describe y se realiza una explicación general del proyecto de tesis que se presenta, de tal manera que el lector pueda generarse una idea del contenido.

En el capítulo dos se mencionan los principales conceptos que serán necesarios para entender secciones posteriores como problemas de agrupación, el problema de Bin Packing, los tipos de algoritmos que son utilizados para resolver este problema (exactos y aproximados) y una descripción general sobre los cinco algoritmos seleccionados como caso de estudio.

En el capítulo tres se realiza un análisis comparativo sobre los algoritmos desarrollados para 1D-BPP en la última década, con el fin de comparar las técnicas y estrategias que utilizan, también se presenta un contraste entro los algoritmos seleccionados y la propuesta en el presente trabajo con el fin de puntualizar descriptivamente las aportaciones realizadas y las diferencias.

En el capítulo cuatro se describe de manera detallada el desarrollo de los métodos seleccionados para lograr los objetivos planteados de este trabajo.

En el capítulo cinco se observan las pruebas realizadas mediante las metodologías seleccionadas; los resultados alcanzados con los métodos implementados primero de manera independiente y después en conjunto, así como una descripción comparativa entre el trabajo propuesto y los algoritmos seleccionados como caso de estudio.

En el capítulo 6 que es el último capítulo se llegan a las conclusiones alcanzadas mediante el desarrollo de la presente tesis.

Capítulo 2

Marco Teórico

En esta sección se describen los principales fundamentos para comprender el funcionamiento y el desarrollo de las actividades descritas en esta tesis, estos conceptos están principalmente relacionados con el problema de Bin Packing y los problemas de agrupación.

Se abordan las diferentes clases de algoritmos que se pueden utilizar para resolver este tipo de problemas (exactos, heurísticos y metaheurísticos) y se presentan de manera general cinco de los algoritmos para 1D-BPP más destacados en el estado del arte.

2.1 Problemas de Agrupación

(Falkenauer,1994) describe a los problemas de agrupación como aquellos que consisten en dividir un conjunto U de objetos en una colección de subconjuntos mutuamente disjuntos U_i de U de modo que:

$$\cup U_i = U \tag{2.1}$$

y

$$U_i \cap U_j = \emptyset, i, \neq j \tag{2.2}$$

En 2.1. El subconjunto U_i pertenece al conjunto U.

En 2.2. El subconjunto U_i se intercepta con el subconjunto U_j y es igual a conjunto vacío, *i* y debe ser diferente de *j*.

Este tipo de problemas tienen el objetivo de agrupar a los miembros de un conjunto U en uno o más grupos de objetos con cada objeto en exactamente un grupo, es decir, para encontrar una agrupación de esos objetos.

En la mayoría de los problemas de este tipo, no se permiten todas las agrupaciones posibles ya que una solución al problema debe cumplir con varias restricciones; de lo contrario, la solución no es válida. Esto quiere decir que un

objeto no se puede agrupar con todos los subconjuntos posibles de los objetos restantes.

El objetivo de la agrupación es optimizar una función de costo definida sobre el conjunto de todas las agrupaciones válidas (Falkenauer, 1994).

Algunos de los ejemplos más conocidos de problemas de agrupación son:

Tabla 2.1: Problemas de agrupación más conocidos (Falkenauer, 1994).

Problema	Restricción	función de costo
Bin Packing	Suma de los tamaños de los objetos en cualquier grupo < C	Número de grupos
Workshop Layouting	Número de máquinas en cualquier grupo < C	Tráfico total entre celdas
Graph Coloring	Nodos no conectados en cualquier grupo	Número de grupos

2.2 Problema de empacado de objetos en contenedores

El problema de empacado de objetos en contenedores de una dimensión, mejor conocido como One-dimensional Bin Packing (1D-BPP) es definido como un problema que consiste en almacenar objetos de diferentes tamaños, o pesos, en el menor número de contenedores de tamaño fijo (Quiroz, 2014; Falkenauer, 1996). Éste es un problema clásico de optimización combinatoria NP-duro, considerado intratable pues demanda una gran cantidad de recursos para su solución exacta (Garey&Johnson, 1979).

Dado un conjunto $N = \{1,\ldots,n\}$ de objetos a distribuir en contenedores del mismo tamaño (capacidad), sea

c = capacidad de cada contenedor

w_i = peso del objeto i, tal que $0 < w_i \leq c$ para $1 \leq i \leq n$

1D-BPP consiste en asignar cada objeto a un contenedor de tal forma que la suma de los pesos de los objetos en cada contenedor no exceda c y el número de contenedores m utilizado sea mínimo (Martello&Toth, 1990). Se busca encontrar el menor número de subconjuntos B_j , para $1 \leq j \leq m$, de una partición del conjunto N

$$\bigcup_{j=1}^{m} B_j = N \qquad (2.3)$$

Tal que

$$\sum_{\forall i \in B_j} w_i \leq c \qquad 1 \leq j \leq m, i \in N = \{1, \dots, n\} \qquad (2.4)$$

En 2.3. Representa la unión de $j = 1$ hasta m cuando el subconjunto $B_j = N$

En 2.4. Es la sumatoria para toda i que pertenezca al subconjunto B_j cuando el peso w_i sea menor a la capacidad c.

Los métodos más comúnmente utilizados para la solución del problema de Bin Packing son a través de procedimientos metaheurísticos que incluyan estrategias inteligentes ya que la complejidad del problema hace computacionalmente difícil su solución a través de algoritmos exactos.

2.3 Algoritmos Exactos

Para obtener la solución exacta (óptima) de un problema de optimización se realiza una búsqueda enumerativa dentro del conjunto de todas las soluciones potenciales. En los métodos de enumeración total (algoritmos de fuerza bruta) se evalúa cada solución factible, mientras que en los métodos de enumeración parcial se introducen cotas, pruebas o alguna otra técnica para descartar soluciones sin conocerlas explícitamente. El costo computacional de estos algoritmos suele ser prohibitivo, pero debido a la reducción significativa del tiempo de ejecución, usualmente se utiliza el término algoritmo exacto para hacer referencia a los algoritmos de enumeración parcial (Cruz, 1999). Estos algoritmos como ya se mencionó, hacen una búsqueda exhaustiva en el espacio de soluciones y garantizan que al terminar entregan la solución óptima.

2.4 Algoritmos Aproximados

Un algoritmo aproximado es aquel que es utilizado para encontrar, tal como su nombre lo describe, una solución aproximada para un problema de

optimización. Dentro de estos algoritmos podemos encontrar algoritmos deterministas y no deterministas. A continuación, se describirán algunos de los algoritmos que entran en estas clasificaciones y que se han utilizado para abordar el problema de empacado de objetos.

2.4.1 Deterministas y No Deterministas

Un algoritmo determinista es un algoritmo que se vuelve completamente predictivo si sus entradas son conocidas. Estos algoritmos han sido de los más estudiados a través de la historia y debido a esto suelen ser los más utilizados por su practicidad.

A diferencia de los algoritmos deterministas antes mencionados, los no deterministas conociendo sus entradas o teniendo una misma entrada, ofrecen muchos posibles resultados y por lo tanto no se obtiene una solución única. No se puede saber de antemano cuál será el resultado de su ejecución.

2.4.2 Heurísticos y Metaheurísticos

Las heurísticas son procedimientos simples, a menudo basados en el sentido común, que se supone que obtendrán una buena solución (no necesariamente óptima) a problemas difíciles de un modo sencillo y rápido. Los métodos heurísticos tienen su principal limitación en su incapacidad para escapar de óptimos locales. Esto se debe, fundamentalmente, a que estos algoritmos no utilizan ningún mecanismo que les permita proseguir la búsqueda del óptimo en el caso de quedar atrapados en un óptimo local. Para solventar este problema, se introducen otros algoritmos de búsqueda más inteligentes (denominados metaheurísticas) que evitan, en la medida de lo posible, este problema (Duarte, et. al, 2007).

(Sevaux, et. al, 2010) describe a una *metaheurística* como un marco algorítmico independiente del problema de alto nivel que proporciona un conjunto de pautas o estrategias para desarrollar algoritmos de optimización heurística. Ejemplos notables de metaheurísticas incluyen algoritmos genéticos/evolutivos, búsqueda tabú, recocido simulado y optimización de colonias de hormigas, aunque existen muchos más. Una implementación específica de un problema de

un algoritmo de optimización heurística de acuerdo con las pautas expresadas en un marco metaheurístico también se denomina metaheurística.
Mientras que (Duarte, et. al, 2007) define a las *metaheurísticas* como una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria en los que los heurísticos clásicos no son efectivos. Las metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos.

2.4.3 Evaluación de algoritmos

(García, et. al, 2007) menciona que, debido al conocimiento parcial de un problema y la necesidad de disponer de algoritmos para su resolución, se plantea la cuestión de decidir cuando un algoritmo es mejor que otro. En el caso del uso de metaheurísticas o algoritmos evolutivos esto lo debemos hacer atendiendo a criterios de eficiencia y/o eficacia. Esta cuestión ha dado lugar al creciente interés en el análisis de experimentos en el campo de la inteligencia computacional. Este interés ha traído el uso de inferencia estadística en el análisis de resultados empíricos obtenidos por los algoritmos. La estadística inferencial muestra qué tan bien una muestra de resultados apoya una determinada hipótesis y si las conclusiones obtenidas pueden generalizarse más allá de lo que se probó.

Hay dos tipos principales de prueba estadística en la literatura: pruebas paramétricas y pruebas no paramétricas. La decisión de utilizar la primera o segunda puede depender de las propiedades de la muestra de resultados a analizar (García, et. al, 2010). Una prueba estadística paramétrica supone que los datos provienen de un tipo de distribución de probabilidad y hace inferencias sobre los parámetros de la distribución. Las pruebas no paramétricas no deben ajustarse a ninguna distribución, pueden por tanto aplicarse incluso aunque no se cumplan las condiciones de validez paramétricas. Por otro lado, es necesaria una distinción entre pruebas de comparación múltiple y por pares. Los primeros deben usarse cuando se comparan más de dos métodos y los segundos son procedimientos válidos para comparar dos algoritmos. Para este trabajo se seleccionó una prueba de comparación múltiple; la prueba de Friedman, que es un análogo no paramétrico del análisis de varianza paramétrico bidireccional. El objetivo de esta prueba es determinar si podemos concluir a partir de una muestra de resultados que existe una diferencia entre los efectos del tratamiento

(Friedman, 1937). Y se seleccionó también una prueba por pares; Wilcoxon: Prueba de los rangos con signo, es una prueba no paramétrica de comparación de dos muestras relacionadas y por lo tanto no necesita una distribución específica. Usa más bien el nivel ordinal de la variable dependiente. Se utiliza para comparar dos mediciones relacionadas y determinar si la diferencia entre ellas se debe al azar o no (en este último caso, que la diferencia sea estadísticamente significativa) (Wilcoxon, 1945).

Las pruebas estadísticas presentadas en esta tesis (Sección 5.5) fueron realizadas en una plataforma web llamada STAC, a través de esta plataforma se pueden verificar los resultados obtenidos de los algoritmos aplicando las pruebas estadísticas a los experimentos, que, entre otros usos, apoyan la toma de decisiones como la elección del algoritmo más adecuado (Rodríguez, et. al., 2015). En la figura 2.1 se presenta el proceso asistente para estimar automáticamente la prueba estadística mejor ajustada para los datos proporcionados por el usuario. El proceso de decisión tiene en cuenta los siguientes datos: el número de grupos disponibles (*k*), número de muestras por grupo (*n*), emparejamiento entre grupos, normalidad de cada grupo (probado mediante una prueba de Shapiro-Wilks con alfa 0.1) y homocedasticidad entre grupos (probada mediante la prueba de Levene con alfa 0.1).

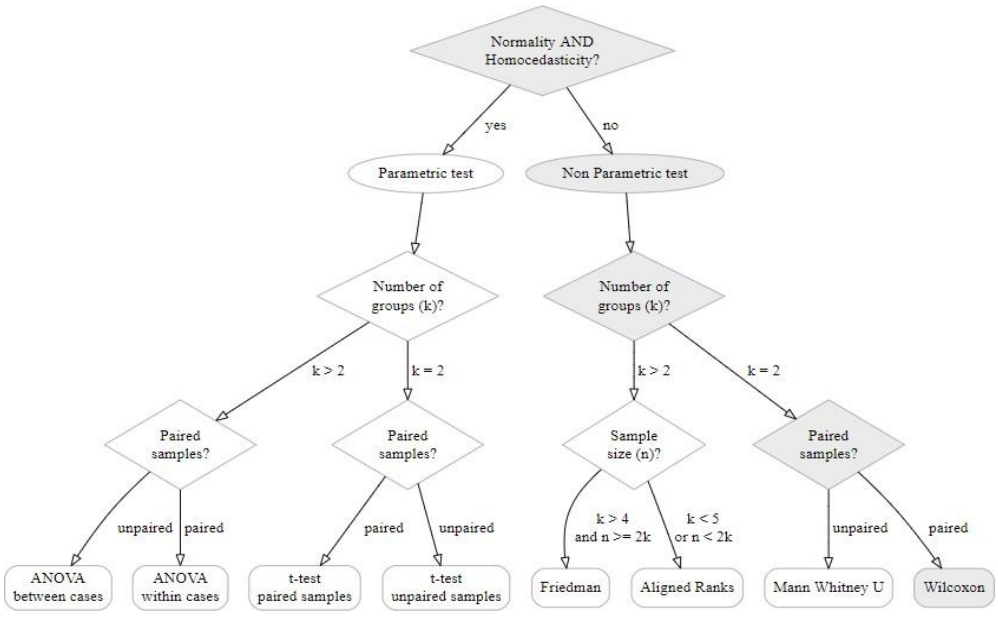

Figura 2.1. Proceso de decisión de pruebas estadísticas en STAC.

2.5 Instancias para Bin Packing

En la literatura existen diversas instancias para el problema de Bin Packing que son utilizadas por los diferentes investigadores para comparar las cualidades de sus estrategias contra las de otros algoritmos del estado del arte. Estas instancias conforman un conjunto de 1615 casos estándar reconocidos por la comunidad científica, sin embargo, con el transcurso de los años y los trabajos realizados se han propuesto otros tres diferentes grupos de instancias, sumando un poco más de 6000 casos de prueba. Dichas instancias pueden ser encontradas en sitios de Internet reconocidos (Unibo).

2.5.1 Instancias BPPLIB*

Para este trabajo se seleccionó un subconjunto de 1615 instancias como casos de prueba, las cuales denominamos BPPLIB*, debido a que es una porción del del conjunto BPPLIB (Delorme, et. al., 2018). Las instancias consideradas se han dividido en cuatro grupos, tomando en cuenta su origen y el sitio del que fueron obtenidas. El primer grupo consiste en dos conjuntos de instancias de prueba propuestas por (Falkenauer, 1996).

1) **Uniform** (Beasley, 1990): 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 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. c

2) **Triplets** (Beasley, 1990): 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, 1997). 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.

1) **Data Set 1** (Klein, 2009): Construidas de forma similar que algunas instancias propuestas por (Martello&Toth, 1990) que resultaron difíciles. Son un conjunto de 720 instancias denotadas con ***n-c-w*** por los datos que maneja: n = 50, 100, 200 y 500, capacidad del contenedor c = 100, 120, 150 y 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.

2) **Data Set 2** (Klein, 2009): Conjunto formado por 480 instancias con pesos generados con una distribución *uniforme*, denotadas por ***n-w-b-r***. 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 caja variara entre tres y nueve, se consideraron otros dos parámetros: $\bar{w}$ que representa el peso medio deseado $\bar{w} = c/3, c/5, c/7, c/9$, y una desviación δ = 20%, 50% y 90% que determina la variación máxima de un peso dado w en relación con $\bar{w}$. Por ejemplo, cuando $\bar{w} = C/5$ y δ = 50% los pesos de los objetos fueron generados de manera aleatoria con una distribución uniforme en el intervalo discreto [100,300]. Combinando las características anteriores se cuenta con 48 clases, cada una con 10 instancias.

3) **Data Set 3** (Klein, 2009): Conjunto formado por una única clase con diez instancias consideradas difíciles, mejor conocidas como ***hard***. Cada instancia posee n = 200 objetos, capacidad de contenedor c = 100000 y pesos distribuidos *uniformemente* entre 20000 y 35000. Los pesos de los objetos se encuentran ampliamente dispersos y el número de elementos por contenedor cae entre tres y cinco.

El tercer grupo incluye tres clases de instancias sugeridas por Schwerin, Wäscher y Gau. Las primeras dos clases was 1 y was 2 se han definido como ***ffd-duras*** (difíciles de resolver por la heurística FFD) (Schwerin, 1997;

Schwerin, 1999). La tercera clase también ha sido considerada difícil (Wäscher, 1996).

1) **Was 1**: Formado por 100 instancias de capacidad *c* = 1000. Cada instancia posee *n* = 100 objetos. Los pesos de los objetos varían entre 150 y 200.

2) **Was 2**: 100 instancias con una capacidad *c* = 1000, cada una con 120 objetos con pesos entre 150 y 200.

3) **Gau 1**: Conjunto de 17 instancias de características variadas. La capacidad de los contenedores es de 10000, el número de objetos *n* varía de 57 a 239 mientras que los pesos se encuentran distribuidos entre 2 y 7332.

El grupo cuatro contiene un conjunto de 28 instancias difíciles que también han sido consideradas en problemas de corte de una dimensión (one-dimensional Cutting Stock Problem). Los resultados óptimos son conocidos para todas las instancias (Belov, 2004).

1) **Hard28**: Son las 28 instancias más difíciles propuestas por (Schoenfield, 2002), dichas instancias no pudieron ser resueltas por algoritmos de corte ni por métodos de reducción. La capacidad del contenedor *c* es 1000 y los pesos de los objetos están uniformemente distribuidos entre 1 y 800. El número de objetos *n* varía entre 160 y 200.

2.5.2 Instancias BPPv_u_c

En esta sección, se presenta el nuevo conjunto de instancias denominadas BPP v_u_c (Carmona-Arroyo, et. al. 2021). Los pesos se generaron en el intervalo [0, $v_u c$] de modo que v_u corresponda al límite superior de los pesos, en relación con la capacidad del contenedor c.

Características de las instancias generadas

a) Las instancias fueron generadas con un óptimo conocido, dónde todos los contenedores en la solución quedan llenos al 100%.

b) Para cada contenedor los pesos de los objetos son generados de manera aleatoria con una distribución uniforme con valores entre 1 y una proporción de la capacidad del contenedor. Los pesos de los objetos están aproximadamente entre 1 y la cantidad indicada por el nombre de la instancia. La clase BPP C entre 1 y C, la clase BBP .5C entre 1 y 0.5C, la clase BPP .75C entre 1 y 0.75C.

Algoritmo de generación de instancias

a) Para cada contenedor, generar pesos de sus objetos con distribución uniforme, si no llenan de manera exacta, el peso del último objeto es la cantidad que falta para llenar el contenedor (complemento).
b) Después, juntar todos los objetos y desordenarlos mediante permutación aleatoria. Se recorre todo el vector de objetos, y objeto por objeto se busca aleatoriamente otro del arreglo con el cual intercambiar, no importa si ya fue intercambiado.

La complejidad del generador es *O(n)*. Esto quiere decir que su orden de complejidad es lineal.

2.6 Revisión sistemática

Una revisión sistemática consiste en una metodología científica específica, y es utilizada para conocer, evaluar e interpretar la información existente en un determinado campo de investigación (Keeke, 2007).

Esta metodología de investigación nació en el campo de la medicina en el que los profesionales empezaron a interesarse por los resultados de sus intervenciones, ya que, aunque había investigación y se disponía de una amplia gama de resultados, estos no estaban sistematizados y el beneficio de la acumulación del conocimiento era limitado (McGuire, 2001).

En la Figura 2.2 se muestra el proceso presentado por (Brereton, et. al., 2007) el cual está adaptado para la Ingeniería de Software y que debe llevarse a cabo para implementar una revisión sistemática.

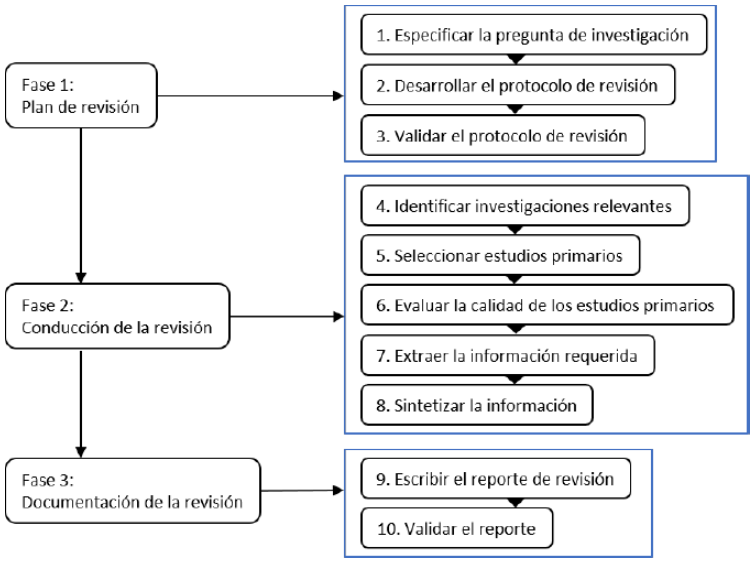

Figura 2.2. Proceso de revisión sistemática.

La Fase 1 da inicio con la definición de una(s) pregunta(s) a la(s) que se dará(n) respuesta como etapa final, después describe la lista de protocolos que van a guiar la búsqueda, así como los criterios de validación. Durante la Fase 2 se lleva a cabo una búsqueda guiada por los parámetros previamente establecidos con el objetivo de identificar los estudios primarios (documentos que serán revisados), realizar la selección de los más relevantes y que den respuesta a la pregunta de investigación; una vez identificados los estudios primarios, se procede a extraer los datos relevantes y que serán la base de la revisión. Finalmente, la Fase 3, consiste en la redacción y validación del reporte, el cual será considerado como un estudio secundario del tema tratado durante el mismo.

