Pontificia Universidad Católica De Valparaíso
Facultad De Ingeniería
Escuela De Ingeniería Informática

RESOLUCIÓN DEL CONTAINER PRE-MARSHALLING PROBLEM UTILIZANDO CUCKOO SEARCH Y BLACK HOLE

HERNÁN ANÍBAL MEZA RIVERA
MATÍAS ANTONIO VILLAR BARRAZA

Profesor guía: **Ricardo Soto De Giorgis**

Profesor Co-referente: **Broderick Crawford Labrín**

Carrera**: Ingeniería de Ejecución en Informática**

2016

Índice

Dedicatoria

Trabajo dedicado a nuestros familiares y amigos, quienes brindaron su apoyo en todo momento de la investigación.

Resumen

El Container Pre-Marshalling Problem (CPMP) es un problema surgido en los terminales portuarios que tiene por objetivo minimizar el número de movimientos necesarios para ordenar un conjunto de pilas de contenedores en función de su salida. Este ordenamiento es clave para la óptima operación del puerto debido a que el orden de llegada de los contenedores al terminal portuario generalmente no coincide con el orden de salida. En este proyecto, el CPMP se resolverá mediante las metaheurísticas Black Hole y Cuckoo Search Algorithm.

Black Hole plantea la asimilación de las soluciones como estrellas en una galaxia; las cuales están en constante movimiento debido a la atracción que sufren por un agujero negro que representa la mejor solución. Por otra parte, Cuckoo Search opera según el hábito de reproducción del pájaro Cucú, el cual va dejando sus huevos en los nidos de otros pájaros siguiendo una trayectoria descrita mediante los vuelos de Levy. Se muestran resultados interesantes obtenidos por ambas técnicas.

Abstract

The Container Pre-Marshalling Problem (CPMP) is a problem emerged in port terminals that aims at minimizing the amount of movements required to sort a stack set of containers according its shipment. This sorting process is key for the optimal port operation since the container arrival generally does not match the shipment schedule. In this project, we solve the CPMP by using Black Hole and Cuckoo Search.

Black Hole simulates the stars in a galaxy; which are in constant motion due to the attraction suffered by a black hole that represents the best solution. On the other hand, Cuckoo Search operates according to the reproduction behavior of the cuckoo bird, which leaves its eggs in the nests of other birds following a path described by Lévy flights. We illustrate interesting results obtained by both techniques.

Lista de Figuras

Lista de Tablas

1 Introducción

La optimización es un área o campo de estudio de la ciencia que evidencia la necesidad natural impuesta por el medio o por la auto-imposición de hacer las cosas de manera más eficiente. La calidad, efectividad y verificación, nacen como conceptos de esta premisa. Con el paso del tiempo el ser humano se ha visto en la necesidad de lograr tal propósito, por lo que la ingeniería y un sin número de disciplinas han puesto su mirada en la creación y aquello que habita en ella, con el fin no solo de apreciar la coherencia del mundo, sino también, de poder replicar las lógicas que utilizan fenómenos físicos, animales, genética e inclusive los insectos. En el presente documento de título el equipo mostrará de forma detallada y concisa la resolución del CPMP haciendo uso de los algoritmos (metaheurísticas) Black Hole y Cuckoo Search, los cuales han tomado como inspiración los principales conceptos que articulan los fenómenos de agujeros negros y los métodos reproductivos del pájaro Cuco, respectivamente; lo anterior con el fin de dar forma a un comportamiento algorítmico que se ha logrado traducir en dos Metaheurísticas.

El Container Pre Marshalling Problem es entendido como el problema nacido a raíz de la necesidad de saber la manera correcta de apilar contendores en los terminales portuarios, si bien tal problema podría resultar en un primer acercamiento una problemática inocente y sin muchas complicaciones, la práctica y el mayor contacto con dicho problema evidencia que a medida que incrementan la cantidad de contenedores a apilar el problema adquiere una complejidad considerable, la cual queda lejos de las capacidades de una persona natural o equipo, en términos de cálculo y tiempo de resolución.

En simples palabras el Pre Marshalling Problem consiste en ordenar los contenedores de un puerto, de tal manera que los contenedores que tengan más prioridad queden sobre los que tienen menos prioridad, ya que los primeros son los que se van a embarcar más prontamente. Esta situación es muy común en los terminales portuarios y además, como esta tarea la hace un operador, este ordena los contenedores a su modo, por lo que se desperdician muchas “recolocaciones”, que a su vez se transforman en tiempo y dinero. El área de estudio del CPMP busca disminuir la cantidad de recolocaciones, para que este ordenamiento se realice con la ayuda de un software y poder así reducir la cantidad de movimientos de grúa que ordenan los contenedores.

Por otro lado, las metaheurísticas son algoritmos iterativos que el caso de BH y CS asimilan el comportamiento de la naturaleza para resolver problemas. Ahora bien, por su lado el algoritmo Black Hole es una metaheurística cuya estructura iterativa está basada en la premisa básica de que un agujero negro coexiste con un número (en este caso) determinado de estrellas, tales estrellas representan soluciones, y el agujero negro es la mejor solución de estas estrellas, la cual acerca a todas aquellas que no resulten tan optimas, con el fin de absorberlas y que el espacio cree nuevas estrellas o soluciones. Resulta importante resaltar que el número de posibles soluciones (estrellas) debe ser constante durante el desarrollo de la metaheurística para su correcto desempeño.

En contraste Cuckoo Search es una metaheurística cuya mecánica cíclica está inspirada en el comportamiento del pájaro Cuco o Cuckoo del inglés, el algoritmo toma como referente el agresivo proceso de reproducción de estas aves para dar solución a diversos problemas; para hacerlo establece el símil entre los huevos o nidos trabajados por el ave y las soluciones de un problema de optimización cualquiera, el ave pone sus huevos en los nidos de otras aves (soluciones), luego por cada iteración se eliminan los huevos que tienen menor probabilidad de nacer, la metaheurística se comporta de esta manera hasta encontrar un buen huevo (solución).

Durante las siguientes páginas del documento se darán a conocer, los objetivos planteados por el equipo para esta entrega del proyecto; así como el estado de arte que existe ante la problemática. Posteriormente se explica en detalle tanto el Container Pre-Marshalling como problema, evidenciando los pasos necesarios para darle solución, así como las nomenclaturas que el equipo y otros investigadores utilizan para referirse a distintos aspectos de la problemática. Posteriormente el informe comienza a describir más en detalle las metaheurísticas utilizadas, ilustrando su comportamiento y el porqué de este. Finalmente se explican aspectos de diseño de las propuestas de los algoritmos implementados, tales como la modularización utilizada, clases o "estructuras de datos" ocupadas, la aplicación de la metaheurísticas utilizadas y algunos de los resultados arrojados que se pudieron conseguir de las instancias propuestas por el profesor guía.

2 Objetivos

2.1 Objetivos Generales

- Resolver el Container Pre-Marshalling Problem mediante las metaheurísticas Black Hole y Cuckoo Search.

2.2 Objetivos Específicos

- Comprender el Container Pre-Marshalling Problem.
- Implementar un algoritmo basado en Black Hole y el algoritmo de Límite Inferior para resolver el Container Pre-Marshalling Problem.
- Implementar un algoritmo basado en Cuckoo Search y el algoritmo de Límite Inferior para resolver el Container Pre-Marshalling Problem.
- Realizar experimentos con las implementaciones realizadas.

3 Estado del Arte

CPMP es un problema de minimizar las recolocaciones que ha sido investigado y resuelto por modelos matemáticos y heurística tradicional, las cuales se mencionan a continuación.

Los primeros acercamientos al CPMP fueron las investigaciones de Kim, Park y Ryu [6] los cuales establecieron los primeros lineamientos para el almacenamiento de contenedores; posteriormente, Robert Stahlbock y Stefan Voβ [1] desarrollaron una heurística que resuelve el problema ligado a una serie de restricciones propuestas. Por su parte Bortfeld describe un árbol heurístico de búsqueda que se utiliza para buscar soluciones al Pre-Marshalling [3] y luego este extiende su trabajo junto a Forster. Este último es codificado en lenguaje C por medio de procedimientos recursivos, para demostrar que poseía mejores resultados que los algoritmos propuestos por Lee y Hsu [3] y Lee y Chao [4].

Lee y Hsu, propusieron un modelo de programación entera para el PMP (Pre-Marshalling Problem) aplicándolo para pequeñas y medianas instancias, Por su parte, Lee propone [5] una heurística de tres fases basada en las reglas para heurísticas que da solución al CPMP.

Caserta, en sus investigaciones, propone otro enfoque heurístico para resolver el CPMP, mirado desde el "Método del Corredor", el cual estipula utilizar un método exacto (Programación Dinámica, por ejemplo) sobre una parte restringida del espacio de solución de un problema dado para así minimizar el espacio de búsqueda, la implementación de Caserta resuelve el problema en base los supuestos de Kim [8]. En referencia a las suposiciones mencionadas por Kim [6] y Caserta [7] se desarrollan tres métodos para resolver el problema de relocalización de contenedores.

Molins presenta una nueva heurística de planificación dominante-dependiente [9], la heurística fue codificada usando la Planificación del Lenguaje Definido Dominante (PDDL). Los resultados de su heurística fueron comparados con los resultados de uno anterior y muestra una significativa disminución en el tiempo de ejecución [10].

Christopher Exposito-Izoquierdo, presenta la primera heurística de primera prioridad baja (LPFH) [11] que usa las suposiciones de Lee y Hsu [4]. También ellos introducen un generador de instancias de PMP. La heurística fue codificada en el lenguaje de programación JAVA y los resultados fueron comparados con los resultados obtenidos por Caserta y Voβ [7] y este fue visto como una heurística Relativamente Perfecta. También los resultados fueron comparados con la solución óptima obtenida por Algoritmo de Búsqueda, comprobando que los resultados tuvieron un buen desempeño con la heurística propuesta.

Finalmente el trabajo de Forster and Bortfeld en el año 2012 genera un algoritmo de búsqueda de árbol para el problema de reubicación de los contenedores que se adapta la estructura básica de un árbol de búsqueda incompleto.

4 Container Pre – Marshalling Problem CPMP

El Container Pre – Marshalling Problem es un problema de optimización que se sitúa en los terminal portuario, en el cual se almacenan una cierta cantidad de contenedores. El contexto de la ubicación de los contenedores, está dividido en bloques, y cada uno de estos consiste en varias bahías; y a su vez cada bahía posee un número determinado de pilas de contenedores. En la vida real, los días en los cuales los contenedores están en el puerto se tienen diversas restricciones, las cuales están en directa relación a la estabilidad de las pilas, el peso del contenedor así como el puerto al cual está destinado; lo anterior ha llevado a que se cree una numeración (prioridad) para grupos con el fin de referenciar a los contenedores que compartan los mismos atributos, con el fin de referirse de forma general a un grupo de contendores, para esta investigación la prioridad está dada únicamente por la urgencia que tendrá un contenedor de ser despachado.

Ahora bien, las recolocaciones de los contenedores son realizados por una grúa mecánica de gran tamaño y el hecho de trasladar uno o varios contenedores de una pila a otra representa un alto costo en tiempo, debido a que es una tarea delicada y que requiere de personal especializado. El párrafo anterior hace referencia cuando se está cargando un barco con contenedores; no obstante, para poder haber llegado a esta instancia, en algún momento la bahía puede haber estado desordenada y por tanto, además del costo de despachar los contenedores, se añade el costo de tener que ordenar la bahía, para un posterior despacho de los contenedores de la manera más eficiente posible. Ante esto el CPMP plantea la necesidad de encontrar una serie de recolocaciones que permitan al operario de la grúa ordenar la bahía en la menor cantidad de recolocaciones posibles.

A lo anterior surge la pregunta ¿Qué es una bahía ordenada? Una bahía ordenada es un conjunto de pilas de contenedores en donde se cumple que cada contenedor tenga abajo, un contenedor de igual o mayor índice numérico y por tanto de menor prioridad; debido a que las prioridades están descritas de la siguiente manera en la figura 4.1.

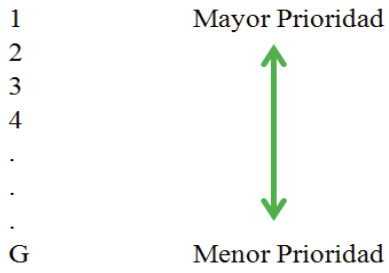


Figura 4.1 Niveles de Prioridad.

Donde G sea la cantidad total de grupos.

