Resolución al Pre-Marshalling Problem utilizando

Cat Swarm Optimization y Harmony Search

Mauricio David Saavedra Castro
Diego Felipe Hidalgo Gallegos

Profesor Guía: **Ricardo Soto de Giorgis**
Profesor co-referente: **Broderick Crawford Labrin**

Ingeniería de ejecución en informática

Valparaíso, Diciembre, 2015

Índice

Índice i

Resumen ii

Abstract ii

Lista de figuras iii

Lista de tablas iv

Lista de abreviaturas v

1 Introducción 1

1.1 Objetivos generales 2
1.2 Objetivos específicos 2

2 Estado del arte 3

3 Pre-marshalling Problem 4

3.1 Definición del problema 4
3.2 Representación del problema 5
3.3 Representación de la solución: 6

4 Cat Swarm Optimization 7

4.1 El gato 7
4.2 Tracing Mode. 8
4.3 Seeking Mode 8
4.4 El algoritmo 9
4.5 Solución inicial inteligente 10
4.5.1 Primer caso 10
4.5.2 Segundo caso 10
4.5.3 Tercer caso 11

5 Harmony Search 13

5.1 Inicializar parámetros del algoritmo 13
5.2 Inicialización de la memoria armónica 14
5.3 Improvisación musical 15
5.4 Actualización de la memoria armónica 16
5.5 Criterio de parada 16

6 Análisis de resultados 17

6.1 Entorno de Desarrollo 17

7 Conclusión 19

8 Referencias 20

Anexo .....

A: Desglose de resultados de los casos realizados por Cat Swarm Optimization . A-1

B: Desglose de resultados de los casos realizados por Harmony Search B-1

Resumen

El objetivo del Pre-Marshalling Problem es minimizar la cantidad de recolocaciones para reorganizar una bahía de contenedores en función de su secuencia de envío. Esta reorganización es clave para la correcta operación de los puertos dado que la secuencia de llegada de contenedores al puerto no es compatible con la secuencia de envío. En este proyecto, se utilizará la Metaheurística Cat Swarm Optimization para la resolución de este problema. Esta técnica se basa en el comportamiento de los gatos y sus dos estados más característicos, el estado de reposo y el de rastreo. La segunda técnica utilizada se denomina Harmony Search, una Metaheurística basada en la improvisación musical. Luego de su ejecución, fue posible determinar que ambos algoritmos ofrecen resultados cercanos al óptimo conocido, en configuraciones simples. Por otro lado, en bahías de mayor complejidad de ordenamiento, no se obtuvieron resultados.

Palabras claves: Problema de Pre-Marshalling, Optimización por Enjambre de Gato, Búsqueda Harmónica.

Abstract

The purpose of the Pre-Marshalling Problem is to minimize the amount of relocations for reorganizing a container bay according to their shipment sequence. This reorganization is key to the proper operation of ports because the arrival sequence of containers to the port does not match their departure sequence. In this project, we employ the Cat Swarm Optimization metaheuristic for the resolution of this problem. This technique is based on the cat behavior and their two most characteristic states, the state of rest and the tracking. The second technique used is called Harmony Search, a metaheuristic based on the musical improvisation. After his execution, it was possible to determine that both algorithms offer results very close of known optimums for simple configurations. Moreover, in more complex bays, no results were obtained.

Keywords: Pre-Marshalling problem, Cat Swarm Optimization, Search Harmonica.

Lista de figuras

Figura 3-1 Puerto de Valparaíso 4
Figura 3-2 Representación de la matriz M 5
Figura 3-3 Vector solución 6
Figura 4-1 Componentes del gato 7
Figura 4-2 Bahía de ejemplo para el caso 1 10
Figura 4-3 Bahía de ejemplo para el caso 2 11
Figura 4-4 Bahía de ejemplo para el caso 3 11
Figura 4-5 Bahía del caso 3 modificada 12
Figura 5-1 Memoria Armónica, donde cada casilla Xn contiene una lista de movimientos de la forma A,B. 14
Figura 5-2 Improvisación musical. 16

Lista de tablas

Tabla 6-1 Casos de prueba analizados 17
Tabla 6-2 Tabla resultado de los casos de prueba 18
Tabla A-1 Resultados del caso CV1 con CSO A-1
Tabla A-2 Resultados del caso CV2 con CSO A-1
Tabla A-3 Resultados del caso BF1 con CSO A-2
Tabla A-4 Resultados del caso BF2 con CSO A-3
Tabla B-1 Resultados del caso CV1 con HS B-1
Tabla B-2 Resultados del caso CV2 con HS B-1
Tabla B-3 Resultados del caso BF1 con HS B-2
Tabla B-4 Resultados del caso BF2 con HS B-3

Lista de abreviaturas

PMP : Pre-Marshalling Problem.
MH : Metaheuristica
ACO : Ant Colony Optimization.
CSO : Cat Swarm Optimization.
SMP : Seeking Memory Pool.
SRD : Search Range of the selected Dimension.
CDC : Count of Dimension to Change.
SPC : Self Position Consideration.
MR : Mixing Ratio.
HS : Harmony Search.
BA : Búsqueda Armónica.
HMCR : Harmony Memory Considering Rate.
PAR : Pitch Adjusting Rate.
BW : Bandwidth.
HM : Harmony Memory.
HMS : Harmony Memory Size.

1 Introducción

El terminal portuario juega un papel fundamental en el intercambio intermodal en las redes globales de transporte de mercancías, debido a esto, se almacenan y retiran diariamente contenedores, obligando a tener un plan de ordenamiento y de ubicación que les permita su pronta salida y no retrasar el proceso. Pero como la configuración de llegada es diferente a la configuración de salida, se solicita el trabajo de una grúa para que reordene los contenedores. Los terminales deben incrementar sus capacidades de almacenamiento y manipulación de contenedores, invirtiendo en nuevas infraestructuras, como también, mejorar la gestión de los recursos materiales y humanos disponibles. Para esto, se necesita contar con un método para minimizar los costes y maximizar la calidad de los servicios que el terminal puerto ofrece, a este conflicto se le denomina Pre-Marshalling Problem.

Debido a lo complejidad que tiene este problema a la hora de encontrar la solución óptima , se recurrió a las Metaheurísticas. Estos son procedimientos de búsqueda que no garantizan la obtención del óptimo del problema considerado, se basan en la aplicación de reglas relativamente sencillas, tratan de huir de óptimos locales orientando la búsqueda en cada momento dependiendo de la evolución del proceso de búsqueda. La lógica de estas técnicas es similar, tiene un punto de partida que, por lo general, es una solución no óptima, y a partir de ella, realizar recolocaciones de acuerdo a cada Metaheuristica utilizada, respetando los parámetros de entrada y criterio de parada, para finalmente obtener como resultado la bahía ordenada.

En el presente informe, se detallarán dos Metaheurísticas. En primer lugar, se abordará la técnica Cat Swarm Optimization, el cual, modela dos estados característicos del gato, la primera, denominada Seeking Mode, en donde, el gato está en reposo, pero sin dejar de estar alerta a su entorno. El otro comportamiento que modela esta Metaheurística, se llama Tracing Mode, aquí el gato se mueve para asechar a sus enemigos o posibles presas.

La segunda MH a detallar en el presente informe se conoce como Harmony Search, el cual es un algoritmo inspirado en la manera en que los músicos buscan la armonía “perfecta” u optima, mediante la improvisación musical. En las prácticas de grupos musicales, cada músico toca una nota aleatoriamente de un rango posible de opciones, creando todos en conjuntos una armonía. Si la armonía entonada es buena, la experiencia es recordada en la memoria de cada miembro, por lo que la posibilidad de tocar una mejor armonía se incrementa para la próxima vez.

