PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

RESOLUCIÓN DEL PRE-MARSHALLING PROBLEM UTILIZANDO BAT ALGORITHM

JAVIER IGNACIO MACHUCA ARREDONDO

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO DE EJECUCIÓN EN INFORMÁTICA

JULIO, 2016

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

RESOLUCIÓN DEL PRE-MARSHALLING PROBLEM UTILIZANDO BAT ALGORITHM

JAVIER IGNACIO MACHUCA ARREDONDO

Profesor Guía: **Ricardo Soto De Giorgis**
Profesor Co-referente: **Ignacio Araya Zamorano**

Carrera: **Ingeniería Ejecución Informática**

Junio, 2016

Índice

Dedicatoria

Agradezco a todas las personas que hacen posible el término de mi carrera profesional, las que me apoyaron en todo momento y aquellos que me acompañaron en este proceso largo y difícil pero a la vez fugaz, a mis padres y a mis compañeros que me siguen acompañando en este nuevo comienzo. Seguiré esforzándome para llegar más allá de lo imaginable y lograr las metas propuestas, con la misma consigna y dedicación en el nombre de las personas que me apoyaron.

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 Bat Algorithm para la resolución de este problema, la cual está inspirada en la ecolocalización de los murciélagos. Se presentan resultados interesantes para distintas configuraciones de bahías.

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 Bat Algorithm metaheuristic for solving this problem, which is inspired on the echolocation behavior of bats. Interesting results are presented for different bay configurations.

Listado de figuras

Listado de Tablas

1 Introducción

En los terminales portuarios de contenedores, se pueden identificar tres áreas diferentes según la función que cumplen. En primer lugar encontramos el muelle, lugar donde se realizan las descargas de los contenedores, luego el área que recibe los contenedores temporalmente para luego moverlos hacia una última área, llamada bahía de contendores que cumple la función de acumular la máxima cantidad de contenedores en el menor área posible.

Al llegar a la bahía, los contenedores se apilan rápidamente según el orden de llegada, el cual, difiere del orden de salida. Es decir, si un contenedor debe ser entregado en brevedad, es posible que se encuentre bajo varios contenedores, produciendo en la extracción del contenedor una pérdida de tiempo. Este problema puede ser resuelto con un reordenamiento de las pilas de contenedores, y para que este proceso sea eficiente, los contenedores que deberían ser extraídos a corto plazo, se encuentren cercanos a las cimas de las pilas, mientras los contenedores que se extraerán en un largo plazo se encuentren en los niveles más bajos de las pilas.

El Pre-Marshalling, es un problema de optimización que busca minimizar la cantidad de recolocaciones hechas en el reordenamiento de las pilas. En este caso posicionar los contenedores de una forma ordenada con la menor cantidad de movimientos que permitan una eficiente extracción de los mismos.

El problema puede tener una extensa cantidad de soluciones válidas, es por esto que las metodologías exactas llevan mucho tiempo en generar soluciones óptimas. Las metaheurísticas, han sido clave en las nuevas formas de hallar las soluciones que permiten desprender resultados muy cercanos a los óptimos, utilizando diferentes fórmulas y probabilidades asociados a una idea, la cual, puede ser al comportamiento de ciertos animales, como el desplazamiento de un felino, la organización de enjambres, como las hormigas, o incluso, el desplazamiento de las estrellas y agujeros negros.

En este documento el problema es abarcado con la metaheurística Bat Algorithm, que se basa en el uso de la técnica de la ecolocalización de los murciélagos, que en general es utilizada por la mayoría de sus distintas especies, especialmente la especie Microchiroptera, esta especie a pesar de ser una de las más pequeñas físicamente, utiliza en gran medida la ecolocalización para de orientarse en los diferentes ambientes que convive y ubicar de su alimento. La técnica de esta especie permite, al igual que un sonar, emitir pulsos que rebotan en los objetos que se encuentran**.** Es por ello, que para representar la técnica es necesario identificar diferentes variables que se utilizan. Las principales variables son la longitud de onda, la frecuencia y pulsaciones por segundo. En la metaheurística son utilizadas como principales valores que ayudan a la búsqueda de soluciones posibles.

En este documento se explicará, a continuación de la introducción un listado de los objetivos propuestos durante la investigación, el estado del arte de Pre-Marchalling Problem, seguido de una explicación detallada del Pre-Marchalling Problem, la implementación con la metaheurística Bat Algorithm, la implementación realizada y por último, los resultados y conclusiones de la línea investigativa.