2.7 Algoritmos destacados para la solución del problema de Bin Packing

Con el transcurso de los años y el avance de la ciencia, se han ido desarrollando nuevas estrategias para la solución de problemas de optimización. Actualmente existen un sin número de algoritmos que abordan y buscan darle solución al problema de Bin Packing. A continuación, se presentan cinco de los algoritmos más relevantes para BPP ya que reportan los resultados más significativos de la literatura.

2.7.1 General Arc Flow Formulation with Graph Compression

Este trabajo fue propuesto por (Brandao&Pedroso, 2013), el cuál es un método exacto basado en una formulación de flujo de arco con restricciones laterales. Es utilizado para resolver problemas de Bin Packing y Cutting Stock. Incluye un algoritmo de compresión de gráficos que reduce el tamaño del gráfico subyacente sustancialmente sin debilitar el modelo. La formulación que utiliza es equivalente a la de (Gilmore&Gomory, 1963). Sin embargo, en lugar de utilizar la generación de columnas en un proceso iterativo, el método construye un gráfico, donde las rutas desde el nodo de origen al destino representan cada patrón de empaque válido.

2.7.2 Grouping Genetic Algorithm with Controlled Gene Transmission

El algoritmo Grouping Genetic Algorithm with Controlled Gene Transmission (GGA-CGT) desarrollado por (Quiroz, 2014), es una metaheurística de empacado inteligente que simplifica y mejora el empacado de los objetos, hace uso de un procedimiento de reacomodo que permite la exploración y explotación del espacio de búsqueda y un conjunto de operadores genéticos de agrupación que promueven la transmisión de los mejores genes en los cromosomas sin perder el equilibrio entre la presión selectiva y diversidad de la población. La transmisión de los mejores genes (los contenedores más llenos) se realiza por medio de un conjunto de operadores genéticos de agrupación, mientras que el proceso evolutivo es equilibrado por medio de una técnica de reproducción que controla la exploración del espacio de búsqueda y evita la convergencia prematura del algoritmo.

ALGORITMO GENÉTICO DE AGRUPACIÓN CON TRANSMISIÓN DE GENES CONTROLADA

Procedimiento GGA-CGT
1 Generar una población inicial con una adaptación y mejora de First Fit;
2 **mientras** *generación* < *max_gen* y Tamaño(*mejor_solución*) > *L*2
3 Seleccionar individuos para cruzar por medio de Selección_Controlada;
4 Aplicar cruzamiento a los individuos seleccionados;
5 Aplicar Reemplazo_Controlado para introducir las crías;
6 Seleccionar individuos y clonar soluciones élite por medio de Selección_Controlada;
7 Aplicar mutación a los mejores individuos;
8 Aplicar Reemplazo_Controlado para introducir los clones;
9 Actualizar la mejor solución;
10 **fin mientras;**
fin GGA-CGT

Algoritmo 1. Procedimiento general del algoritmo GGA-CGT.

2.7.3 Consistent Neighborhood Search for one-dimensional Bin Packing

El algoritmo Consistent Neighborhood Search for one-dimensional Bin Packing (CNS-BP) es una metaheurística propuesta por (Buljubašić&Vasquez, 2016) que consiste en realizar una búsqueda local para derivar una solución factible con un número dado de contenedores, *m*, comenzando desde *m = UB-1*, donde *UB* es un límite superior obtenido mediante el uso de una variante de la heurística clásica First Fit. Hace uso de una reducción para simplificar el problema y después aplica dos búsquedas locales: Búsqueda Tabú y Hill Climbing Descent. Este algoritmo muestra resultados exitosos en sus experimentaciones ya que obtiene soluciones iguales o mejores a los demás algoritmos en el estado del arte, incluso las instancias Hard28 que son consideradas una de las más difíciles dentro de BPP.

ALGORITMO CNS-BP

1 Remover los objetos pares (i,j) tal que $w_i + w_j = C$
2 Calcular el límite inferior LB
3 Barajear aleatoriamente el conjunto de elementos
4 $S \leftarrow$ Solución completa obtenida por First Fit
5 $m \leftarrow$ Número de contenedores en S
6 **mientras** m > LB **y** el tiempo límite no exceda **hacer**
7 $m \leftarrow m - 1$
8 Construir la solución parcial P con m-2 contenedores * Elimina 3 contenedores de S
9 $S' \leftarrow CNS(P)$ * Intentar encontrar la solución completa con m contenedores
10 **Si** la solución S' no está completa **entonces** termina
11 $S \leftarrow S'$
12 **retorna** S
13 **fin mientras**

Algoritmo 2. Procedimiento general del algoritmo CNS-BP.

2.7.4 Perturbation-SAWMBS

Es una metaheurística propuesta por (Fleszar&Charalambous, 2011) la cual se extiende del procedimiento Perturbation-MBS' (Fleszar&Hindi, 2002) que originalmente está compuesto por una versión modificada de la heurística MBS de (Gupta&Ho, 1999), una búsqueda de vecindad variable y límites inferiores. Esta nueva versión del algoritmo utiliza una estrategia para controlar el peso promedio de los objetos que son empacados en cada contenedor.

ALGORITMO Perturbation-SAWMBS

1 Ejecutar MTRP
2 Remover los objetos empacados por MTRP
3 Calcular el límite de relajación lineal $LB = [\frac{1}{c}\sum w_i]$
4 Empacar los elementos con First Fit Decreasing *Usando reducciones simples
5 Empacar los elementos con SAWB2F *Usando reducciones simples
6 Empacar los elementos con SAWMBS *Usando reducciones simples
7 Adoptar las mejores 3 soluciones
8 Ejecutar *k* iteraciones de Perturbation-SAWMBS *Ejecuta *k* veces de la línea 1-7

Algoritmo 3. Procedimiento general del algoritmo Perturbation-SAWMBS.

2.7.5 Hybrid Improvement Heuristic for the Bin Packing Problem

El algoritmo Hybrid Improvement Heuristic for the Bin Packing Problem (HI-BP) propuesto por (Alvim, et al., 2004), es una meheurística de mejora híbrida que cuenta con el uso de estrategias de límite inferior, retoman la estrategia dual utilizada por (Scholl et al., 1997), así como las técnicas de reducción de (Martello&Toth, 1990) e incluye un proceso de mejora que utiliza la búsqueda tabú.

ALGORITMO HI-BP

Procedimiento HI-BP
1 Calcular la solución $S_{FBD} = \{B_1, \ldots, B_{z(FBD)}\}$ usando el algoritmo BFD;
2 Dado $UB \leftarrow z(S_{FBD})$;
3 **Si** $L_1 = UB$ **entonces** regresa S_{FBD};
4 **Si** $L_2 = UB$ **entonces** regresa S_{FBD};
5 Calcular el límite superior $\boldsymbol{L_3}$ y dejar $S_{MTRP} = (B_1, \ldots, B_b)$ ser la solución parcial con b contenedores creados por el procedimiento de reducción MTRP;
6 **Si** $L_3 = b$ **entonces** regresa S_{MTRP};
7 **Si** $L_3 = UB$ **entonces** regresa S_{FBD};
8 Eliminar de N los objetos acomodados en la solución parcial S_{MTRP};
9 Actualizar los límites $L_3 \leftarrow L_3 - b$ y $UB \leftarrow UB - b$; introducir los clones;
10 Calcular el límite superior $L_*^{(20)}$y L_σy el conjunto $LB, nbins \leftarrow max\{L_3, L_*^{(20)}$y $L_\sigma\}$;
11 **Si** nbins = UB **entonces hacer** $S \leftarrow S_{FBD}$; **devolver** S; **fin**;
12 *Infactible* ← $Verdadero$;
13 **Mientras** $nbins < UB$ **hacer;**
14 **Para** *k* = 1,…,MaxTrials e *Infactible* ← $Verdadero$ **hacer**
15 Selecciona una heurística de H = {DB3FD, DBFD, DSSFD, LPT};
16 Construir una solución $S = \{B_1, \ldots, B_{z(FBD)}\}$ a DBP;
17 **Si** S es factible para BP **entonces hacer**
18 Dar *Infactible* ← $Falso$ y $UB \leftarrow z(S)$;
19 **Si no hacer**
20 $S \leftarrow redistribución(S)$
21 **Si** S es factible para BP **entonces hacer**
22 Dar *Infactible* ← $Falso$ y $UB \leftarrow z(S)$;
23 **Si no hacer**
24 $S \leftarrow Búsqueda\ Tabú(S)$
25 **Si** S es factible **entonces hacer**
26 Dar *Infactible* ← $Falso$ y $UB \leftarrow z(S)$;
27 **fin si**
28 **fin para**
29 **Si** $infactible = Verdadero$ **entonces** $nbins \leftarrow nbins + 1$;
30 **fin mientras**
31 **Si No** es infactible **entonces devuelve** $S \cup S_{MTRP}$;
32 **Si no** devolver $\boldsymbol{S_{BFD}}$;
Termina HI-BP

Algoritmo 4. Procedimiento general del algoritmo HI-BP

Capítulo 3

Estado del arte

En este capítulo se ofrece un análisis comparativo de los principales trabajos relacionados con el problema de Bin Packing en la última década, presentando las diferentes técnicas/estrategias que se han utilizado. Posteriormente, se presenta una breve revisión de las características principales que utilizan los mejores cinco algoritmos que se seleccionaron como caso de estudio y que fueron descritos en el capítulo anterior, realizando una comparación con el trabajo propuesto en esta tesis.

3.2 Comparación de estrategias de algoritmos para Bin Packing de la última década

A través de los años se ha mostrado un incremento considerable en el desarrollo de procedimientos para la solución de problemas de optimización en aplicaciones prácticas. En la actualidad existen diversos algoritmos para la resolución del problema de empacado de objetos de una dimensión, en esta sección se presenta una comparación empírica de los algoritmos que se encuentran en la literatura desde la actualidad a diez años atrás. El objetivo de este apartado es identificar cuáles componentes y/o técnicas fueron utilizadas para cada uno de los algoritmos y cuáles de estas aportan más al desempeño de los mismos.

Se seleccionaron diecisiete algoritmos de la literatura especializada y se analizaron sus trabajos con el fin de reportar los diferentes enfoques que utilizan, como: métodos matemáticos tradicionales, programación dinámica, heurísticas simples, métodos enumerativos, metaheurísticas, entre otros. En la tabla 3.2 se enlistan aquellas técnicas/estrategias más representativas de una muestra de siete de los diecisiete algoritmos seleccionados, la comparativa completa se encuentra en el anexo A.

Tabla 3.2 Comparación de estrategias de algoritmos para BPP en la última década.

Trabajo	Algoritmo	Año	Técnica/Estrategia
Hyper-Heuristic Classifier for 1D-BPP (Sim&Hart, 2012)	HHC-BP	2012	Hiper Heurística con módulo clasificador (KNN).
Efficient Hybrid Grouping Heuristics for the Bin Packing Problem (Cruz-Reyes, et. al., 2012)	HGGA-BP	2012	Algoritmo Genético Híbrido
Evolving Bin Packing Heuristic Using Micro-Differential Evolution with Indirect Representation (Sotelo-Figueroa, et. al., 2013)	μ*DE*	2013	Algoritmo de evolución diferencial y representación indirecta
A Memory-Integrated Artificial Bee Algorithm For 1D-BPP (Trugul, et.al. ,2014)	MEABC	2014	Algoritmo Colonia de Abejas Artificiales (ABC) y búsqueda local
Solving bin packing problem with a hybrid genetic algorithm for VM placement in cloud (Kaaouache&Bouamama, 2015)	HGBF_BP	2015	Algoritmo Genético Híbrido
A Novel Grouping Genetic Algorithm for the 1D-BPP on GPU (Czachórski, et. al., 2016)	1D-BPP-CUDA	2016	Algoritmo genético en GPU (Graphics Processing Unit) usando CUDA.
Cooperative parallel grouping genetic algorithm with controlled gene transmission for the 1D-BPP (Kucukyilma&Kiziloz, 2018)	IPGGA	2018	Algoritmo genético de agrupación paralela cooperativa

La comparación de estrategias de los trabajos del estado del arte enfocados a BPP en la última década reveló que, con el transcurso de los años se han desarrollado estrategias o técnicas más sofisticadas y/o novedosas que pueden aplicarse a los algoritmos con el fin de mejorar su desempeño, un ejemplo de esto son los modelos de isla, la programación paralela, redes neuronales, entre otros. Sin embargo, también existen estrategias que se han mantenido con el paso del tiempo y son fundamentales para abordar este tipo de problemas de optimización, como lo son los algoritmos evolutivos.

3.3 Contraste entre la propuesta y los algoritmos destacados para Bin Packing

Como se mencionó en la sección previa, existen diversos algoritmos para la resolución de 1D-BPP, no obstante, tras realizar una revisión sistemática, se seleccionaron 5 algoritmos como caso de estudio para este trabajo, ya que muestran los mejores resultados reportados en la literatura. La tabla 3.3 resume las técnicas principales que son utilizadas por estos algoritmos, incluyendo el trabajo propuesto, así como las instancias evaluadas y el número total de soluciones reportadas en cada uno de ellos.

Tabla 3.3: Contraste entre la propuesta y los algoritmos más destacados para BPP.

Algoritmo	**Estrategia Principal**	**Metaheurística**	**Técnicas**			**Instancias Evaluadas**		**Caracterización del desempeño**
			Reducción	**Límites**	**Diversificación**	**BPPLIB***	BPP$v_{u_}c$	
General Arc-Flow Formulation (Brandao&Pedroso, 2013)	flujo de arco y compresión de gráficos					1615		
CNS_BP (Buljubašić&Vasquez, 2016)	Búsqueda Local	✓	✓	4		1612		
Pert.-SAWMBS (Fleszar&Charalambous, 2011)	Búsqueda Local	✓	✓	1		1590		
HI_BP (Alvim, et. al., 2004)	Heurística Híbrida	✓	✓	3		1587		
GGA-CGT (Quiroz, 2014)	Algoritmo Genético	✓		1		1602		✓
Este trabajo **(GGA-CGT/D)**	Algoritmo Genético con diversificación	✓	✓	5	✓	1615	✓	✓
BPPLIB* (Uniform, Triplet, Data Set 1,2 y 3, Was 1 y 2, Gau 1, Hard28) **BPP$v_{u_}c$** (BPP.25, BPP.5, BPP.75, BPP1)								

En la tabla anterior, se puede observar que el trabajo propuesto está conformado de diferentes técnicas que lo ayudan a superar a los algoritmos heurísticos destacados, resolviendo a la optimalidad todo el subconjunto de instancias BPPLIB* y resolviendo un mayor número de instancias del conjunto BPP$v_{u_}c$ que los mejores trabajos de la literatura. El detalle de los resultados se puede encontrar en el Anexo B.

Capítulo 4

Metodología

En este capítulo, primero se presentan las métricas seleccionadas para la caracterización de instancias que se utilizan en este trabajo. También se realiza una descripción general de los métodos desarrollados para el rediseño del algoritmo GGA-CGT y posteriormente se presentan a detalle cada uno de los métodos implementados en la nueva versión del algoritmo.

4.1 Métricas seleccionadas para caracterización de instancias

Es sabido que, factores como el número de objetos, la tendencia central de sus pesos y su distribución, afectan el grado de dificultad de una instancia de BPP. De esta forma, diferentes autores han propuesto conjuntos de índices para la caracterización de BPP. Para modelar la estructura de una instancia de BPP, Cruz (Cruz, 2004) formuló un conjunto de índices de dificultad. Por otro lado, en el trabajo de Álvarez (Álvarez, 2006) se propone un conjunto de índices propósito general, basados en estadística descriptiva y Quiroz (Quiroz, 2014) propone un nuevo índice de caracterización llamado Gap. Para este trabajo se seleccionaron 3 indicadores, los cuáles son estudiados y utilizados en (Quiroz, 2014) para caracterizar una instancia mostrando buenos resultados, estos son; Gap, t, Rango y el estadístico de correlación, los cuales se describen en la tabla 4.1. Las pruebas realizadas con los indicadores se presentan en el anexo C.

Tabla 4.1: Indicadores seleccionados para este trabajo.

Autor	Ecuación	Descripción
Quiroz, 2014	$Gap\_óptimo = 1 - \frac{S}{mc}$	*Gap* determina cuánto espacio quedará libre en los contenedores de la solución óptima, donde *m* = número de contenedores en la solución óptima, *S* = suma de los pesos de todos los objetos y *c* = capacidad de los contendores.
Cruz, 2004	$t = \frac{\sum_{i=1}^{n} w_i / n}{c}$	*t* es el índice de capacidad ocupada por un objeto promedio, donde *wi* = *peso del objeto i*, *c* = capacidad del contenedor.
Álvarez, 2006	$R = max(w) - min(w)$	*R* es la amplitud del rango de distribución de pesos de los objetos, donde *w* = pesos de los objetos, *max()* = función que devuelve el peso máximo de todos los objetos, *min()* = función de devuelve el peso menor de todos los objetos.
Pearson, 1895	$C = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2}\sqrt{(y-\bar{y})^2}}$	La *correlación* es un índice que puede utilizarse para medir el grado de relación de dos variables.

4.2 Diseño de estrategias y análisis de su impacto en el desempeño

Con la revisión de la literatura, se identificaron algunas de las estrategias más representativas en los algoritmos seleccionados como caso de estudio. Estas estrategias se tomaron como ejemplo para implementarlas en uno de los algoritmos (GGA-CGT) y ver su impacto en el desempeño de manera aislada y después en conjunto. En la tabla 4.2 se enlistan estas estrategias y su descripción detallada. En la primera columna se encuentra la estrategia seleccionada y en la segunda columna su descripción y referencia al algoritmo que originalmente la utiliza.

Tabla 4.2. Estrategias representativas seleccionadas para su implementación en el algoritmo GGA-CGT (Quiroz, 2014).

Estrategia	Descripción
Método para calcular los límites inferiores	Del algoritmo CNS_BP (Buljubašić&Vasquez, 2016) se implementó el cálculo de algunos límites inferiores y se seleccionó el mejor para cada instancia. $L_1 = \lceil \frac{\sum_{i=1}^{n} w_i}{C} \rceil$ (Scholl,Klein&Jürgens, 1997), $L_*^{(p)}$ (Fakete&Schepers, 2001) con p = 10, L_2, L_3 (Martello&Toth, 1990) y L_P (Alvim, 2004).
Método de reducción de instancias para simplificar el problema	De cada instancia se removieron aquellos objetos pares (i, j) que llenan en su totalidad la capacidad del contenedor $(w_i + w_j = C)$. (Buljubašić&Vasquez, 2016)
Método de diversificación	Para incrementar la diversificación y disminuir la intensificación en el algoritmo, se diseñó un método de reinicio, el cual iterativamente genera nuevas soluciones iniciales que reemplazan a un porcentaje de la población cada cierto número de generaciones. Se consideró la propuesta de (Schiavinotto&Stützle, 2004).

Como se mencionó con anterioridad, cada una de las estrategias seleccionadas se desarrollaron en el algoritmo GGA-CGT. Cada método se implementó de manera aislada, esto quiere decir que, para cada método se creó una versión independiente del algoritmo, generando tres versiones diferentes (Límites, reducción y diversificación) para medir el impacto en el desempeño de cada método desarrollado. Después de realizar las experimentaciones correspondientes, se creó el algoritmo robusto, en donde se incluyen los tres métodos seleccionados. Las experimentaciones y resultados se detallan en el capítulo 5.

4.3 Rediseño del algoritmo GGA-CGT

En esta sección se presenta una comparación de diseño entre el algoritmo GGA-CGT original y la versión propuesta (GGA-CGT/D). También se describen de manera detallada cada uno de los métodos implementados.

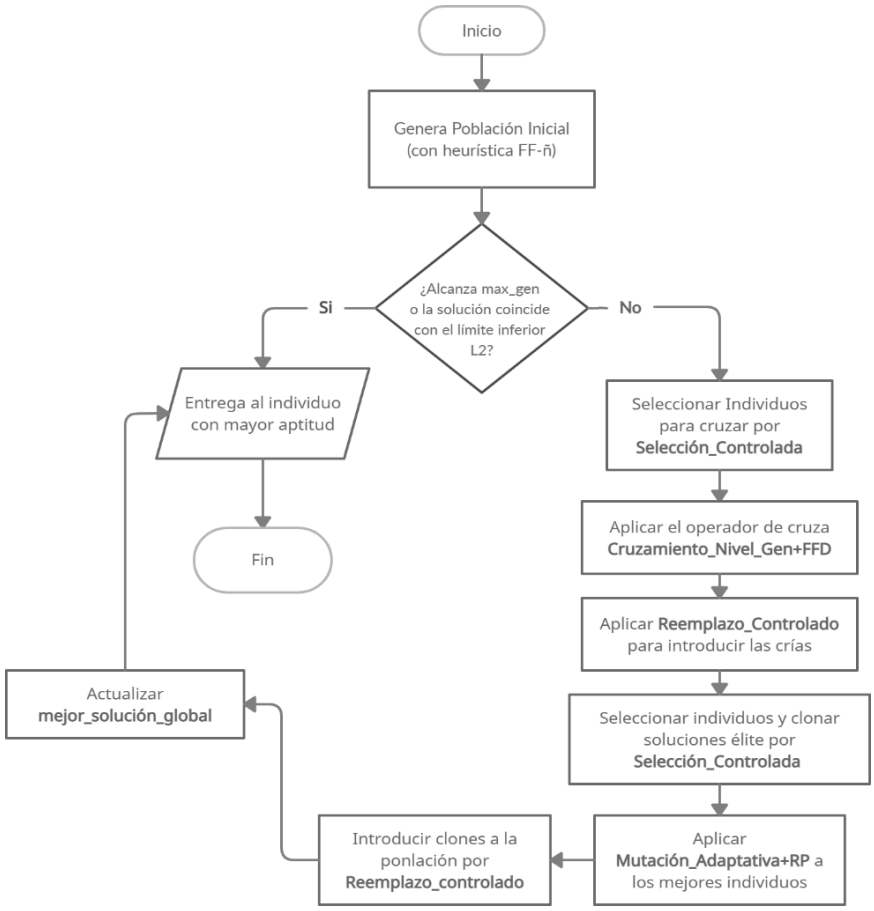

Figura 4.1. Estructura general del algoritmo GGA-CGT original.