Ahora bien, el modelamiento del problema indica que los contenedores deben tener numeración descendente, la cual hace referencia a la prioridad o urgencia con la cual van a ser despachados; en pocas palabras a mayor índice de prioridad en el contenedor, la urgencia por sacar dicho contenedor es menor; de esta manera en una columna correctamente ordenada un contenedor cuya prioridad es 7 o 9 no puede estar arriba de un contenedor cuya prioridad es 5. Empíricamente a la vez se entiende que si se diera el siguiente caso: (Figura 4.2), toda la columna a excepción del primer contenedor se consideraría incorrecta; no obstante, lo anterior, no indica que una columna; donde hayan dos o más contenedores que estén juntos verticalmente y cuyas prioridades sean idénticas; no sea una configuración aceptable. (Figura 4.3).

Figura 4.2 Pila Desordenada.

Figura 4.3 Pila Ordenada.

Finalmente, a la hora de presentar la solución al CPMP, esta debe ser evidenciada mediante una serie de recolocaciones del tipo (Columna Origen, Columna Destino), además la cantidad de recolocaciones idealmente debe ser igual al límite inferior (cantidad mínima de recolocaciones que se requieren para solucionar el CPMP), o bien no pasarse mucho de este.

La siguiente figura mostrará una bahía desordenada, figura 4.4, en ella se puede apreciar que los contenedores desordenados son los de la columna tres, ya que el primer contenedor, tienen prioridad dos, y sobre él tiene dos contenedores más, con una prioridad mucho mayor (11 y 5 respectivamente), lo que significa que estos dos tienen que ser reposicionados para que la bahía quede ordenada.

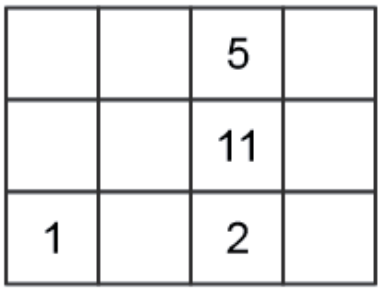


Figura 4.4 Bahía Desordenada.

Una posible solución (ver figura 4.5) es mover el contendedor de prioridad 5 hacia la columna dos, y posteriormente mover el contenedor de prioridad 11 hacia la columna cuatro, resultando una bahía ordenada, porque cada contenedor no tiene un contenedor con menor prioridad sobre él.

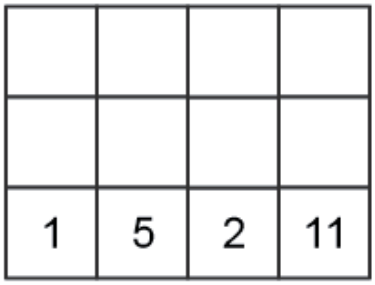


Figura 4.5 Bahía Ordenada.

4.1 Notaciones

En este apartado se explican las notaciones utilizadas en el CPMP en la tabla 4.1. Primeramente se definen las siglas utilizadas para hacer referencia a aspectos como las dimensiones de una bahía determinada. Posteriormente se explica la notación de una recolocación.

Tabla 4.1 Notaciones CPMP

Sigla	**Significado**
L	Matriz o Bahía
S	Numero de Stacks (Pilas de contenedores). Donde (s=1,…,S)
H	H = Altura máxima de las pilas. Donde (h=1,…,H)
L(h,s)	Referencia a un contenedor en particular.

4.2 Calcular el Fitness

El fitness es esencial en la resolución del problema, es el primer paso para comenzar a modelar y entregar una buena solución al Pre Marshalling. Por este motivo, es muy importante elegir una buena técnica para encontrar el mejor fitness, o un fitness adecuado.

Este se calculará mediante el límite inferior, el cual da como resultado la menor cantidad mínima de recolocaciones para que la bahía quede ordenada, por ejemplo en el siguiente caso el límite inferior será 5 (ver figura 4.6):

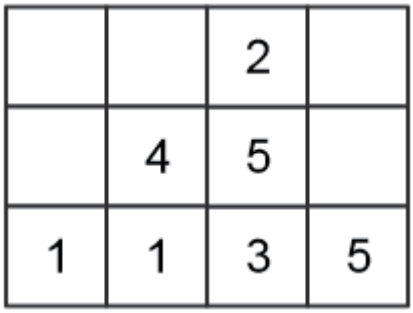


Figura 4.6 Límite Inferior Bahía Desordenada.

Y ordenada la bahía quedará de la siguiente forma. (Figura 4.7):

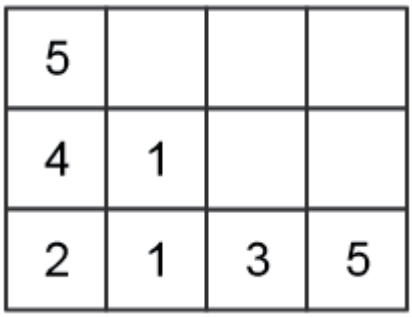


Figura 4.7 Límite Inferior Bahía Ordenada.

Cabe mencionar que en la práctica, el valor arrojado límite inferior en ocasiones es difícil de conseguir, en otras palabras, la cantidad de recolocaciones en las que se consigue ordenar la bahía es mayor. Resulta conveniente advertir esto debido a que a simple vista en el problema ejemplificado en las figuras 4.6 y 4.7; este puede ser resuelto con la cantidad de pasos otorgados en el límite inferior, pero en un problema con 20 o más contenedores por ejemplo, la complejidad de conseguir cumplir el límite inferior aumenta considerablemente.

Ahora bien, para obtener el límite inferior se necesita formar una tabla tomando como referencia una bahía desordenada. Para ello se generaran d(g), D(g), sp(g), Sp(g), Ds(g) cuyos significados se explican en la tabla 4.2.

Tabla 4.2 Notaciones Límite Inferior

Sigla	Significado
g	Es cada prioridad del contenedo, ejemplo g=1 son los contenedores con prioridad uno.
d(g)	Es la cantidad de contenedores (de prioridad g) mal posicionados. La llamaremos Demanda
D(g)	Es la demanda acumulada.
sp(g)	Es la cantidad de bloques hacia arriba (de abajo hacia arriba, en la columna), desde el ultimo contenedor ordenado.
Sp(g)	Es la oferta potencial acumulada.
Ds(g)	Es la demanda acumulada [D(g)] menos la Oferta potencial acumulada [Sp(g)].

Si tomamos como bahía desordenada el ejemplo de la figura 4.6, se obtendrá como resultado una tabla como la siguiente (ver Tabla 4.3):

Tabla 4.3 Cálculo para el Límite Inferior.

Elemento del Grupo (g)	Demanda d(g)	Demanda Acumulada D(g)	Oferta Potencial sp(g)	Oferta Potencial Acumulada Sp(g)	Demanda Acumulada por Exceso Ds(g)
5	1	1	2	2	-1
4	1	2	0	2	0
3	0	2	2	4	-2
2	1	3	0	4	-1
1	0	3	4	8	-5

Luego al tener esta tabla lista se procede a calcular n_{Bx} mediante la formula 4.2.1

$$n_{Bx} = n_b + \min\{n_b(s) | \mathrm{s} = 1,2,\dots,\mathrm{S}\}$$

(4.2.1)

Donde n_{Bx} es la suma de todos los contenedores mal posicionados en la bahía, más la mínima prioridad de contenedores que están ubicados en la columna S. Mediante esta definición la ecuación quedaría como: $n_{bx} = 3 + 1$ Luego se calcula el Ngx, mediante la siguiente fórmula:

$$n_{Gx} = \sum_{s=1}^{n(s,Gx)} n^{g^*}(s)$$

(4.2.2)

Hay que tomar en cuenta que:

- Se debe satisfacer: $Ds\,(g^*) \geq Ds\,(g)$ con g=1,…, G.
- Para la función de la sumatoria, se deben considerar elementos $(g) < (g^*)$. puesto que son potenciales pilas GX cuando están ubicadas en lo alto de estas.

Finalmente se calcula la suma de los dos resultados anteriores (4.11), resultando como límite inferior 5.

$$n_m = n_{Gx} + n_{Bx}$$

(4.2.3)

5 Metaheurísticas utilizadas

5.1.1 Black Hole

El algoritmo Black Hole es una metaheurística basada en poblaciones de soluciones, al igual que otros algoritmos del mismo tipo, se enfoca en el planteamiento de un conjunto de soluciones distribuidas en el espacio de búsqueda de forma aleatoria, así como en la evaluación metódica de dichas soluciones tomando como valor de referencia el fitness de cada solución; lo cual es en pocas palabras el valor asociado a la función objetivo del problema de optimización. En esta investigación se utilizó una variante de la metaheurística tradicional de Black Hole para abordar el CPMP y ayudar a minimizar el número de recolocación de contenedores; ya que el original plantea mecánicas más propias de algoritmos como PSO. Es así como la variante mencionada describe su utilización en forma sencilla de la siguiente manera:

1. Inicializar una población de estrellas (Vectores de Recolocaciones) de un tamaño igual al Límite Inferior (LB).

Bucle Iterativo

2. Por cada estrella es necesario almacenar el fitness asociado a realizar las recolocaciones de la solución.
3. Seleccionar la mejor estrella según su valor de fitness como Agujero Negro.
4. Cambiar la ecuación de movimiento a todas las estrellas con la excepción del Black Hole según la ecuación 5.1.1.1
5. Si una estrella tiene menor fitness que el Agujero Negro, se han de permutar (Actualizar Black Hole).
6. Si una estrella cruza el horizonte de eventos (“R”), se ha de reemplazar la estrella.
7. Si se cumple un criterio de término como demasiadas iteraciones o un buen fitness del Black Hole, se ha de terminar el bucle.

Terminar Bucle

8. Evidenciar solución.

Lo anterior explica de forma procedimental, pero genérica el comportamiento de la metaheurística Black Hole. Lo anterior se toma como principal referente en la implementación de la solución del CPMP.

5.1.2 Movimiento de las Estrellas

Luego de la inicialización de las estrellas y la evaluación de los fitness, y por tanto la asignación del Black Hole, las estrellas comienzan a moverse debido a la atracción que ejerce el agujero negro sobre estas. El movimiento del que se habla, no está dado de forma aleatoria en su totalidad; debido a que existe una correlación metafórica (física en la realidad) entre la fuerza de atracción del agujero y las estrella. Dicho movimiento de las estrellas esta descrito formalmente por el siguiente modelo matemático.

$$x_i(t+1) = x_i(t) + rand_i(x_{BH} - x_i(t))$$

(5.1.1.1)

Donde $x_i(t)$ y $x_i(t+1)$ es la estrella "i" en las iteraciones "t" y "t+1" del bucle de la metaheurística respectivamente. Lo cual explica que la futura "posición" o recolocaciones de una estrella, serán los valores actuales de la estrella, más la resta vectorial entre sí misma y el Black Hole x_{bh} , escalada por un valor randómico que esta entre 0 y 1. En consecuencia, en la práctica esto se traduce en que la ecuación de movimiento afecta al vector de recolocaciones en cuanto al origen y destino

Por otro lado, existe la alta probabilidad de que una estrella cruce el horizonte de eventos; el cual se describe como el radio de atracción del agujero negro; por norma general todo ente u objeto que cruce tal horizonte, es succionado por el agujero y destruido. De la misma forma, existe para la metaheurística la consideración de este fenómeno bajo la siguiente formula.

$$R = \frac{\sum_{i=1}^{N} Fitness_i}{Fitness_{BH}}$$

(5.1.1.2)

Es con el factor R que luego se plantea la siguiente evaluación: Si la solución resultante de la resta vectorial entre el Black Hole y la estrella "i" da un menor fitness que el factor R, la estrella o solución "i" se eliminará y se creará otra que la reemplace.

5.1.3 Black Hole aplicado al CPMP

Tabla 5.1 Aplicación de Black Hole como Forma de Resolución del CPMP

Concepto Black Hole	Concepto CPMP	Implementación Realizada
Estrella	Solución de Recolocaciones Consecutivas	Arreglo Bidimensional
Ecuación de Movimiento	Alterar una Solución	Ecuación
Fitness	Valor de Función Objetivo	Límite Inferior de la Bahía
Variable	Recolocación	Una Recolocación (Una columna del Vector)
Dominio	$IN \in \{1,..,Cant\ Columnas\}$	$IN \in \{1,..,Cant\ Columnas\}$
Restricción	Números menores a 1, Decimales	Números menores a 1, Decimales
Dimensión	Tamaño de la Solución	Tamaño o Largo del Vector de Recolocación

Como se aprecia en la Tabla 5.1 para entregar una solucion concreta al CPMP haciendo uso del lenguaje de programacion Java, para esto se homologaron los conceptos descritos en la primera columna pertenecientes a Black Hole con los con los conceptos de la segunda columna del CPMP. A lo anterior se le dio un carácter concreto en la implementacion mediante aspectos de diseño que seran explicados mas adelante en la seccion 6. A continuacion en la figura 5.1se muestra en un diagrama de flujo simplificado el como se aplico Black Hole a la problemática estudiada.