2 Objetivos Planteados

En esta sección se describen los objetivos que se propusieron en este trabajo. Se comenzará definiendo a grandes rasgos el objetivo general, en donde se plantea primero la meta que se debe lograr y posteriormente se definen los objetivos específicos, que desarrollan las ideas que se segregan, de manera más detallada, los diferentes puntos que abarcan el objetivo principal.

2.1 Objetivo General

Resolver el Container Pre-Marshalling Problem utilizando la metaheurística Bat Algorithm.

2.2 Objetivos Específicos

A continuación se mostrarán los objetivos específicos que se plantean en esta investigación:

- Comprender a cabalidad el Container Pre-Marshalling Problem.
- Comprender la Metaheurística Bat Algorithm.
- Realizar una implementación de Bat Algorithm para solucionar el Pre-Marshalling Problem.
- Realizar experimentos y analizar resultados de la implementación realizada.

3 Estado del arte

En primera instancia el Pre-Mashalling Problem es analizado por Robert Stahlbock y Stefan Voβ, quienes desarrollan una heurística para que el problema, dado a ciertas restricciones, sea resuelto [1], que posteriormente actualizan los conocimientos del tema en otra publicación [2].

Lee y Hsu, propusieron un modelo de programación entera para el PMP (Pre- Marchalling Problem) aplicándolo para pequeñas y medianas instancias, es decir, el tamaño de instancias es representado según la medición de la cantidad de contenedores, pilas y su altura máxima. Sin embargo, en grandes instancias solo era posible proveer una solución si se minimizaba los tiempos de computación. Además los autores no muestran comparaciones entre este método y otro método.

Bortfeld describe una heurística de búsqueda en árboles, que busca soluciones del Pre-Marchalling [3] y luego a Bortfeld extiende su trabajo junto a Foster. El árbol es codificado en lenguaje C por medio de procedimientos recursivos, que demostraban mejores resultados que los algoritmos propuestos por Lee, Hsu [4] y Chao [5].

Luego Lee propone [5] una heurística de tres fases basada en las reglas para heurísticas que resuelve problema de apilado de contenedores. Los experimentos computacionales mostraban que, esta metodología propuesta podía resolver instancias con más de 700 contenedores.

En referencia a las suposiciones hechas por Kim [6] y Caserta [7] se desarrollan tres métodos para resolver el problema de relocalización de contenedores. El primero de ellos es desarrollado de un modelo de programación entera, la cual, es capaz de resolver pequeñas instancias del problema. Debido al gran tiempo de computación, el modelo fue simplificado por supuestos realistas para poder resolver casos de mediano tamaño. Por la gran cantidad de casos problemáticos, los autores generaron reglas heurísticas basadas en el cálculo de una puntuación por pila, la cual ayuda a determinar cuándo una reubicación de un contenedor puede ser realizada. Esta heurística fue desarrollada en lenguaje de programación C++.

Caserta, es quien propone otro enfoque heurístico para resolver el PMP, mirado desde el Método del Corredor, el cual estipula utilizar un método exacto (por ejemplo Programación dinámica) sobre una parte restringida del espacio de solución de un problema dado para así minimizar el espacio de búsqueda. Ellos utilizaron un algoritmo de DP (Programación dinámica) modificado como un método exacto con CM (Método del Corredor). El cual, resuelve el problema en base los supuestos de Kim [8] y fue codificado en C++ para realizar las experimentaciones correspondientes. Los resultados del algoritmo de Kim y Hong, mostró una disminución en la cantidad de movimientos en todas las instancias, pero cuando eran instancias de mayor tamaño, considerando también que el tiempo de trabajo del algoritmo aumentaba en para instancias más grandes.

Molins presenta una nueva heurística de planificación dominante-dependiente [9], la heurística fue codificada usando la Planificación del Lenguaje Definido Dominante (PDDL). La ventaja de este lenguaje es que es capaz de representar caracteres físicos en los objetos bajo

estudio. Los resultados de la heurística es comparada con los antes expuestos, demostrando una significativa disminución en el tiempo de ejecución [10].