La Figura 4.1 presenta el algoritmo GGA-CGT. El proceso inicia generando una población *P* de individuos con la heurística de empacado FF-ñ. Luego, durante un máximo de *max_gen* generaciones, algunos individuos de *P* son seleccionados para ser recombinados y mutados en dos fases. En la primera fase *nc* individuos son seleccionados con la estrategia *Selección_Controlada* para aplicarles el operador de cruza *Cruzamiento_Nivel_Gen+FFD*; en seguida, la estrategia *Reemplazo_Controlado* es utilizada para introducir la descendencia. En la segunda fase, de acuerdo con una *max_edad* predefinida, los individuos del grupo élite son clonados y los mejores nm individuos son seleccionados para aplicarles el operador *Mutación_Adaptativa+RP*; posteriormente, los clones

son introducidos a la población por medio de la estrategia *Reemplazo_Controlado*. El algoritmo se detendrá antes de alcanzar el número máximo de generaciones *max_gen* en caso de encontrar una solución cuyo tamaño coincida con el límite inferior L2 de (Martell&Toth, 1990). El resultado final del algoritmo es el individuo con la mayor aptitud de todo el proceso evolutivo.

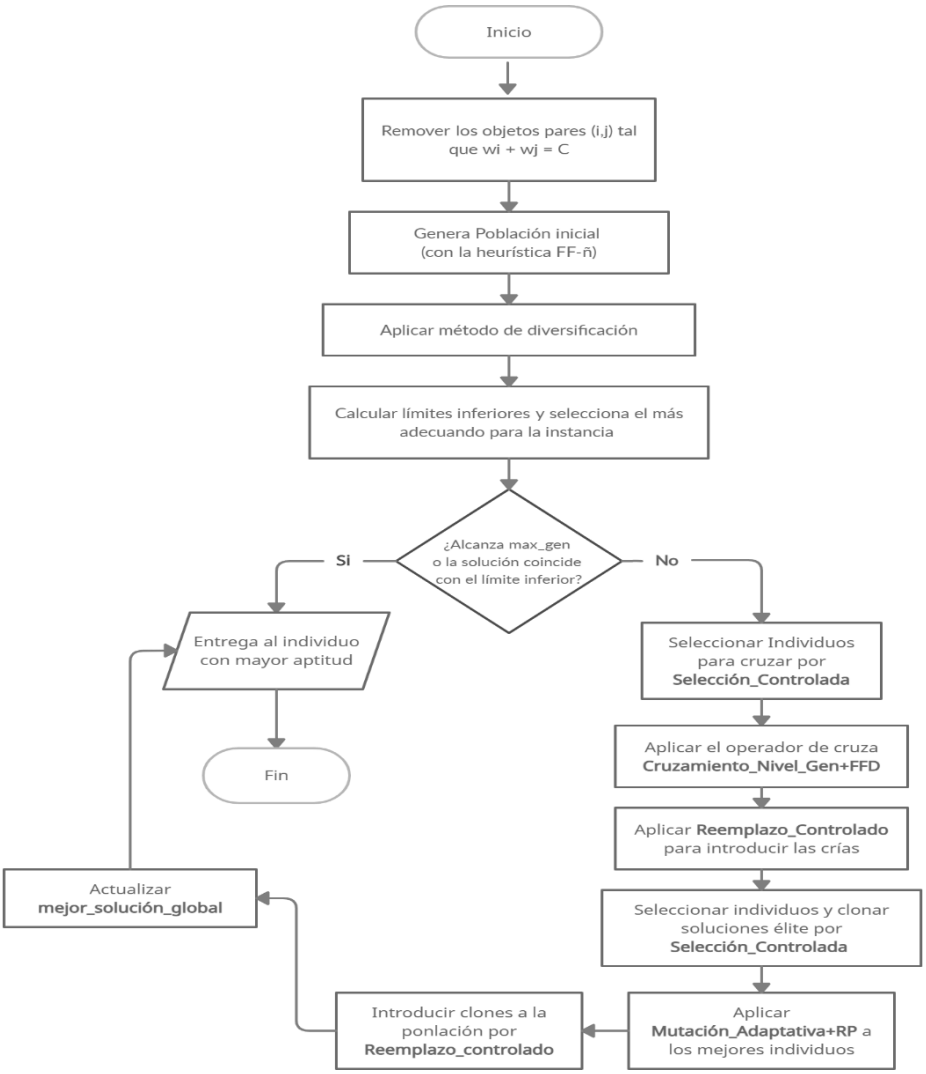

Figura 4.2. Estructura general del algoritmo GGA-CGT/D (versión propuesta).

La figura 4.2 presenta el algoritmo GGA-CGT/D. La idea principal de la versión propuesta era mantener la esencia del algoritmo GGA-CGT original, por lo tanto, los métodos añadidos no afectan el núcleo del algoritmo genético. El proceso comienza aplicando un procedimiento de reducción simple que consta

de eliminar todos los objetos pares *i,j* que al sumarse su total sea igual a la capacidad del contenedor *c*, después con el problema reducido se genera una población *P* de individuos con la heurística de empacado FF-ñ. Luego, a esa población generada se le aplica el método de diversificación, que consiste en reiniciar un porcentaje de la población cada cierto número de generaciones. La configuración seleccionada para este método es de reiniciar el 10% de la población que equivale a 10 individuos a la primera mitad de las generaciones y se reinicia el 15% de la población (15 individuos) en la mitad restante. Estos nuevos individuos generados aleatoriamente sustituyen a los peores en cada mitad. Después de este proceso se calculan los límites inferiores (L_1, $L^{(p)}_*$ L_2, L_3 y L_P) y se selecciona el mejor límite para cada instancia. Posteriormente el algoritmo continúa su proceso original.

4.4 Métodos propuestos para GGA-CGT

En esta sección se describen los métodos implementados en la versión propuesta del algoritmo GGA-CGT. Como principal aportación, se propone lo siguiente: un método para el cálculo de cinco diferentes límites inferiores, un método de reducción de instancias para reducir el problema y un método que brinda diversidad al algoritmo.

4.4.1 Método para el cálculo de algunos límites inferiores

Como es sabido, el límite inferior de una instancia de BPP, indica el número de contenedores que al menos deberán utilizarse para colocar los *n* objetos de la instancia (Pérez-Ortega, et. al., 2016). Si el límite inferior coincide con el número de contenedores óptimo que una instancia debe utilizar, sería de mucha utilidad, por lo que conseguir un límite inferior lo más cercano al óptimo es importante. Una solución obtenida de BPP al aplicar una heurística se puede decir que es la óptima si coincide con el valor de alguno de los límites inferiores.

Los investigadores han propuesto varios métodos para calcular el límite inferior de una instancia, algunos de ellos se aproximan más al número de contenedores de la solución óptima que otros.

Para este trabajo se implementó un método usado en el algoritmo CNS_BP (Buljubašić&Vasquez, 2016), este calcula diferentes límites inferiores. El límite

inferior usado para cada una de las instancias fue el mejor de los siguientes cinco:

$L_1 = \lceil \frac{\sum_{1=1}^{n} w_i}{C} \rceil$ propuesto por (Scholl,Klein&Jürgens, 1997), $L_*^{(p)}$ (Fakete&Schepers, 2001) con p = 10, L_2 , L_3 (Martello&Toth, 1990) y L_P (Alvim, 2004). Cabe mencionar que cuatro de los límites, a excepción de L_2, fueron implementados originalmente en CNS_BP. La figura 4.3 presenta el método utilizado en esta propuesta.

Método para cálculo de límites inferiores

```
Procedimiento LB
1    calcular L1, L*(p), L2, L3, LP
2    si L1 > L2 entonces lb ← L1 si no lb ← L2 fin si;
3    si L*(p) > lb entonces lb ← L*(p) si no lb fin si;
4    si L3 > lb entonces lb ← L3 si no lb fin si;
5    si LP > lb entonces lb ← LP si no lb fin si;
6    retorna lb
fin LB
```

Figura 4.3. Procedimiento para cálculo de límites inferiores.

Este método se compone de dos etapas: primero, se calcula de manera independiente cada límite inferior, es decir, que para el cálculo de cada uno de ellos se realiza su método correspondiente; segundo, el procedimiento LB es el encargado de seleccionar el mejor de los límites. Se realiza la llamada a cada uno de los métodos que calculan a sus respectivos límites (línea 1), posteriormente se comparan L_1 y L_2, el mejor se asigna a la variable *lb* para continuar con la evaluación (línea 2). Se realiza el mismo proceso con los límites pendientes de evaluar, el mejor irá reemplazando al anterior guardado en *lb* (línea 3-5). Al llegar a su fin, el proceso regresará *lb* el cuál contendrá al mejor límite calculado (línea 6-7).

4.4.2 Método de reducción de instancias para simplificar el problema

A través de los años, los investigadores dedicados a problemas de optimización han desarrollado diversos métodos de reducción, algunos más complejos y sofisticados que otros. Estos métodos han sido de gran ayuda, ya que, al simplificar un problema, se puede trabajar enfocándose en una parte del mismo.

La versión del algoritmo que aquí se propone, hace uso de un procedimiento de reducción simple que se utiliza en CNS_BP (Buljubašić&Vasquez, 2016). Este procedimiento consiste en fijar las asignaciones de todos los pares de objetos que pueden llenar en su totalidad a un contenedor. Más precisamente, una vez que se identifica un conjunto de pares de elementos (i, j) tales que $(w_i + w_j = C)$, el problema se puede reducir eliminando esos elementos (o estableciendo sus asignaciones). Esta misma reducción se ha utilizado en diversos artículos sobre BPP. La figura 4.4 presenta un ejemplo del funcionamiento del método de reducción.

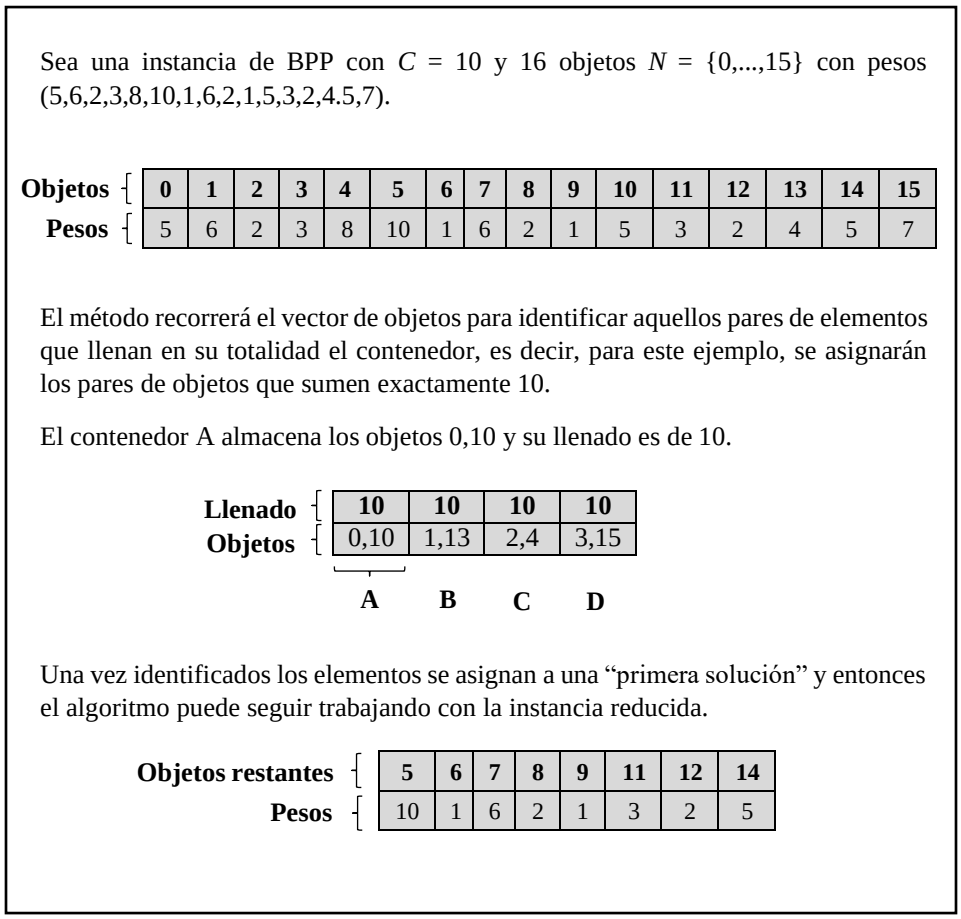

Figura 4.4. Ejemplo del funcionamiento del método de reducción de instancias para simplificar el problema.

4.4.3 Método de diversificación

Para incrementar la diversificación y disminuir la intensificación en el algoritmo, se diseñó un método de reinicio, el cual iterativamente genera nuevas soluciones iniciales de manera aleatoria que reemplazan a un porcentaje de la población cada cierto número de generaciones.

Se consideró la propuesta de (Schiavinotto&Stützle, 2004) en la que utilizan un mecanismo de diversificación que se activa si el valor medio de la función objetivo de la población no ha cambiado durante varios pasos. En este caso, generan una nueva población inicial aleatoria, manteniendo solo el mejor individuo en general. Para este trabajo se adaptó el método de acuerdo a las necesidades del algoritmo, como se mencionó con anterioridad, nuestro método se basa en generar nuevas soluciones aleatorias que sustituyen a los peores individuos en un porcentaje dado sobre la población "*porcentaje_p"*, cada cierto porcentaje de generaciones indicado "*porcentaje_mxg"*. Tras realizar diversas experimentaciones, *porcentaje_p* se estableció en 10% y 15%, por lo tanto, se decidió crear *porcentaje_p1* para la primera configuración y *porcentaje_p2* para la segunda, mientras que *porcentaje_mxg* se estableció en un 50%. Con estas configuraciones, el algoritmo mejoró considerablemente el número de soluciones óptimas encontradas. Otros parámetros utilizados por este método son; el tamaño de la población *P* y el número máximo de generaciones *max_gen*. En la figura 4.5 se presenta el método de diversificación desarrollado.

Método de diversificación

```
Procedimiento Diversificación
1    porcentaje_p1 ← 0.10*P, porcentage_p2 ← 0.15*P;
2    porcentaje_mxg ← 0.5*max_gen;
3    para i ← 0 hasta max_gen hacer
4        si generación = porcentaje_mxg entonces    (generación: variable global que cuenta el # de generaciones)
5            Población ← SeleccionaPeor(Población , porcentaje_p1);
6            nuevos_individuos ← GenerarSolucionAleatoria(porcentaje_p1);
7            Población ← Población ∪ {nuevos_individuos};
8            retorna Población ;
9        fin si
10       si generación = max_gen entonces
11           Población ← SeleccionaPeor(Población , porcentaje_p2);
12           nuevos_individuos ← GenerarSolucionAleatoria(porcentaje_p1);
13           Población ← Población ∪ {nuevos_individuos};
14           retorna Población ;
15       fin si
16   fin para
fin Diversificación
```

Figura 4.5. Procedimiento para agregar diversificación al algoritmo.

El procedimiento comienza estableciendo el porcentaje de la población a reiniciar, así como el número de generación en que se realizará cada reinicio. (línea 1-2). Se itera hasta alcanzar el máximo número de generaciones indicado en la configuración del algoritmo (línea 3), si la generación actual es igual al número de generación en donde se realizará el reinicio, se seleccionan los peores individuos a sustituir de la población inicial generada, el parámetro *porcentaje_p1* en esta función representa el número de individuos que se van a sustituir. Posteriormente se generan aleatoriamente los nuevos individuos, se insertan y se retorna la población actualizada (líneas 4-9). Este mismo proceso se realiza cuando el número de generación actual es igual al número máximo de generaciones, ya que es el otro extremo de generaciones al que se le aplicará el reinicio (líneas 10-15).

Capítulo 5

Experimentación y Resultados

En esta sección se presentan los experimentos realizados para observar el impacto en el desempeño de los métodos implementados en el algoritmo GGA-CGT primero de manera independiente y posteriormente en conjunto, con el fin de comparar su desempeño contra los mejores enfoques en la literatura. También se presenta un análisis con soporte estadístico para evaluar la versión propuesta.

Los experimentos que aquí se presentan fueron realizados en el Clúster de la institución, que cuenta con sistema operativo CentOS release 6.7, versión 5.1.0 de gcc y procesador Intel Xeon E5-2650 2.30GHz. El lenguaje usado para la compilación del programa es C++ y las instancias utilizadas para su evaluación fueron el conjunto BPPLIB* (Uniform, Triplet, Data Set 1,2 y 3, Was 1 y 2, Gau 1, Hard28) y BPP$v_{u-}c$ (BPP.25, BPP.5, BPP.75, BPP1) descritas en la sección 2.5.

Es también pertinente mencionar que, para cada experimento, se realizó una sola corrida por configuración, ya que en algunos casos el número de generaciones es muy grande y el tiempo de ejecución para algunas instancias varía de entre 1 a 5 días.

5.1 Experimento 1: Cálculo de diferentes límites inferiores

Objetivo: Observar independientemente el impacto en el desempeño del método para el cálculo de límites inferiores (L_1, $L^{(p)}_*$, L_2, L_3 y L_P) para las instancias BPPLIB* y BPP $v_{u-}c$. Comparar los resultados con la versión original del algoritmo GGA-CGT y los dos mejores del estado del arte; CNS_BP (Buljubašić&Vasquez, 2016) y General Arc-Flow Formulation with Graph Compression (Brandao&Pedroso, 2013) que es un algoritmo exacto.

Configuración del experimento

En la tabla 5.1 se describe la configuración usada en el algoritmo GGA-CGT para la versión de límites en el conjunto de instancias BPPLIB*.

Tabla 5.1. Configuración del algoritmo GGA-CGT para la versión de límites en las instancias BPPLIB*.

Configuración del algoritmo GGA-CGT Límites para instancias BPPLIB*	
Tamaño de la población |*P*|	100
Número máximo de generaciones *max_gen*	1000
Número de individuos a cruzar *nc*	20
Número de individuos a mutar *nm*	83
El número de contenedores a eliminar *nb* en el operador de mutación para las soluciones no clonadas es calculado utilizando una tasa de cambio k	1.3
Número de individuos en el grupo élite |*B*|	10
El número de contenedores a eliminar *nb* en el operador de mutación cuando la solución fue clonada es calculado utilizando una tasa de cambio *k*	4
Edad máxima para que un individuo pueda ser clonado *max_edad*	10
Semilla *seed*	1

En la tabla 5.2 se presenta el parámetro *max_gen* (número máximo de generaciones) como configuración, ya que es el único parámetro que se incrementa a un millón de generaciones para evaluar el algoritmo GGA-CGT para la versión de límites con el conjunto de instancias BPP$v_{u-}c$.

Tabla 5.2. Configuración del parámetro *max_gen* en el algoritmo GGA-CGT para la versión de límites en las instancias BPP$v_{u-}c$.

Configuración del algoritmo GGA-CGT Límites para instancias BPP$v_{u-}c$	
Número máximo de generaciones *max_gen*	1000000

Resultados

Al término de la implementación de cada una de las estrategias mencionadas en la sección 4.2, se realizaron las respectivas pruebas de manera aislada en el algoritmo GGA-CGT, es decir, se generó una versión del algoritmo para cada una de las estrategias propuestas y se ejecutaron de manera individual para observar su efecto en el desempeño. Como se mencionó con anterioridad, se realizó una sola corrida para cada de una de las experimentaciones aquí presentadas, en otras palabras, las instancias se evaluaron una sola vez por algoritmo. A continuación, se presentan los resultados obtenidos para la versión "límites" del algoritmo con el conjunto de instancias BPPLIB*. La tabla 5.3 muestra para cada clase de instancias, el número de instancias y, para cada enfoque de solución, el número de instancias en las que el algoritmo obtiene una solución óptima (Columna Opt.) y el tiempo medio de ejecución, medido en segundos (Columna T). La última columna (% mejora) presenta cuánto mejoró la versión propuesta con respecto al algoritmo original en número de óptimos encontrados. En el anexo D se encuentran los experimentos que se presentan en esta tesis con el fin de reportar de manera detallada los tiempos de ejecución como lo es el tiempo mínimo, máximo, mediana y media para cada familia de instancias.

Tabla 5.3. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de límites y los dos mejores del estado del arte para las instancias BPPLIB*.

Clase	Inst.	Gral. Arc-Flow Formulation (Exacto)		CNS_BP		GGA-CGT (Original)		GGA-CGT (Versión Límites)		
		Opt.	T(s)	Opt.	T(s)	Opt.	T(s)	Opt.	T(s)	% mejora
Uniform	80	80	0.96	80	0.06	80	1.87	80	1.21	0
Triplets	80	80	1.41	80	0.09	80	2.87	80	1.65	0
Data Set 1	720	720	0.54	720	0.08	720	18.61	720	1.84	0
Data Set 2	480	480	67.53	480	0.04	480	6.96	480	0.72	0
Data Set 3	10	10	21.48	10	0.01	10	45.18	10	7.64	0
Was 1	100	100	0.82	100	0.00	100	0.07	100	0.04	0
Was 2	100	100	0.76	100	0.02	100	14.46	100	8.27	0
Gau 1	17	17	1665.84	17	2.62	16	4.19	17	0.99	5.88
Hard28	28	28	**53.75**	20	6.34	16	23.20	17	12.23	3.57
Total	**1615**	**1615**	**1813.09**	**1607**	**9.26**	**1602**	**117.40**	**1604**	**34.60**	**0.12**
Gran Total			**4452.00**		**1753.00**		**1900.00**		**1323.00**	

De la tabla anterior se puede observar que la versión “límites” del algoritmo, logra resolver todas las instancias Gau_1 y 17 instancias de la familia Hard28, mejorando en un 0.12% respecto al algoritmo original, sin embrago, el método implementado de manera independiente sigue quedando por debajo de los otros dos algoritmos de la literatura.