Ambos algoritmos tuvieron resultados similares, logrando llegar al óptimo en varias configuraciones simples. No obstante, en instancias complejas, ambas Metaheurísticas no lograron llegar a una solución, por lo que, se debió abordar de manera más especifica la calidad de los movimientos obtenidos y como mejorar los resultados en las instancias finales de cada Metaheurística.

1.1 Objetivos generales

Resolver el PMP utilizando dos Metaheurísticas, Cat Swarm Optimization y Harmony Search.

1.2 Objetivos específicos

Los objetivos propuestos para la realización del presente proyecto, son los siguientes:

1. Comprender el Pre-Marshalling Problem.
2. Comprender el Cat Swarm Optimization y Harmony Search.
3. Implementar y adaptar las Metaheurísticas al problema.
4. Realizar experimentos para lograr optimizar ambos algoritmos.

2 Estado del arte

El Pre-Marshalling Problem no es un conflicto que carezca de solución, debido a que, se han realizados diferentes modelos matemáticos y Heurísticas tradicionales, obteniendo resultados positivos, aunque en ciertos casos alejados del óptimo, las cuales se mencionan a continuación.

Entre alguno de los primeros acercamientos a la resolución de este problema encontramos a Robert Stahlbock y Stefan Voβ, los cuales desarrollaron una heurística que resuelve el problema mediante una heurística de cuatro etapas, en donde para cada una de estas fases, se pueden aplicar varias heurísticas [1]. Por otro lado, Caserta [2], en sus investigaciones, propone otro enfoque para resolver el PMP. Utilizando un método exacto sobre una parte restringida del espacio de solución del problema, con el fin de minimizar el espacio de búsqueda.

Expósito – Izquierdo presenta la primera heurística de prioridad baja o Lowest Priority First Heuristic [3], la cual describe que, si los contenedores con las prioridades más bajas se colocan en la parte inferior de las pilas, entonces no tendrán que ser reubicados en el futuro, ya que no constituyen casillas mal ubicadas. Un contenedor mal ubicado se encuentra por encima de uno o más contenedores con mayor prioridad o por encima de cualquier otro contenedor mal ubicado. Los resultados de esta heurística fueron comparados con Caseta y Voβ, fue visto como una heurística eficiente, en comparación a las otras dos.

Y por otro lado, Bortfeld y Forster propusieron la resolución a esta problemática usando un árbol de búsqueda [4]. Este árbol, se basa en un ordenamiento natural de recolocaciones posibles y aplica un esquema de ramificación adecuado, empleando secuencias de movimiento individuales prometedoras. Los resultados obtenidos por este algoritmo, que fueron comparados con las configuraciones anteriormente nombrados, obtuvieron soluciones con 0,1% menos reubicaciones.

Debido a su complejidad, no se ha logrado alcanzar el óptimo mediante heurísticas, es por eso que se está llevando a cabo la investigación para implementar Metaheurísticas que puedan resolver este problema y obtener mejores resultados. Actualmente, se está trabajando en diferentes Metaheurísticas para atacar al Pre-Marshalling Problem, entre ellos, está la metaheurística Firefly [5], la cual se basa en 3 comportamientos de las luciérnagas. Una luciérnaga, al ser unisexual puede ser atraída por cualquier otra luciérnaga, el insecto menos brillante será atraído por uno más brillante, pero la intensidad del brillo disminuye a medida que aumenta la distancia mutua. Por último, si no hay luciérnagas más brillantes, se moverá al azar. El "Bat Algorithm" [6] se basó en las características de eco-localización o sonar biológico de los murciélagos, de esta forma, pueden conocer y elegir el camino a tomar.

Otra Metaheurística que se está desarrollando para resover el PMP, es el Artificial Bee Colony [7], como su nombre lo indica, se basa en el comportamiento de una colonia de abejas, el cual, consta de tres tipos de abeja, la empleadas, las exploradoras y las abejas observadoras. Todas ellas trabajan en conjunto para encontrar la comida.

3 Pre-marshalling Problem

3.1 Definición del problema

El PMP es un conflicto enfocado en la reubicación de contendores en el patio de contenedores y que puede causar innumerables inconvenientes para el funcionamiento general del terminal si no es tratada de una manera adecuada. El proceso de almacenaje comienza con la llegada de los contenedores al puerto, ya sea por medio marítimo o terrestre, donde son derivados al patio terminal de contenedores para ser apilado junto a los demás. En este lugar, los contenedores permanecen durante un tiempo variante antes de ser trasportados a la litera del barco de trasporte que le corresponde, dejando el puerto. El patio terminal, se divide en diferentes bloques, que a su vez se componen de varias bahías, todas las bahías del mismo bloque tienen la misma cantidad de pilas, y en cada pila, existe la misma cantidad de ranuras para almacenar los contenedores.

Figura 3-1 Puerto de Valparaíso

Una grúa se encarga de apilar y retirar los contenedores, moviéndolos unitariamente por toda la bahía y solamente teniendo acceso directo al primero de cada pila, para luego obtener el contenedor deseado, el cual tiene una prioridad, que indica el orden de salida. El problema de este sistema, deriva en que, la configuración de llegada de los contenedores no es el mismo en que éstos se retiran del terminal, provocando que la grúa realice varios movimientos de reubicación hasta lograr su objetivo, retrasando la salida de los buques portacontenedores o medios terrestres de transporte, creando un efecto negativo en la eficiencia del terminal. Es en este punto, en donde se requiere una forma de optimizar el trabajo realizado, obteniendo una

configuración ordenada de la bahía, en el cual, no existan contenedores de mayor prioridad sobre uno menor y realizando el menor número de movimientos.

3.2 Representación del problema

El Pre-Marshalling Problem utiliza 3 supuestos:

1. El problema se limita a una sola bahía contenedor.
2. Todos los contenedores en la bahía tienen las mismas dimensiones.
3. Las prioridades de contenedores se conocen de antemano. Cada contenedor tiene una prioridad bien definido que represente su orden de recuperación, de tal forma que, cuanto menor es el índice mayor es la prioridad.

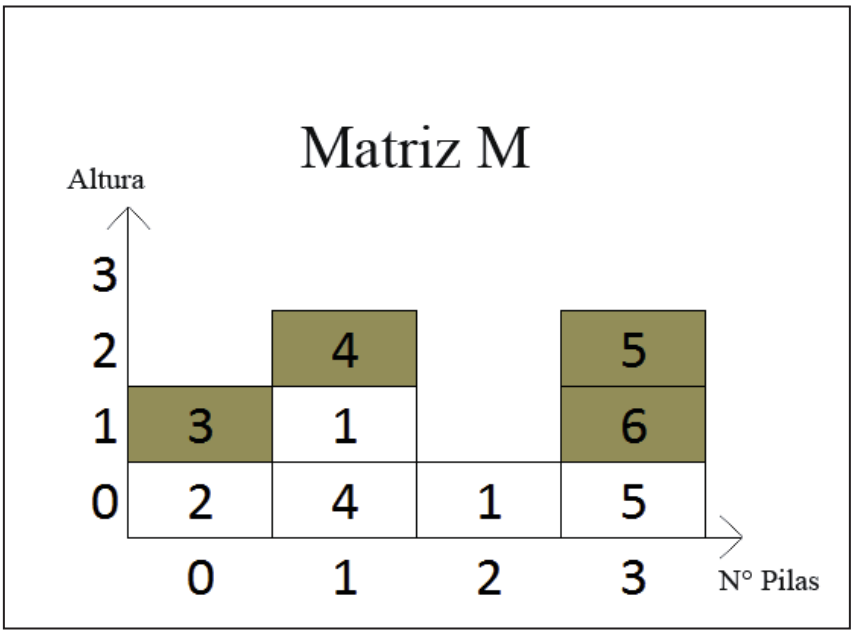

Figura 3-2 Representación de la matriz M

La matriz M cuenta con *P* pilas, de una altura *H, C* contenedores y *K* prioridad; los contenedores destacados están mal ubicados.