También Molins trata de integrar el PMP con el problema de ubicación de atrancamiento (BAP) y el Problema de asignación de muelle y grúa (QCAP). Primero ellos resuelven el PMP con un método desarrollado anteriormente [9]. Una vez resuelto integraron problema BAP+QCAP, entonces ellos propusieron un planificador que integra la solución de los dos problemas, y el terminal de operadores están para decidir cuál solución es la más apropiada en relación a la función multi-objeto.

Exposito-Izquierdo, presenta la primera heurística de 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. Tras ver los resultados comparados con los anteriores obtenidos por Caserta y Voβ [7], fue visto como una heurística relativamente perfecta.

El Desarrollo de Forster y Bortfeld[12] en el año 2012 generan un algoritmo de árbol de búsqueda para el Pre-Marshalling Problem, que adapta la estructura básica de un árbol de búsqueda incompleto, asimismo utiliza un esquema de clasificación del movimiento más fino, reglas diferentes para ramificación y de delimitación, es por esto que requiere una codiciosa heurística adicional.

Los modelos matemáticos son capaces de resolver solo pequeñas instancias, mientras en instancias más extensas es apropiado utilizar una heurística, en el caso de esta investigación se utiliza Lower Bound, publicado por Bortfeldt y Foster en el año 2010, enriquecido con las investigaciones expuestas [12].

4 Pre-Marshalling Problem (PMP)

En esta sección se formulará el contexto donde surge el problema, la representación utilizada, cuales son las definiciones de; Una configuración, una prioridad de un contenedor, una recolocación, el concepto de fitness, cual su utilidad para el problema, y por último, las consideraciones de que se deben tener en cuenta al trabajar con el Pre-Marshalling Problem.

4.1 Contexto

El Pre-Marshalling es un problema de optimización que busca minimizar la cantidad de recolocaciones de los contenedores, para que puedan salir de manera ordenada y expedita.

El problema surge en terminales portuarios, los cuales, poseen un terreno limitado para manejar grandes cantidades de contenedores. Por lo cual, es necesario conocer las diferentes estructuras de que permiten ordenar las aglomeraciones de contenedores. El acumulado de superposición de contenedores es llamado pila, las que son agrupadas en un bloque. Mientras que una bahía, es una hilera de contenedores de un bloque. Este último concepto es esencial, debido a que el Pre-Marshalling Problem, busca un orden en una bahía y no en un bloque. Esta organización puede visualizada en la figura 4.1.

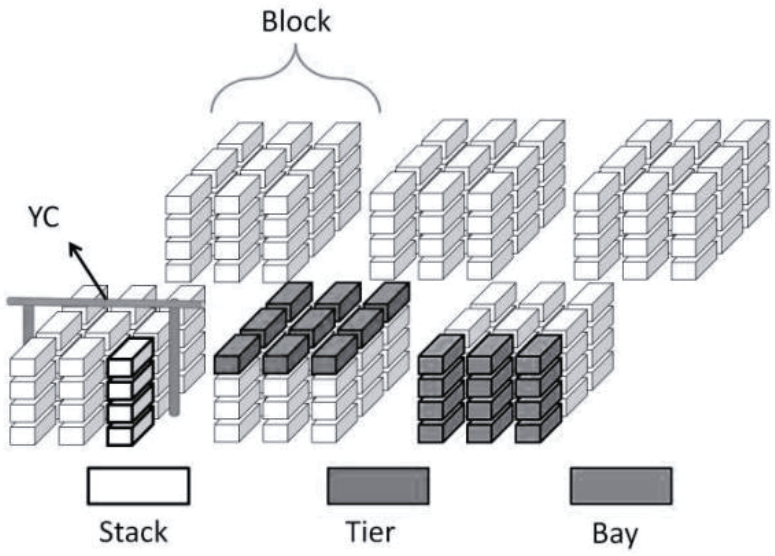

Figura 4.1 Bloques de contenedores.

La organización de la bahía es establecida por el orden de llegada de los contenedores, lo cual, difiere al orden que deben ser despachados. Cabe destacar, que el tiempo utilizado en cada recolocación de un contenedor tiene un valor para la empresa portuaria, al igual que, la distribución de cada uno a su lugar de destino. Es por esto que, se busca minimizar la cantidad de movimientos de cada contenedor, lo cual se traduce en un uso óptimo del tiempo, y a su vez, un menor costo para la empresa portuaria. Por último, redefiniendo el problema, el Pre-Marshalling Problem busca minimizar la cantidad de recolocaciones realizadas en una bahía para cumplir con la secuencia de salida más óptima.