A continuación, se presentan los resultados obtenidos para la versión “límites” del algoritmo con el conjunto de instancias BPP$v_{u-}c$. Estos resultados son presentados en cuatro tablas independientes, ya que estas instancias están conformadas por cuatro clases distintas (BPP.25, BPP.5, BPP.75, BPP1), las cuáles se detallaron en la sección 2.5. Cada tabla muestra la capacidad del contenedor (Columna C) y el número de instancias, para cada enfoque de solución, el número de instancias en las que el algoritmo obtiene una solución óptima (Columna Opt.) y el tiempo medio de ejecución, medido en segundos (Columna T). La última columna (% mejora) presenta cuánto mejoró la versión propuesta con respecto al algoritmo original en número de óptimos encontrados.

Tabla 5.4. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de límites y los dos mejores del estado del arte para la clase BPP.25 de las instancias BPP$v_{u-}c$.

BPP.25										
C	**Inst.**	**Gral. Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT (Versión Límites)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.96	100	0.00	100	0.00	100	0.00	0
1000	100	100	1.09	100	0.01	100	0.02	100	0.00	0
10000	100	100	3.86	100	0.28	100	0.25	100	0.15	0
100000	100	100	9.15	100	5.97	100	6.23	100	4.27	0
1000000	100	100	76.84	100	51.64	100	54.12	100	45.29	0
10000000	100	100	3152.96	100	1721.10	100	1938.60	100	1625.09	0
100000000	100	87	5982.07	61	2164.52	6	2611.22	10	2218.19	4
Total	**700**	**687**	**9226.94**	**661**	**3943.53**	**606**	**4610.44**	**610**	**3893.00**	**0.57**
Gran Total			**301613.00**		**114076.00**		**118663.00**		**113963.00**	

En la tabla 5.4 se muestran los resultados obtenidos por los algoritmos en la clase BPP.25 de las instancias BPP$v_{u-}c$. La versión desarrollada “Límites”, logra resolver 610 de un total de 700 instancias, aumentando un 0.57% la eficacia del algoritmo original, sin embrago, se muestra inferior en número de óptimos alcanzados con respecto a los algoritmos de la literatura.

Tabla 5.5. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de límites y los dos mejores del estado del arte para la clase BPP.5 de las instancias BPP$v_{u_}c$.

BPP.5										
C	**Inst.**	**Gral. Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT (Versión Límites)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.75	100	0.00	100	0.01	100	0.01	0
1000	100	100	1.40	100	0.09	100	0.12	100	0.08	0
10000	100	100	3.93	100	2.12	100	2.05	100	1.93	0
100000	100	96	624.08	91	447.99	89	462.09	90	431.87	1
1000000	100	34	1448.21	28	981.14	1	907.70	8	876.23	7
10000000	100	21	1187.95	16	962.97	8	941.99	11	903.12	3
100000000	100	75	1092.36	42	892.17	35	878.22	38	795.75	3
Total	**700**	**526**	**4358.68**	**477**	**3286.48**	**433**	**3192.17**	**447**	**3008.99**	**0.57**
Gran Total			**536297.00**		**382551.00**		**372600.00**		**351458.00**	

De la tabla 5.5 se pueden observar los resultados obtenidos por los algoritmos en la clase BPP.5. La versión "Límites", resuelve 447 instancias, encontrando 14 óptimos más que el algoritmo GGA-CGT original y nuevamente aumenta un 0.57% su eficacia. No obstante, esta versión sigue posicionándose como la tercera en cuestión de instancias resueltas de forma óptima.

Tabla 5.6. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de límites y los dos mejores del estado del arte para la clase BPP.75 de las instancias BPP$v_{u_}c$.

BPP.75										
C	**Inst.**	**Gral. Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT (Versión Límites)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.82	98	0.02	95	0.03	97	0.02	2
1000	100	100	1.54	97	0.52	99	0.70	99	0.46	0
10000	100	31	7.25	19	3.86	0	5.05	12	3.99	12
100000	100	26	8.68	21	6.09	0	5.75	1	4.88	1
1000000	100	22	4.10	13	2.05	0	2.20	7	1.89	7
10000000	100	37	3.71	24	1.24	0	1.57	5	1.03	5
100000000	100	42	4.04	36	0.98	0	1.46	2	0.56	2
Total	**700**	**358**	**30.13**	**308**	**14.75**	**194**	**16.77**	**223**	**12.83**	**4.14**
Gran Total			**5872.00**		**1976.00**		**2047.00**		**1754.00**	

La tabla 5.6 nos muestra los resultados obtenidos por los algoritmos en la clase BPP.75. La versión "Límites", en esta ocasión resuelve 29 instancias más que el algoritmo GGA-CGT original y mejora un 4.14%, sin embargo, los dos algoritmos de la literatura siguen mostrando mejores resultados.

Tabla 5.7. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de límites y los dos mejores del estado del arte para la clase BPP1 de las instancias BPP$v_{u-}c$.

BPP1										
C	**Inst.**	**Gral. Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT (Versión Límites)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.47	100	0.04	100	0.06	100	0.04	0
1000	100	100	2.41	100	0.82	100	0.93	100	0.75	0
10000	100	64	565.73	58	328.65	49	379.11	54	337.25	5
100000	100	87	369.74	69	273.98	73	293.66	76	254.09	3
1000000	100	100	131.09	99	92.71	94	95.96	98	90.09	4
10000000	100	100	32.47	100	16.10	100	19.55	100	17.74	0
100000000	100	100	19.46	100	6.25	100	8.22	100	5.88	0
Total	**700**	**651**	**1121.37**	**626**	**718.56**	**616**	**797.49**	**628**	**705.83**	**1.71**
Gran Total			**104852.00**		**73421.00**		**84534.00**		**72505.00**	

De la tabla 5.7 se observan los resultados obtenidos por los algoritmos en la clase BPP1. La versión "Límites", resuelve un total de 628 instancias, encontrando 12 óptimos más que el algoritmo GGA-CGT original y aumenta un 1.71% su eficacia. También podemos ver que, para esta clase, nuestra versión logra superar por 2 instancias resueltas de manera óptima al algoritmo CNS_BP.

Conclusión del experimento

Con la experimentación realizada para la versión "límites", se puede concluir que este método de manera independiente mejora el número de instancias resueltas de manera óptima en un 2.11% con respecto al total de instancias que logra resolver el algoritmo GGA-CGT. Sin embargo, para tres de las cuatro clases evaluadas de BPP$v_{u-}c$, la versión propuesta se muestra inferior a los dos algoritmos evaluados de la literatura en número de óptimos encontrados.

5.2 Experimento 2: Reducción de instancias para simplificar el problema

Objetivo: Observar independientemente el impacto en el desempeño del método de reducción implementado, para las instancias BPPLIB* y BPP$v_{u_}c$. Comparar los resultados con la versión original del algoritmo GGA-CGT y los dos mejores del estado del arte; CNS_BP (Buljubašić&Vasquez, 2016) y General Arc-Flow Formulation with Graph Compression (Brandao&Pedroso, 2013) que es un algoritmo exacto.

Configuración del experimento

En la tabla 5.8 se describe la configuración usada en el algoritmo GGA-CGT para la versión de límites en el conjunto de instancias BPPLIB*.

Tabla 5.8. Configuración del algoritmo GGA-CGT para la versión de reducción en las instancias BPPLIB*.

Configuración del algoritmo GGA-CGT Reducción para instancias BPPLIB*	
Tamaño de la población |*P*|	100
Número máximo de generaciones *max_gen*	1000
Número de individuos a cruzar *nc*	20
Número de individuos a mutar *nm*	83
El número de contenedores a eliminar *nb* en el operador de mutación para las soluciones no clonadas es calculado utilizando una tasa de cambio k	1.3
Número de individuos en el grupo élite |*B*|	10
El número de contenedores a eliminar *nb* en el operador de mutación cuando la solución fue clonada es calculado utilizando una tasa de cambio *k*	4
Edad máxima para que un individuo pueda ser clonado *max_edad*	10
Semilla *seed*	1

En la tabla 5.9 se presenta el parámetro *max_gen* (número máximo de generaciones) como configuración, ya que es el único parámetro que se

incrementa a un millón de generaciones para evaluar el algoritmo GGA-CGT para la versión de reducción con el conjunto de instancias BPP$v_{u-}c$.

Tabla 5.9. Configuración del parámetro *max_gen* en el algoritmo GGA-CGT para la versión de reducción en las instancias BPP$v_{u-}c$.

Configuración del algoritmo GGA-CGT Reducción para instancias BPP$v_{u-}c$	
Número máximo de generaciones *max_gen*	1000000

Resultados

En esta sección se presentan los resultados obtenidos para la versión "Reducción" del algoritmo con el conjunto de instancias BPPLIB*. Se realizó una sola corrida para cada de una de las experimentaciones aquí presentadas, es decir, las instancias se evaluaron una sola vez por algoritmo. La tabla 5.10 muestra para cada clase de instancias, el número de instancias y, para cada enfoque de solución, el número de instancias en las que el algoritmo obtiene una solución óptima (Columna Opt.) y el tiempo medio de ejecución, medido en segundos (Columna T). La última columna (% mejora) presenta cuánto mejoró la versión propuesta con respecto al algoritmo original en número de óptimos encontrados.

Tabla 5.10. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de reducción y los dos mejores del estado del arte para las instancias BPPLIB*.

Clase	**Inst.**	**Gral. Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT (Versión Reducción)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
Uniform	80	80	0.96	80	0.06	80	1.87	80	2.35	0
Triplets	80	80	1.41	80	0.09	80	2.87	80	3.44	0
Data Set 1	720	720	0.54	720	0.08	720	18.61	720	18.85	0
Data Set 2	480	480	67.53	480	0.04	480	6.96	480	7.13	0
Data Set 3	10	10	21.48	10	0.01	10	45.18	10	47.25	0
Was 1	100	100	0.82	100	0.00	100	0.07	100	0.19	0
Was 2	100	100	0.76	100	0.02	100	14.46	100	16.14	0
Gau 1	17	17	1665.84	17	2.62	16	4.19	17	4.73	5.88
Hard28	28	28	**53.75**	20	6.34	16	23.20	20	25.19	14.29
Total	**1615**	**1615**	**1813.09**	**1607**	**9.26**	**1602**	**117.40**	**1607**	**125.26**	
Gran Total			**4452.00**		**1753.00**		**1900.00**		**2082.00**	**0.31**

De la tabla anterior se puede observar que la versión "Reducción", logra resolver todas las instancias Gau_1 y 20 instancias de la familia Hard28 (3 instancias más que GGA-CGT), mejorando en un 0.31% respecto al algoritmo original e iguala al algoritmo CNS_BP en número de instancias resueltas de manera óptima. No obstante, el algoritmo exacto se muestra superior resolviendo todo el conjunto de instancias BPPLIB*.

A continuación, se presentan los resultados obtenidos para la versión "Reducción" del algoritmo con el conjunto de instancias BPP$v_{u-}c$. Cada tabla muestra la capacidad del contenedor (Columna C) y el número de instancias, para cada enfoque de solución, el número de instancias en las que el algoritmo obtiene una solución óptima (Columna Opt.) y el tiempo medio de ejecución, medido en segundos (Columna T). La última columna (% mejora) presenta cuánto mejoró la versión propuesta con respecto al algoritmo original en número de óptimos encontrados.

Tabla 5.11. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de reducción y los dos mejores del estado del arte para la clase BPP.25 de las instancias BPP$v_{u-}c$.

BPP.25										
C	**Inst.**	**Gral. Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT (Versión Reducción)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.96	100	0.00	100	0.00	100	1.46	0
1000	100	100	1.09	100	0.01	100	0.02	100	1.73	0
10000	100	100	3.86	100	0.28	100	0.25	100	2.35	0
100000	100	100	9.15	100	5.97	100	6.23	100	8.46	0
1000000	100	100	76.84	100	51.64	100	54.12	100	68.64	0
10000000	100	100	3152.96	100	1721.10	100	1938.60	100	2106.27	0
100000000	100	87	5982.07	61	2164.52	6	2611.22	18	3874.53	12
Total	**700**	**687**	**9226.94**	**661**	**3943.53**	**606**	**4610.44**	**618**	**6063.44**	**1.71**
Gran Total			**301613.00**		**114076.00**		**118663.00**		**195286.00**	

En la tabla 5.11 se muestran los resultados obtenidos por los algoritmos en la clase BPP.25 de las instancias BPP $v_{u-}c$. La versión "Reducción", logra resolver 618 de un total de 700 instancias, aumentando un 1.71% la eficacia del algoritmo original, sin embrago, se muestra inferior en número de óptimos alcanzados con respecto a los algoritmos de la literatura.

Tabla 5.12. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de reducción y los dos mejores del estado del arte para la clase BPP.5 de las instancias BPP$v_{u-}c$.

BPP.5										
C	**Inst.**	**Gral. Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT (Versión Reducción)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.75	100	0.00	100	0.01	100	0.67	0
1000	100	100	1.40	100	0.09	100	0.12	100	1.17	0
10000	100	100	3.93	100	2.12	100	2.05	100	2.98	0
100000	100	96	624.08	91	447.99	89	462.09	94	564.92	5
1000000	100	34	1448.21	28	981.14	1	907.70	13	1276.34	12
10000000	100	21	1187.95	16	962.97	8	941.99	14	1125.84	6
100000000	100	75	1092.36	42	892.17	35	878.22	41	976.27	6
Total	**700**	**526**	**4358.68**	**477**	**3286.48**	**433**	**3192.17**	**462**	**3948.20**	**2.71**
Gran Total			**536297.00**		**382551.00**		**372600.00**		**425624.00**	

De la tabla 5.12 se pueden observar los resultados obtenidos por los algoritmos en la clase BPP.5. La versión "Reducción", resuelve 462 instancias, encontrando 29 óptimos más que el algoritmo GGA-CGT original y aumenta un 2.71% su eficacia. No obstante, esta versión sigue posicionándose como la tercera en cuestión de instancias resueltas de forma óptima.

Tabla 5.13. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de reducción y los dos mejores del estado del arte para la clase BPP.75 de las instancias BPP$v_{u-}c$.

BPP.75										
C	**Inst.**	**Gral. Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT (Versión Reducción)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.82	98	0.02	95	0.03	100	0.38	5
1000	100	100	1.54	97	0.52	99	0.70	100	2.75	1
10000	100	31	7.25	19	3.86	0	5.05	32	7.98	32
100000	100	26	8.68	21	6.09	0	5.75	3	7.23	3
1000000	100	22	4.10	13	2.05	0	2.20	11	4.35	11
10000000	100	37	3.71	24	1.24	0	1.57	18	3.03	18
100000000	100	42	4.04	36	0.98	0	1.46	6	3.47	6
Total	**700**	**358**	**30.13**	**308**	**14.75**	**194**	**16.77**	**270**	**29.20**	**10.86**
Gran Total			**5872.00**		**1976.00**		**2047.00**		**4318.00**	

La tabla 5.13 nos muestra los resultados obtenidos por los algoritmos en la clase BPP.75. La versión "Reducción", en esta ocasión resuelve 76 instancias más que el algoritmo GGA-CGT original y mejora un 10.86%, sin embargo, los dos algoritmos de la literatura siguen mostrando mejores resultados.

Tabla 5.14. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de reducción y los dos mejores del estado del arte para la clase BPP1 de las instancias BPP$v_{u_}c$.

BPP1										
C	**Inst.**	**Gral. Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT (Versión Reducción)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.47	100	0.04	100	0.06	100	0.51	0
1000	100	100	2.41	100	0.82	100	0.93	100	2.98	0
10000	100	64	565.73	58	328.65	49	379.11	62	429.11	13
100000	100	87	369.74	69	273.98	73	293.66	80	413.63	7
1000000	100	100	131.09	99	92.71	94	95.96	100	201.87	6
10000000	100	100	32.47	100	16.10	100	19.55	100	28.24	0
100000000	100	100	19.46	100	6.25	100	8.22	100	14.43	0
Total	**700**	**651**	**1121.37**	**626**	**718.56**	**616**	**797.49**	**642**	**1090.79**	**3.71**
Gran Total			**104852.00**		**73421.00**		**84534.00**		**102179.00**	

De la tabla 5.14 se observan los resultados obtenidos por los algoritmos en la clase BPP1. La versión "Reducción", resuelve un total de 642 instancias, encontrando 26 óptimos más que el algoritmo GGA-CGT original y aumenta un 3.71% su eficacia. También podemos ver que, para esta clase, nuestra versión logra superar por 16 instancias resueltas de manera óptima al algoritmo CNS_BP.

Conclusión del experimento

Con la experimentación realizada para la versión "Reducción", se puede concluir que este método de manera independiente mejora el número de instancias resueltas de manera óptima en un 5.11% con respecto al total de instancias que logra resolver el algoritmo GGA-CGT y a su vez, esta versión mejora un 3% los resultados obtenidos por la versión "Límites" y supera al algoritmo CNS_BP en la clase BPP1, sin embargo, para las tres clases restantes de BPP$v_{u_}c$ se muestra inferior a los dos algoritmos evaluados de la literatura en número de óptimos encontrados.

5.3 Experimento 3: Método de diversificación

Objetivo: Observar independientemente el impacto en el desempeño del método de diversificación implementado, para las instancias BPPLIB* y BPPv_u_c. Comparar los resultados con la versión original del algoritmo GGA-CGT y los dos mejores del estado del arte; CNS_BP (Buljubašić&Vasquez, 2016) y General Arc-Flow Formulation with Graph Compression (Brandao&Pedroso, 2013) que es un algoritmo exacto.

Configuración del experimento

Para alcanzar los resultados obtenidos, se realizaron diversas pruebas con diferentes configuraciones de diversificación; reiniciando el 5%, 10%, 15%, 50% y 100% de la población cada 50 generaciones, dando mejores resultados en todos los casos la configuración del 10%. En la tabla 5.15 se describe la configuración usada en el algoritmo GGA-CGT para la versión de diversificación en el conjunto de instancias BPPLIB*.

Tabla 5.15. Configuración del algoritmo GGA-CGT para la versión de diversificación en las instancias BPPLIB*.

Configuración del algoritmo GGA-CGT Diversificación para instancias BPPLIB*	
Tamaño de la población |*P*|	100
Número máximo de generaciones *max_gen*	1000
Porcentaje de Diversificación	10% de |*P*| cada 50 generaciones
Número de individuos a cruzar *nc*	20
Número de individuos a mutar *nm*	83
nb para las soluciones no clonadas *k*	1.3
Número de individuos en el grupo élite |*B*|	10
nb cuando la solución fue clonada *k*	4
Edad máxima para que un individuo pueda ser clonado *max_edad*	10
Semilla *seed*	1

En la tabla 5.16 se presentan los parámetros *max_gen* (número máximo de generaciones) y Porcentaje de Diversificación como configuración, ya que son los únicos que se modifican para la evaluación el algoritmo GGA-CGT para la versión de diversificación con el conjunto de instancias BPPv_u_c.

Tabla 5.16. Configuración en el algoritmo GGA-CGT para la versión de diversificación en las instancias BPP$v_{u-}c$.

Configuración del algoritmo GGA-CGT Diversificación para instancias BPP$v_{u-}c$	
Número máximo de generaciones *max_gen*	1000000
Porcentaje de Diversificación	10% de |*P*| cada 1000 generaciones

Resultados

En esta sección se presentan los resultados obtenidos para la versión "Diversificación" del algoritmo con el conjunto de instancias BPPLIB*. Se realizó una sola corrida para cada de una de las experimentaciones aquí presentadas, es decir, las instancias se evaluaron una sola vez por algoritmo. La tabla 5.17 muestra para cada clase de instancias, el número de instancias y, para cada enfoque de solución, el número de instancias en las que el algoritmo obtiene una solución óptima (Columna Opt.) y el tiempo medio de ejecución, medido en segundos (Columna T). La última columna (% mejora) presenta cuánto mejoró la versión propuesta con respecto al algoritmo original en número de óptimos encontrados.

Tabla 5.17. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de diversificación y los dos mejores del estado del arte para las instancias BPPLIB*.

Clase	**Inst.**	**Gral. Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT (Versión Diversificación)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
Uniform	80	80	0.96	80	0.06	80	1.87	80	2.35	0
Triplets	80	80	1.41	80	0.09	80	2.87	80	3.44	0
Data Set 1	720	720	0.54	720	0.08	720	18.61	720	18.85	0
Data Set 2	480	480	67.53	480	0.04	480	6.96	480	7.13	0
Data Set 3	10	10	21.48	10	0.01	10	45.18	10	47.25	0
Was 1	100	100	0.82	100	0.00	100	0.07	100	0.19	0
Was 2	100	100	0.76	100	0.02	100	14.46	100	16.14	0
Gau 1	17	17	1665.84	17	2.62	16	4.19	17	4.73	5.88
Hard28	28	28	**53.75**	20	6.34	16	23.20	28	22.1	42.86
Total	**1615**	**1615**	**1813.09**	**1607**	**9.26**	**1602**	**117.40**	**1615**	**123.27**	**0.81**
Gran Total			**4452.00**		**1753.00**		**1900.00**		**2085.00**	

De la tabla anterior se puede observar que la versión “Diversificación”, logra resolver todo el conjunto de instancias BPPLIB*, mejorando en un 0.81% respecto al algoritmo original, supera a CNS_BP e iguala al algoritmo exacto (General Arc-Flow Formulation) en número de instancias resueltas de manera óptima.