3.3 Representación de la solución:

El formato de la solución al problema se establece en una secuencia de *N* variables de la forma (a,b), donde el primer término es la pila de origen, seleccionando el contenedor que este en la cima y el segundo término es la pila de destino.

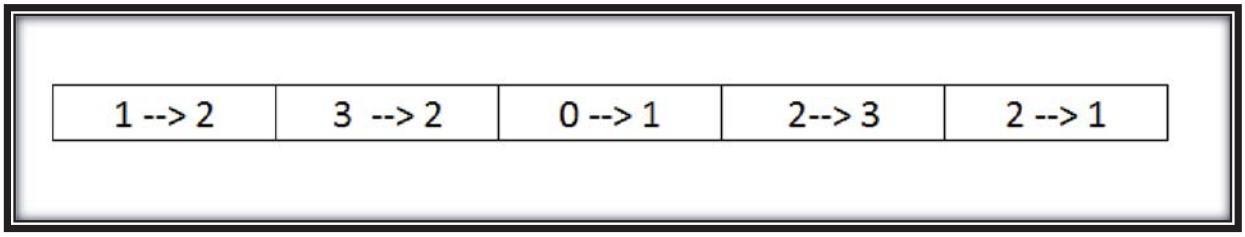

Figura 3-3 Vector solución

4 Cat Swarm Optimization

Pata resolver este problema, se ha establecido la utilización de la Metaheurística llamada “Cat Swarm Optimization”, creado por Shu-Chuan Chu y Pei-Wei Tsai [8], el cual está basado en el comportamiento de los gatos. Estos animales, tiene dos conductas bien definidas, la primera es la habilidad de la caza, presentes en todos los felinos, aunque en el caso del gato doméstico, esta habilidad se redujo, al igual que la considerable curiosidad por los objetos en movimientos. A pesar de ello, el gato pasa la mayor parte del tiempo en un estado de reposo, su segunda conducta, pero pese a ello, estos animales tienen un elevado nivel de alerta, incluso estando descansando. Del mismo modo que los gatos, CSO tiene estos dos estados, definidos como Tracing Mode y Seeking mode respectivamente.

4.1 El gato

Lo primero que se decide, es la cantidad de gatos a utilizar; cada gato tiene:

- La posición del gato, representado por el vector de parámetros de tipo [a , b].
- Las velocidades de cada dimensión con sus respectivos límites; una dimensión del gato es una variable de su vector de movimiento.
- Una variable de ajuste (fitness function), que indique la cercanía del resultado óptimo.
- Un Flag de estado, que definirá si está en modo “Tracing” o “Seeking”.

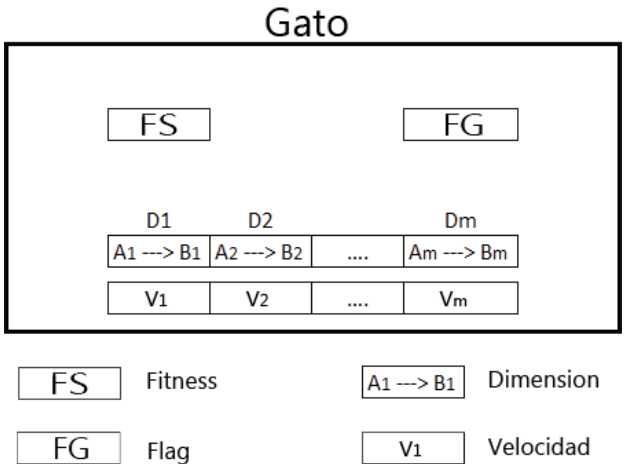

Figura 4-1 Componentes del gato

4.2 Tracing Mode.

Este estado modela el caso del gato cuando está localizando y acechando al objetivo, el animal se mueve de acuerdo a la velocidad de cada dimensión. Es posible definir este proceso en 3 pasos.

1. Se modifican las velocidades de cada dimensión *(Vk,d)* según la siguiente función

$$Vk,d = Vk,d + R * C \cdot (Xbest,d - Xk,d), d = 1,2,\ldots,M \qquad (4.2.1)$$

Xbest es la posición del mejor gato, el cual, tiene el mejor valor de fitness; mientras que *Xk,d* es la posición del *gatoK*, C es una constante y *R* es un valor aleatorio entre *1* y *0*.

2. Se comprueban que las velocidades no sobrepasen el límite establecido, de ocurrir, se corrige, ubicándolo en el límite superior.

3. Actualizar la posición de gato, mediante la siguiente función.

$$Xk,d = Xk,d + Vk,d, \quad d = 1,2,\ldots,M \qquad (4.2.2)$$

4.3 Seeking Mode

Modela el comportamiento del gato mientras descansa, pero atento y mirando a su alrededor, en la búsqueda de su siguiente posición a mover, al igual que en la vida real, la Metaheurística cae con mayor frecuencia este estado. Este sub-modelo utiliza 4 factores esenciales:

- "Seeking Memory Pool" (SMP), la cual se utiliza para definir el tamaño de la búsqueda de cada gato.
- El rango de búsqueda de la dimensión seleccionada (SRD), declara la relación mutativa para las dimensiones seleccionadas.
- La cantidad de dimensiones a cambiar (CDC).
- La consideración de la propia posición (SPC), es un booleano que indica si la posición actual del gato es un punto candidato.

El Seeking Mode se describe en 5 pasos que realizará cada gato en este comportamiento.

1. Realizar *J* copias de la posición del gato, donde *J = SMP*. Si el valor de SPC es verdadero, dejar *J = SMP - 1*, significando que es un gato candidato.
2. Por cada copia, de acuerdo al CDC, aleatoriamente aumentar o disminuir el porcentaje del SRD.
3. Calcular el valor fitness de todos los puntos candidatos.

4. Si todos los valores fitness no son exactamente iguales, calcular la probabilidad de selección de cada punto candidato en la ecuación (5.3.1). En caso contrario, establecer este valor como 1.
5. Mover la posición del gato hacia un punto candidato aleatoriamente.

$$Pi = \frac{|FS_i - FS_b|}{FS_{max} - FS_{min}}, Donde \ 0 < i < j \qquad (4.3.1)$$

Si la finalidad de la función fitness es encontrar la mínima solución, *FSb = FSmax,* en caso contrario, *FSb = FSmin.*

4.4 El algoritmo

Para combinar los dos métodos en el algoritmo y teniendo en cuenta que los gatos pasan la mayor parte del tiempo descansando, se tiene una funcionalidad llamada relación de mezcla (MR), que define cuál de los dos comportamientos realizará el gato en cada iteración. Con esto, cada gato tiene un 90% de probabilidad de realizar el modo de Seeking. La Metaheurística se desarrolla en 6 pasos:

Se crean los *N* gatos.