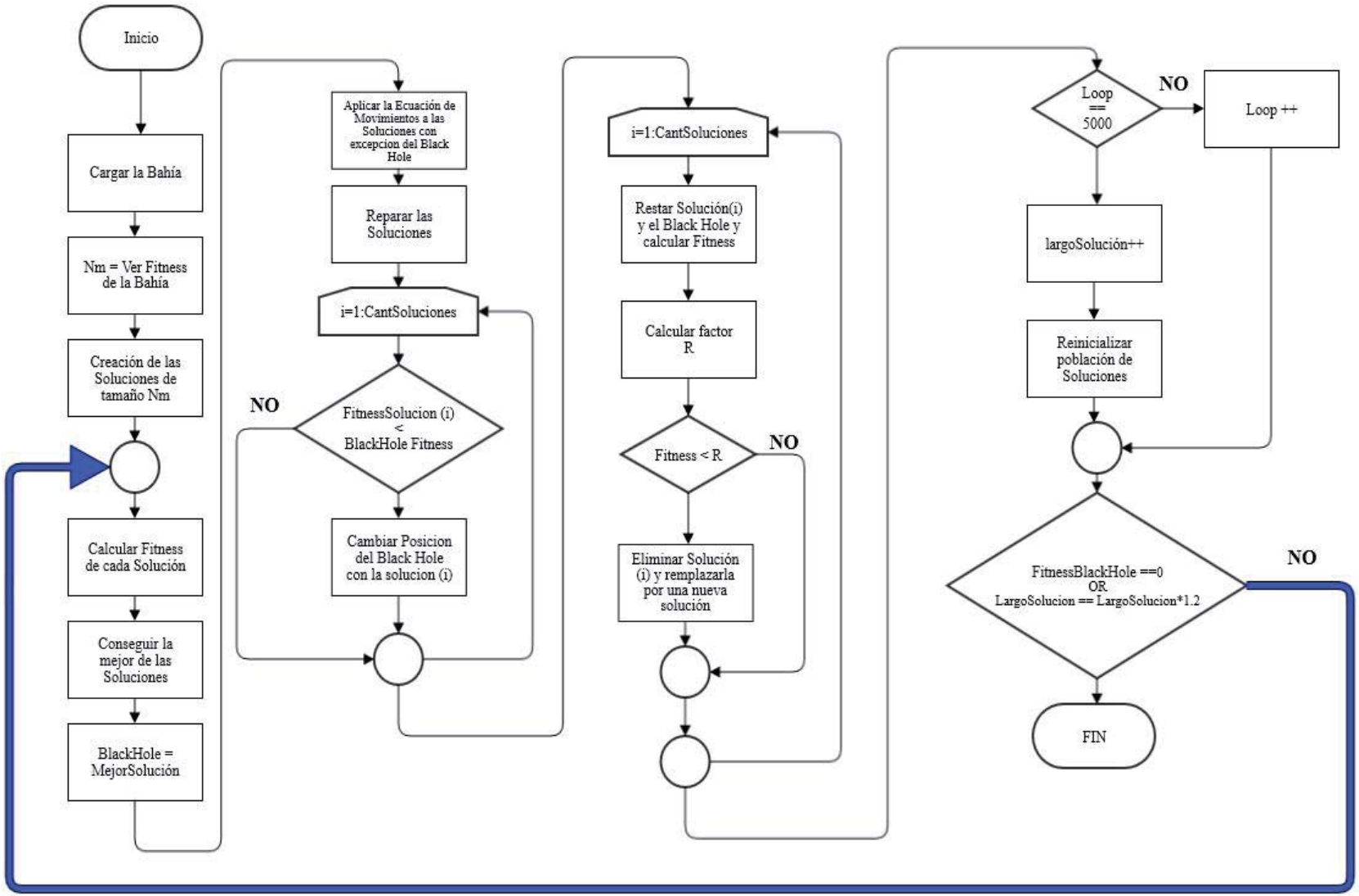


Figura 5.1 Diagrama de Flujo Black Hole.

5.2.1 Cuckoo Search

El algoritmo Cuckoo Search al igual que Black Hole, es una metaheurística basada en poblaciones de soluciones representadas de forma metafórica por Huevos del pájaro Cuckoo, En esta investigación se utilizó una mecánica basada en las ecuaciones de movimiento de los "Vuelos de Lévy" para representar el desplazamiento del ave en función de donde dejará los huevos. Lo anterior sirve para abordar el CPMP y ayudar a minimizar el número de recolocación de contenedores. Su comportamiento se describe en forma sencilla de la siguiente manera:

1. Inicializar una población de Huevos (Vectores de Recolocaciones en este caso) de un tamaño igual al Límite Inferior (LB en este caso).

Bucle Iterativo

2. Por cada Huevo es necesario almacenar el fitness asociado a realizar las recolocaciones de la solución.
3. Se elige un huevo mediante Vuelo de Lévy (Aplicación de la Ecuación 5.2.2.2 sobre el vector)
4. Se elige un huevo al azar Hj
5. Si Hj tiene un mayor fitness que el Huevo escogido mediante Lévy, se reemplaza Hj por este último.
6. Se eliminan una proporción "Pa" de malas soluciones, y son reemplazadas por nuevos Huevos.
7. Si se cumple un criterio de término como demasiadas iteraciones o un buen fitness de la mejor solución, se ha de terminar el bucle.
8. Determinar mejor Huevo de la dimensión actual.

Terminar Bucle

9. Evidenciar solución.

Lo anterior explica de forma procedimental, pero genérica el comportamiento de la metaheurística Cuckoo Search. Lo anterior se toma como principal referente en la implementación de la solución del CPMP.

5.2.2 Movimientos Cuckoo Search

Los movimientos del Cuckoo Search ocurren después de inicializar la población de huevos o posibles soluciones, es en aquel momento cuando se genera una nueva solución candidata mediante un Vuelo de Lévy.

Diversos animales realizan sus movimientos con un cierto patrón, este movimiento es llamado "Vuelo de Lévy". Ocupando este principio, el ave Cuco pone sus huevos siguiendo esta lógica en los nidos de las otras aves.

Utilizando un lenguaje más técnico, un Vuelo de Lévy es un movimiento "aleatorio", que varía dependiendo de una variable. Su fórmula se expresa de la siguiente manera:

$$t^{-\lambda}$$

(5.2.2.1)

Centrándose en el problema, la variable "t" es un valor que en el caso del Cuckoo Search se dejará en uno (más adelante se explicara que "t" es idéntico a alpha), la variable "λ" (lambda), es un número que fluctúa entre uno y tres, sin considerar los límites (el uno y el tres respectivamente). Por lo tanto con la variable lambda se van a generar movimientos infinitos, ya que entre dichos números hay un espacio infinito.

Ahora que ya está explicado el funcionamiento de los Vuelos de Lévy, se procederá a explicar el movimiento del ave Cuckoo, el cual viene dado por la siguiente fórmula.

$$x_i^{(t+1)} = x_i^t + \alpha_i \, Lévy(\lambda)_i$$

(5.2.2.2)

Dónde: α = igual a uno

La fórmula anterior se traduce en que la nueva solución candidata ($x_i^{(t+1)}$)S será igual a una posible solución (x_i^t) más la suma de, la multiplicación matricial de alpha (Que tiene valor uno) y el Vuelo de Lévy que se explicó anteriormente.

Por lo tanto una nueva solución se va a crear utilizando una solución existente que se le aplicará el Vuelo de Lévy, generando como resultado una nueva solución candidata.

5.2.3 Cuckoo Search aplicado al CPMP

Tabla 5.2 Aplicación de Cuckoo Search como Forma de Resolución del CPMP

Concepto Cuckoo Search	Concepto CPMP	Implementación Realizada
Huevo, Nido	Solución de Recolocaciones Consecutivas	Arreglo Bidimensional
Lévy Flights	Alterar una Solución	Ecuación
Fitness	Valor de Función Objetivo	Límite Inferior de la Bahía
Variable	Recolocación	Una Recolocación (Una columna del Vector de Solución)
Dominio	$IN \in \{1,..,Cant\ Columnas\}$	$IN \in \{1,..,Cant\ Columnas\}$
Restricción	Números menores a 1, Decimales	Números menores a 1, Decimales
Dimensión	Tamaño de la Solución	Tamaño o Largo del Vector de Recolocación

Como se aprecia en la Tabla 5.2 y al igual que la Tabla 5.1 para entregar una solucion concreta al CPMP mediante una la implementacion de Java. Para este caso en particular se homologaron los conceptos descritos en la primera columna pertenecientes a Cuckoo Search con los los conceptos de la segunda columna propios del CPMP. A lo anterior se le dio un carácter concreto en la implementacion mediante aspectos de diseño que seran explicados mas adelante en la seccion 6. A continuacion en la figura 5.1se muestra en un diagrama de flujo simplificado el como se aplico Cuckoo Search a la problemática estudiada.

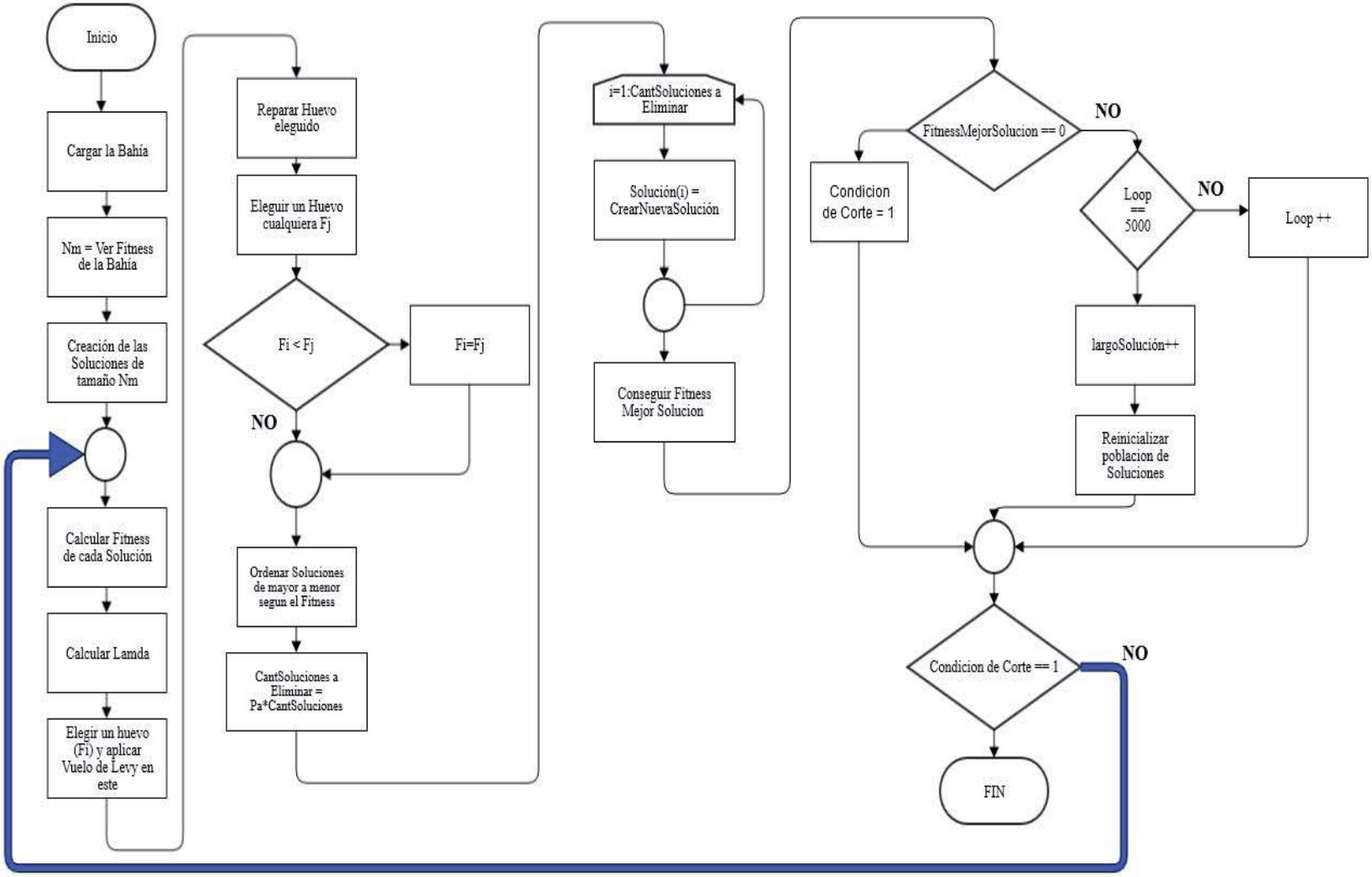


Figura 5.2 Diagrama de Flujo Cuckoo Search.