4.2 Representación del Problema

A continuación se definirán los conceptos más importantes que ayudan a la representación del problema.

4.2.1 Configuración

Cómo es posible entender, el Pre-Marshalling busca un orden en la estructura llamada bahía, la cual, debe tener ciertos parámetros que no son modificables. Puesto que, estos valores al ser alterados pueden desplegarse a otros problemas, por lo que, su fin es generar un margen o delimitación del problema.

El primer parámetro de una bahía es la altura, denotada por H, que representa la cantidad de contenedores máximos que son posibles a tener en las pilas. El segundo, corresponde a la cantidad de pilas, denotada por S, que refiere al ancho máximo de contenedores de una bahía. Como tercer y último parámetro, se encuentra la cantidad de contenedores, denotada por C, la cual no debe variar en el proceso del ordenamiento, es decir, ninguno de los contenedores puede ser extraído desde la bahía hasta que finalice el proceso de ordenamiento de la estructura. Uno de los conceptos no medidos, es el bloque, el cual es entendido como el espacio disponible para que un contenedor pueda ser situado.

Cuando en una bahía se realiza una recolocación cambia su distribución visible, no obstante, se mantienen los parámetros de esta, por lo tanto, un estado de cambio de la bahía es llamado configuración.

Ejemplo de una configuración con H igual 4 y S igual 5:

1		3		
3	1	5	1	
6	3	3	9	
6	2	4	8	3

Figura 4.2 Configuración de Contenedores.

4.2.2 Prioridad

La prioridad es un valor asignado a cada contenedor, que no puede ser modificado y expone la urgencia en su entrega. La cifra varía en una escala de números enteros positivos, si está es cercana 0, simboliza una alta prioridad, de lo contrario, si este es un valor elevado, simboliza una baja prioridad. Es por esto que, durante el reordenamiento de los contenedores, se busca que los contenedores con una alta prioridad se encuentren en las cimas de las pilas y los que poseen una baja prioridad se ubiquen en los niveles más bajos de las pilas, es decir, un contenedor se encuentra "bien ubicado" si no superpone a otro con un valor menor, de lo contrario, el contenedor está "mal ubicado" y todos los que los superiores también se clasifican en ese estado.

Por ejemplo, en la figura 4.3, en el lado derecho es posible ver una pila desordenada, visibilizando contenedores que poseen un valor de prioridad alto están superpuestos bajo uno de menor valor, por ende, el contenedor y los que están encima se encuentran mal ubicados. Por otro lado, la pila izquierda ningún contenedor con un valor alto se encuentra sobre uno que posea un menor valor.

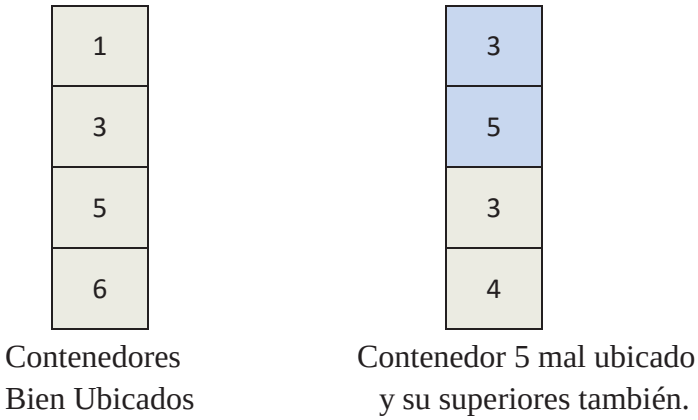

Figura 4.3 Prioridad de Contenedores.

4.2.3 Recolocación

Una recolocación es considerada como el desplazamiento de un contenedor desde una pila a otra dentro de una bahía. La recolocación es representado por dos enteros, el primero es el número de la pila inicial y el segundo la pila final.
Ejemplo: Recolocación (3,5) es representado en la figura 4.2.2.