1. Aleatoriamente, se incorporan en el espacio de solución e inician los valores del gato, se designa la velocidad, validando el rango máximo de esta. Después, de acuerdo al MR, se establece el estado de cada gato.
2. Evaluar los valores fitness de cada gato y mantener la posición del mejor gato.
3. Mover cada gato, de acuerdo a su variable de estado, derivarlos a Seeking o Tracing, realizando las sentencias respectivas.
4. Se vuelven a escoger los gatos, cambiando sus variables de estados aleatoriamente y otra vez, derivar a los gatos a sus correspondientes estados.
5. Verificar la condición de término, el cual es obtener un gato con fitness 0, lo que significa, que aquel gato encontró una secuencia de movimientos que ordene la bahía; en caso contrario, se debe volver al paso 3.
6. Cuando el algoritmo alcance las 5.000 iteraciones, se aumentará la cantidad de dimensiones.

Si el algoritmo llega a aumentar el vector de solución en un 150% del límite inferior inicial y no fue capaz de encontrar solución, se procederá a la fase final del algoritmo, en el cual, si el mejor gato fue actualizado más de una vez, se considerará al mejor gato actual como solución. En caso contrario, la ejecución no tendrá solución

4.5 Solución inicial inteligente

Con la finalidad de mejorar el rendimiento del Cat Swarm Optimization, se ha establecido una modificación en su configuración inicial, el cual se refiere a cambiar su inicialización aleatoria por una configuración inicial inteligente. Este nuevo procedimiento, optimizará el rendimiento al realizar una evaluación por casos en cada recolocación, llenando de una forma más eficiente cada dimensión del vector de movimiento. Existen 3 casos, los cuales, serán explicados a continuación; seguido de un ejemplo para cada uno.

4.5.1 Primer caso

En este primer caso, se busca la pila con el contenedor mayor mal ubicado en la cima, luego se evalúa la posibilidad de recolocarlo en alguna pila, en la cual, quede bien ubicado.

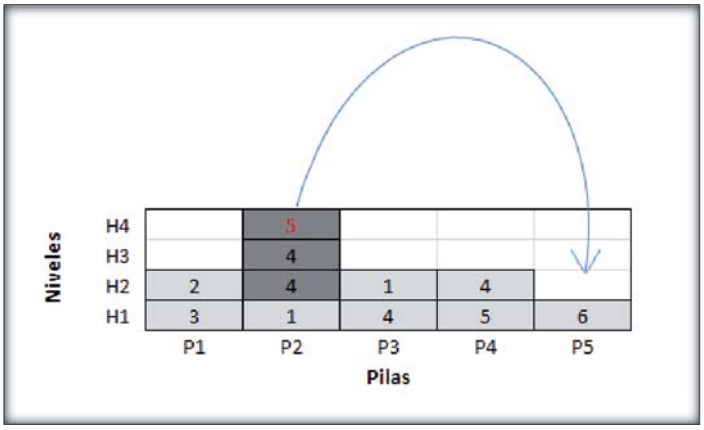

Figura 4-2 Bahía de ejemplo para el caso 1

En este ejemplo, el contenedor mal ubicado, de prioridad 5, es el mayor de los mal ubicados y tiene la opción de moverse a la pila P5 y cambiar su estado a “bien ubicado”.

4.5.2 Segundo caso

Si el primer caso no es posible, es decir, si el mayor mal ubicado no puede ser recolocado de manera correcta, se busca la posibilidad de mover el segundo contenedor mal ubicado, el tercero y así consecutivamente, hasta lograr que un mal ubicado sea recolocado. De no ser posible, se acude al tercer caso.

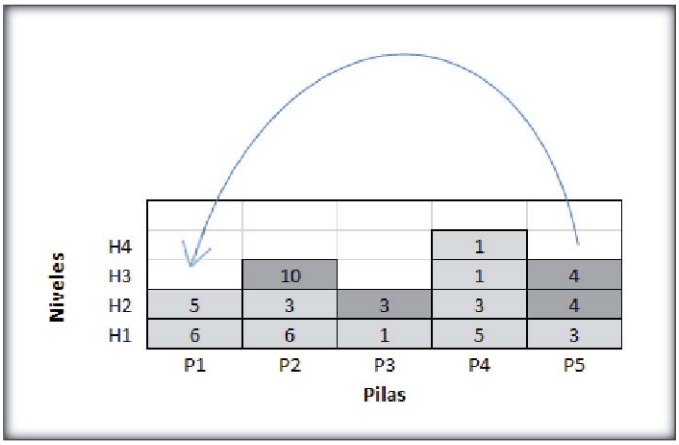

Figura 4-3 Bahía de ejemplo para el caso 2

En este ejemplo, el contenedor de prioridad 10, es el mayor mal ubicado, pero no existe alguna pila en donde quede bien ubicado. Se sigue buscando otro contenedor y se encuentra el de la pila P5, el cual, puede ser recolocado a la pila P1.

4.5.3 Tercer caso

Este caso, a diferencia de los dos anteriores, involucra más de un movimiento de contenedores. Se elige la pila con menor número de contenedores y que no contenga al contenedor con menor prioridad en la cima (mayor mal ubicado). Uno a uno, los contenedores son recolocados en las otras pilas, tratando de no aumentar el número de mal ubicados y de no impedir el movimiento del contenedor mayor mal ubicado.

Este caso se repetirá hasta vaciar por completo la pila o se cumpla los requerimientos para el primer caso, es decir, que en la pila que se encuentra vaciando, el contenedor de la cima tenga una prioridad menor que la del peor mal ubicado.

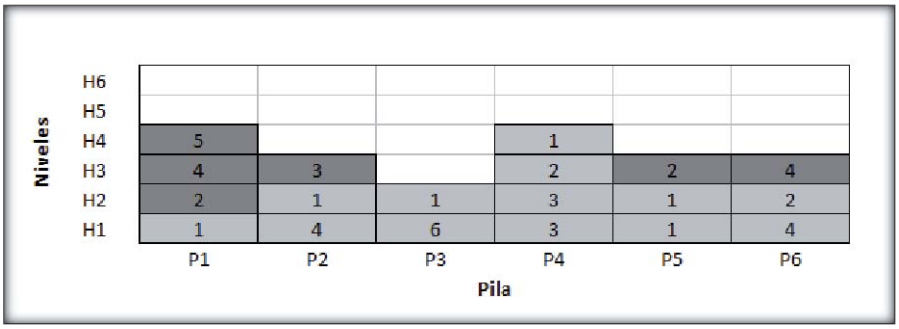

Figura 4-4 Bahía de ejemplo para el caso 3

En este ejemplo, se busca la pila menor, resultando ser la pila P3, la cual, será vaciada por completo o hasta que la prioridad del contenedor que está en la cima de la pila P3 sea

menor a la del contenedor mayor mal ubicado. La primera recolocación a efectuarse será de la pila P3 a la P4 (3 --> 4), debido a que priorizará una buena recolocación, en la medida de lo posible.