A continuación, se presentan los resultados obtenidos para la versión “Diversificación” del algoritmo con el conjunto de instancias BPP$v_{u_}c$. Cada tabla muestra la capacidad del contenedor (Columna C) y el número de instancias, para cada enfoque de solución, el número de instancias en las que el algoritmo obtiene una solución óptima (Columna Opt.) y el tiempo medio de ejecución, medido en segundos (Columna T). La última columna (% mejora) presenta cuánto mejoró la versión propuesta con respecto al algoritmo original en número de óptimos encontrados.

Tabla 5.18. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de diversificación y los dos mejores del estado del arte para la clase BPP.25 de las instancias BPP$v_{u_}c$.

BPP.25										
C	**Inst.**	**Gral. Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT (Versión Diversificación)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.96	100	0.00	100	0.00	100	0.01	0
1000	100	100	1.09	100	0.01	100	0.02	100	0.05	0
10000	100	100	3.86	100	0.28	100	0.25	100	0.24	0
100000	100	100	9.15	100	5.97	100	6.23	100	6.88	0
1000000	100	100	76.84	100	51.64	100	54.12	100	57.08	0
10000000	100	100	3152.96	100	1721.10	100	1938.60	100	2014.75	0
100000000	100	87	5982.07	61	2164.52	6	2611.22	34	2798.77	28
Total	**700**	**687**	**9226.94**	**661**	**3943.53**	**606**	**4610.44**	**634**	**4877.78**	**4**
Gran Total			**301613.00**		**114076.00**		**118663.00**		**117028.00**	

En la tabla 5.18 se muestran los resultados obtenidos por los algoritmos en la clase BPP.25 de las instancias BPP$v_{u_}c$. La versión “Diversificación”, logra resolver 634 de 700 instancias, aumentando un 4% la eficacia del algoritmo original, sin embrago, se muestra inferior en número de óptimos alcanzados con respecto a los algoritmos de la literatura.

Tabla 5.19. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de diversificación y los dos mejores del estado del arte para la clase BPP.5 de las instancias BPP$v_{u_}c$.

BPP.5										
C	**Inst.**	**Gral. Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT (Versión Diversificación)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.75	100	0.00	100	0.01	100	0.02	0
1000	100	100	1.40	100	0.09	100	0.12	100	0.17	0
10000	100	100	3.93	100	2.12	100	2.05	100	2.74	0
100000	100	96	624.08	91	447.99	89	462.09	100	488.21	11
1000000	100	34	1448.21	28	981.14	1	907.70	32	914.05	31
10000000	100	21	1187.95	16	962.97	8	941.99	29	997.32	21
100000000	100	75	1092.36	42	892.17	35	878.22	55	884.55	20
Total	**700**	**526**	**4358.68**	**477**	**3286.48**	**433**	**3192.17**	**516**	**3287.07**	**11.86**
Gran Total			**536297.00**		**382551.00**		**372600.00**		**380296.00**	

De la tabla 5.19 se pueden observar los resultados obtenidos por los algoritmos en la clase BPP.5. La versión "Diversificación", resuelve 516 instancias, encontrando 83 óptimos más que el algoritmo GGA-CGT original y aumenta un 11.86% su eficacia, así mismo, supera al algoritmo CNS_BP en número de instancias resueltas de manera óptima.

Tabla 5.20. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de diversificación y los dos mejores del estado del arte para la clase BPP.75 de las instancias BPP$v_{u_}c$.

BPP.75										
C	**Inst.**	**Gral. Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT (Versión Diversificación)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.82	98	0.02	95	0.03	100	0.04	5
1000	100	100	1.54	97	0.52	99	0.70	100	0.72	1
10000	100	31	7.25	19	3.86	0	5.05	39	4.93	39
100000	100	26	8.68	21	6.09	0	5.75	21	6.01	21
1000000	100	22	4.10	13	2.05	0	2.20	18	2.46	18
10000000	100	37	3.71	24	1.24	0	1.57	22	2.17	22
100000000	100	42	4.04	36	0.98	0	1.46	17	1.43	17
Total	**700**	**358**	**30.13**	**308**	**14.75**	**194**	**16.77**	**317**	**17.75**	**17.57**
Gran Total			**5872.00**		**1976.00**		**2047.00**		**2206.00**	

La tabla 5.20 nos muestra los resultados obtenidos por los algoritmos en la clase BPP.75. La versión "Diversificación", aumenta significativamente el número de óptimos alcanzados con respecto al algoritmo original, resolviendo 317 instancias, por lo cual mejora en un 17.57% la eficacia de GGA-CGT. Nuevamente logra superar al algoritmo CNS_BP, en esta ocasión por 9 instancias.

Tabla 5.21. Resultados obtenidos por el algoritmo GGA-CGT original, la versión de diversificación y los dos mejores del estado del arte para la clase BPP1 de las instancias BPP$v_{u-}c$.

BPP1										
C	**Inst.**	**Gral. Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT (Versión Diversificación)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.47	100	0.04	100	0.06	100	0.06	0
1000	100	100	2.41	100	0.82	100	0.93	100	0.92	0
10000	100	64	565.73	58	328.65	49	379.11	72	381.65	23
100000	100	87	369.74	69	273.98	73	293.66	89	295.88	16
1000000	100	100	131.09	99	92.71	94	95.96	100	98.76	6
10000000	100	100	32.47	100	16.10	100	19.55	100	18.87	0
100000000	100	100	19.46	100	6.25	100	8.22	100	9.01	0
Total	**700**	**651**	**1121.37**	**626**	**718.56**	**616**	**797.49**	**661**	**805.15**	**6.42**
Gran Total			**104852.00**		**73421.00**		**84534.00**		**85097.00**	

De la tabla 5.21 se observan los resultados obtenidos por los algoritmos en la clase BPP1. La versión "Diversificación", resuelve un total de 661 instancias, encontrando 45 óptimos más que el algoritmo GGA-CGT original y aumenta un 6.42% su eficacia. También podemos ver que, para esta clase, nuestra versión logra superar a los dos algoritmos de la literatura; por 10 instancias a General Arc-Flow F. y por 35 a CNS_BP.

Conclusión del experimento

Con la experimentación realizada para la versión "Diversificación", se puede concluir que este método de manera independiente mejora el número de instancias resueltas de manera óptima en un 9.97% con respecto al total de instancias que logra resolver el algoritmo GGA-CGT y a su vez, este método mejora los resultados obtenidos para las versiones anteriores; "Límites" y "Reducción" en 7.86% y 4.86% respectivamente. Esta propuesta resuelve en su

totalidad el conjunto de instancias BPPLIB* y supera en tres de las cuatro clases de BPP$v_{u_}c$ a los algoritmos de la literatura. Esto se debe a que el método desarrollado disminuye la intensificación del algoritmo y enriquece la diversificación de los individuos.

5.4 Experimento 4: GGA-CGT con los métodos propuestos en conjunto (GGA-CGT/D)

Objetivo: Observar el impacto en el desempeño de los métodos propuestos en conjunto, para las instancias BPPLIB* y BPP$v_{u_}c$. Comparar los resultados con la versión original del algoritmo GGA-CGT y los dos mejores del estado del arte; CNS_BP (Buljubašić&Vasquez, 2016) y General Arc-Flow Formulation with Graph Compression (Brandao&Pedroso, 2013) que es un algoritmo exacto.

Configuración del experimento

Para alcanzar los resultados obtenidos, se realizaron diversas pruebas con diferentes configuraciones de diversificación, las más significativas fueron las siguientes:

a) Reiniciar el 10% de la población cada 10% de generaciones.
b) Reiniciar el 10% y 15% de la población cada mitad de las generaciones.
c) Reiniciar el 5% y 15% de la población cada mitad de las generaciones.
d) Reiniciar el 5%, 10% y 15% de la población cada tercio de las generaciones.

Cabe mencionar que todas estas configuraciones se probaron en conjunto con los otros métodos, dando los mejores resultados en todas las ocasiones la configuración b (Reiniciar el 10% y 15% de la población cada mitad de las generaciones).

La figura 5.1 muestra el comportamiento algorítmico para cada una de las configuraciones utilizadas.

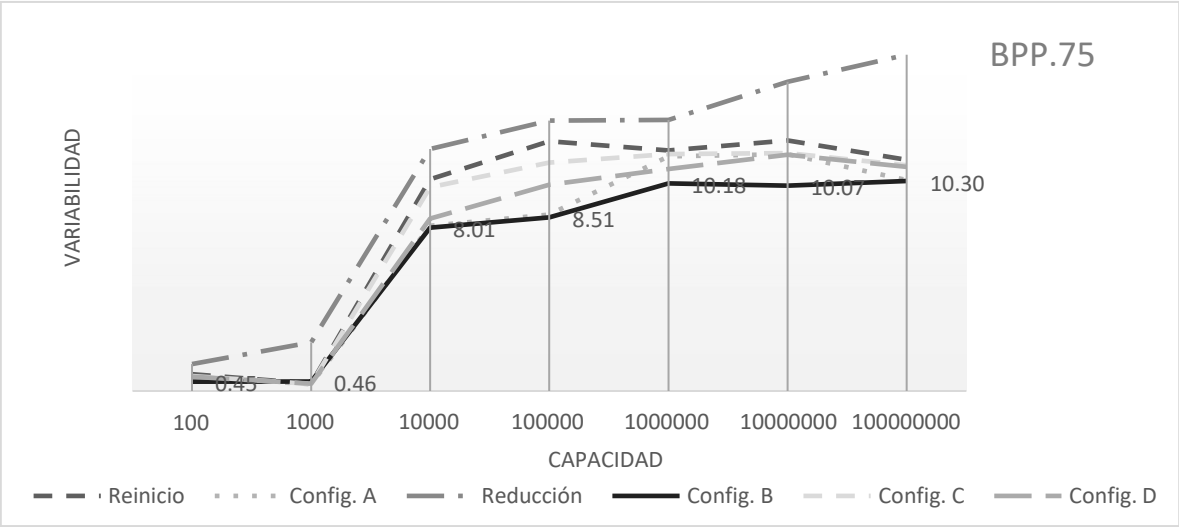

Figura 5.1 Fitness repetido en promedio para las diferentes configuraciones en la clase BPP.75.

La gráfica representa el número de fitness repetido en promedio para las diferentes capacidades de contendor en la clase BP.75. La línea más alta representa el método de reducción de manera independiente y puede observarse que para esta versión del algoritmo existe un mayor número de fitness repetido. La segunda línea más alta representa el método de diversificación de manera independiente y se observa como disminuyen los fitness repetidos en comparación con la anterior. No obstante, al aplicar ambos métodos en conjunto se va disminuyendo el número de repetidos hasta llegar a la mejor configuración que es: b) Reiniciar el 10% y 15% de la población cada mitad de las generaciones.

Esto podría indicarnos de cierta manera que reducir el número de fitness repetidos podría mejorar el desempeño del algoritmo, sin embargo, esto no es contundente ya que también influye la calidad de solución del individuo que se sustituye.

La figura 5.2 muestra un ejemplo de variabilidad cuando se realiza el reinicio de la población en la mejor configuración. Se puede observar que disminuye de manera significativa el número de fitness repetidos al aplicar el reinicio.

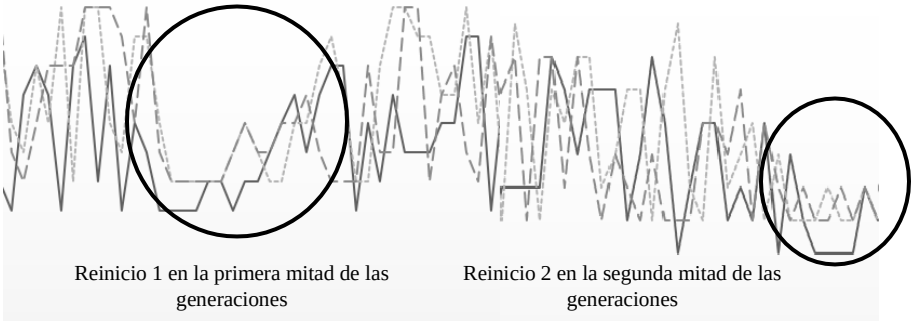

Figura 5.2 Ejemplo de variabilidad cuando se realiza el reinicio de la población en la mejor configuración.

En la tabla 5.22 se describe la configuración usada en el algoritmo GGA-CGT/D para las instancias BPPLIB*.

Tabla 5.22. Configuración del algoritmo GGA-CGT/D para las instancias BPPLIB*.

Configuración del algoritmo GGA-CGT/D para instancias BPPLIB*	
Tamaño de la población $\|P\|$	100
Número máximo de generaciones *max_gen*	1000
Porcentaje de Diversificación	10% y 15% de $\|P\|$ a ½ de *max_gen*
Número de individuos a cruzar *nc*	20
Número de individuos a mutar *nm*	83
El número de contenedores a eliminar *nb* en el operador de mutación para las soluciones no clonadas es calculado utilizando una tasa de cambio k	1.3
Número de individuos en el grupo élite $\|B\|$	10
El número de contenedores a eliminar *nb* en el operador de mutación cuando la solución fue clonada es calculado utilizando una tasa de cambio *k*	4
Edad máxima para que un individuo pueda ser clonado *max_edad*	10
Semilla *seed*	1

En la tabla 5.23 se presentan los parámetros *max_gen* (número máximo de generaciones) y Porcentaje de Diversificación como configuración, ya que son los únicos que se modifican para la evaluación el algoritmo GGA-CGT para la versión de diversificación con el conjunto de instancias BPP$v_{u-}c$.

Tabla 5.23. Configuración del algoritmo GGA-CGT/D para las instancias BPP$v_{u-}c$.

Configuración del algoritmo GGA-CGT/D para instancias BPP$v_{u-}c$	
Número máximo de generaciones *max_gen*	1000000000
Porcentaje de Diversificación	10% y 15% de |*P*| a ½ de *max_gen*

Para estas últimas instancias se decidió realizar 1 billón de generaciones con el fin de mantener en igualdad de circunstancias al algoritmo exacto (General Arc-Flow Formulation) y a la versión propuesta, ya que el tiempo que tarda el exacto en evaluar cada clase de instancias es entre tres y cuatro días.

Resultados

Para investigar la eficacia de la versión propuesta GGA-CGT/D, se compararon los resultados obtenidos con la versión propuesta (GGA-CGT/D) con los obtenidos por los mejores enfoques en la literatura: General Arc-Flow Formulation with Graph Compression (Brandao&Pedroso, 2013), CNS_BP (Buljubašić&Vasquez, 2016) y el algoritmo original de GGA-CGT (Quiroz, 2014) para las instancias BPPLIB*. Se realizó una sola corrida para cada de una de las experimentaciones aquí presentadas, es decir, las instancias se evaluaron una sola vez por algoritmo.

La tabla 5.24 muestra, para cada clase de instancias, el número de instancias y, para cada enfoque de solución, el número de instancias en las que el algoritmo obtiene una solución óptima (Columna Opt.) y el tiempo medio de ejecución, medido en segundos (Columna T). La última columna (% mejora) presenta cuánto mejoró la versión propuesta con respecto al algoritmo original en número de óptimos encontrados. Como se mencionó en el experimento 1, en el anexo D se presentan de manera detallada los tiempos para las experimentaciones realizadas; mínimos, máximos, mediana y media para cada familia de instancias.

Tabla 5.24. Resultados obtenidos por el trabajo propuesto contra el algoritmo GGA-CGT original y los dos mejores del estado del arte para las instancias BPPLIB*.

Clase	Inst.	General Arc-Flow Formulation (Exacto)		CNS_BP		GGA-CGT (Original)		GGA-CGT/D (Versión propuesta)		
		Opt.	T(s)	Opt.	T(s)	Opt.	T(s)	Opt.	T(s)	% mejora
Uniform	80	80	0.96	80	0.06	80	1.87	80	2.28	0
Triplets	80	80	1.41	80	0.09	80	2.87	80	2.74	0
Data Set 1	720	720	0.54	720	0.08	720	18.61	720	19.57	0
Data Set 2	480	480	67.53	480	0.04	480	6.96	480	11.23	0
Data Set 3	10	10	21.48	10	0.01	10	45.18	10	47.10	0
Was 1	100	100	0.82	100	0.00	100	0.07	100	0.12	0
Was 2	100	100	0.76	100	0.02	100	14.46	100	16.24	0
Gau 1	17	17	1665.84	17	2.62	16	4.19	17	8.97	5.89
Hard28	28	28	53.75	20	6.34	16	23.20	28	26.22	42.86
Total	**1615**	**1615**	**1813.09**	**1607**	**9.26**	**1602**	**117.40**	**1615**	**134.47**	**0.81**
Gran Total			**4452.00**		**1753.00**		**1900.00**		**2114.00**	

De la tabla 5.24, puede observarse la superioridad de la versión propuesta GGA-CGT/D sobre la clase de instancias Hard28, que contiene las instancias más difíciles para los algoritmos heurísticos conocidos, posicionándose como el mejor algoritmo metaheurístico, ya que logra superar los resultados obtenidos por CNS_BP, GGA-CGT e iguala al algoritmo exacto en número de soluciones y reduce el tiempo de ejecución total a casi un 50%. También se observa que el trabajo propuesto mejora en un 5.89% el desempeño del algoritmo GGA-CGT original para las instancias Gau1 y 42.86% en las instancias Hard28. Mejorando en un total de 0.81% el desempeño del algoritmo para el subconjunto de instancias BPPLIB*.

A continuación, se presentan los resultados obtenidos por GGA-CGT/D, el algoritmo original y los mejores del estado del arte para las instancias BPP$v_{u-}c$. Cada tabla muestra la capacidad del contenedor (Columna C) y el número de instancias, para cada enfoque de solución, el número de instancias en las que el algoritmo obtiene una solución óptima (Columna Opt.) y el tiempo medio de ejecución, medido en segundos (Columna T). La última columna (% mejora) presenta cuánto mejoró la versión propuesta con respecto al algoritmo original en número de óptimos encontrados.

Tabla 5.25. Resultados obtenidos por el trabajo propuesto contra el algoritmo GGA-CGT original y los dos mejores del estado del arte para la clase de instancias BPP.25.

BPP.25										
C	**Inst.**	**General Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT/D (Versión propuesta)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.96	100	0.00	100	0.00	100	0.00	0
1000	100	100	1.09	100	0.01	100	0.02	100	0.02	0
10000	100	100	3.86	100	0.28	100	0.25	100	0.29	0
100000	100	100	9.15	100	5.97	100	6.23	100	6.47	0
1000000	100	100	76.84	100	51.64	100	54.12	100	57.95	0
10000000	100	100	3152.96	100	1721.1	100	1938.60	100	1941.19	0
100000000	100	87	5982.07	61	2164.52	6	2611.22	89	4025.19	83
Total	**700**	**687**	**9226.94**	**661**	**3943.53**	**606**	**4610.44**	**689**	**6031.10**	**11.85**
Gran Total			**301613.00**		**114076.00**		**118663.00**		**253651.00**	

De la tabla 5.25, se observa que la versión propuesta supera a todos los algoritmos seleccionados, encontrando el mayor número de soluciones con respecto a los demás del caso de estudio para la clase BPP.25. GGA-CGT/D mejora en un 11.85% la eficacia del algoritmo original.

Tabla 5.26. Resultados obtenidos por el trabajo propuesto contra el algoritmo GGA-CGT original y los dos mejores del estado del arte para la clase de instancias BPP.5.

BPP.5										
C	**Inst.**	**General Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT/D (Versión propuesta)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.75	100	0.00	100	0.01	100	0.03	0
1000	100	100	1.40	100	0.09	100	0.12	100	0.18	0
10000	100	100	3.93	100	2.12	100	2.05	100	2.14	0
100000	100	96	624.08	91	447.99	89	462.09	100	611.92	11
1000000	100	34	1448.21	28	981.14	1	907.70	46	1176.18	45
10000000	100	21	1187.95	16	962.97	8	941.99	34	1073.11	26
100000000	100	75	1092.36	42	892.17	35	878.22	63	995.85	28
Total	**700**	**526**	**4358.68**	**477**	**3286.48**	**433**	**3192.17**	**543**	**3859.40**	**15.71**
Gran Total			**536297.00**		**382551.00**		**372600.00**		**381981.00**	

De la tabla 5.26, se observa que la versión propuesta supera a todos los algoritmos seleccionados, encontrando el mayor número de soluciones con respecto a los demás del caso de estudio para la clase BPP.5. GGA-CGT/D mejora en un 15.71% la eficacia del algoritmo original.

Tabla 5.27. Resultados obtenidos por el trabajo propuesto contra el algoritmo GGA-CGT original y los dos mejores del estado del arte para la clase de instancias BPP.75.

BPP.75										
C	**Inst.**	**General Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT/D (Versión propuesta)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	1.99	98	0.02	95	0.03	100	1.95	5
1000	100	100	2.74	97	0.52	99	0.70	100	4.09	1
10000	100	38	130.56	19	3.86	0	5.05	48	172.24	48
100000	100	31	225.00	21	6.09	0	5.75	34	201.84	34
1000000	100	34	1902.87	13	2.05	0	2.20	39	2621.10	39
10000000	100	43	2971.90	24	1.24	0	1.57	39	3257.41	39
100000000	100	50	3824.04	36	0.98	0	1.46	42	3963.18	42
Total	**700**	**396**	**9059.10**	**308**	**14.75**	**194**	**16.77**	**402**	**10221.81**	**29.71**
Gran Total			**300123.00**		**1976.00**		**2047.00**		**319275.00**	

Tabla 5.28. Resultados obtenidos por el trabajo propuesto contra el algoritmo GGA-CGT original y los dos mejores del estado del arte para la clase de instancias BPP1.