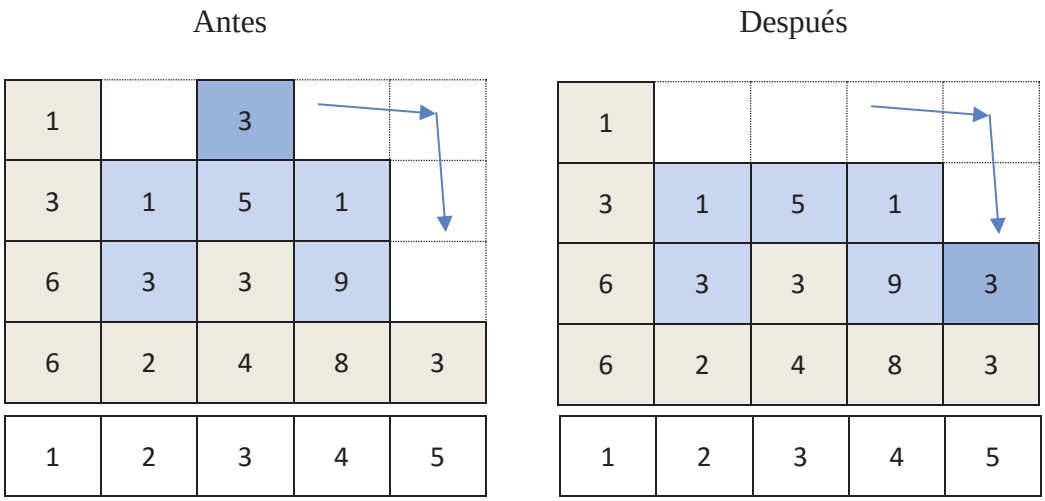

Figura 4.4 Recolocación de pila 3 a pila 5.

Existen ciertas condiciones para que una recolocación represente un desplazamiento real de un contenedor, o sea una recolocación no puede:

- Tener una pila inicial vacía.
- Tener una pila final que esté llena.
- Realizarse sobre la misma pila, en otras palabras, no es posible que la pila inicial sea igual a la pila final.

4.2.4 Fitness

Existen varios métodos evaluadores, también llamados Fitness, estos cumplen una función importante en el desarrollo de una solución, debido a que, permiten obtener la mínima cantidad de recolocaciones que una bahía debe sufrir para estar llegar a una configuración con un orden óptimo.

El mejor método de evaluación según la literatura para el Pre-Marshalling Problem, es llamado Lower bound [12] o Límite Inferior, es de considerar que, el valor obtenido es una referencia, es decir, el número de recolocaciones para llegar a una solución son mayores o iguales al límite inferior. De esta forma, cuando se desarrolla la implementación de soluciones, esta cifra es calculada en cada configuración, con la cual es posible verificar si esta distribución de contenedores se aproxima a una buena solución.

Por ejemplo la configuración de la figura 4.4 es posible ver la bahía desordenada la cual posee un límite inferior igual a 11, mientras que en la figura 4.5, posible ver la misma bahía pero ordenada y un límite inferior igual a 0.

1		3		
3	1	5	1	
6	3	3	9	
6	2	4	8	3

Figura 4.5 Configuración Desordenada.

1				
3	1	3	1	
6	2	5	8	3
6	3	5	9	3

Figura 4.6 Configuración Ordenada.

El Lower Bound [12] tiene un proceso dividido en tres fases, desde donde cada una es posible obtener un valor:

$nBx = nb + \min\{nb(s) | s = 1,2, \dots, S\}$.
$nGx = \sum n^{g*}(s)^{n(s,Gx)}$, $s=1$.
$nm = nGx + nBx$.

nBx: Se selecciona la pila que contenga la menor cantidad de contenedores mal posicionados y esa cantidad de contenedores es sumada la cantidad de contenedores mal posicionados en toda bahía.
nGx: El número de contenedores bien posicionados desde la primera pila hasta una cierta pila. Desde este paso se obtiene g* (prioridad del óptimo) y su valor de Ds(g).
nm: Es la suma de la cantidad de contenedores mal posicionados (nBx) y la cantidad de contenedores bien posicionados (nGx).

Por ejemplo, utilizando el ejemplo de la figura 4.5, podemos encontrar contenedores desordenados pintados de color más oscuro, son clasificados como contenedores mal posicionados, y los contenedores pintados de color más claro, son clasificados como contenedores bien posicionados, según las especificaciones de prioridad anteriormente mencionadas.

En la fase i se obtiene, 6 el número de contenedores mal posicionados (contenedores más oscuros) más el número de contenedores mínimo de contenedores mal posicionados en las pilas de la bahía.

Durante la fase ii utiliza una tabla en la que se tabulan los valores obtenidos desde la bahía.