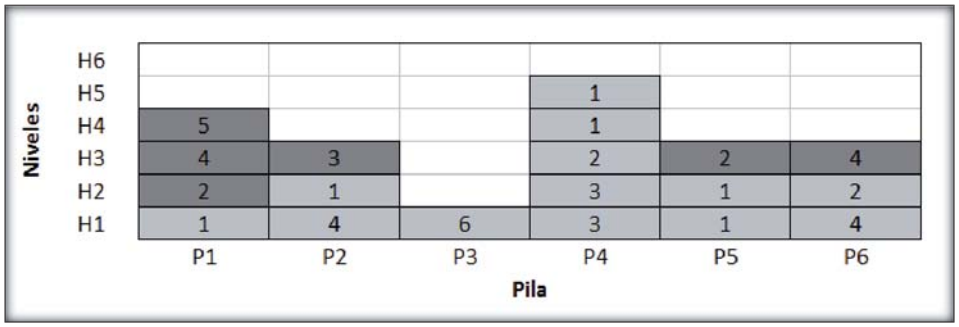

Figura 4-5 Bahía del caso 3 modificada

Después, se valida que el nuevo contenedor en la cima de la pila a vaciar tenga una prioridad mayor a la del mal ubicado. En este caso, esto se cumple. El mayor mal ubicado se encuentra en la pila P1y posee una prioridad de 5, mientras que la pila P3 tiene un 6. El caso termina y se prosigue a realizar el caso 1.

5 Harmony Search

El Harmony Search (HS) o Búsqueda Armónica [9,10] es un algoritmo Metaheurístico basado en la improvisación musical, la cual ocurre cuando un músico busca la armonía óptima. De manera que en el HS la solución es similar a una armonía, mientras que los otros operadores simulan el proceso de improvisación musical. En comparación con otros algoritmos, la BA representa muchas ventajas por utilizar pocos parámetros, además de su rapidez a la hora de entregar una solución.

Contrastándolo con el problema, en el HS el conjunto de músicos representa las variables de decisión, el rango de afinación es equivalente al rango de valores que puede tomar nuestras variables, la armonía representa el vector solución, la práctica de los músicos es igual a la cantidad de iteraciones que se realicen y la experiencia está representada por la memoria armónica.

Los procedimientos que sigue esta heurística son:

1. Inicializar parámetros del algoritmo
2. Inicializar la memoria armónica.
3. Improvisación musical.
4. Actualización memoria armónica.
5. Repetir pasos 3 y 4 hasta que se cumpla el criterio de parada.

5.1 Inicializar parámetros del algoritmo

El algoritmo comienza con la inicialización de parámetros, los cuales son: la tasa de exploración (HMCR), la cual representa la probabilidad de construir una nueva solución a partir de los datos almacenados en la memoria armónica. La razón de ajuste de tono (PAR), que representa la probabilidad de modificar una solución previamente generada. Por último, el ancho de desplazamiento (BW), que representa la magnitud de la modificación mencionada anteriormente.

5.2 Inicialización de la memoria armónica

Antes de inicializar la memoria armónica (HM), se debe establecer el tamaño de la memoria (HMS). Luego se inicializa la HM, en donde cada casilla representará un vector *Xi*, el cual contendrá la solución número *i*, la cual será generada usando la fórmula:

$$X_i = \mathrm{L}(\mathrm{j}) + [\mathrm{U}(\mathrm{j}) - \mathrm{L}(\mathrm{j})] * \mathrm{Rand}(0,1), \mathrm{Con\ i} = 1,2 \dots, \mathrm{HMS}\ ; \mathrm{j} = 1,2, \dots, \mathrm{n} \quad (5.2.1)$$

Donde *Rand (0,1)* es un número generado aleatoriamente del rango de valores existentes entre 0 y 1. La memoria armónica (HM) almacenara los HMS vectores generados inicialmente quedando de la siguiente forma:

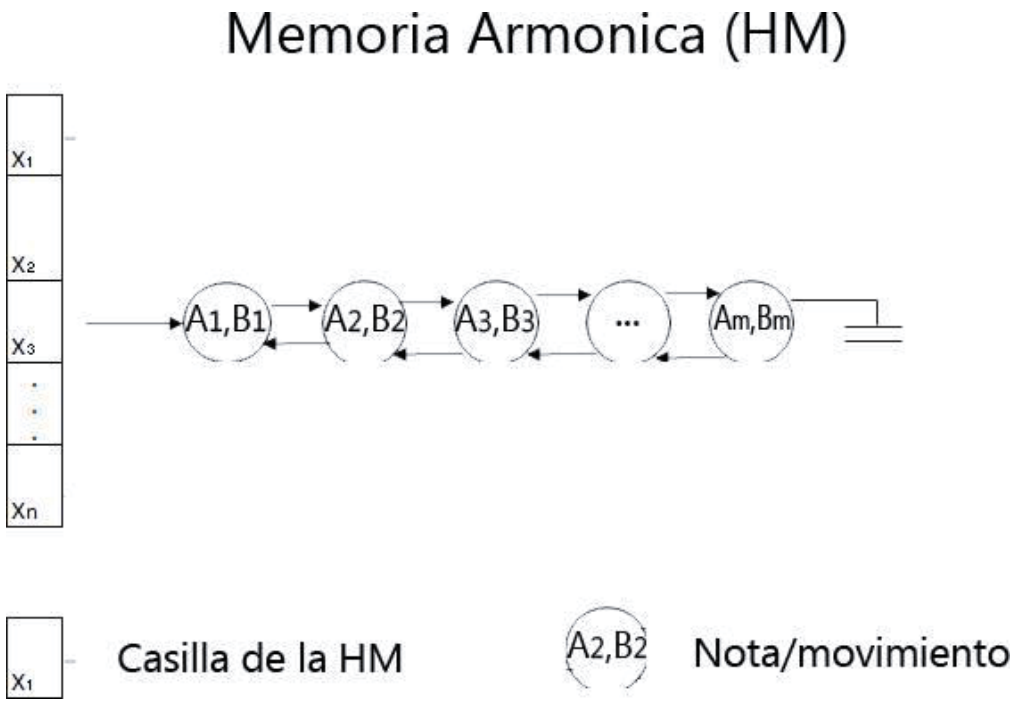

Figura 5-1 Memoria Armónica, donde cada casilla Xn contiene una lista de movimientos de la forma A,B.

5.3 Improvisación musical

El proceso de crear una nueva armonía se llama improvisación. El nuevo vector de soluciones se genera considerando 3 factores: examinado de memoria, re-inicialización aleatoria y ajuste de tono. En la etapa del examinado de memoria, el primer movimiento en el vector solución X_1 (Fig. 5-1), es elegido de cualquiera de los primeros movimientos ya existentes en la HM. Para esta inicialización se crea un numero aleatorio r (Ecu. 5.3.1) entre los rangos [0,1]. Si r es menor al HMCR, el movimiento se inicializa con el método anteriormente explicado, de no ser así, el movimiento se obtiene a partir de un valor aleatorio reinicializado con la misma fórmula que se inicializa la HM. Los valores de las otras variables $X_2, X_3, \ldots X_n$. (Fig. 5-1) se seleccionan de la misma manera.

De esta forma, el examinado de memoria y la re-inicialización aleatoria puede modelarse de la siguiente manera:

If $(r < HMCR)$

$$X_s\ (\mathrm{j}) = \mathrm{Xa}\ (\mathrm{j}) \tag{5.3.1}$$

Else

$$X_s\ (j) = L(j) + [U(j) - L(j)] * r\ , Con\ r\ \in Rand\ (0{,}1)$$

Para cada $X_{s(j)}$,se le aplica un ajuste de tono, el cual está definido por la razón de ajuste de tono PAR y el ancho de desplazamiento BW. El parámetro PAR es una variable probabilística que determina la posibilidad de ajustar o no el movimiento, sumando valores vecinos, probabilidad la cual se determina al iniciar los parámetros de la Metaheuristica. Se compara una variable $u\ \in Rand\ (0{,}1)$ y se evalúa si su valor cumple con la probabilidad anteriormente mencionada. Si se debe ajustar el tono (probabilidad PAR), se procederá a ajustar el movimiento actual, de no ser así (probabilidad 1-PAR), no se modifica y se evalúa la probabilidad de ajustar el próximo movimiento. El parámetro BW $\in Rand\ (-1{,}1)$ determina si se debe sumar el vecino izquierdo o el vecino derecho.