6 Diseño e Implementación

En este apartado se explica con mayor detalle algunas de los aspectos de diseño utilizados a la hora de implementar los algoritmos para dar solución al CPMP.

6.1 Bahia de Entrada

La Figura 6.1 evidencia una bahia desordenada y sera la instancia a resolver del CPMP para esto, la implementacion desarrollada toma un archivo de texto plano como entrada de extension ".bay" donde se evidencian los *stacks* o columnas de la bahia; posteriormente la configuracion de contendores (bahía) es pasada a un arreglo bidimensional de forma matricial.

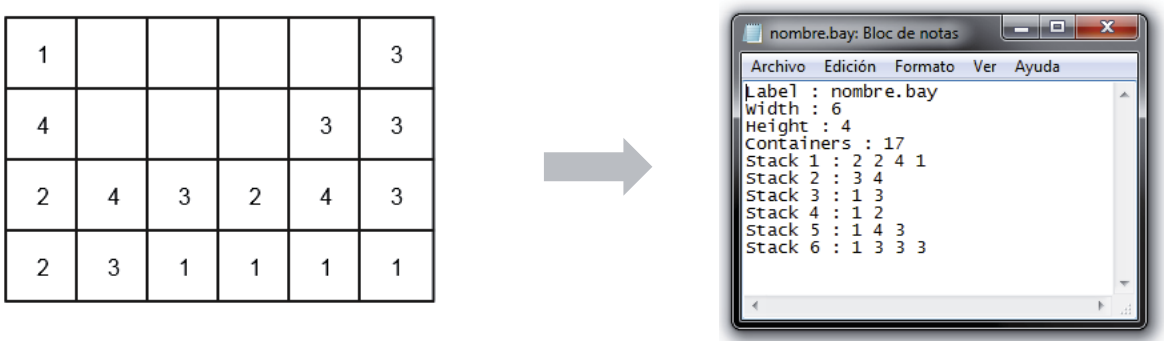


Figura 6.1 Representación de Bahía en Programa.

6.2 Poblacion de Soluciones

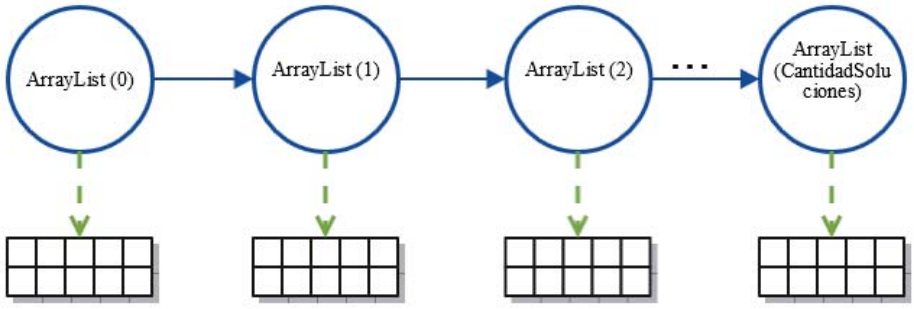


Figura 6.2 Representación de la Población de Soluciones.

Como se puede apreciar en la figura 6.2; la población de soluciones que manejan las implementaciones de ambas metaheurísticas están representadas mediante ArrayList; las cuales son un tipo Arreglo/Lista de tamaño dinámico que provee el lenguaje Java; el arreglo en cuestión puede contener dentro de cada celda cualquier tipo de dato, arreglo u objeto; en este caso para el diseño de la implementación nodo del ArrayList contiene un vector de recolocación y el largo de dicho ArrayList está dado por la cantidad de soluciones, lo cual permite un manejo relativamente sencillo del conjunto de soluciones.

6.3 Vector de Recolocaciones

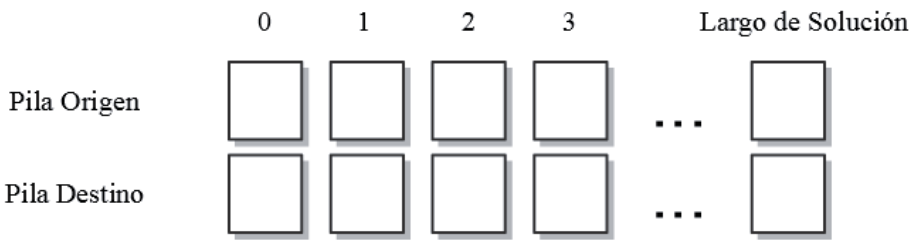


Figura 6.3 Representación Vector de Recolocaciones.

La Figura 6.3 explica a nivel de diseño como son representadas las soluciones (Vectores de Recolocaciones) que entrega el programa; huevos y estrellas en Cuckoo Search y Black Hole respectivamente, cabe destacar que estos movimientos de contenedores están relacionados de forma secuencial, pero no representan necesariamente una solución concreta al CPMP que deje el límite inferior de la bahía en cero. Finalmente es preciso mencionar que el largo del vector es en una primera instancia el valor del límite inferior asociado a una bahía, no obstante este largo se irá incrementando a las 5.000 iteraciones en ambas metaheurísticas. Ahora bien sin importar el largo del vector de recolocaciones, se considera una solución concreta a cualquier vector que luego de ser aplicado en la bahía deje el límite inferior en cero.

Profundizando más en el arreglo propiamente tal; la primer fila del arreglo simboliza la columna de origen desde donde se sacará un contenedor, de igual manera la segunda fila representa la columna de destino donde se depositará el contenedor previamente mencionado.

6.4 Diseño de la Creación de Soluciones Iniciales

Para comenzar la creación de soluciones la función desarrollada en java exige ciertos elementos como argumentos de entrada, estos son en resumidas cuentas: la bahía y sus dimensiones, la cantidad de recolocaciones que tendrá la solución y el número de prioridades presentes en la bahía. Luego se irá progresivamente creando recolocaciones secuencialmente factibles y en las cuales se valida que no ocurran movimientos circulares; esto es, movimientos del tipo (1 a 4) y luego (4 a 1). Entrando en detalle para asignar una única recolocación de contenedores al vector de recolocaciones se aplican los criterios de “Bueno Bueno”, “Malo Bueno”, “Bueno a Vacío” y “Movimiento Cualquiera”. Posterior a haber evaluado los criterios se elige cual tipo de recolocación dará un mejor fitness.

“Bueno Bueno” representa aquella recolocación, en la cual el contenedor de más arriba de la columna de origen tiene bajo éste, un contenedor de igual o menor prioridad, siendo el contenedor de más arriba trasladado a una columna donde quede encima de otro contenedor con igual o menor prioridad. Cabe destacar que lo anterior no implica necesariamente que lo que este debajo del penúltimo contenedor este ordenado.

“Malo Bueno” representa aquella recolocación, en la cual el contenedor de más arriba de la columna de origen tiene bajo este, un contenedor de mayor prioridad, siendo el contenedor de más arriba trasladado a una columna donde quede encima de otro contenedor con igual o menor prioridad.

“Bueno a Vacío” representa aquella recolocación que podría ser considerada un recolocación “Bueno Bueno” pero que el equipo lo separa como criterio por un tema netamente de la implementación de código; en resumidas cuentas, representa aquella recolocación, en la cual el contenedor de más arriba de la columna de origen tiene bajo este, un contenedor de igual o menor prioridad y lo deposita en una columna vacía.

El “Movimiento Cualquiera” es aquel movimiento que se implementó bajo la premisa de que no se encontrara alguna recolocación factible mediante los criterios anteriores y se tuviera que forzosamente que buscar una recolocación de forma randómica; cabe destacar que esto último podría desencadenar una recolocación “Malo Bueno”.

6.5 Modularización

En este apartado de procederán a explicar el funcionamiento de algunas de las piezas de software que desarrollaron y que tienen el propósito de dar solución al Pre-Marshalling Problem, aplicando las heurísticas Black Hole y Cuckoo Search.

Los algoritmos se explicarán de igual manera a cómo se comporta el programa, es decir, se explicarán los algoritmos de forma narrativa y no técnica.

6.5.1 blackHole.java

Black Hole se declara como función estática bajo la clase BH.java del programa; comienza primeramente recibiendo todos los parámetros necesarios desde script.java, los cuales son básicamente la cantidad de soluciones, la lista de soluciones y la bahía con sus dimensiones y su límite inferior. Posteriormente también instancia un arreglo que almacenará el fitness asociado a cada solución del ArrayList

Luego se procede a comenzar el bucle de la metaheurística, el cual se termina cuando el fitness (mínima cantidad de movimientos para que la bahía quede ordenada) sea igual a cero, es decir que la bahía está completamente ordenada o bien hasta que se haya aumentado un 20% el tamaño de las soluciones que se están manejando.

Para comenzar dentro del ciclo iterativo se calcula el fitness de todas las soluciones y se almacenan en un vector auxiliar con valores de fitness. Luego se elige como Black Hole a la solución que tenga el fitness numéricamente más bajo, en otras palabras, el mejor encontrado.

Luego se aplica la fórmula de movimiento explicada anteriormente, esta fórmula puede entregar valores decimales que no están dentro del dominio del problema, por lo que se ha de reparar la solución para que quede dentro del “rango” mediante el criterio explicado en la sección 6.4. Para implementar esto último se crearon dos funciones, reparar recolocación y reparador solución. Reparar recolocación arregla los movimientos que están fuera del dominio

de la bahía, es decir que si la bahía tiene 5 columnas y un movimiento quedó desde la columna 1 a la 10, arregla el valor 10 para que dicho valor este dentro de las 5 columnas válidas. Reparador solución comprueba si los movimientos que se van a hacer, son posibles de realizar consecutivamente, por ejemplo si se va a mover un contenedor de una columna vacía, lo que hace esta función es arreglar dicho problema lógico y transforma esa solución en una realmente factible. Una vez que la solución está validada y es realmente factible se calcula su fitness asociado.

Luego prosigue la permutación de ubicaciones; esto es cuando una estrella tiene menor fitness que el Black Hole se cambian de ubicación, lo mismo pasa con el vector de fitness, los dos cambian sus posiciones en el vector.

Si una estrella cruza el horizonte de eventos, es decir que sea menor al horizonte de eventos (R), esta es reemplazada por una estrella nueva, cuando esta es creada se calcula su respectivo fitness.

Finalmente se establece un criterio de término, cuando el fitness del Black Hole es igual a cero se termina el bucle iterativo y se evidencia la solución, de no ser así se repite una cantidad determinada de veces (5000 para las evaluaciones realizadas), cuando se alcanza ese límite se inicializa de nuevo el Black Hole sólo que con soluciones de tamaño más grandes, es decir, si se tenía una solución de largo 20, ahora se creará una solución de largo 21. Este proceso se repite como se explicó previamente hasta que se encuentre una solución con fitness igual a cero o hasta que se aumente un 20% el largo de la solución en relación al límite inferior original de la bahía.

6.5.2 CuckooSearch.java

Primero que todo Cuckoo Search se declara como función estática bajo la clase CS.java del programa; Cuckoo Search comienza de igual manera que Black Hole, primeramente recibe todos los parámetros necesarios desde script.java, los cuales son básicamente la cantidad de soluciones, la lista de soluciones y la bahía con sus dimensiones y su límite inferior.

Luego se creará un vector auxiliar que almacenará el fitness asociado a cada solución; para lo cual evidentemente el algoritmo comienza calculando el límite inferior de cada solución. Además se crea una constante que formará la fórmula de lévy flights, esta contante es alpha, la cual se explicó anteriormente en la sección 5.2.2. Además se instancia otra variable que responde a la probabilidad de que el pájaro dueño del nido encuentre el huevo que puso el Cucko y lo elimine. En otras palabras es la probabilidad de cambiar las soluciones.