- La primera columna (g) referirá a las un listado de todas las prioridades, desde la más alta a la más baja.
- En la columna contigua (d(g)), la "demanda" se denota por la cantidad de contenedores mal posicionados de la prioridad g.
- En la siguiente columna (D(g)) se genera la acumulación de las demandas (d(g)).
- En la columna adyacente (sp(g)) corresponde a el número de espacios de ofertas potenciales para los contenedores con la prioridad g.
- La columna siguiente (Sp(g)) corresponde a la acumulación del número de espacios de ofertas (sp(d))
- Y por último, (Ds(g)) esta representa la demanda acumulada(D(g)) menos la oferta potencial acumulada(Sp(g)). De esta forma podemos visualizar, para la configuración de la figura 4.5, la tabla que se menciona.

g	$d(g)$	$D(g)$	$sp(g)$	$Sp(g)$	$Ds(g)$
9	1	1	0	0	1
8	0	1	3	3	-2
6	0	1	0	3	-2
5	1	2	0	3	-1
4	0	2	0	3	-1
3	1	3	3	6	-3
2	0	3	3	9	-6
1	0	0	0	9	-6

Tabla 4.1 Tabla paso ii Lower Bound.

De la tabla 4.1 se obtiene la prioridad que posee el mayor Ds(g), en este caso 9 con valor 1. Para obtener la cantidad de iteraciones de la sumatoria se obtiene de:

$$(s, Gx) = max\ (0, [Ds(g*)/H])$$

De esta forma tenemos que H = 4 y Ds(g*)=1, tenemos una división de 1: 4= 0,25 a este valor se le aproxima al entero superior, es decir, 1.

Entonces la sumatoria se realiza desde la pila 0 hasta la pila 1, donde buscaremos los contenedores menores a 9 los cuales son 1, 3, 6, 6 de la pila 0 y 2 de la pila 1, que en su total es de 5 por lo tanto $nGx = 5$.

Y como último en el paso iii se realiza la suma de los dos valores obtenidos anteriormente:

$$nm = nGx + nBx$$
$$nm = 5 + 6$$
$$nm = 11$$

Por lo tanto 11 es el Lower Bound o la mínima cantidad de recolocaciones a realizar para que la bahía esté organizada.

4.2.5 Otras consideraciones importantes

El problema posee ciertas restricciones, de esta forma, esta definición permitirá no alejarse de lo que se desea representar, además de fijar reglas que acotaran los términos de recolocación, junto con, las propiedades de los contenedores.

Las consideraciones son:

- Las bahías tienen un número fijo de altura (cantidad de contenedores apilados) y pilas, además las recolocaciones se efectúan dentro de estas, ya que, al moverlas a otra bahía se estaría, manejando otro problema distinto al Pre-Marshalling Problem.
- En una bahía, no se requerirá tener contenedores en todas sus pilas.
- Todos los contenedores son del mismo tamaño y peso similar, en el caso contrario, esto dificultaría el proceso de las recolocaciones, obligando a tomar medidas de seguridad debidas.
- Las prioridades de los contenedores suelen darse números enteros y con anterioridad a la recolocación.
- La bahía debe tener un porcentaje mínimo de posiciones libres para realizar las recolocaciones, esto otorga la posibilidad de realizar recolocaciones dentro de la bahía.

5 Bat Algorithm

En esta investigación, la implementación de la búsqueda de soluciones es enfocada en una metaheurística. Una metaheurística es un método inexacto, que realiza una indagación de soluciones por medio un proceso repetitivo de algoritmos y procedimientos abstractos, los cuales, poseen un bajo tiempo en su ejecución. De esta manera, internamente funciona de diferentes maneras, según el algoritmo utilizado, pero todas examinan el dominio del problema de diferentes formas con el fin de mejorar de soluciones en cada ciclo.

La metahurística utilizada en este trabajo es llamada Bat Algorithm, enfocada en el comportamiento de enjambre de los murciélagos, de esta forma, es considerando ciertas características que los representa, las que son adaptadas y utilizadas en la indagación de soluciones.