El ajuste de tono se realiza de la siguiente manera:

Con probabilidad PAR

$$X_s(j) = X_s(j) + Rand\ (0{,}1) * BW \tag{5.3.2}$$

Con probabilidad (1-PAR)

$$X_s(j) = X_s(j)$$

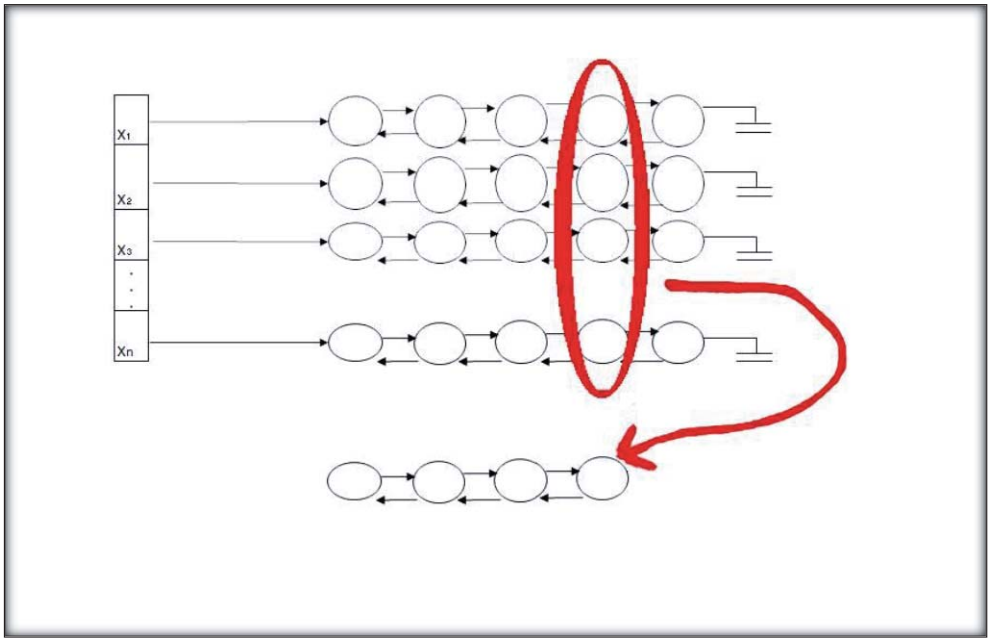

Figura 5-2 Improvisación musical.

5.4 Actualización de la memoria armónica

Luego de que el vector de improvisación es generado, la memoria HM se actualiza mediante la comparación entre el peor vector de la memoria y la improvisación. Si la improvisación tiene una menor cantidad de movimientos que el peor vector de la memoria, se reemplaza la solución en la memoria, de lo contrario, el contenido de la HM permanece sin cambios. Como la HM solo almacenara las mejores improvisaciones que se vayan generando, a medida que avancen las iteraciones, el tamaño de los vectores de solución en la memoria irán disminuyendo, por lo que al final convergerán a un solo valor.

5.5 Criterio de parada

Se repiten los pasos mencionados en 5.3 y 5.4 hasta que algún vector de improvisación obtenga un fitness igual a 0, encontrando la solución óptima. Cuando la Metaheurística alcance un múltiplo de 5.000 iteraciones, se aumentará la cantidad de dimensiones.

6 Análisis de resultados

En esta sección, se presentarán los resultados analizados por ambas Metaheurísticas. En un primer lugar, se describe el entorno de desarrollo de las pruebas en términos de casos, luego se muestran los resultados del prototipo, y por último, identificar los problemas ocurridos al solucionar el PMP con este algoritmo.

6.1 Entorno de Desarrollo

Se ha realizado el análisis de 4 casos de prueba, CV1, CV2, BF1 y BF2, los dos primeros casos, constan de 10 instancias, mientras que los dos restantes contienen 20. Se ejecutaron las Metaheurísticas 10 veces por instancia.

Tabla 6-1 Casos de prueba analizados

Caso de Prueba	No. de Instancias	No. de Pilas	Altura de la Pila	No. de contenedores	No. de índices de prioridad	No. de contenedores mal ubicados
CV1	10	3	9	9	9	5,40
CV2	10	4	16	16	16	10,60
BF1	20	16	5	48	10	29,00
BF2	20	16	5	48	10	36,00

En el caso de Cat Swarm Optimization, se ocuparon 15 gatos para estos casos de prueba, por otro lado, en Harmony Search se utilizaron 10 músicos. En ambos, se aumentó el largo del vector de solución en una unidad por cada cinco mil iteraciones, hasta un máximo de 150% del límite inferior.

Tabla 6-2 Tabla resultado de los casos de prueba

Caso de Prueba	Promedio del Límite Inferior	Número de movimientos			
		Autor	Óptimo Conocido	CSO	HS
CV1	7,9	21,3	10,1	12,57	10,28
CV2	13,6	28,27	19,1	65,25	25,7
BF1	29,05	29,1	29,05	-	-
BF2	36	36	36	-	-

En ambas Metaheurísticas se logró encontrar solución en los casos CV1 y CV2, no obstante en los dos restantes casos de pruebas, no fue posible.

7 Conclusión

El PMP es una problemática de optimización que cuenta con distintos métodos ya planteados para su resolución, y que han resuelto el problema, entregando resultados considerables, promoviendo el interés de continuar con el trabajo, pese a no obtener resultados óptimos. A medida que se continuó con la investigación y experimentación de adaptar ambas Metaheurísticas al Pre-Marshalling, se fueron encontrando diferentes dificultades, tanto en Cat Swarm Optimization, como en Harmony Search. Pero pese a ello, se ha logrado comprender el proceso de los algoritmos, logrando crear dos programas aceptables. A diferencia del CSO, el Harmony Search no tuvo mayor problema al momento de adaptarlo al PMP, ya que requiere una menor cantidad de parámetros para su ejecución, por lo que, su tiempo de respuesta es más eficiente y su cercanía al óptimo es mayor, con respecto a las soluciones entregadas por la otra Metaheurística.

Gracias a la Solución inicial inteligente, se ha logrado crear una inicialización más eficiente para Cat Swarm Optimization, pero pese a ello, no ha logrado el mejoramiento esperado. Otro cambio significativo en la implementación de este algoritmo, fue la utilización del mejor gato al final cada ejecución, en el caso de no haber encontrado solución. No se ha podido comprobar el rendimiento completo de esta función, debido a los escasos casos de pruebas que se pudo realizar para esta entrega, pero se tiene en mente continuar con esta investigación.

Por otra parte, en HS, se ha regulado el ajuste de tono, con lo cual, se busca refinar los valores obtenidos anteriormente, en otras palabras, encontrar valores cercanos a los mejores ya obtenidos. Esto da pie a que las mejores soluciones locales sean guardadas, mientras la aleatoriedad se encarga de explorar, de forma eficiente, el espacio de búsqueda.