Al comenzar el ciclo iterativo de Cuckoo Search, primero, se identifica la mejor solución, es decir, la que tiene el menor fitness. Posteriormente se crea una nueva solución con Vuelos de Lévy, primero se calcula el valor lambda, que es un valor entre 1 y 3 y se crea la nueva solución, como se explicó en el apartado 5.2.2. Al igual que en el Black Hole se necesita reparar la recolocación y la solución entera bajo los criterios explicados en la sección 6.4. Cuando la solución es realmente factible se calcula su fitness asociado. Posteriormente se elige una solución aleatoria que se comparará con la nueva solución que se creó con Lévy

Flights, si el fitness de la solución creada es menor a la solución elegida aleatoriamente, esta última reemplaza a la solución elegida al azar.

Luego se ordena el vector de soluciones de manera que queden las peores soluciones (mayor fitness) al principio y las mejores al final (menor fitness), en otras palabras se ordenaran de mayor a menor fitness. Posteriormente en función de la probabilidad de que cierta cantidad de huevos no sobrevivan, se elimina la proporción de soluciones más malas (fitness mayor), y se crearan inmediatamente nueva soluciones con su fitness respectivo.

Finalmente se encuentra la solución que contenga el menor fitness, si esta solución tiene fitness igual a cero se termina el ciclo iterativo y se evidencia la solución, de no ser así, pasado una cierta cantidad de iteraciones (para los casos de prueba 5000), se aumenta el tamaño de la solución, al igual que se hace en la implementación de Black Hole. Este proceso se repite hasta que se encuentre una solución con fitness igual a cero o bien hasta que se haya aumentado un 20% más del largo del vector de recolocaciones en relación al límite inferior original de la bahía.

6.5.3 Reparación de Soluciones

La reparación de soluciones tiene como objetivo reparar o arreglar una solución que no sea factible de realizar o cuyos valores estén fuera del dominio de la instancia, por ejemplo si el movimiento generado por la ecuación de movimiento tiene como origen o destino una columna mayor a la definida por la bahía, que el movimiento tenga como destino una pila llena, o que el movimiento tenga como origen una pila vacía.

Estos 2 escenarios posibles, indefinen la solución, es decir, que posterior a haber aplicado la ecuación de movimiento, el vector de recolocaciones resultante no es válido; es debido a esto que se han desarrollado las funciones de Reparar Recolocación y Reparar Solución, con el fin de concretar vectores de recolocaciones que puedan ordenar la bahía satisfactoriamente.

6.5.3.1 Reparar Recolocación

La función toma como parámetro una solución a la cual se le haya aplicado una ecuación de movimiento, o bien haya sido alterada mediante resta vectorial; esto dependiendo de la metaheurística y de en qué parte del código sea invocada. Para luego verificar si la operación a la que estuvo sujeta la solución dejó al vector de recolocaciones con valores inválidos. El criterio utilizado para determinar hasta este punto si los valores son válidos o no, es verificar si pertenecen al dominio, en otras palabras, que los valores sean números enteros entre 1 y el número máximo de pilas posibles en la bahía (número de stacks); además se verifica que el destino y origen de una recolocación no sean iguales.

Posterior a las verificaciones ya mencionadas, en caso de estar mal algún origen o destino de alguna recolocación, la función repara dicho valor por alguno que este dentro del dominio. Cabe destacar que esta función no asegura que el movimiento pueda ser realizado, esto último es verificado por “repararSolucion”.

6.5.3.2 Reparar Solución

La función toma como parámetro la solución a reparar cuyos valores ya han sido constatados por la función "repararRecolocacion"; para luego verificar si las recolocaciones que tiene el vector pueden ser ejecutadas de forma consecutiva, por ejemplo, si la columna de origen está vacía, o la columna de destino está llena. En el caso de que esto no pueda ser posible en alguna recolocación, la función repara dicho movimiento de contenedores por uno que se pueda realizar. Para hacer esto la función analiza recolocación por recolocación que está mal; es decir analiza si el origen como el destino es correcto; ante lo cual existen tres posibles escenarios.

- Origen y Destino Mal: En este caso lo que el algoritmo hace es buscar una recolocación valida y factible mediante los criterios explicados en la sección 6.4.

- Origen Mal: El algoritmo procede a buscar solo una columna de origen factible sin importar como afectara está el fitness, el criterio no puede ser ejecutado en su totalidad debido a que ya está predefinido el destino.

- Destino Mal: El algoritmo procede a buscar solo una columna de destino factible sin importar como afectara está el fitness; el criterio no puede ser ejecutado en su totalidad debido a que ya está predefinido el origen.

7 Experimentos

En el siguiente apartado se describen los resultados Obtenidos por las metaheurísticas aplicadas al CPMP tomando como referencias: la calidad de las soluciones conseguidas según su fitness, el número de iteraciones necesarias para lograr dichas soluciones, la cantidad de soluciones evaluadas por las heurísticas y en las distintas instancias aplicadas (configuraciones de bahía). La tabla 7.1 explica los factores considerados para la experimentación. Estos factores serán utilizados para poder explicar de mejor forma las tablas 7.4 y 7.5.

Tabla 7.1 Factores Considerados Experimentación

Elemento	Significado
Nombre Carpeta	Indica el nombre de la carpeta de instancias, se ha decidido agruparlas así debido a que son más 200 instancias y resulta poco comprensible a la lectura verlas una por una.
Optimo Conocido	Cantidad mínima de recolocaciones (mejor fitness) que dan solución al problema.
Lower Bound	Promedio de los limites inferiores de las bahías de una carpeta
Optimo Logrado	Valor mínimo conseguido de recolocaciones por las metaheurísticas Black Hole y Cuckoo Search que dan solución al CPMP.
Limite Largo Solución	Largo de solución máximo con el que se ejecutaron las metaheurísticas. Este límite es el óptimo conocido más un 30%.
Cant. Rec.	Cantidad de recolocaciones necesarias que quedan para dar solución al problema.
TS	Resolución de las instancias utilizando Tabú Search.
BF	Resolución de las instancias utilizando el algoritmo de los autores Andreas Bortfeldt, Florian Forster.

La tabla 7.2 explica la configuración inicial para dar paso a la experimentación. La cantidad de soluciones iniciales van a ser la cantidad de soluciones con las que va a trabajar la metaheurística para resolver el problema, en este caso son 20 soluciones. El número de iteraciones por largo de vector va a ser la cantidad de repeticiones que se ejecutará la metaheurística antes de aumentar el largo de la solución. Finalmente cada instancia se ejecutará tres veces, para poder entender el comportamiento de la metaheurística.

Tabla 7.2 Configuración de Variables para la Experimentación

Cantidad de Soluciones Iniciales	20
Numero de Iteraciones por Largo de Vector	5000
Número de Ejecuciones por Instancia	3

En la siguiente tabla 7.3 se describe la cantidad de instancias por cada carpeta evaluada.

Tabla 7.3 Cantidad de Instancias por Carpeta.

Nombre carpeta	Cantidad de Instancias por Carpeta
BF1 a BF9	20
CV1 a CV4	10

Para obtener los resultados expuestos se ejecutó cada instancia, (bahía o caso de prueba) un total de 3 veces, de las cuales se calculó el promedio de los resultados de dichas ejecuciones. Se realiza el mismo procedimiento para cada instancia de la carpeta. Luego con los resultados obtenidos de cada instancia de problema se procede a calcular el promedio por carpeta.

7.1 Resultados

A continuación se visualiza la tabla con los resultados obtenidos por Black Hole en las carpetas BF1 a BF9. La primera columna indica el nombre de cada carpeta ejecutada. La segunda columna indica el óptimo conocido en resolver la instancia, es decir en cuántas recolocaciones se logró resolver la carpeta en promedio. La tercera columna indica el límite inferior en promedio de las bahías, es decir la mínima cantidad de recolocaciones en el que se puede resolver la carpeta. La Cuarta columna indica el límite del largo de solución, es decir la cantidad máxima de recolocaciones a la cual se restringieron las soluciones para que la metaheurística termine si no encontraba un óptimo, en este caso ninguna se pudo resolver. La quinta columna indica el óptimo logrado por las metaheurísticas del estudio. La última columna indica la cantidad de recolocaciones necesarias que faltan para obtener un óptimo.

Tabla 7.4 Resultados Instancias Andreas Bortfeldt, Florian Forster Utilizando Black Hole.

Nombre Carpeta	Óptimo Conocido (Promedio)	Lower Bound (Promedio)	Límite Largo Solución	Óptimo Logrado (Promedio)	Cant. Rec. (Promedio)
BF1	29	29.05	38	--	10.54
BF2	36	36.00	47	--	16.88
BF3	29	29.10	38	--	10.99
BF4	36	36.00	47	--	16.94
BF5	42	40.85	55	--	31.19
BF6	49	48.60	64	--	38.26
BF7	43	41.25	56	--	32.68
BF8	50	48.80	65	--	39.31
BF9	52	49.75	68	--	40.66

En la tabla 7.4 se puede observar que Black Hole al aumentar el límite inferior o lower bound, cada vez se va alejando más del óptimo, es decir, a mayor complejidad más difícil va a resultar el resolver la instancia. Comparando los óptimos conocidos, Black Hole está muy

lejos de lograrlos, eso puede ser porque la metaheurística no es la apropiada para este problema.

La siguiente tabla (tabla 7.5) muestra los resultados obtenidos en las carpetas BF1 a las BF9 aplicando la metaheurística Cuckoo Search.

Tabla 7.5 Resultados Instancias Andreas Bortfeldt, Florian Forster Utilizando Cuckoo Search.

Nombre Carpeta	Óptimo Conocido (Promedio)	Lower Bound (Promedio)	Limite Largo Solución	Óptimo Logrado (Promedio)	Cant. Rec. (Promedio)
BF1	29	29.05	38	--	11.32
BF2	36	36.00	47	--	14.45
BF3	29	29.10	38	--	11.13
BF4	36	36.00	47	--	14.32
BF5	42	40.85	55	--	24.66
BF6	49	48.60	64	--	31.56
BF7	43	41.25	56	--	26.09
BF8	50	48.80	65	--	32.40
BF9	52	49.75	68	--	33.97

En la tabla anterior se puede observar que Cuckoo Search al aumentar el límite inferior o Lower Bound cada vez se va alejando más del óptimo, (al igual que Black Hole) es decir, a mayor complejidad más difícil va resultar resolver la instancia. Comparando los óptimos conocidos, Cuckoo Search está muy lejos de lograrlos, eso puede ser porque la metaheurística no es la apropiada para este problema.

Tabla 7.6 Resultados Instancias M. Caserta, y S. Voß Utilizando Black Hole.

Nombre Carpeta	Óptimo Conocido (Promedio)	Lower Bound (Promedio)	Limite Largo Solución	Óptimo Logrado (Promedio)	Cant. Rec. (Promedio)
CV1	10.10 (TS)	7.90	X	13.9	-
CV2	19.10 (BF)	13.60	25	-	5.0
CV3	30.40(BF)	21.50	39	-	10.3
CV4	44.40(BF)	28.60	57	-	20

La tabla 7.6 muestra los resultados obtenidos mediante Black Hole aplicado en las carpetas de los autores M. Caserta, y S. Voß. Para la carpeta CV1 No se utilizó límite inferior, para lo cual se obtuvo uno óptimo logrado, de 13,9, es decir que en 13, 9 recolocaciones en promedio de las diez instancias de la carpeta se logró encontrar un óptimo.

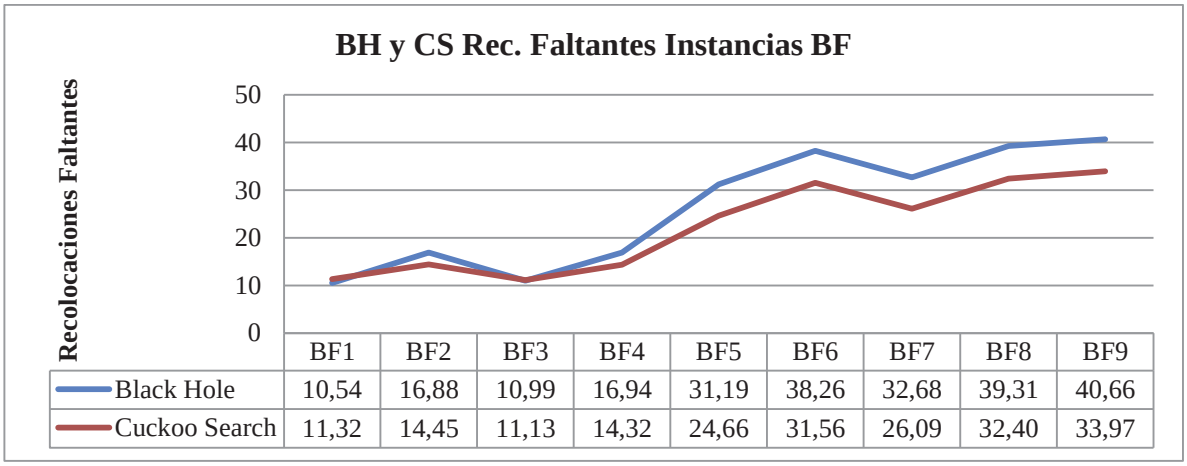
Tabla 7.7 Resultados Instancias M. Caserta, y S. Voß Utilizando Cuckoo Search.