BPP1										
C	**Inst.**	**General Arc-Flow Formulation (Exacto)**		**CNS_BP**		**GGA-CGT (Original)**		**GGA-CGT/D (Versión propuesta)**		
		Opt.	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**% mejora**
100	100	100	0.47	100	0.04	100	0.06	100	0.08	**0**
1000	100	100	2.41	100	0.82	100	0.93	100	1.08	**0**
10000	100	64	565.73	58	328.65	49	379.11	81	387.17	**32**
100000	100	87	369.74	69	273.98	73	293.66	94	302.98	**21**
1000000	100	100	131.09	99	92.71	94	95.96	100	104.15	**6**
10000000	100	100	32.47	100	16.10	100	19.55	100	21.09	**0**
100000000	100	100	19.46	100	6.25	100	8.22	100	13.20	**0**
Total	**700**	**651**	**1121.37**	**626**	**718.56**	**616**	**797.49**	**675**	**829.73**	**8.43**
Gran Total			**104852.00**		**73421**		**84534.00**		**87084.00**	

De la tabla 5.27, se observa que la versión propuesta supera a todos los algoritmos seleccionados, encontrando el mayor número de soluciones con respecto a los demás del caso de estudio para la clase BPP.75. GGA-CGT/D mejora en un 29.71% la eficacia del algoritmo original. Mientras que en la tabla 5.28, podemos ver nuevamente que la versión propuesta supera a todos los algoritmos seleccionados, encontrando el mayor número de soluciones con respecto a los demás del caso de estudio para la clase BPP1. GGA-CGT/D mejora en un 8.43% la eficacia del algoritmo original.

Conclusión del experimento

Con la experimentación realizada, se puede concluir que el trabajo propuesto mejora considerablemente el número de instancias resueltas de manera óptima. El algoritmo GGA-CGT/D logra resolver todo el subconjunto de instancias de prueba de BPPLIB* reduciendo el tiempo de ejecución del algoritmo exacto a casi el 50%, también obtuvo el mayor número de soluciones entre los algoritmos evaluados para las instancias BPP$v_{u-}c$. Resuelve a la optimalidad 3924 de 4415 instancias evaluadas, mejorando la eficacia total del algoritmo original en un 10.71%. De esta manera, la versión propuesta se posiciona como el mejor metaheurístico de entre los algoritmos seleccionados como caso de estudio.

5.5 Experimento 5: Análisis estadístico del desempeño de GGA-CGT/D

Objetivo: Analizar mediante soporte estadístico el desempeño de la versión que aquí se propone para realizar una comparación con los mejores enfoques en la literatura.

Configuración del experimento

La tabla 5.29 muestra la tasa de solución obtenida y la aptitud para cada una de las familias de las instancias BPPLIB* (Columna 2 y 3 dentro de cada algoritmo), este primer valor es calculado dividiendo el número de soluciones óptimas encontradas por cada algoritmo entre el número de óptimos en la instancia, mientras que la aptitud es un promedio de las instancias por familia. Cabe mencionar que el valor en la tasa de solución y la aptitud se multiplican por -1, ya que al ser un problema de maximización debe cambiarse el signo para convertirlo en minimización debido a que STAC así lo requiere. Para este conjunto de instancias se realizó la prueba de Friedman (Friedman, 1937) en

donde se evalúan a dos algoritmos (CNS_BP y General Arc-Flow Formulation) contra la propuesta, y la prueba de Wilcoxon (Wilcoxon, 1945) en donde se evalúa GGA-CGT/D contra CNS_BP y General Arc-Flow Formulation de manera independiente.

Tabla 5.29. Tasa de solución y valor de aptitud para las instancias BPPLIB* en cada uno de los algoritmos a evaluar en STAC.

Clase	Inst.	CNS_BP			Gral. Arc-Flow Formulation			GGA-CGT/D (Versión propuesta)		
		Opt.	Tasa de Sol.	Aptitud	Opt.	Tasa de Sol.	Aptitud	Opt.	Tasa de Sol.	Aptitud
Uniform	80	80	-1.00	-1.00	80	-1.00	-1.00	80	-1.00	-1.00
Triplets	80	80	-1.00	-1.00	80	-1.00	-1.00	80	-1.00	-1.00
Data Set 1	720	720	-1.00	-1.00	720	-1.00	-1.00	720	-1.00	-1.00
Data Set 2	480	480	-1.00	-1.00	480	-1.00	-1.00	480	-1.00	-1.00
Data Set 3	10	10	-1.00	-1.00	10	-1.00	-1.00	10	-1.00	-1.00
Was 1	100	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00
Was 2	100	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00
Gau 1	17	17	-1.00	-1.00	17	-1.00	-1.00	17	-1.00	-1.00
Hard28	28	20	-0.71	-0.9876	28	-1.00	-1.00	28	-1.00	-1.00
Total	**1615**	**1607**			**1615**			**1615**		
Gran Total										

La tabla 5.30 presenta la tasa de solución y la aptitud para cada una de las familias de las instancias BPP$v_{u_}c$ (Columna 2 y 3 respectivamente dentro de cada algoritmo). Al igual que la configuración anterior, para este conjunto de instancias se realizó la evaluación para los dos datos mencionados (tasa de solución y aptitud) mediante la prueba de Friedman (Friedman, 1937) en donde se evalúan a dos algoritmos (CNS_BP y General Arc-Flow Formulation) contra la propuesta, y la prueba de Wilcoxon (Wilcoxon, 1945) en donde se evalúa GGA-CGT/D contra CNS_BP y General Arc-Flow Formulation de manera independiente. Para estas instancias se presentan 28 valores en total, ya que existen cuatro clases y siete capacidades distintas de contenedor en cada una de ellas, se debe considerar que los valores de aptitud a evaluar son promedios de las instancias por familias.

Tabla 5.30. Tasa de solución y valor de aptitud para las instancias BPPv_u_c en cada uno de los algoritmos a evaluar en STAC.

Clase	C	CNS_BP			Gral. Arc-Flow Formulation			GGA-CGT/D (Versión propuesta)		
		Opt.	Tasa de Sol	Aptitud	Opt.	Tasa de Sol	Aptitud	Opt.	Tasa de Sol	Aptitud
BPP.25	100	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00
	1000	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00
	10000	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00
	100000	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00
	1000000	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00
	10000000	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00
	100000000	61	-0.61	-0.9787	87	-0.87	-0.9914	89	-0.89	-0.9931
BPP.5	100	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00
	1000	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00
	10000	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00
	100000	91	-0.91	-0.9979	96	-0.96	-0.9976	100	-1.00	-1.00
	1000000	28	-0.28	-0.9657	34	-0.34	-0.9772	46	-0.46	-0.9825
	10000000	16	-0.16	-0.9416	21	-0.21	-0.9698	34	-0.34	-0.9786
	100000000	42	-0.42	-0.9733	75	-0.75	-0.9919	63	-0.63	-0.9880
BPP.75	100	98	-0.98	-0.9977	100	-1.00	-1.00	100	-1.00	-1.00
	1000	97	-0.97	-0.9964	100	-1.00	-1.00	100	-1.00	-1.00
	10000	19	-0.19	-0.9470	38	-0.38	-0.9883	48	-0.48	-0.9885
	100000	21	-0.21	-0.9570	31	-0.31	-0.9836	34	-0.34	-0.9855
	1000000	13	-0.13	-0.9408	34	-0.34	-0.9871	39	-0.39	-0.9863
	10000000	24	-0.24	-0.9623	43	-0.43	-0.9886	39	-0.39	-0.9868
	100000000	36	-0.36	-0.9754	50	-0.50	-0.9912	42	-0.42	-0.9872
BPP1	100	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00
	1000	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00
	10000	58	-0.58	-0.9752	64	-0.64	-0.9854	81	-0.81	-0.9968
	100000	69	-0.69	0.9800	87	-0.87	-0.9975	94	-0.94	-0.9990
	1000000	99	-0.99	-0.9976	100	-1.00	-1.00	100	-1.00	-1.00
	10000000	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00
	100000000	100	-1.00	-1.00	100	-1.00	-1.00	100	-1.00	-1.00

Resultados

En la tabla 5.31 se observan los resultados estadísticos obtenidos para los algoritmos evaluados en la prueba de Friedman con un valor de significancia de 0.05 para las instancias BPPLIB* con los valores de tasa de solución y aptitud.

Tabla 5.31. Resultados estadísticos para los mejores algoritmos con la tasa de solución y aptitud para las instancias BPPLIB*.

Prueba de Friedman (nivel de significancia 0.05)			
Estadístico	**P-Value**	**Resultado**	
0.07477	0.92828	H0 se acepta	
Rank			
GGA_CGT/D (Versión propuesta)	1.94		
General Arc-Flow Formulation	1.94		
CNS_BP	2.11		
Comparación			
Algoritmos	**Estadístico**	**P-Value Ajustado**	**Resultado**
GGA_CGT/D vs CNS_BP	0.35355	1	H0 se acepta
GGA-CGT/D vs General Arc-Flow	0.00	1	H0 se acepta
CNS_BP vs General Arc-Flow	0.35355	1	H0 se acepta

Con los resultados obtenidos se puede observar que en los algoritmos GGA-CGT/D y General Arc-Flow Formulation no existe diferencia estadísticamente para las instancias BPPLIB*, sin embargo, ambos algoritmos muestran ser superiores a CNS_BP .

En la tabla 5.32 se presentan los valores obtenidos para el algoritmo GGA-CGT/D vs CNS_BP y GGA-CGT/D vs General Arc-Flow Formulation con los valores de la tasa de solución y aptitud de las instancias BPPLIB* para la prueba Wilcoxon con un nivel de significancia de 0.05.

Tabla 5.32. Resultados estadísticos para GGA-CGT/D vs CNS_BP y GGA-CGT/D vs General Arc Flow F. con la tasa de solución y aptitud para las instancias BPPLIB*.

Prueba de Wilcoxon (nivel de significancia 0.05)		
GGA-CGT/D vs CNS_BP		
Estadístico	**P-Value**	**Resultado**
0	0.0001170508	H0 se rechaza
Prueba de Wilcoxon (nivel de significancia 0.05)		
GGA-CGT/D vs General Arc-Flow Formulation		
Estadístico	**P-Value**	**Resultado**
21	0.21061862	H0 se acepta

Para determinar si estadísticamente existe diferencia entre GGA-CGT/D y CNS_BP; GGA-CGT/D y General Arc-Flow Formulation, se utilizó la prueba de Wilcoxon, la cual nos indica que existe evidencia para concluir que el algoritmo GGA-CGT/D es superior a CNS_BP, mediante la obtención de un P-Value de 0.0001170508. Sin embargo, también nos muestra nuevamente que el algoritmo GGA-CGT/D está en igualdad con respecto al algoritmo exacto en las instancias BPPLIB*.

En la tabla 5.33 se presentan los resultados estadísticos obtenidos para los algoritmos evaluados en la prueba de Friedman con un valor de significancia de 0.05 para las instancias BPP$v_{u-}c$ con los valores de tasa de solución.

Tabla 5.33. Resultados estadísticos para los mejores algoritmos con la tasa de solución para las instancias BPP$v_{u-}c$.

Prueba de Friedman (nivel de significancia 0.05)			
Estadístico	**P-Value**	**Resultado**	
7.91629	0.00097	H0 se rechaza	
Rank			
GGA_CGT/D (Versión propuesta)	1.625		
General Arc-Flow Formulation	1.83929		
CNS_BP	2.53571		
Comparación			
Algoritmos	**Estadístico**	**P-Value Ajustado**	**Resultado**
GGA_CGT/D vs CNS_BP	3.40758	0.00131	H0 se rechaza
CNS_BP vs General Arc-Flow	2.6058	0.01833	H0 se rechaza
GGA-CGT/D vs General Arc-Flow	0.8018	0.42268	H0 se acepta

En la tabla anterior podemos observar que para las instancias BPP$v_{u-}c$ con el valor de tasa de solución el algoritmo GGA-CGT/D es superior estadísticamente a CNS_BP, ya que al compararlos presenta un P-Value de 0.00131 a favor de GGA-CGT y ranquea en la primera posición con 1.625.

En la tabla 5.34 se presentan los resultados estadísticos obtenidos para los algoritmos evaluados en la prueba de Friedman con un valor de significancia de 0.05 para las instancias BPP$v_{u-}c$ con los valores de aptitud.

Tabla 5.34. Resultados estadísticos para los mejores algoritmos con los valores de aptitud para las instancias BPP$v_{u-}c$.

Prueba de Friedman (nivel de significancia 0.05)			
Estadístico	**P-Value**	**Resultado**	
6.56005	0.00282	H0 se rechaza	
Rank			
GGA_CGT/D (Versión propuesta)	1.66071		
General Arc-Flow Formulation	1.83929		
CNS_BP	2.5		
Comparación			
Algoritmos	**Estadístico**	**P-Value Ajustado**	**Resultado**
GGA_CGT/D vs CNS_BP	3.14032	0.00506	H0 se rechaza
CNS_BP vs General Arc-Flow	2.47217	0.02686	H0 se rechaza
GGA-CGT/D vs General Arc-Flow	0.6682	0.50404	H0 se acepta

En la tabla anterior podemos observar que para las instancias BPP$v_{u-}c$ con el valor de aptitud el algoritmo GGA-CGT/D es superior estadísticamente a CNS_BP, ya que presenta un P-Value de 0.00506 a favor de GGA-CGT/D y ranquea en la primera posición con 1.66071. Curiosamente al compararlo con el algoritmo exacto no se rechaza la hipótesis de que GGA-CGT sea mejor, a pesar de que en número de instancias resueltas la versión propuesta es superior.

En la tabla 5.35 se presentan los valores obtenidos para el algoritmo GGA-CGT/D vs CNS_BP y GGA-CGT/D vs General Arc-Flow Formulation con los valores de la tasa de solución de las instancias BPP $v_{u-}c$ para la prueba Wilcoxon con un nivel de significancia de 0.05. Mientras que en la tabla 5.36 se presenta la misma configuración, pero para los valores de aptitud.

Tabla 5.35. Resultados estadísticos para GGA-CGT/D vs CNS_BP y GGA-CGT/D vs General Arc Flow F. con la tasa de solución para las instancias BPP$v_{u-}c$.

Prueba de Wilcoxon (nivel de significancia 0.05)		
GGA-CGT/D vs CNS_BP		
Estadístico	**P-Value**	**Resultado**
0	0.000654958	H0 se rechaza
Prueba de Wilcoxon (nivel de significancia 0.05)		
GGA-CGT/D vs General Arc-Flow Formulation		
Estadístico	**P-Value**	**Resultado**
19.5	0.12594383	H0 se acepta

Tabla 5.36. Resultados estadísticos para GGA-CGT/D vs CNS_BP y GGA-CGT/D vs General Arc Flow F. con los valores de aptitud para las instancias BPPv_u_c.

Prueba de Wilcoxon (nivel de significancia 0.05)		
GGA-CGT/D vs CNS_BP		
Estadistico	**P-Value**	**Resultado**
0	0.000654958	H0 se rechaza
Prueba de Wilcoxon (nivel de significancia 0.05)		
GGA-CGT/D vs General Arc-Flow Formulation		
Estadistico	**P-Value**	**Resultado**
24	0.239316541	H0 se acepta

En las últimas dos tablas presentadas se compara al trabajo propuesto contra CNS_BP y General Arc Flow Formulation con los valores de tasa de solución y aptitud para las instancias BPPv_u_c. En ambos resultados se observa que estadísticamente el algoritmo GGA-CGT/D es superior a CNS_BP con un P-Value de 0.00065 en ambas ocasiones, sin embargo, ocurre el mismo efecto en el que se rechaza la hipótesis de que GGA-CGT/D es mejor que el algoritmo exacto a pesar de mostrar inferioridad hablando de número de soluciones óptimas encontradas.

Conclusiones del experimento

Con la experimentación anterior, se puede concluir que la versión propuesta (GGA-CGT/D) es superior a CNS_BP estadísticamente hablando y es superior a General Arc-Flow Formulation en número de instancias resueltas para el caso de las BPPv_u_c. La comparación estadística entre metaheurísticos revela que el algoritmo propuesto es superior. La comparación estadística con el algoritmo exacto revela que ambos algoritmos alcanzan desempeños similiares. Aunque el soporte estadístico dicte que ambos algoritmos están en condiciones muy parecidas, la comunidad científica le ha dado mucho peso al hecho de que un algoritmo resuelva más instancias que otro, ya que ha sido un método que se ha utilizado en diversos trabajos (Alvim et.al, 2004; Fleszar&Charalambous, 2011; Brandao&Pedroso, 2013; Quiroz, 2014; Buljubašić&Vasquez, 2016) como medida de desempeño para la evaluación de algoritmos.

Capítulo 6

Conclusiones

En este capítulo se describen las principales aportaciones de esta investigación. Cabe destacar que se cumplió satisfactoriamente el objetivo general del proyecto, ya que se logró estudiar tanto teórica como experimentalmente los algoritmos seleccionados del estado del arte enfocados al 1D-BPP. Así como los objetivos específicos, implementando las estrategias más representativas de los algoritmos seleccionados en la versión propuesta e incluyendo la comparación con soporte estadístico de los algoritmos originales y el actualizado (Sección 5.4 y 5.5). También se presentan algunas posibilidades de trabajo futuro.

6.1 Aportaciones

A continuación, se listan las principales contribuciones de la tesis:

1. Este documento aporta el desarrollo de una versión mejorada del algoritmo genético GGA-CGT. Esta nueva versión cuenta con la inclusión de tres diferentes métodos que ayudan a mejorar significativamente el desempeño del algoritmo original (ver sección 4.4 y 5.4).
2. Un análisis comparativo de algoritmos para 1D-BPP desarrollados en la última década, con el fin de reportar los diferentes enfoques, técnicas y estrategias que se han desarrollado a través de los años y que pueden aplicarse a los algoritmos con el fin de mejorar su desempeño (ver sección 3.1).
3. El análisis y descripción del funcionamiento del algoritmo GGA-CGT original y la nueva versión propuesta GGA-CGT/D, así como la descripción detallada de los métodos implementados en el desarrollo de éste (ver capítulo 4). El método de límites logró mejorar la eficacia del algoritmo original en un 1.38%, mientras que el método de reducción y diversificación mejoraron 3.35% y 5.23% respectivamente. Por lo tanto, el método que más contribuyó a la mejora del algoritmo fue el método de diversificación. (resultados detallados en capítulo 5)

4. Se presentan los resultados de las diversas experimentaciones realizadas a lo largo del desarrollo de esta propuesta (métodos implementados de manera independiente y en conjunto) y la comparación contra los resultados de los algoritmos de la literatura, que hasta nuestro conocimiento son los más destacados por su desempeño (ver capítulo 5).
5. De acuerdo con los resultados experimentales se obtuvo lo siguiente:

    a) GGA-CGT/D mejora considerablemente el número de instancias resueltas de manera óptima, logra resolver en su totalidad el subconjunto de instancias de prueba de BPPLIB* (1615 instancias) reduciendo el tiempo de ejecución del algoritmo exacto a casi el 50%.
    b) Obtuvo el mayor número de soluciones entre los algoritmos evaluados para las instancias BPP$v_{u-}c$ resolviendo un total de 2309 de 2800 instancias.
    c) Mejora el desempeño del algoritmo original (GGA-CGT) y el del heurístico CNS_BP en un 10.71% y 5.55% respectivamente, mientras que para el algoritmo exacto mejora un 1.11%
    d) El número de instancias resueltas a la optimalidad muestra correspondencia con las medidas de la complejidad de la literatura estudiadas, como:
        - Bin capacity
        - Range in which the weights are distributed (ver sección 4.1 y anexo C).

    e) GGA-CGT/D logra posicionarse como el mejor de los algoritmos metaheurísticos ganándole estadísticamente a CNS_BP con un P-Value de 0.000654958.

6.2 Difusión de la investigación

A partir del trabajo realizado se derivaron las siguientes participaciones en eventos y publicaciones:

- Artículos en capítulos de libro:

  González-San Martín J.E., Cruz-Reyes L., Dorronsoro B., Quiroz-Castellanos M., & Rangel-Valdez N. (2020). Estudio comparativo de heurísticas para el problema de empacado de objetos de una dimensión. En *La investigación científica y tecnológica impulsando la creatividad*

para innovar. (pp. 485-496). Academia Mexicana Multidisciplinaria A.C.

- Participación en eventos:

  1. Exposición del trabajo "Estudio formal de metaheurísticas y heurísticas para el problema de empacado de objetos de una dimensión" en el *7º Encuentro de Jóvenes Investigadores de Tamaulipas organizado por el COTACYT*. Noviembre de 2019.

  2. Artículo: "Estudio comparativo de heurísticas para el problema de empacado de objetos de una dimensión" para Evento *AMM (Academia Mexicana Multidisciplinaria)*. Agosto de 2020.

  3. Artículo "The Bin Packing Optimization Problem: Algorithm Analysis and Open Problems" para Evento *NEO 2020 (Numerical and Evolutionary Optimization 2020)*. Noviembre de 2020.

6.3 Trabajos futuros

En base a la experiencia obtenida con el desarrollo de este trabajo se pudieron identificar algunas líneas de investigación y actividades que podrían realizarse en un futuro:

1) Diseño de nuevos indicadores de desempeño basado en información de la literatura, por ejemplo: (Quiroz, 2014) menciona que muchas instancias son difíciles cuando el peso promedio de los objetos ocupa alrededor del 0.3 de la capacidad del contenedor.
2) Desarrollo de un método de control de la diversidad basado en información del proceso evolutivo, por ejemplo, el número de fitness repetido (ver sección 5.4).
3) Experimentación calculando diferentes tiempos de ejecución:
   a. Cuando se encuentra el límite inferior.
   b. Cuando encuentra la mejor solución.
   c. Cuanto tarda en evaluar todas las condiciones.
   d. Tiempo de reloj usado por el algoritmo exacto.
4) Integración de procesos de construcción (Alvim, 2004) y destrucción (Buljubašić&Vasquez, 2016).