Luego de haber implementado las Metaheurísticas, se ha logrado avanzar en la optimización de ambos. Entre los 2 algoritmos, HS ha demostrado ser más eficiente que CSO, debido a que en los 2 primeros casos de pruebas, logró obtener un resultado más cercano al óptimo. No obstante, los tiempos de respuesta en ambos, sigue siendo un conflicto a la hora de realizar un considerable número de casos de pruebas. Aún queda mucho por optimizar, por lo que, ambos algoritmos pueden ser mejorados, a medida que se analicen todos los distintos tipos de escenarios que se presentan en determinadas configuraciones.

8 Referencias

[1] R. Jovanovic, M. Tuba and S. Voß, "A multi-heuristic approach for solving the pre-marshalling problem", ***Cent Eur J Oper Res***, vol. 10, pp. 1-28, 2015.

[2] M. Caserta and S. Voß, "A Corridor Method-Based Algorithm for the Pre-marshalling Problem", ***Applications of Evolutionary Computing***, vol. 5484, pp. 788-797, 2009.

[3] C. Expósito, B. Melian and M. Moreno, "Una heurística eficaz para problemas de pre-marshalling en una terminal marítima de contenedores", ***VIII Congreso español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB-2012),*** 2012.

[4] A. Bortfeldt and F. Forster, "A tree search procedure for the container pre-marshalling problem", ***European Journal of Operational Research***, vol. 217, no. 3, pp. 531-540, 2012.

[5] M. Sayadi, R. Ramezanian and N. Ghaffari-Nasab, "A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems", ***IJIEC***, vol. 1, no. 1, pp. 1-10, 2010.

[6] X. Yang and X. He, "Bat algorithm: literature review and applications", ***International Journal of Innovative Computing, Information and Control (IJBIC)***, vol. 5, no. 3, p. 141, 2013.

[7] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", ***Journal of Global Optimization***, vol. 39, no. 3, pp. 459-471, 2007.

[8] S. Chu and P. Tsai, "Computational Intelligence Based on the Behavior of Cats". ***International Journal of Innovative Computing, Information and Control (IJICIC)***, vol. 3, no. 1, pp. 163-173, 2007.

[9] C. COBOS, J. PÉREZ and D. ESTUPIÑAN. Una revisión de la búsqueda armónica. ***Avances en Sistemas e Informática***, vol. 8, no 2, pp. 67-80, 2011

[10] K. Lee and Z. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice", ***Computer Methods in Applied Mechanics and Engineering***, vol. 194, no. 36-38, pp. 3902-3933, 2005.

Anexo

A: Desglose de resultados de los casos realizados por Cat Swarm Optimization

Tabla A-1 Resultados del caso CV1 con CSO

Nombre instancia	Límite inferior	Numero de recolocaciones										Promedio de instancia	Promedio Caso de prueba
		Ejec. 1	Ejec. 2	Ejec. 3	Ejec. 4	Ejec. 5	Ejec. 6	Ejec. 7	Ejec. 8	Ejec. 9	Ejec. 10		
CV1_1	9	15	14	15	14	12	11	15	11	15	16	13,8	12,57
CV1_2	7	8	9	8	8	8	9	8	9	8	9	8,4	
CV1_3	9	13	14	12	14	14	11	12	13	12	14	12,9	
CV1_4	9	13	12	12	17	13	15	17	13	17	18	14,7	
CV1_5	9	19	19	14	20	16	15	19	17	18	14	28,5	
CV1_6	7	8	9	9	9	9	8	8	8	8	9	8,5	
CV1_7	6	8	8	8	8	8	8	8	8	8	8	8	
CV1_8	9	14	12	11	14	13	13	14	14	14	13	13,2	
CV1_9	7	9	10	9	9	9	10	9	9	11	9	9,4	
CV1_10	7	8	9	8	8	9	8	8	8	8	9	8,3	

Tabla A-2 Resultados del caso CV2 con CSO

Nombre instancia	Límite inferior	Numero de recolocaciones										Promedio de instancia	Promedio Caso de prueba
		Ejec. 1	Ejec. 2	Ejec. 3	Ejec. 4	Ejec. 5	Ejec. 6	Ejec. 7	Ejec. 8	Ejec. 9	Ejec. 10		
CV2_1	14	108	64	81	93	90	108	87	42	93	94	86	65,25
CV2_2	13	60	52	47	44	47	82	60	60	52	83	58,7	
CV2_3	7	10	12	9	11	10	10	10	10	11	10	10,3	
CV2_4	14	104	81	84	126	107	121	94	98	94	95	100,4	
CV2_5	14	50	81	77	72	88	50	79	87	80	94	75,8	
CV2_6	13	76	78	78	84	45	64	107	100	46	70	74,8	
CV2_7	16	80	47	59	55	67	72	66	69	60	71	64,6	
CV2_8	13	55	45	58	39	51	57	51	56	58	42	51,2	
CV2_9	16	61	68	48	54	56	67	40	63	49	48	55,4	
CV2_10	16	74	88	88	84	74	73	72	59	64	77	75,3	

Tabla A-3 Resultados del caso BF1 con CSO

Nombre instancia	Límite inferior	Numero de recolocaciones										Promedio de instancia	Promedio Caso de prueba
		Ejec. 1	Ejec. 2	Ejec. 3	Ejec. 4	Ejec. 5	Ejec. 6	Ejec. 7	Ejec. 8	Ejec. 9	Ejec. 10		
BF1_1	19	19	19	19	19	19	19	19	19	19	19	19	
BF1_2	29	x	x	x	x	x	x	x	x	x	x	x	
BF1_3	30	x	x	x	x	x	x	x	x	x	x	x	
BF1_4	31	31	31	31	31	31	31	31	31	31	31	31	
BF1_5	30	30	30	30	30	30	30	30	30	30	30	30	
BF1_6	31	31	31	31	31	31	31	31	31	31	31	31	
BF1_7	29	x	x	x	x	x	x	x	x	x	x	x	
BF1_8	29	29	29	29	29	29	29	29	29	29	29	29	
BF1_9	30	x	x	x	x	x	x	x	x	X	X	x	
BF1_10	31	31	31	31	31	31	31	31	31	31	31	31	X
BF1_11	31	x	x	x	x	x	x	x	x	x	x	x	
BF1_12	30	x	x	x	x	x	x	x	x	x	x	x	
BF1_13	30	30	30	30	30	30	30	30	30	30	30	30	
BF1_14	32	32	32	32	32	32	32	32	32	32	32	32	
BF1_15	32	32	32	32	32	32	32	32	32	32	32	32	
BF1_16	29	29	29	29	29	29	29	29	29	29	29	29	
BF1_17	29	x	x	x	x	x	x	x	x	x	x	x	
BF1_18	29	x	x	x	x	x	x	x	x	x	x	x	
BF1_19	30	x	x	x	x	x	x	x	x	x	x	x	
BF1_20	30	x	x	x	x	x	x	x	x	x	x	x	