Nombre Carpeta	Óptimo Conocido (Promedio)	Lower Bound (Promedio)	Limite Largo Solución	Óptimo Logrado (Promedio)	Cant. Rec. (Promedio)
CV1	10.10 (TS)	7.90	X	17.37	-
CV2	19.10 (BF)	13.60	25	-	5.98
CV3	30.40(BF)	21.50	39	-	11.1
CV4	44.40(BF)	28.60	57	-	17.27

Como se aprecia en la tabla 7.7, la misma situación ocurre con Cuckoo Search el cual logro encontrar un óptimo en 17,27 recolocaciones en la carpeta CV1.

7.2 Comparación Resultados

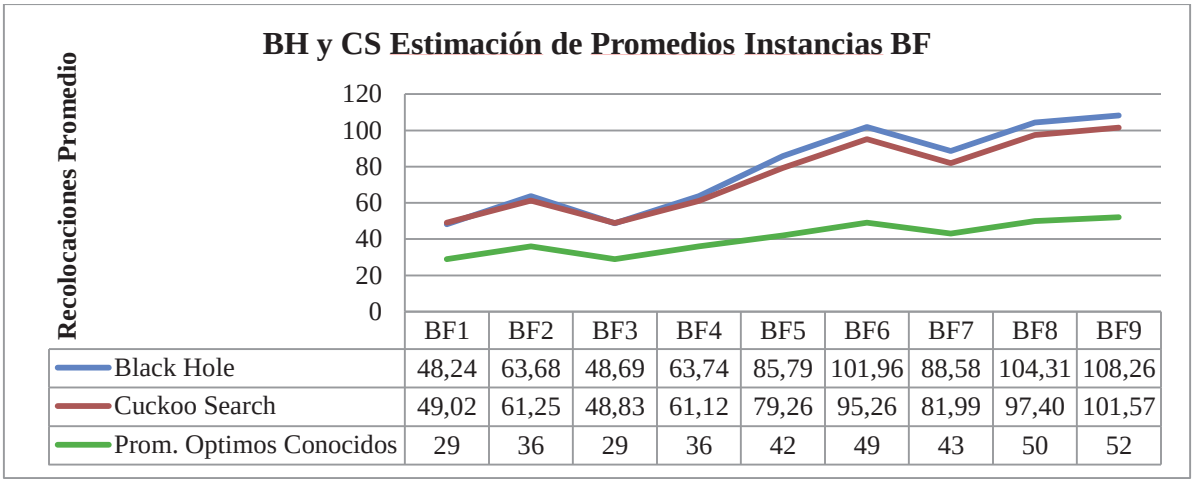
El siguiente gráfico representa un resumen de las tablas 7.4 y 7.5. La línea azul y roja representan la cantidad de recolocaciones que hacen falta para encontrar un óptimo para la metaheurísticas Black Hole y Cuckoo Search respectivamente; bajo el grafico se evidencian los resultados para una mayor claridad.



	BF1	BF2	BF3	BF4	BF5	BF6	BF7	BF8	BF9
Black Hole	10,54	16,88	10,99	16,94	31,19	38,26	32,68	39,31	40,66
Cuckoo Search	11,32	14,45	11,13	14,32	24,66	31,56	26,09	32,40	33,97

Figura 7.1 Grafico Comparación Rec. Faltantes Instancias BF

En el grafico 7.1 se puede apreciar claramente que Black Hole presenta un mejor desempeño en las instancias pequeñas que pertenecen a BF1; es así, como además podemos afirmar que en instancias de Límite Inferior bajo, BH por lo general muestra mejores resultados que Cuckoo Search, en otras palabras está más cerca de solucionar las instancias de problema que Cuckoo Search bajo el setéo establecido. No obstante Cuckoo Search presenta un mejor desempeño en general que Black Hole de manera sostenida en instancias de problema más complejas de mayor límite inferior, presentando vectores de recolocación que dejan la bahía más cerca de estar completamente ordenada que los que genera Black Hole.



	BF1	BF2	BF3	BF4	BF5	BF6	BF7	BF8	BF9
Black Hole	48,24	63,68	48,69	63,74	85,79	101,96	88,58	104,31	108,26
Cuckoo Search	49,02	61,25	48,83	61,12	79,26	95,26	81,99	97,40	101,57
Prom. Optimos Conocidos	29	36	29	36	42	49	43	50	52

Figura 7.2 Grafico Comparación Estimación de Promedios Instancias BF

En la Figura 7.2 La línea verde representa el promedio conocido de óptimos, es decir el promedio en el cual se logró resolver la carpeta por otros autores. Por su parte, la línea azul y roja representan la cantidades estimadas de movimientos en los cuales se hará efectivo el ordenamiento total de una bahía mediante las metaheurísticas Black Hole y Cuckoo Search respectivamente; la cantidades mencionada fueron calculadas tomando en consideración los datos de la Figura 7.1 más el largo máximo de prueba usado en los vectores de recolocación en cada carpeta. Cabe recalcar que estos datos son solo estimaciones, por lo cual al momento de hacer una ejecución completa de los softwares desarrollados estos podrían arrojar resultados mejores o peores que los explicitados en el Grafico 7.2; esto se debe en gran medida a los componentes aleatorios que manejan las metaheurísticas investigadas .

Ahora bien, a partir del gráfico anterior se puede observar que tanto Cuckoo Search como Black Hole estuvieron casi a la misma cantidad de recolocaciones en promedio de resolver cada instancia, esto se traduce que ambas metaheurísticas se comportan de similar manera frente al problema CPMP obteniendo resultados similares. Pero ambas están muy por sobre los óptimos conocidos, esto se pude deber a que las metaheurísticas no son las más apropiadas para resolver el problema de Container Pre Marshalling Problem, No obstante Cuckoo Search presenta unos posibles mejores resultados, lo cual está en estricta relación con la conclusión presentada en el párrafo de explicación de la Figura 7.1.

Tabla 7.9 Comparación de Resultados Instancias CV1 por varios Autores.

	Límite Inferior	CV	BF	TS	CS	Bh
CV1	7.90	21,3	10.5	10.1	17.36	13.9

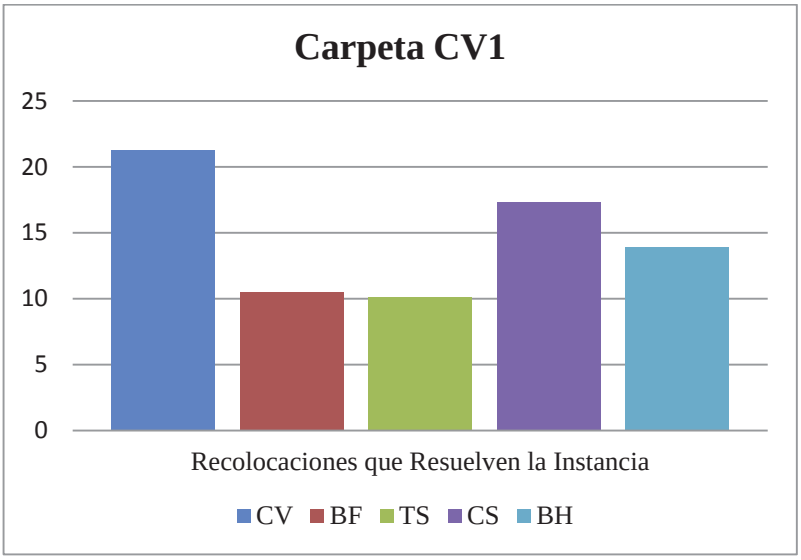


Figura 7.3 Comparación instancia CV1

En la imagen 7.2 en conjunto a la tabla 7.9 se puede observar el comportamiento que tuvieron las metaheurísticas Black Hole (BH) y Cuckoo Search (CS) comparado con otros autores, M. Caserta, y S. Voß (CV), Bortfeldt, Florian Forster (BF), Taboo Search (TS). Se puede apreciar que tanto Cuckoo Search como Black Hole, encontraron un óptimo en 17,36 y 13,9 recolocaciones respectivamente superando a Caserta, y S. Voß, los creadores de estas instancias, los cuales lograron el óptimo en 21,3 recolocaciones.

Aunque es preciso mencionar que los resultados están lejos de ser los mejores, ya que Tabu Search obtuvo un óptimo en 10,1 recolocaciones siendo la metaheurística que mejor se comportó para esta instancia. En conclusión ambas metaheurísticas se comportan de forma similar con las instancias probadas, en otras palabras, ambas están lejos de acercarse a una solución con las restricciones que se utilizaron. Cabe mencionar que no todas las metaheurísticas son las apropiadas para resolver un problema en específico, existen algunas que funcionan mejor que otras para un determinado problema.

8 Conclusiones

A modo de conclusión, resulta pertinente mencionar que el Container Pre-Marshalling Problem es un problema real que se encuentra principalmente en los terminales portuarios de los distintos puertos de las ciudades del mundo, por lo que es un interesante fenómeno logístico que se presta para la continua investigación con fines de encontrar nuevas técnicas que sean útiles para su resolución. Siguiendo con la misma línea de pensamiento; a través de este investigación como equipo hemos podido observar que utilizar una metaheurística para solucionar un problema, independiente sea su índole, resulta ser una propuesta realmente interesante, ya que se aplica la lógica que habita hasta en las cosas más sencillas o sutiles, lo cual nos muestra que incluso los problemas más complejos pueden ser abordados desde la creatividad que hay detrás de un método matemático.

Al comienzo de una investigación de esta índole todo parece desafiante; enfrentar y entender una metaheurística tiene una gran complejidad, ya que, si bien se entiende el comportamiento del método y el problema, la asociación del problema con la metaheurística es el elemento más complicado de realizar, ya que el nivel de abstracción que se le exige al Ingeniero Informático para la interrelación de conceptos es alto. Lo anterior es acrecentado aún más si el problema es propuesto a informáticos ya que es de un área relativamente ajena, no obstante el equipo también considera y reconoce a este hecho como un reto interesante que no solo representa un desafío a nuestra versatilidad como diseñadores de software, sino que además nos entrega la oportunidad de demostrarse a nosotros mismos que podemos llegar a ser profesionales competentes una vez egresados.

En conclusión y centrándonos un poco más en lo técnico solo queda decir que el problema se resolvió primeramente con Black Hole y luego mediante la metaheurística Cuckoo Search, cabe destacar que esta última implementación costo algo menos que la de Black Hole, debido a que ya se tenía conocimiento y ese nivel de abstracción necesario para poder unir la heurística con el problema; Además se reutilizó la mayoría del código, porque al ser el mismo problema, sólo cambia la metaheurística, lo cual aliviana la curva de aprendizaje.

Si bien se cometieron errores de asimilación de conceptos en un primer momento; el equipo se siente conforme e incluso orgulloso de haber sabido respetar la tarea encomendada y poder solucionarla para entregar un producto final que ya tiene resultados concretos. Esto sumado al hecho de haber tenido que volver a implementar ambas metaheurísticas desde el lenguaje Matlab al lenguaje Java en busca de un mejor performance basándonos únicamente en aspectos de diseño; nos hace sentir mayor confianza, ya que ante un problema que a la vista de otros les podría haber parecido insalvable, nosotros persistimos y logramos concretar dichas implementaciones en un tiempo razonable, lo cual ha sido sin lugar a dudas, una gran experiencia y que creemos que corresponde a lo que debiera ser una memoria de título.

Para finalizar solo agradecer a todos aquellos que nos acompañaron durante el trayecto, así como a nuestros profesores quienes nos lo señalaron.

9 Referencias

[1] R. Stahlbock, and S. Voβ, "Operations research at container terminals: A literature update," OR spectrum, vol. 30, pages 1-52, 2008.

[2] Bortfeldt, "A heuristic for the container premarshalling problems," In Proceedings of the 3rd International Conference on Computer and IT Applications in Maritime Industries, pages 419-429, 2004.