5) Solución aproximada del modelo matemático basado en flujo de arco (Brandao&Pedroso, 2013):
    a. Codificación de las variables enteras del modelo matemático en cromosoma.
    b. Método para convertir las instancias de BPP en las variables que se codifiquen en el problema (variables del punto a).
    c. Método para interpretar la solución que está codificada en el cromosoma e interpretarla en Bins.
    d. Un algoritmo evolutivo general.
6) Desarrollo de una hiper-heurística para seleccionar entre dos metaheurísticas diferentes (Sim&Hart, 2013), (Chaurasia& Kim 2019).
7) Integración de diferentes funciones de aptitud:
    a. Generar una nueva a partir de las de los casos de estudio o
    b. Seleccionar la que mejor se adapte a cierta instancia del problema.
8) Proceso de caracterización de instancias en el proceso de búsqueda de algoritmos metaheurísticos.

Bibliografía

(Abdel et.al., 2018) Abdel-Basset M., Manogaran G., Abdel-Fatah ., L., y Mirjalili S., “An improved nature inspired meta-heuristic algorithm for 1-D bin packing problems”, *Pers. Ubiquitous Comput.*, vol. 22, núm. 5–6, oct. 2018, pp. 1117–1132, doi: 10.1007/s00779-018-1132-7.

(Almeida& Steiner, 2015) Almeida R. y Steiner M. T. A., “Resolution of 1-D Bin Packing Problem using Augmented Neural Networks and Minimum Bin Slack”, presentado en *2015 Latin America Congress on Computational Intelligence (LA-CCI)*, Curitiba, Brazil, 2015, pp. 1–6, doi: 10.1109/LA-CCI.2015.7435943.

(Alvim, et al., 2004) Alvim C.F Adriana, Ribeiro Celso C., Glover Fred, and Aloise Dario J. 2004. “A Hybrid Improvement Heuristic for the One-Dimensional Bin Packing Problem.” *Journal of Heuristics* 10 (March): 205–29. https://doi.org/https://doi.org/10.1023/B:HEUR.0000026267.44673.ed.

(Beasley, 1990) Beasley J. E. OR-librery: Distributing test problems by electrnic mail. Jonural of the Operational Research Society, 41: 1069-1072, 1990. Disponible en internet vía http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.html

(Belov, 2004) Belov G. Problems, "Models and Algorithms in One and Two Dimensional Cutting". PHD thesis. Fakultät Mathematik und Naturwissenschaften der Technischen Universität Dresden, 2004.

(Brandao&Pedroso, 2013) Brandao Filipe, and Pedroso Joao Pedro. 2013. “Bin Packing and Related Problems: General Arc-Flow Formulation with Graph Compression.” Portugal: Universidade Do Porto.

(Brereton, et. al., 2007) Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. 2007. "Lessons from applying the systematic literature review process within the software engineering domain". *The Journal of Systems and Sofware, 80*, 571-583.

(Buljubašić&Vazquez, 2016) Buljubašić Mirsad, and Vasquez Michel. 2016. “Consistent Neighborhood Search for One-Dimensional Bin Packing and Two-Dimensional Vector Packing.” *Computers & Operations Research* 76 (December): 12–21. https://doi.org/10.1016/j.cor.2016.06.009.

(Carmona-Arroyo, el. al., 2021) Carmona-Arroyo G., Vázquez-Aguirre J.B., Quiroz-Castellanos M. (2021) One-Dimensional Bin Packing Problem: An Experimental Study of Instances Difficulty and Algorithms Performance. In: Castillo O., Melin P. (eds) Fuzzy Logic Hybrid Extensions of Neural and Optimization Algorithms: Theory and Applications. Studies in Computational Intelligence, vol 940. Springer, Cham. https://doi.org/10.1007/978-3-030-68776-2_10

(Cruz, 1999) Cruz Reyez Laura. 1999. “Automatización del diseño de la fragmentación vertical y ubicación de bases de datos distribuidas utilizando métodos heurísticos y exactos.” Instituto Tecnológico y de Estudios Superiores de Monterrey.

(Cruz-Reyes et.al., 2012) Cruz-Reyes L., Quiroz-Castellanos M., Alvim A. C., Fraire H., Gómez-Santillán C., y Torres-Jimenez J., “Efficient Hybrid Grouping Heuristics for the Bin Packing Problem”, *Comput. Sist.*, vol. 16, núm. 3, 2012.

(Czachórski et. al, 2016) Czachórski T., Gelenbe E.,. Grochla K, y Lent R., Eds., *Computer and Information Sciences: 31st International Symposium, ISCIS 2016, Kraków, Poland, October 27–28, 2016, Proceedings*, vol. 659. Cham: Springer International Publishing, 2016.

(Delorme, et. al., 2018) Delorme, M., Iori, M. & Martello, S. BPPLIB: a library for bin packing and cutting stock problems. Optim Lett 12, 235–250 (2018). https://doi.org/10.1007/s11590-017-1192-z

(Díaz, et al., 1996) Díaz Fernández A., González Velerde J.L, Laguna M., Moscato P., Glover F., and Ghaziri M. H. 1996. *Optimización Heurística y redes neuronales*. Paraninfo. Madrid.

(Dokeroglu&Cosar, 2014) Dokeroglu T. y Cosar A., “Optimization of one-dimensional Bin Packing Problem with island parallel grouping genetic algorithms”, *Comput. Ind. Eng.*, vol. 75, sep. 2014, pp. 176–186, doi: 10.1016/j.cie.2014.06.002.

(Duarte, et al., 2007) Duarte A., Pantrigo J.J, Gallego M.. 2007. *Metaheuristicas*. Universidad Rey Juan Carlos.

(Falkenauer,1994) Falkenauer Emanuel. 1994. “A New Representation and Operators for Genetic Algorithms Applied to Grouping Problems.” *Evolutionary Computation* 2: 123–44. https://doi.org/https://doi.org/10.1162/evco.1994.2.2.123.

(Falkenauer,1996) Falkenauer Emanuel. 1996. “A Hybrid Grouping Genetic Algorithm for Bin Packing” 2 *(Journal of Heuristics*): 5–30.
(Fekete&Schepers, 2001). Fekete, S. P, Schepers, J. 2001. "New classes of fast lower bounds for bin packing problems". *Mathematical Programming* 91, 11_31.

(Fleszar&Charalambous, 2011) Fleszar Krzysztof, and Charalambous Christoforos. 2011. “Average-Weight-Controlled Bin-Oriented Heuristics for the One-Dimensional Bin-Packing Problem.” *European Journal of Operational Research* 210 (2): 176–84. https://doi.org/10.1016/j.ejor.2010.11.004.

(Fleszar&Hindi, 2002) Fleszar Krzysztof, and Hindi Khalil S. 2002. “New Heuristics for One-Dimensional Bin-Packing.” *Computers & Operations Research* 29: 821–39.

(Friedman,1937) Friedman, M. 1937. “The use of ranks to avoid the assumption of normality implicit in the analysis of variance” *Journal of the American Statistical Association* 32, 674–701.

(García, et.al, 2007) García S., Molina D., Lozano M. and Herrera F. 2007. "Un estudio experimental sobre el uso de test no paramétricos para analizar el comportamiento de los algoritmos evolutivos en problemas de optimización". In *Actas del Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*,(MAEB07) (pp. 275-285).

(García, et.al, 2010) García, S., Fernández, A., Luengo, J., and Herrera, F. (2010). "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power". *Information sciences*, 180(10), 2044-2064.

(Garey&Johnson, 1979) Garey M., and Johnson D. 1979. “Computers and Intractability: A Guide to the Theory of NP-Completeness.”

(Gilmore&Gomory, 1963) Gilmore P. C., and Gomory R. E. 1963. “A Linear Programming Approach to the Cutting Stock Problem—Part II,” no. *Annals of Operations Research*: 863–88. https://doi.org/https://doi.org/10.1287/opre.11.6.863.

(Kaaouache&Bouamama, 2015) Kaaouache M. A. and Bouamama S., “Solving bin Packing Problem with a Hybrid Genetic Algorithm for VM Placement in Cloud”, *Procedia Comput. Sci.*, vol. 60, 2015, pp. 1061–1069, doi: 10.1016/j.procs.2015.08.151.

(Kantorovich, 1960) Kantorovich, L. V. 1960. “Mathematical Methods of Organizing and Planning Production.” *Management Science* 6 (4): 366–422. https://doi.org/10.1287/mnsc.6.4.366.

(Keeke, 2007) Keeke, S. 2007. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE.

(Klein, 2009) Klein R., Scholl A. *Bin Packing*. Disponible en internet vía http://www.wiwi.uni-jena.de/Entscheidung/binpp/

(Kucukyilmaz& Kiziloz, 2018) Kucukyilmaz T. and Kiziloz H. E., “Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem”, *Comput. Ind. Eng.*, vol. 125, nov. 2018, pp. 157–170, doi: 10.1016/j.cie.2018.08.021.

(Martello&Toth, 1990) Martello Silvano, and Toth Paolo. 1990. “Lower Bounds and Reduction Procedures for the Bin Packing Problem.” *Discrete Applied Mathematics*, 59–70. https://doi.org/10.1016/0166-218X(90)90094-S.

(Newell, Shaw,&Simon, 1963) Newell A., Shaw J.C., and Simon H. 1963. “A Empirical Explorations with the Logic Theory Machine: A Case History in Heuristics In Feigenbaum and Feldman,” 109–33.

(Laveb&Chenche, 2012) Layeb A. y Chenche S., “A Novel GRASP Algorithm for Solving the Bin Packing Problem”, *Int. J. Inf. Eng. Electron. Bus.*, vol. 4, núm. 2, abr. 2012, pp. 8–14, doi: 10.5815/ijieeb.2012.02.02.

(Layeb& Boussalia, 2012) Layeb A. and Boussalia S. R, “A Novel Quantum Inspired Cuckoo Search Algorithm for Bin Packing Problem”, *Int. J. Inf. Technol. Comput. Sci.*, vol. 4, núm. 5, may 2012, pp. 58–67, doi: 10.5815/ijitcs.2012.05.08.

(Pérez-Ortega, et. al., 2016) Pérez-Ortega J., Castillo-Zacatelco H., Vilariño-Ayala D., Mexicano-Santoyo A., Zavala-Díaz J. C., Martínez-Rebollar A. and Estrada-Esquivel H. 2016. “A New Heuristic Strategy for the Bin Packing Problem” Ing. invest. y tecnol. vol.17 no.2, Junio 2016.

(Quiroz, 2014) Quiroz Castellanos Marcela. 2014. “Caracterización del proceso de optimización de algoritmos heurísticos aplicados del problema de empacado de objetos en contenedores.” Tesis, Instituto Tecnológico de Tijuana.

(Quiroz, 2018) Quiroz-Castellanos, M., "Entendiendo el proceso de optimización de algoritmos heurísticos", *XXVII Escuela Nacional de Optimización y Análisis Numérico*, 2018

(McGuire, 2001) McGuire, J. 2001. "What works in correctional intervention?Evidence and Practical Implications". *OffenderRehabilitation in Practice*, 25-43.

(Rodríguez, et. al., 2015) Rodríguez-Fdez I., Canosa A, Mucientes M., Bugarín A., " STAC: a web platform for the comparison of algorithms using statistical tests ", *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Istanbul, 2015, pp. 1-8, doi: 10.1109/FUZZ-IEEE.2015.7337889.

(Schiavinotto, T., & Stützle, T., 2004). "The linear ordering problem: Instances, search space analysis and algorithms". *Journal of Mathematical Modelling and Algorithms*, *3*(4), 367-402.

(Schoenfield, 2002) Schoenfield J. E. *Fast, exact solution of open bin packing problems without linear programming*. Draft, US Army Space & Missile Defense Command, Huntsville, Alabama, USA, 2002.

(Scholl,Klein&Jurgens, 1997) Scholl, A., Klein, R., Jurgens, C., 1997. Bison: "A fast hybrid procedure for exactly solving the one-dimensional bin packing problem". *Computers & Operations Research* 24 (7), 627 _ 645.

(Schwerin, 1997) Schwerin P., Wäscher G. *The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP*. International Transactions in Operational Research, 4: 337-389, 1997.

(Schwerin, 1999) Schwerin P., Wäscher G. *A new lower bound for the bin-packing problem and its integration to MTP*. Pesquisa Operacional, 19: 111-129, 1999.

(Sevaux, et. al, 2010) Sevaux M., Rossi A., Coussy P., Söresnen K. *Metaheuristics*. 2010.

(Sim&Hart, 2012) Sim K., Hart E., y Paechter B., “A Hyper-Heuristic Classifier for One Dimensional Bin Packing Problems: Improving Classification Accuracy by Attribute Evolution”, en *Parallel Problem Solving from Nature - PPSN XII*, vol. 7492, 2012, pp. 348–357.

(Sim&Hart, 2013) Sim K. y Hart E., “Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model”, presentado en *GECCO ’13:*

Proceedings of the 15th annual conference on Genetic and evolutionary computation, Amsterdam, The Netherlands, 2013, pp. 1549–1556, doi: https://doi.org/10.1145/2463372.2463555.

(Sotelo-Figueroa, et. al., 2013) Sotelo-Figueroa M.A., Soberanes H.J.P., Carpio J.M., Fraire Huacuja H.J., Reyes L.C., Soria Alcaraz J.A. (2013) "Evolving Bin Packing Heuristic Using Micro-Differential Evolution with Indirect Representation". In: Castillo O., Melin P., Kacprzyk J. (eds) *Recent Advances on Hybrid Intelligent Systems. Studies in Computational Intelligence*, vol 451. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-33021-6_28

(Talbi, 2009) Talbi E. G. 2009. "Metaheuristics: from design to implementation". Vol. 74. Universidad de Lille: John Wiley & Sons.

(Unibo) UNIBO 2020, Operation Research Group Bologna, BPPLIB – A Bin Packing Problem Library, Disponible en internet vía http://or.dei.unibo.it/library/bpplib

(Tugrul et.al, 2014) Tugrul B., Mehmet A., y Muharrem D., "A memory-integrated artificial bee algorithm for 1-D bin packing problems", presentado en *CIE44 & IMSS'14 Proceedings*, Istanbul, Turkey, 2014, pp. 1023–1034.

(Valério de Carvalho, 1999) Valério de Carvalho, J. M. 1999. "Exact solution of bin-packing problems using column generation and branch-and bound" 86 *(Annals of Operations Research)*: 629–59.

(Wäscher, 1996) Wäscher G., Gau T. "Heuristics for the one-dimensional cutting stock problem: A computational study". *OR Spektrum*, 18: 131-144, 1996.

(Wilcoxon, 1945) Wilcoxon, F. (1945). "Individual comparisons by ranking methods". *Biometrics bulletin*, 80-83.

Anexo A: Comparación de estrategias de algoritmos para BPP en la última década

En este anexo se presenta una tabla integradora del tema presentado en el Capítulo 3. En la tabla A.1 se muestra una comparativa de estrategias que componen a diecisiete algoritmos de la literatura desarrollados en la última década y que se enfocan en el problema de empacado de objetos de una dimensión.

Tabla A.1. Estrategias principales en algoritmos de la literatura desarrollados en la última década.

Trabajo	Algoritmo	Año	Técnica/Estrategia
Hybrid Improvement Heuristic for 1D-BPP (Alvim, et.al., 2004)	HI_BP	2004	Heurística Híbrida
Average-weight-controlled bin-oriented heuristics for the 1D-BPP (Fleszar&Charalambous, 2011)	Pert-SAWMBS	2011	Búsqueda Local
Hyper-Heuristic Classifier for 1D-BPP (Sim&Hart, 2012)	HHC-BP	2012	Hiper Heurística con módulo clasificador (KNN).
A Novel GRASP Algorithm for Solving the Bin Packing Problem (Layeb&Chenche, 2012)	N_GRASP	2012	Algoritmo GRASP y Búsqueda Local
A Novel Quantum Inspired Cuckoo Search Algorithm for BPP (Layeb&Boussalia, 2012)	QICSABP	2012	Búsqueda Local
Efficient Hybrid Grouping Heuristics for the Bin Packing Problem (Cruz-Reyes, et. al., 2012)	HGGA-BP	2012	Algoritmo Genético Híbrido
General Arc-Flow Formulation with Graph Compression (Brandao&Pedroso, 2013)	Gral. Arc-Flow F.	2013	Flujo de arco y compresión de gráficos
Generating Single and Multiple Cooperative Heuristics for 1D-BPP Using a Single Node Genetic Programming Island Model (Sim&Hart, 2013)	SNGP-BP	2013	Hiper heurística, Programación Genética de Nodo Único y modelos de isla.
Evolving Bin Packing Heuristic Using Micro-Differential	µ*DE*	2013	Algoritmo de evolución diferencial y representación indirecta

Evolution with Indirect Representation (Sotelo-Figueroa, et. al., 2013)			
A Memory-Integrated Artificial Bee Algorithm For 1D-BPP (Trugul, et.al. ,2014)	MEABC	2014	Algoritmo Colonia de Abejas Artificiales (ABC) y búsqueda local
Optimization of 1D-PP with island parallel grouping genetic algorithms (Dokeroglu&Cosar, 2014)	Parallel Falk-MBS-BFD	2014	Metaheurística genética de agrupación evolutiva con computación paralela
Grouping Genetic Algorithm with Controlled Gene Transmission (Quiroz, 2014)	GGA-CGT	2014	Algoritmo Genético
Augmented Neural Networks and Minimum Bin Slack for 1D-BPP (Almeida&Steiner, 2015)	AugNN-MBS	2015	Redes neuronales
Solving bin packing problem with a hybrid genetic algorithm for VM placement in cloud (Kaaouache&Bouamama, 2015)	HGBF_BP	2015	Algoritmo Genético Híbrido
A Novel Grouping Genetic Algorithm for the 1D-BPP on GPU (Czachórski, et. al., 2016)	1D-BPP-CUDA	2016	Algoritmo genético en GPU (Graphics Processing Unit) usando CUDA.
Consistent Neighborhood Search for One-Dimensional Bin Packing (Buljubašić&Vasquez, 2016)	CNS_BP	2016	Búsqueda Local
Improved Lévy-based whale optimization algorithm (Abdel-Basset, et. al., 2018)	ILWOA	2018	Algoritmo de optimización de ballenas (WOA) y vuelo de Lévy
Cooperative parallel grouping genetic algorithm with controlled gene transmission for the 1D-BPP (Kucukyilma&Kiziloz, 2018)	IPGGA	2018	Algoritmo genético de agrupación paralela cooperativa

En la primera columna se encuentra el nombre del trabajo y sus respectivos autores, en “Algoritmo” se encuentra el nombre abreviado de los algoritmos, en la tercera columna podemos encontrar el año de publicación de cada trabajo y en la última columna tenemos las estrategias o técnicas más representativas que fueron utilizadas en cada uno de los algoritmos.

Anexo B: Resultados reportados en la literatura de los algoritmos destacados para BPP

En la tabla B.1 se muestran los resultados detallados de los cinco algoritmos destacados para Bin Packing y el trabajo propuesto para el subconjunto de instancias BPLIB*.

Tabla B.1. Resultados reportados en la literatura de los algoritmos destacados para BPP.

Clase	**Inst.**	**HI_BP**		**Pert-SAWMBS**		**Gral. Arc-Flow F.**		**CNS_BP**		**GGA-CGT**		**GGA-CGT/D** (Este trabajo)	
		Opt.	**T(s)**	**Opt.**	**T(s).**	**Opt.**	**T(s).**	**Opt.**	**T(s)**	**Opt.**	**T(s)**	**Opt.**	**T(s)**
Uniform	80	80	0.03	79	0.00	80	1.36	80	0.07	80	0.23	80	2.35
Triplets	80	80	0.98	80	0.00	80	3.66	80	0.02	80	0.41	80	3.44
Set_1	720	720	0.19	720	0.01	720	0.15	720	0.07	720	0.35	720	18.85
Set_2	480	480	0.01	480	0.00	480	43.43	480	0.03	480	0.12	480	7.13
Set_3	10	10	4.60	10	0.16	10	12.17	10	0.00	10	1.99	10	47.25
Was_1	100	100	0.02	100	0.00	100	0.67	100	0.00	100	0.00	100	0.19
Was_2	100	100	0.02	100	0.01	100	0.57	100	0.00	100	1.07	100	16.14
Gau_1	17	12	0.60	16	0.04	17	2458.32	17	2.68	16	0.27	17	4.73
Hard28	28	5	-	5	0.24	28	29.69	25	7.21	16	2.40	28	22.10
Total	**1615**	**1587**		**1590**		**1615**		**1612**		**1602**		**1615**	
Equipos		Pentium IV 1.7 GHz		Intel core2 Q8200 2.33GHz		2 x Quad-Core Intel Xeon at 2.66GHz		Intel i7-3770 3.40GHz		Core2 Duo CE6300 1.86GHz		Intel Xeon E5-2650 2.30GHz	

En la primera columna se encuentran los nombres de cada familia de instancias, seguido del número de instancias a resolver dando un total de 1615 casos de prueba que componen al subconjunto BPPLIB*. Después de presentan los algoritmos más destacados del estado del arte con dos columnas: “Opt.” representa el número de soluciones optimas que pudo encontrar cada algoritmo, “T(s)” representa el tiempo de ejecución promedio expresado en segundos, para cada familia de instancias. En la fila “Total” se muestra el número total de soluciones optimas obtenidas por cada algoritmo. La última fila “Equipos” muestra las diferentes condiciones en donde fueron ejecutados los algoritmos, esto se encuentra reportado en cada uno de los trabajos seleccionados.

Anexo C: Resultados de los indicadores Gap, t y rango para las instancias evaluadas

En este anexo se presentan los valores obtenidos por los indicadores GAP, t y rango para las instancias BPPLIB* y BPPv_u_c. El resultado es el promedio de instancias por familia y cada indicador se obtuvo con las ecuaciones dadas en la tabla C.1. En la primera columna se encuentran las familias de instancias evaluadas del subconjunto BPPLIB*, en la segunda, tercera y cuarta columna se encuentran los valores promedio por familia del indicador GAP, t y rango respectivamente.