Tabla A-4 Resultados del caso BF2 con CSO

Nombre instancia	Límite inferior	Numero de recolocaciones										Promedio de instancia	Promedio Caso de prueba
		Ejec. 1	Ejec. 2	Ejec. 3	Ejec. 4	Ejec. 5	Ejec. 6	Ejec. 7	Ejec. 8	Ejec. 9	Ejec. 10		
BF2_1	38	38	38	38	38	38	38	38	38	38	38	38	X
BF2_2	38	38	38	38	38	38	38	38	38	38	38	x	
BF2_3	37	37	37	37	37	37	37	37	37	37	37	x	
BF2_4	39	39	39	39	39	39	39	39	39	39	39	39	
BF2_5	37	37	37	37	37	37	37	37	37	37	37	37	
BF2_6	37	x	x	x	x	x	x	x	x	x	x	0	
BF2_7	36	x	x	x	x	x	x	x	x	x	x	x	
BF2_8	36	36	36	36	36	36	36	36	36	36	36	36	
BF2_9	38	38	38	38	38	38	38	38	38	38	38	x	
BF2_10	38	38	38	38	38	38	38	38	38	38	38	38	
BF2_11	38	38	38	38	38	38	38	38	38	38	38	x	
BF2_12	39	39	39	39	39	39	39	39	39	39	39	x	
BF2_13	37	37	37	37	37	37	37	37	37	37	37	37	
BF2_14	39	39	39	39	39	39	39	39	39	39	39	39	
BF2_15	37	37	37	37	37	37	37	37	37	37	37	37	
BF2_16	38	38	38	38	38	38	38	38	38	38	38	38	
BF2_17	36	x	x	x	x	x	x	x	x	x	x	x	
BF2_18	36	36	36	36	36	36	36	36	36	36	36	x	
BF2_19	37	37	37	37	37	37	37	37	37	37	37	x	
BF2_20	38	38	38	38	38	38	38	38	38	38	38	x	

B: Desglose de resultados de los casos realizados por Harmony Search

Tabla B-1 Resultados del caso CV1 con HS

Nombre instancia	Límite inferior	Numero de recolocaciones										Promedio de instancia	Promedio Caso de prueba
		Ejec. 1	Ejec. 2	Ejec. 3	Ejec. 4	Ejec. 5	Ejec. 6	Ejec. 7	Ejec. 8	Ejec. 9	Ejec. 10		
CV1_1	9	11	13	10	9	12	11	9	11	10	9	10,5	10,27666667
CV1_2	7	8	9	7	8	7	7	8	9	7	9	7,9	
CV1_3	9	10	12	9	11	9	9	12	9	12	10	10,3	
CV1_4	9	11	9	12	9	13	15	9	10	13	9	11	
CV1_5	9	13	12	14	9	10	15	14	9	12	10	19,66666667	
CV1_6	7	8	9	7	9	9	8	8	7	8	9	8,2	
CV1_7	6	8	6	8	7	6	8	6	8	7	6	7	
CV1_8	9	14	12	11	9	13	13	10	10	14	13	11,9	
CV1_9	7	8	8	7	9	7	10	7	9	11	10	8,6	
CV1_10	7	7	9	8	7	9	8	7	8	7	7	7,7	

Tabla B-2 Resultados del caso CV2 con HS

Nombre instancia	Límite inferior	Numero de recolocaciones										Promedio de instancia	Promedio Caso de prueba
		Ejec. 1	Ejec. 2	Ejec. 3	Ejec. 4	Ejec. 5	Ejec. 6	Ejec. 7	Ejec. 8	Ejec. 9	Ejec. 10		
CV2_1	14	29	16	49	16	32	38	31	15	25	24	27,5	25,7
CV2_2	13	25	18	16	33	35	19	21	28	34	17	24,6	
CV2_3	7	10	12	7	11	10	7	10	10	11	10	9,8	
CV2_4	14	17	25	23	28	36	41	16	23	26	29	26,4	
CV2_5	14	17	19	33	21	26	30	18	30	18	25	23,7	
CV2_6	13	15	25	21	24	13	21	13	16	16	18	18,2	
CV2_7	16	17	23	34	25	18	21	36	17	25	71	28,7	
CV2_8	13	55	45	58	39	51	57	51	56	58	42	51,2	
CV2_9	16	21	23	37	24	23	17	19	20	18	18	22	
CV2_10	16	19	17	21	19	24	38	47	18	29	17	24,9	

Tabla B-3 Resultados del caso BF1 con HS

Nombre instancia	Límite inferior	Numero de recolocaciones										Promedio de instancia	Promedio Caso de prueba
		Ejec. 1	Ejec. 2	Ejec. 3	Ejec. 4	Ejec. 5	Ejec. 6	Ejec. 7	Ejec. 8	Ejec. 9	Ejec. 10		
BF1_1	19	19	19	19	19	19	19	19	19	19	19	19	
BF1_2	29	x	x	x	x	x	x	x	x	x	x	x	
BF1_3	30	x	x	x	x	x	x	x	x	x	x	x	
BF1_4	31	31	31	31	35	31	33	31	31	31	31	31,6	
BF1_5	30	30	32	30	30	32	30	31	30	33	31	30,9	
BF1_6	31	31	33	32	31	35	31	32	33	31	31	32	
BF1_7	29	x	x	x	x	x	x	x	x	x	x	x	
BF1_8	29	29	30	29	29	31	29	29	30	29	31	29,6	
BF1_9	30	x	x	x	x	x	x	31	x	X	X	x	
BF1_10	31	31	31	33	31	37	31	34	31	35	31	32,5	X
BF1_11	31	x	x	x	x	x	x	x	x	x	x	x	
BF1_12	30	x	x	x	x	x	x	x	x	x	x	x	
BF1_13	30	30	30	30	33	36	30	30	35	30	30	31,4	
BF1_14	32	32	32	32	32	32	32	32	32	32	32	32	
BF1_15	32	32	32	32	32	32	32	32	32	32	32	32	
BF1_16	29	29	29	29	29	29	29	29	29	29	29	29	
BF1_17	29	29	x	x	29	x	29	30	29	29	x	x	
BF1_18	29	x	x	x	x	x	x	x	x	x	x	x	
BF1_19	30	x	x	x	x	x	x	x	x	x	x	x	
BF1_20	30	30	37	35	30	31	30	32	36	31	31	x	

Tabla B-4 Resultados del caso BF2 con HS

Nombre instancia	Límite inferior	Numero de recolocaciones										Promedio de instancia	Promedio Caso de prueba
		Ejec. 1	Ejec. 2	Ejec. 3	Ejec. 4	Ejec. 5	Ejec. 6	Ejec. 7	Ejec. 8	Ejec. 9	Ejec. 10		
BF2_1	38	38	38	38	38	38	38	38	38	38	38	38	X
BF2_2	38	38	38	38	38	38	38	38	38	38	38	x	
BF2_3	37	37	37	37	37	37	37	37	37	37	37	x	
BF2_4	39	39	39	39	39	39	39	39	39	39	39	39	
BF2_5	37	x	x	x	x	x	x	x	x	x	x	0	
BF2_6	37	x	x	x	x	x	x	x	x	x	x	0	
BF2_7	36	x	x	x	x	x	x	x	x	x	x	x	
BF2_8	36	36	36	40	36	36	36	39	36	38	37	37	
BF2_9	38	38	38	38	38	39	38	38	40	39	38	x	
BF2_10	38	38	38	39	38	38	38	40	38	39	39	38,5	
BF2_11	38	x	x	x	x	x	38	x	x	x	x	x	
BF2_12	39	39	40	39	39	40	39	41	39	39	39	x	
BF2_13	37	37	37	37	37	37	37	37	37	37	37	37	
BF2_14	39	39	39	39	39	39	39	39	39	39	39	39	
BF2_15	37	x	x	x	x	x	x	x	x	x	x	0	
BF2_16	38	38	40	41	39	38	42	38	46	38	38	39,8	
BF2_17	36	x	x	x	x	x	x	x	x	x	x	x	
BF2_18	36	38	36	36	39	36	38	36	36	37	37	x	
BF2_19	37	37	37	39	37	37	38	37	38	39	37	x	
BF2_20	38	38	38	38	40	38	38	41	38	43	38	x	