[3] Y. Lee, and S. Chao, "A neighborhood search heuristic for pre-marshalling export containers," European Journal Of Operational Research, vol. 196, pages 468-574, 2009.

[4] Y. Lee, and N. Hsu, "An optimization model for the container pre-marshalling problem," Computer & Operations Research, vol. 34, pages 3295-3313, 2007.

[5] Y. Lee, and S. Chao, "A neighborhood search heuristic for pre-marshalling export containers," European Journal Of Operational Research, vol. 196, pages 468-574, 2009.

[6] K. H. Kim, Y. M. Park, and K. R. Ryu, "Deriving decision rules to locate export containers in container yards," European Journal of Operational Research, vol. 124, pages 89-101, 2000.

[7] M. Caserta, and S. Voß, "A corridor method-based algorithm for the pre-marshalling problem," Lecture Notes in Computer Science, vol. 5484, pages 788-797, 2009.

[8] KH. Kim, and GP. Hong, "A heuristic rule for relocating blocks," Computers & Operations Research, vol. 33, pages 940-954, 2006.

[9] M. Molins, M. Salido, and F. Barber, " Domain dependent planning heuristics for locating containers in maritime terminals," IEA/AIE, pages 742-751, 2010**.**

[10] M. Molins, M. Salido, F. Barber, "Intelligent planning for allocating containers in maritime terminals,"Expert Systems with Applications, vol. 39, pages 978-989, 2012.

[11] Exposito-Izoquierdo, N. Melian-Batista, and M. Morena-Vaga, "Pre-marshalling problem" "Heuristic solution method and instance generator", Expert Systems with Applications, vol. 39, pages 8337-8349, 2012.

[12] Andreas Bortfeldt, Florian Forster, "A tree search procedure for the container pre-marshalling problem", Journal European of Operational Research, Vol. 217, pages 531–540, 2012.

[13] Xin-She Yang, "Bat Algorithm: Literature Review and Applications", School of Science and Technology, Middlesex University, The Burroughs, London NW4 4BT, United Kingdom, Vol. 5, pages 141–149, 2013.

[14] R. Y. M. Nakamura, L. A. M. Pereira, K. A. Costa, D. Rodrigues, J. P. Papa, "BBA: A Binary Bat Algorithm for Feature Selection", XXV SIBGRAPI Conference on Graphics, Patterns and Images, pages 291-297, 2012

[15] Abdolreza Hatamlou. Black hole: A new heuristic optimization approach for data clustering. **Information Sciences.** Vol. 222, pages 175-184, 2012

[16] X.-S. Yang, S. Deb, "Cuckoo search via Lévy flights", in: Proc. Of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), India. IEEE Publications, USA, pages 210-214, 2009

Anexos

A: Glosario de Términos

Bahía: Instancia del CPMP o configuración de contenedores a dar solución.

Black Hole: metaheurística basada en poblaciones de soluciones.

Container Pre- Marshalling Problem: Problema de minimización de recolocaciones de contenedores en un puerto.

Cuckoo Search: metaheurística basada en poblaciones de soluciones.

Fitness: Valor asociado al cálculo del Límite Inferior.

Lévy Flight: También conocidos en su traducción al español “Vuelos de Lévy”, es la ecuación de movimiento de la metaheurística Cuckoo Search.

Límite Inferior: Cantidad mínima posible de recolocaciones de contenedores.

B: Lista de Siglas

BF: Instancias propuestas por los autores A. Bortfeldt y F. Forster.

BH: Black Hole.

Cant. Rec.: Cantidad de recolocaciones necesarias que quedan para dar solución al problema.

CPMP: Container Pre-Marshalling Problem.

CS: Cuckoo Search.

CV: Instancias propuestas por los autores M. Caserta y S. Voß.

LB: Límite Inferior del inglés “Lower Bound”.

LC: Instancias propuestas por los autores Y. Lee y S. Chao.

Nm: Límite Inferior del inglés “Lower Bound”.

Pa: Proporción.

R: Horizonte de eventos.

TS: Taboo Search

C: Especificación Resultados por Instancia

A continuación se muestran los resultados obtenidos en cada instancia con la configuración determinada en el informe.

Para entender las tablas se entrega un ejemplo:

Tabla C.1 Tabla de Ejemplo de Resultados.

BH	**29**		BF1	
Instancia	Límite Inferior	Largo Max Solución Probado	Optimo Logrado	Cant. Mov.
BF1_1	29	38	--	10,6
				10,6

BH: Nombre de los autores de los casos de prueba.
29: Límite inferior en promedio de las instancias de la carpeta.
BF1: nombre carpeta.
BF1_1: Nombre de una instancia o bahía.
Límite Inferior: Limite inferior de una instancia.
Largo Max Solución Probado: Largo de solución máximo con el que se realizaron las pruebas.
Optimo Logrado: Optimo logrado por la metaheurística.
Cant. Mov: Cantidad de recolocaciones necesarias para resolver el problema, al alcanzar el Largo max. Solución Probada.

A. Bortfeldt y F. Forster. Utilizando Black Hole:

Tabla C.2 Resultados BH Instancias carpeta BF1.

BH	29		BF1	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF1_1	29	38	--	10,6
BF1_2	29	38	--	8,93
BF1_3	30	39	--	12,46
BF1_4	29	38	--	11
BF1_5	29	38	--	10,63
BF1_6	29	38	--	11,33
BF1_7	29	38	--	10,83
BF1_8	29	38	--	8,96
BF1_9	29	38	--	9,86
BF1_10	29	38	--	9,73
BF1_11	29	38	--	14,4
BF1_12	29	38	--	10
BF1_13	29	38	--	9,96
BF1_14	29	38	--	10,4
BF1_15	29	38	--	8,8
BF1_16	29	38	--	8,96
BF1_17	29	38	--	9,83
BF1_18	29	38	--	10,8
BF1_19	29	38	--	12,46
BF1_20	29	38	--	10,76
				10,535

Tabla C.3 Resultados BH Instancias carpeta BF2.

BH	36		BF2	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF2_1	36	47	--	13,33
BF2_2	36	47	--	18
BF2_3	36	47	--	15,88
BF2_4	36	47	--	13,72
BF2_5	36	47	--	17,22
BF2_6	36	47	--	13,36
BF2_7	36	47	--	15,38
BF2_8	36	47	--	12,86
BF2_9	36	47	--	21,86
BF2_10	36	47	--	14,75
BF2_11	36	47	--	21,55
BF2_12	36	47	--	13,08
BF2_13	36	47	--	19,86
BF2_14	36	47	--	20,69
BF2_15	36	47	--	17,47
BF2_16	36	47	--	13,88
BF2_17	36	47	--	19,88
BF2_18	36	47	--	19,19
BF2_19	36	47	--	15,5
BF2_20	36	47	--	20,11
				16,8785

Tabla C.4 Resultados BH Instancias carpeta BF3.

BH	**29**		BF3	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF3_1	29	38	--	10,86
BF3_2	29	38	--	12,1
BF3_3	29	38	--	13,06
BF3_4	29	38	--	9,4
BF3_5	29	38	--	10,66
BF3_6	30	39	--	10,26
BF3_7	29	38	--	9,96
BF3_8	29	38	--	9,9
BF3_9	29	38	--	12,7
BF3_10	29	38	--	12,8
BF3_11	31	38	--	6,8
BF3_12	29	38	--	11,8
BF3_13	29	38	--	9,1
BF3_14	29	38	--	15,4
BF3_15	29	38	--	8,93
BF3_16	29	38	--	10,16
BF3_17	29	38	--	12,3
BF3_18	29	38	--	6,6
BF3_19	29	38	--	14,83
BF3_20	29	38	--	12,16
				10,989

Tabla C.5 Resultados BH Instancias carpeta BF4.

BH	**36**		BF4	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF4_1	36	47	--	13,55
BF4_2	36	47	--	17
BF4_3	36	47	--	20,33
BF4_4	36	47	--	17,72
BF4_5	36	47	--	16,83
BF4_6	36	47	--	20,69
BF4_7	36	47	--	15
BF4_8	36	47	--	16,86
BF4_9	36	47	--	15,86
BF4_10	36	47	--	12,27
BF4_11	36	47	--	12,38
BF4_12	36	47	--	19,41
BF4_13	36	47	--	18,72
BF4_14	36	47	--	14,86
BF4_15	36	47	--	15,86
BF4_16	36	47	--	16,52
BF4_17	36	47	--	14,3
BF4_18	36	47	--	19,44
BF4_19	36	47	--	20,22
BF4_20	36	47	--	20,94
				16,938

Tabla C.6 Resultados BH Instancias carpeta BF5.

BH	**42**		BF5	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF5_1	43	55	--	34,05
BF5_2	41	55	--	33,35
BF5_3	42	55	--	29,76
BF5_4	44	55	--	30
BF5_5	41	55	--	34,21
BF5_6	42	55	--	33,55
BF5_7	39	55	--	29,61
BF5_8	41	55	--	30
BF5_9	43	55	--	32,2
BF5_10	43	55	--	28,15
BF5_11	39	55	--	33,57
BF5_12	42	55	--	30,5
BF5_13	40	55	--	32,22
BF5_14	42	.	--	30,64
BF5_15	41	55	--	30,15
BF5_16	42	55	--	26,15
BF5_17	44	55	--	33,25
BF5_18	43	55	--	33,43
BF5_19	41	55	--	28,27
BF5_20	41	55	--	31,08
				31,1855

Tabla C.7 Resultados BH Instancias carpeta BF6.

BH	**49**		BF6	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF6_1	48	64	--	37,8
BF6_2	48	64	--	38,37
BF6_3	48	64	--	37,43
BF6_4	49	64	--	40,04
BF6_5	49	64	--	38,72
BF6_6	48	64	--	38,41
BF6_7	48	64	--	40,29
BF6_8	50	64	--	35,41
BF6_9	48	64	--	36,98
BF6_10	50	64	--	39,78
BF6_11	48	64	--	37,27
BF6_12	48	64	--	40,47
BF6_13	48	64	--	37,83
BF6_14	49	64	--	38,41
BF6_15	49	64	--	37,27
BF6_16	48	64	--	40,84
BF6_17	48	64	--	38,13
BF6_18	48	64	--	38,57
BF6_19	48	64	--	36,33
BF6_20	48	64	--	36,88
				38,2615

Tabla C.8 Resultados BH Instancias carpeta BF7.

BH	**43**		BF7	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF7_1	42	56	--	28,75
BF7_2	42	56	--	34,26
BF7_3	43	56	--	36,94
BF7_4	41	56	--	31,83
BF7_5	46	56	--	34,8
BF7_6	39	56	--	32,18
BF7_7	42	56	--	35,38
BF7_8	42	56	--	31,38
BF7_9	46	56	--	30,9
BF7_10	41	56	--	32,64
BF7_11	42	56	--	32,21
BF7_12	43	56	--	32,06
BF7_13	41	56	--	34,78
BF7_14	45	56	--	30,2
BF7_15	41	56	--	33,06
BF7_16	43	56	--	32,97
BF7_17	41	56	--	36,48
BF7_18	43	56	--	28,85
BF7_19	44	56	--	31,26
BF7_20	40	56	--	32,7
				32,6815

Tabla C.9 Resultados BH Instancias carpeta BF8.

BH	**50**		BF8	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF8_1	49	65	--	41,62
BF8_2	49	65	--	41,79
BF8_3	48	65	--	38,2
BF8_4	50	65	--	37,77
BF8_5	49	65	--	37,94
BF8_6	48	65	--	40,31
BF8_7	48	65	--	39,39
BF8_8	48	65	--	39,85
BF8_9	48	65	--	40,92
BF8_10	48	65	--	39,98
BF8_11	51	65	--	39,75
BF8_12	48	65	--	37,85
BF8_13	48	65	--	39,15
BF8_14	48	65	--	40,18
BF8_15	48	65	--	39,2
BF8_16	48	65	--	40,83
BF8_17	49	65	--	37,27
BF8_18	49	65	--	36,57
BF8_19	48	65	--	41,53
BF8_20	49	65	--	36,03
				39,3065

Tabla C.10 Resultados BH Instancias carpeta BF9.

BH	52		BF9	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF9_1	49	68	--	41,2
BF9_2	49	68	--	37,03
BF9_3	48	68	--	41,54
BF9_4	50	68	--	39,19
BF9_5	49	68	--	38,85
BF9_6	48	68	--	43,46
BF9_7	48	68	--	42,45
BF9_8	48	68	--	41,91
BF9_9	48	68	--	39,41
BF9_10	48	68	--	38,41
BF9_11	51	68	--	38,98
BF9_12	48	68	--	40,96
BF9_13	48	68	--	39,05
BF9_14	48	68	--	40,66
BF9_15	48	68	--	42,24
BF9_16	48	68	--	42,39
BF9_17	49	68	--	42,31
BF9_18	49	68	--	36,77
BF9_19	48	68	--	44,68
BF9_20	49	68	--	41,8
				40,66

M. Caserta y S. Voß. Utilizando Black Hole

Tabla C.11 Resultados BH Instancias carpeta CV1.