Tabla C.1. Resultados de los indicadores GAP, t y rango para el subconjunto de instancias BPPLIB*.

Instancias	GAP	t	Rango
Uniform	0.006083524	0.402344972	79.825
Triplets	0	0.333333333	24.65625
Data Set 1	0.050214106	0.487881539	82.08055556
Data Set 2	0.028810045	0.195439788	203.2166667
Data Set 3	0.021180488	0.27503025	14884.5
Was 1	0.029876667	0.1746222	49.66
Was 2	0.045513182	0.17498925	49.73
Gau 1	0.007958993	0.157748256	3686.176471
Hard28	0.00319348	0.386731677	746.25

Tabla C.2. Resultados de los indicadores GAP, t y rango para las instancias BPPv_u_c.

Instancias	GAP	t	Rango
BPP	0	0.36826834	15690523.4
BPP.25	0	0.11588729	3935377.12
BPP.5	0	0.2138609	7856500.9
BPP.75	0	0.30137238	11752197.1

Realizando una revisión preliminar de las tablas C.1 y C.2 se encontró lo siguiente. Las familias subrayas en amarillo obtuvieron un Gap de 0 o cercano a 0 y un rango intermedio, casualmente estas instancias fueron las más retadoras para los algoritmos. No obstante, existen otras instancias que obtuvieron un Gap de 0, como es el caso de las tripletas y las instancias BPPv_u_c. Esto nos dice que este indicador (Gap) puede servirnos como una guía para conocer la

dificultad de una instancia, sin embargo, no es contundente y cabe la posibilidad de que existan otros factores que podrían indicarnos dicha dificultad.

En la tabla C.3 se presentan los resultados de correlación entre los indicadores GAP, t y rango para las familias que obtuvieron un índice de correlación más alto, las celdas marcadas en amarillo corresponden a estos valores.

Tabla C.3. Matriz de correlación entre los indicadores GAP, t y rango para el conjunto de instancias de las familias, Data Set 1, 3, Was 1 y Hard28.

Correlación				
Data Set 1		**GAP**	**t**	**Rango**
	GAP	1	0.783	-0.480
	t	0.783	1	-0.515
	Rango	-0.480	-0.515	1
Data Set 3		**GAP**	**t**	**Rango**
	GAP	1	0.199	0.358
	t	0.199	1	0.663
	Rango	0.358	0.663	1
Gau 1		**GAP**	**t**	**Rango**
	GAP	1	0.425	0.287
	t	0.425	1	0.943
	Rango	0.287	0.943	1
Hard28		**GAP**	**t**	**Rango**
	GAP	1	0.424	0.089
	t	0.424	1	0.657
	Rango	0.089	0.657	1

Este análisis de correlación permite identificar índices con un alto grado de asociación entre ellos, que posiblemente miden un mismo factor. En la tabla C.4 se presenta de manera detallada los valores para cada indicador (GAP, r y rango) obtenidos para una muestra de las instancias evaluadas. Dicha muestra corresponde a un número de 31 instancias.

Tabla C.4. Resultados de los indicadores GAP, t y rango para una amuestra de 31 instancias.

INDICADORES				
	Instancias	**GAP**	**t**	**Rango**
1	N1c1w1_a.txt	0.0264	0.4868	96
2	N2c2w4_b.txt	0.104305556	0.537416667	70
3	N3c1w2_n.txt	0.082573529	0.62385	80
4	N4c3w2_b.txt	0.002495895	0.404986667	80
5	N1w1b1r6.txt	0.014058824	0.33522	119
6	N2w3b2r1.txt	0.005214286	0.13927	138
7	N3w4b2r5.txt	0.010391304	0.113805	108
8	N4w3b3r0.txt	0.000625	0.14391	250
9	Hard4.txt	0.026193509	0.27753485	14861
10	t60_10.txt	0	0.333333333	24.1
11	t120_15.txt	0	0.333333333	23.7
12	t249_04.txt	0	0.333333333	24.9
13	t501_17.txt	0	0.333333333	24.8
14	u120_04.txt	0.019466667	0.408555556	79
15	u250_14.txt	0.008333333	0.396666667	80
16	u500_13.txt	0.002108844	0.391173333	80
17	u1000_19.txt	0.001666667	0.399333333	80
18	hBPP742.txt	0.001	0.3996	788
19	hBPP359.txt	0.015118421	0.415838889	793
20	BPP20.txt	0.022666667	0.17592	50
21	BPP_91.txt	0.050727273	0.174033333	49
22	TEST0044.txt	0.000078571	0.085359146	2450
23	TEST0065.txt	0.0626625	0.249956667	4820
24	BPP_100_26.txt	0	0.384615385	99
25	BPP_100000_79.txt	0	0.4	98078
26	BPP.25_10000_47.txt	0	0.125	2498
27	BPP.25_100000000_99.txt	0	0.118110236	24937797
28	BPP.5_100000_14.txt	0	0.214285714	49813
29	BPP.5_10000000_56.txt	0	0.198675497	4967066
30	BPP.75_100000_12.txt	0	0.310344828	74759
31	BPP.75_100000000_17.txt	0	0.274390244	74739463

Anexo D: Tiempo de ejecución detallado de la experimentación final

En este anexo se presenta un detalle de los tiempos de ejecución para cada una de las experimentaciones presentadas en la sección 5. En las tablas siguientes se muestra para cada clase de instancias, el número de instancias y, para cada enfoque de solución, el número de instancias en las que el algoritmo obtiene una solución óptima (Columna Opt.), el tiempo mínimo, máximo, mediana y media (expresado en segundos) por familia de instancias (Columna Tiempo). La última columna (% mejora) presenta cuánto mejoró la versión propuesta con respecto al algoritmo original en número de óptimos encontrados.

Tabla D.1. Resultados obtenidos con la versión Límites del algoritmo GGA-CGT para las instancias BPPLIB*.

Tabla D.2. Resultados obtenidos con la versión Reducción del algoritmo GGA-CGT para las instancias BPPLIB*.

Clase	Inst.	Gral. Arc-Flow Formulation (Exacto)					CNS_BP					GGA-CGT (Original)					GGA-CGT (Versión Reducción)					
		Opt.	Tiempo (s)				Opt.	Tiempo (s)				Opt.	Tiempo (s)				Opt.	Tiempo (s)				
			Min	Max	Mediana	Media		Min	Max	Mediana	Media		Min	Max	Mediana	Media		Min	Max	Mediana	Media	% Mejora
Uniform	80	80	0.12	3.10	0.31	0.96	80	0.01	0.30	0.04	0.06	80	0.00	4.28	0.89	1.87	80	0.00	4.59	1.19	2.35	0
Triplets	80	80	0.04	5.84	0.41	1.41	80	0.02	0.18	0.03	0.09	80	0.02	5.12	0.58	2.87	80	0.00	6.28	1.25	3.44	0
Data Set 1	720	720	0.01	1.63	0.26	0.54	720	0.00	0.32	0.03	0.08	720	0.00	191.03	0.02	18.61	720	0.00	31.46	0.02	18.85	0
Data Set 2	480	480	0.04	851.10	12.9	67.53	480	0.03	0.14	0.02	0.04	480	0.00	58.51	2.07	6.96	480	0.00	37.02	0.01	7.13	0
Data Set 3	10	10	9.01	28.80	12.17	21.48	10	0.00	0.02	0.009	0.01	10	0.15	48.73	22.19	45.18	10	0.03	64.17	22.41	47.25	0
Was 1	100	100	0.33	1.75	0.51	0.82	100	0.00	0.00	0.00	0.00	100	0.00	0.09	0.01	0.07	100	0.00	0.26	0.08	0.19	0
Was 2	100	100	0.39	1.71	0.49	0.76	100	0.00	0.04	0.012	0.02	100	0.00	16.24	7.58	14.46	100	0.00	23.05	4.61	16.14	0
Gau 1	17	17	1.23	15,432.27	407.64	1665.84	17	1.82	11.70	1.56	2.62	16	0.00	3.18	0.11	4.19	17	0.00	6.51	1.39	4.73	5.88
Hard28	28	28	3.47	170.32	10.42	53.75	20	2.01	31.50	4.87	6.34	16	1.24	27.52	14.09	23.20	20	0.64	38.16	11.54	25.19	14.29
Total	**1615**	**1615**				**1813.09**	**1607**				**9.26**	**1602**				**117.40**	**1607**				**125.26**	**0.31**
Gran Total						**4452**					**1753**					**1900**					**2082**	

Tabla D.3. Resultados obtenidos con la versión Diversificación del algoritmo GGA-CGT para las instancias BPPLIB*.

Clase	Inst.	Gral. Arc-Flow Formulation (Exacto)					CNS_BP					GGA-CGT (Original)					GGA-CGT (Versión Diversificación)					
		Opt.	Tiempo (s)				Opt.	Tiempo (s)				Opt.	Tiempo (s)				Opt.	Tiempo (s)				
			Min	Max	Mediana	Media		Min	Max	Mediana	Media		Min	Max	Mediana	Media		Min	Max	Mediana	Media	% Mejora
Uniform	80	80	0.12	3.10	0.31	0.96	80	0.01	0.30	0.04	0.06	80	0.00	4.28	0.89	1.87	80	0.00	4.59	1.19	2.35	0
Triplets	80	80	0.04	5.84	0.41	1.41	80	0.02	0.18	0.03	0.09	80	0.02	5.12	0.58	2.87	80	0.00	6.28	1.25	3.44	0
Data Set 1	720	720	0.01	1.63	0.26	0.54	720	0.00	0.32	0.03	0.08	720	0.00	191.03	0.02	18.61	720	0.00	31.46	0.02	18.85	0
Data Set 2	480	480	0.04	851.10	12.9	67.53	480	0.03	0.14	0.02	0.04	480	0.00	58.51	2.07	6.96	480	0.00	37.02	0.01	7.13	0
Data Set 3	10	10	9.01	28.80	12.17	21.48	10	0.00	0.02	0.009	0.01	10	0.15	48.73	22.19	45.18	10	0.03	64.17	22.41	47.25	0
Was 1	100	100	0.33	1.75	0.51	0.82	100	0.00	0.00	0.00	0.00	100	0.00	0.09	0.01	0.07	100	0.00	0.26	0.08	0.19	0
Was 2	100	100	0.39	1.71	0.49	0.76	100	0.00	0.04	0.012	0.02	100	0.00	16.24	7.58	14.46	100	0.00	23.05	4.61	16.14	0
Gau 1	17	17	1.23	15,432.27	407.64	1665.84	17	1.82	11.70	1.56	2.62	16	0.00	3.18	0.11	4.19	17	0.00	6.51	1.39	4.73	5.88
Hard28	28	28	3.47	170.32	10.42	53.75	20	2.01	31.50	4.87	6.34	16	1.24	27.52	14.09	23.20	28	0.64	34.42	9.06	22.1	42.86
Total	**1615**	**1615**				**1813.09**	**1607**				**9.26**	**1602**				**117.40**	**1615**				**123.27**	**0.81**
Gran Total						**4452**					**1753**					**1900**					**2081**	

Tabla D.4. Resultados obtenidos con el algoritmo GGA-CGT/D (versión propuesta) para las instancias BPPLIB*.

Clase	Inst.	Gral. Arc-Flow Formulation (Exacto)					CNS_BP					GGA-CGT (Original)					GGA-CGT/D (Propuesta)					
		Opt.	Tiempo (s)				Opt.	Tiempo (s)				Opt.	Tiempo (s)				Opt.	Tiempo (s)				
			Min	Max	Mediana	Media		Min	Max	Mediana	Media		Min	Max	Mediana	Media		Min	Max	Mediana	Media	% Mejora
Uniform	80	80	0.12	3.10	0.31	0.96	80	0.01	0.30	0.04	0.06	80	0.00	4.28	0.89	1.87	80	0.00	4.46	0.58	2.28	0
Triplets	80	80	0.04	5.84	0.41	1.41	80	0.02	0.18	0.03	0.09	80	0.02	5.12	0.58	2.87	80	0.00	4.22	0.46	2.74	0
Data Set 1	720	720	0.01	1.63	0.26	0.54	720	0.00	0.32	0.03	0.08	720	0.00	191.03	0.02	18.61	720	0.00	35.16	0.01	19.57	0
Data Set 2	480	480	0.04	851.10	12.9	67.53	480	0.03	0.14	0.02	0.04	480	0.00	58.51	2.07	6.96	480	0.00	41.28	0.01	11.23	0
Data Set 3	10	10	9.01	28.80	12.17	21.48	10	0.00	0.02	0.009	0.01	10	0.15	48.73	22.19	45.18	10	0.00	59.93	19.84	47.10	0
Was 1	100	100	0.33	1.75	0.51	0.82	100	0.00	0.00	0.00	0.00	100	0.00	0.09	0.01	0.07	100	0.00	0.21	0.04	0.12	0
Was 2	100	100	0.39	1.71	0.49	0.76	100	0.00	0.04	0.012	0.02	100	0.00	16.24	7.58	14.46	100	0.00	25.13	4.83	16.24	0
Gau 1	17	17	1.23	15,432.27	407.64	1665.84	17	1.82	11.70	1.56	2.62	16	0.00	3.18	0.11	4.19	17	0.00	10.47	2.18	8.97	5.89
Hard28	28	28	3.47	170.32	10.42	53.75	20	2.01	31.50	4.87	6.34	16	1.24	27.52	14.09	23.20	28	0.29	42.01	13.25	26.22	42.86
Total	**1615**	**1615**				**1813.09**	**1607**				**9.26**	**1602**				**117.40**	**1615**				**134.47**	**0.81**
Gran Total						**4452**					**1753**					**1900**					**2114**	

En las tablas siguientes se presentan los resultados obtenidos para las cuatro clases de instancias BPP$v_{u_}c$ (BPP.25, BPP.5, BPP.75 y BPP1). Se muestra la capacidad del contenedor (Columna C) y el número de instancias, para cada enfoque de solución, el número de instancias en las que el algoritmo obtiene una solución óptima (Columna Opt.), el tiempo mínimo, máximo, mediana y media (expresado en segundos) por capacidad del contenedor (Columna Tiempo). La última columna (% mejora) presenta cuánto mejoró la versión propuesta con respecto al algoritmo original en número de óptimos encontrados.

Tabla D.5. Resultados obtenidos con el algoritmo GGA-CGT/D (versión propuesta) para la clase BPP.25 de las instancias BPP$v_{u-}c$.

BPP.25																						
C	Inst.	Gral. Arc-Flow Formulation (Exacto)					CNS_BP					GGA-CGT (Original)					GGA-CGT/D (Versión propuesta)					
		Opt.	Tiempo (s)				Opt.	Tiempo (s)				Opt.	Tiempo (s)				Opt.	Tiempo (s)				
			Min	Max	Mediana	Media		Min	Max	Mediana	Media		Min	Max	Mediana	Media		Min	Max	Mediana	Media	% Mejora
100	100	100	0.00	1.58	0.35	0.96	100	0.00	0.01	0.00	0.00	100	0.00	0.01	0.00	0.00	100	0.00	0.01	0.00	0.00	0
1000	100	100	0.01	1.96	0.68	1.09	100	0.00	0.02	0.00	0.01	100	0.00	0.04	0.01	0.02	100	0.00	0.05	0.01	0.02	0
10000	100	100	0.53	4.94	3.16	3.86	100	0.06	0.47	0.26	0.28	100	0.10	0.41	0.25	0.25	100	0.10	0.56	0.28	0.29	0
100000	100	100	6.27	18.34	9.04	9.15	100	4.69	12.07	5.25	5.97	100	5.72	11.21	5.78	6.23	100	5.89	13.62	5.82	6.47	0
1000000	100	100	49.23	268.25	65.21	76.84	100	10.27	197.21	38.33	51.64	100	11.88	207.07	42.77	54.12	100	12.34	209.61	45.36	57.95	0
10000000	100	100	975.91	5871.04	1246.20	3152.96	100	636.18	3087.49	651.83	1721.1	100	650.96	32837.3	789.47	1938.6	100	628.76	3477.2	771.09	1941.19	0
100000000	100	87	3118.91	8608.17	5439.52	5982.07	61	1826.98	4193.70	2033.84	2164.52	6	2378.51	4762.54	2607.96	2611.22	89	2592.65	6043.9	3976.5	4025.19	83
Total	**700**	**687**				**9226.9**	**661**				**3943.5**	**606**				**4610.4**	**689**				**6031.10**	**11.9**
Gran Total						**301613**					**114076**					**118663**					**253651**	

Tabla D.6. Resultados obtenidos con el algoritmo GGA-CGT/D (versión propuesta) para la clase BPP.5 de las instancias BPP$v_{u-}c$.

BPP.5																
C	Inst.	Gral. Arc-Flow Formulation (Exacto)					CNS_BP					GGA-CGT (Original)				GGA-CGT/D (Ve
		Opt.	Tiempo (s)				Opt.	Tiempo (s)				Opt.	Tiempo (s)			
			Min	Max	Mediana	Media		Min	Max	Mediana	Media		Min	Ma	Me	
100	100	100	0.00	1.09	0.38	0.75	100	0.00	0.01	0.00	0.0					
1000	100	100	0.37	1.96	0.61	1.4	100	0.03	0.14							
10000	100	100	2.49	6.82	3.16	3.93	100	1.10								
100000	100	96	9.63	6174.09	265.22	624.08	9									
1000000	100	34	816.11	2015.32	1217.81											
10000000	100	21	69.35	2147.2												
100000000	100	75	53													
Total	**700**															
Gran																

Tabla D.7. Resultados obtenidos con el algoritmo GGA-CGT/D (versión propuesta) para la clase BPP.75 de las instancias BPP$v_{u-}c$.

BPP.75																						
C	Inst.	Gral. Arc-Flow Formulation (Exacto)					CNS_BP					GGA-CGT (Original)					GGA-CGT/D (Versión propuesta)					
		Opt.	Tiempo (s)				Opt.	Tiempo (s)				Opt.	Tiempo (s)				Opt.	Tiempo (s)				
			Min	Max	Mediana	Media		Min	Max	Mediana	Media		Min	Max	Mediana	Media		Min	Max	Mediana	Media	% Mejora
100	100	100	0.07	3.85	1.16	1.99	98	0.00	0.06	0.02	0.02	95	0.00	0.11	0.03	0.03	100	0.00	2.19	1.07	1.95	5
1000	100	100	1.04	4.52	2.39	2.74	97	0.14	1.06	0.45	0.52	99	0.28	1.14	0.70	0.70	100	0.46	5.38	3.21	4.09	1
10000	100	38	51.22	258.34	119.15	130.56	19	0.58	13.21	3.17	3.86	0	1.09	15.98	4.42	5.05	48	86.11	237.03	125.53	172.24	48
100000	100	31	138.56	530.64	216.01	225.00	21	0.65	32.18	4.36	6.09	0	0.39	19.65	4.92	5.75	34	112.85	416.71	184.22	201.84	34
1000000	100	34	154.19	5781.83	453.25	1902.87	13	0.04	10.74	1.49	2.05	0	0.09	9.70	1.68	2.20	39	140.18	5082.15	179.37	2621.1	39
10000000	100	43	1392.07	4522.03	1078.41	2971.9	24	0.37	5.26	1.07	1.24	0	0.42	5.81	1.46	1.57	39	1852.49	4703.28	1396.4	3257.41	39
100000000	100	50	1564.18	5232.7	1408.96	3824.04	36	0.22	4.19	0.74	0.98	0	0.34	4.62	1.32	1.46	42	1276.20	5016.1	1644.8	3963.18	42
Total	**700**	**396**				**9059.10**	**308**				**14.75**	**194**				**16.77**	**402**				**10222**	**29.7**
Gran Total						**300123**					**1976**					**2047.00**					**319275**	

Tabla D.8. Resultados obtenidos con el algoritmo GGA-CGT/D (versión propuesta) para la clase BPP1 de las instancias BPP$v_{u-}c$.

BPP1																						
C	Inst.	Gral. Arc-Flow Formulation (Exacto)					CNS_BP					GGA-CGT (Original)					GGA-CGT/D (Versión propuesta)					
			Tiempo (s)					Tiempo (s)					Tiempo (s)					Tiempo (s)				
		Opt.	Min	Max	Mediana	Media	Opt.	Min	Max	Mediana	Media	Opt.	Min	Max	Mediana	Media	Opt.	Min	Max	Mediana	Media	% Mejora
100	100	100	0.12	1.13	0.42	0.47	100	0.00	0.14	0.01	0.04	100	0.01	0.17	0.06	0.06	100	0.00	0.21	0.08	0.08	**0**
1000	100	100	1.28	3.67	1.38	2.41	100	0.28	2.26	0.64	0.82	100	0.37	2.35	0.93	0.93	100	0.56	2.98	1.02	1.08	**0**
10000	100	64	1.05	7.66	201.59	565.73	58	0.32	1501.19	98.53	328.65	49	0.35	17298.32	107.21	379.11	81	0.47	1984.01	119.25	387.17	**32**
100000	100	87	1.19	4051.73	1.12	369.74	69	0.21	1674.82	23.71	273.98	73	0.30	17118.43	15.50	293.66	94	0.51	2053.82	47.31	302.98	**21**
1000000	100	100	0.56	2522.35	34.19	131.09	99	0.26	17160.11	3.03	92.71	94	0.33	17469.94	3.47	95.96	100	0.39	1907.63	19.24	104.15	**6**
10000000	100	100	1.07	760.47	11.73	32.47	100	0.15	502.94	2.16	16.1	100	0.19	560.45	2.49	19.55	100	0.23	591.04	4.16	21.09	**0**
100000000	100	100	0.51	398.04	9.62	19.46	100	0.21	120.72	0.97	6.25	100	0.19	138.67	1.96	8.22	100	0.17	156.31	3.28	13.2	**0**
Total	**700**	**651**				**1121.4**	**626**				**718.56**	**616**				**797.49**	**675**				**829.73**	**8.43**
Gran Total						**104852**					**73421**					**84534**					**87084**	