BH	10		CV1	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
CV1_1	8	X	20,33	--
CV1_2	6	X	10	--
CV1_3	7	X	13,33	--
CV1_4	7	X	19,33	--
CV1_5	8	X	19,33	--
CV1_6	6	X	11,33	--
CV1_7	5	X	10	--
CV1_8	7	X	12,33	--
CV1_9	6	X	12,67	--
CV1_10	6	X	10,33	--
			13,90	

Tabla C.12 Resultados BH Instancias carpeta CV2.

BH	19		CV2	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
CV2_1	12	25	--	6,0
CV2_2	12	25	13,3	--
CV2_3	8	25	--	0,4
CV2_4	15	25	--	5,5
CV2_5	12	25	--	5,6
CV2_6	11	25	--	5,4
CV2_7	13	25	--	5,9
CV2_8	11	25	--	4,5
CV2_9	16	25	--	5,1
CV2_10	13	25	--	6,66
				5,0

Tabla C.13 Resultados BH Instancias carpeta CV3.

BH	30		CV3	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
CV3_1	20	39	--	10,5
CV3_2	20	39	--	8,8
CV3_3	17	39	--	8,9
CV3_4	19	39	--	9,3
CV3_5	20	39	--	9,7
CV3_6	20	39	--	11,0
CV3_7	22	39	--	14,1
CV3_8	20	39	--	11,0
CV3_9	20	39	--	12,0
CV3_10	17	39	--	7,9
				10,3

Tabla C.14 Resultados BH Instancias carpeta CV4.

BH	44		CV4	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
CV4_1	22	57	--	13,5
CV4_2	29	57	--	16,1
CV4_3	29	57	--	26,2
CV4_4	30	57	--	24,9
CV4_5	32	57	--	22,8
CV4_6	29	57	--	25,8
CV4_7	23	57	--	15,4
CV4_8	26	57	--	15,3
CV4_9	27	57	--	21,2
CV4_10	25	57	--	19,2
				20,0

A. Bortfeldt y F. Forster. Utilizando Cuckoo Search:

Tabla C.15 Resultados CS Instancias carpeta BF1

CS	29		BF1	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF1_1	29	38	--	11,53
BF1_2	29	38	--	9,76
BF1_3	30	39	--	13,13
BF1_4	29	38	--	12,06
BF1_5	29	38	--	11,36
BF1_6	29	38	--	12,06
BF1_7	29	38	--	11,8
BF1_8	29	38	--	9,66
BF1_9	29	38	--	10,46
BF1_10	29	38	--	10,6
BF1_11	29	38	--	13,77
BF1_12	29	38	--	10,8
BF1_13	29	38	--	10,53
BF1_14	29	38	--	11,53
BF1_15	29	38	--	9,6
BF1_16	29	38	--	9,93
BF1_17	29	38	--	10,46
BF1_18	29	38	--	11,86
BF1_19	29	38	--	13,73
BF1_20	29	38	--	11,7
				11,32

Tabla C.16 Resultados CS Instancias carpeta BF2.

CS	36		BF2	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF2_1	36	47	--	13,08
BF2_2	36	47	--	14,47
BF2_3	36	47	--	13,69
BF2_4	36	47	--	13,41
BF2_5	36	47	--	14,55
BF2_6	36	47	--	12,38
BF2_7	36	47	--	14,5
BF2_8	36	47	--	12,69
BF2_9	36	47	--	16,63
BF2_10	36	47	--	13,55
BF2_11	36	47	--	16,41
BF2_12	36	47	--	12,88
BF2_13	36	47	--	16,27
BF2_14	36	47	--	15,58
BF2_15	36	47	--	14,41
BF2_16	36	47	--	14,02
BF2_17	36	47	--	15,94
BF2_18	36	47	--	15,25
BF2_19	36	47	--	14,02
BF2_20	36	47	--	15,19
				14,446

Tabla C.17 Resultados CS Instancias carpeta BF3.

CS	29		BF3	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF3_1	29	38	--	11,6
BF3_2	29	38	--	12,83
BF3_3	29	38	--	13,43
BF3_4	29	38	--	10,46
BF3_5	29	38	--	11,16
BF3_6	30	39	--	11,13
BF3_7	29	38	--	10,76
BF3_8	29	38	--	8,7
BF3_9	29	38	--	8,03
BF3_10	29	38	--	13,9
BF3_11	31	38	--	7,83
BF3_12	29	38	--	12,5
BF3_13	29	38	--	9,96
BF3_14	29	38	--	12,92
BF3_15	29	38	--	9,83
BF3_16	29	38	--	10,83
BF3_17	29	38	--	13,1
BF3_18	29	38	--	7,43
BF3_19	29	38	--	13,25
BF3_20	29	38	--	12,96
				11,1305

Tabla C.18 Resultados CS Instancias carpeta BF4.

CS	36		BF4	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF4_1	36	47	--	13,55
BF4_2	36	47	--	14,27
BF4_3	36	47	--	15,63
BF4_4	36	47	--	14,27
BF4_5	36	47	--	14,08
BF4_6	36	47	--	15,5
BF4_7	36	47	--	13,3
BF4_8	36	47	--	13,63
BF4_9	36	47	--	13,8
BF4_10	36	47	--	12,11
BF4_11	36	47	--	13,19
BF4_12	36	47	--	16,25
BF4_13	36	47	--	15,83
BF4_14	36	47	--	14,47
BF4_15	36	47	--	13,5
BF4_16	36	47	--	14,38
BF4_17	36	47	--	13,33
BF4_18	36	47	--	14,88
BF4_19	36	47	--	15,11
BF4_20	36	47	--	15,41
				14,3245

Tabla C.19 Resultados CS Instancias carpeta BF5.

CS	42		BF5	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF5_1	43	55	--	27,74
BF5_2	41	55	--	26,57
BF5_3	42	55	--	22,28
BF5_4	44	55	--	22,95
BF5_5	41	55	--	27,45
BF5_6	42	55	--	26,19
BF5_7	39	55	--	23,51
BF5_8	41	55	--	23,84
BF5_9	43	55	--	26,15
BF5_10	43	55	--	21,75
BF5_11	39	55	--	27,16
BF5_12	42	55	--	23,58
BF5_13	40	55	--	25,53
BF5_14	42	55	--	24,57
BF5_15	41	55	--	23,13
BF5_16	42	55	--	20,13
BF5_17	44	55	--	27,05
BF5_18	43	55	--	27,3
BF5_19	41	55	--	21,58
BF5_20	41	55	--	24,81
				24,66

Tabla C.20 Resultados CS Instancias carpeta BF6.

CS	**49**		BF6	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF6_1	48	64	--	30,76
BF6_2	48	64	--	31,8
BF6_3	48	64	--	30,52
BF6_4	49	64	--	32,91
BF6_5	49	64	--	31,27
BF6_6	48	64	--	32,09
BF6_7	48	64	--	32,82
BF6_8	50	64	--	28,76
BF6_9	48	64	--	30,92
BF6_10	50	64	--	33,05
BF6_11	48	64	--	30,78
BF6_12	48	64	--	33,62
BF6_13	48	64	--	30,93
BF6_14	49	64	--	31,78
BF6_15	49	64	--	30,94
BF6_16	48	64	--	34,53
BF6_17	48	64	--	30,92
BF6_18	48	64	--	32,22
BF6_19	48	64	--	29,98
BF6_20	48	64	--	30,62
				31,56

Tabla C.21 Resultados CS Instancias carpeta BF7.

CS	43		BF7	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF7_1	42	56	--	20,97
BF7_2	42	56	--	28
BF7_3	43	56	--	30,88
BF7_4	41	56	--	25,12
BF7_5	46	56	--	28,57
BF7_6	39	56	--	25,93
BF7_7	42	56	--	30,19
BF7_8	42	56	--	23,07
BF7_9	46	56	--	24,41
BF7_10	41	56	--	27,02
BF7_11	42	56	--	25,73
BF7_12	43	56	--	25,56
BF7_13	41	56	--	26,63
BF7_14	45	56	--	23,75
BF7_15	41	56	--	26,04
BF7_16	43	56	--	25,91
BF7_17	41	56	--	30,33
BF7_18	43	56	--	22,25
BF7_19	44	56	--	24,51
BF7_20	40	56	--	26,95
				26,09

Tabla C.22 Resultados CS Instancias carpeta BF8.

CS	50		BF8	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF8_1	49	65	--	34,09
BF8_2	49	65	--	34,51
BF8_3	48	65	--	31,29
BF8_4	50	65	--	31,74
BF8_5	49	65	--	30,83
BF8_6	48	65	--	34,07
BF8_7	48	65	--	32,01
BF8_8	48	65	--	33,12
BF8_9	48	65	--	35,05
BF8_10	48	65	--	32,13
BF8_11	51	65	--	32,79
BF8_12	48	65	--	31,04
BF8_13	48	65	--	32,15
BF8_14	48	65	--	32,88
BF8_15	48	65	--	32,01
BF8_16	48	65	--	34,46
BF8_17	49	65	--	30,24
BF8_18	49	65	--	30,16
BF8_19	48	65	--	34,6
BF8_20	49	65	--	28,81
				32,40

Tabla C.23 Resultados CS Instancias carpetas BF9.

CS	52		BF9	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
BF9_1	49	68	--	35.58
BF9_2	49	68	--	30.43
BF9_3	48	68	--	33.88
BF9_4	50	68	--	32.26
BF9_5	49	68	--	32.40
BF9_6	48	68	--	37.03
BF9_7	48	68	--	35.76
BF9_8	48	68	--	34.60
BF9_9	48	68	--	33.09
BF9_10	48	68	--	32.69
BF9_11	51	68	--	32.41
BF9_12	48	68	--	33.38
BF9_13	48	68	--	30.86
BF9_14	48	68	--	33.95
BF9_15	48	68	--	35.94
BF9_16	48	68	--	35.58
BF9_17	49	68	--	35.50
BF9_18	49	68	--	30.68
BF9_19	48	68	--	38.04
BF9_20	49	68	--	35.35
				33.97

Tabla C.24 Resultados CS Instancias carpeta CV1.

CS	10		CV1	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
CV1_1	8	X	22,33	--
CV1_2	6	X	11	--
CV1_3	7	X	17,3	--
CV1_4	7	X	25.33.	--
CV1_5	8	X	28	--
CV1_6	6	X	13..66	--
CV1_7	5	X	13,33	--
CV1_8	7	X	18	--
CV1_9	6	X	17,66	--
CV1_10	6	X	11,33	--
			17,36875	

Tabla C.25 Resultados CS Instancias carpeta CV2.

CS	19		CV2	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
CV2_1	12	X	77,66	6,7
CV2_2	12	25	--	3,6
CV2_3	8	25	14,3	--
CV2_4	15	25	--	6,3
CV2_5	12	25	--	6,2
CV2_6	11	25	--	6,0
CV2_7	13	25	--	6,7
CV2_8	11	25	--	5,1
CV2_9	16	25	--	5,9
CV2_10	13	25	--	7,25
				6,0

Tabla C.26 Resultados CS Instancias carpeta CV3.

CS	30		CV3	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
CV3_1	20	39	--	11,4
CV3_2	20	39	--	9,6
CV3_3	17	39	--	9,6
CV3_4	19	39	--	10,1
CV3_5	20	39	--	10,8
CV3_6	20	39	--	11,8
CV3_7	22	39	--	14,6
CV3_8	20	39	--	11,8
CV3_9	20	39	--	12,8
CV3_10	17	39	--	8,6
				11,10

Tabla C.27 Resultados CS Instancias carpeta CV4.

CS	**44**		CV4	
Instancia	Límite Inferior	Largo Max Solución Probado	Óptimo Logrado	cant. Mov.
CV4_1	22	57	--	14,3
CV4_2	29	57	--	15,1
CV4_3	29	57	--	21,0
CV4_4	30	57	--	20,3
CV4_5	32	57	--	17,7
CV4_6	29	57	--	20,1
CV4_7	23	57	--	15,4
CV4_8	26	57	--	14,7
CV4_9	27	57	--	17,6
CV4_10	25	57	--	16,7
				17,27