5.1 Definición

Las especies de murciélagos, poseen una de las características destacable llamado ecolocalización, que consiste en que el animal es capaz de generar ondas de sonido de alta frecuencia que rebotan en los obstáculos para volver a los mismos y darle una orientación en el medio. Otra de sus particularidades, que la técnica anteriormente mencionada es utilizada, en mayor medida, por las especies de menor tamaño. Las especies más grandes llegan a pesar un kilogramo y a medir, entre los extremos de sus alas, dos metros, mientras que las de menor volumen llegan a medir menos de dos centímetros y alcanzar los dos gramos.

Existen diversas variaciones de la metaheurística, ahora bien, la versión estándar de esta se basada en la ecolocalización o bio-sonar caracterizado la especie Microchiroptera. La ecolocalización permite a esta especie de murciélagos emitir pulsaciones de sonar muy fuertes que rebotan en los objetos circundantes. Estos pulsos varían en sus características y se puede relacionar con sus estrategias de caza. Gran parte de los murciélagos utilizan señales de frecuencia modulada entre 25 a 150 Hertz para barrer su entorno en una octava por segundo, y cada pulso dura entre 8 a 10 milésimas de segundo. Normalmente esta especie de murciélagos emite entre 10 a 20 ráfagas de sonidos por segundo las que pueden aumentar hasta 200 por segundo cuando se encuentra sobre su presa. Teniendo en cuenta que la velocidad del sonido es de v igual 340 mts/s, la longitud de onda λ de ráfagas de sonido ultrasónicas con una frecuencia f constante está dada por λ es igual a v /f, está en el intervalo de 2 mm a 14 mm para el típico rango de frecuencia de 25 kHz a 150 kHz. Curiosamente, estas longitudes de onda están en el mismo orden de sus tamaños de presa.

Los murciélagos pueden utilizar un retardo entre sus orejas junto con la sonoridad, permitiéndoles un sentido del entorno en tres dimensiones. Características de la ecolocalización que pueden vincularse con una función objetivo de un problema de optimización. Los murciélagos pueden utilizar un retardo entre sus orejas junto con la sonoridad, permitiéndoles un sentido del entorno en tres dimensiones. Características de la ecolocalización que pueden vincularse con una función objetivo de un problema de optimización.

5.2 Reglas a considerar

Según las investigaciones realizadas de la ecolocalización por parte de Yang (2010), al desarrollar el Bat Algorithm formula las tres siguientes reglas:

1) Todos los murciélagos utilizan la ecolocalización para detectar su ubicación, conocer la diferencia entre alimento, comida y los obstáculos que se presenten en su camino.

2) Los murciélagos vuelan randomicamente con una velocidad vi en una posición xi con una frecuencia $fmin$ variando la longitud de onda y el volumen para buscar a su presa. Ellos pueden ajustar su longitud de onda o frecuencia de sus pulsos y ajustar el rango de emisión $r \in [0,1]$, dependiendo de la proximidad de su objetivo.

3) Se el volumen varía de forma decreciente desde un $A0$ positivo a un valor mínimo constante $\mathcal{A}_{mink}$.

5.3 Movilidad del Bat

Cada Bat es representado por ***i*** y ***t*** representa la iteración en que se encuentra el proceso. Entre uno de todos los Bats, existe el de la mejor solución actual, de él se utiliza uno de sus parámetros, su posición representado por $x*$. Utilizando esta información es posible visualizar la movilidad de los Bats por medio de las siguientes ecuaciones:

$$fi = fmin + (fmax - fmin)\beta, \quad (1)$$

$$vti = vt-1i + (xt-1i - x*)fi, \quad (2)$$

$$xti = xt-1i + vti, \quad (3)$$

Donde β es un vector aleatorio dibujado por una distribución uniforme, por otro lado, las variables de frecuencia poseen cifras desde un $fmin$ igual a 0 hasta un $fmax$ igual a $O(1)$ este último valor puede ser modificado según la amplitud del problema. Inicialmente, cada Bat se le asigna randomicamente una frecuencia la cual se dibuja por una distribución uniforme desde $[fmin, fmax]$. Por esta razón, este algoritmo puede ser considerado como un algoritmo de modulación de frecuencia para proveer un balance de combinaciones entre explotación y de exploración. El volumen y el rango de pulso de emisión generan un mecanismo automático de auto-Zoom en una región prometedora.

5.4 Manejo de Volumen y pulsos de emisión

Con el fin de proporcionar un mecanismo eficaz que controle la explotación y exploración, permitiendo también cambiar a la etapa de explotación cuando sea necesario, debemos variar un volumen Ai y una tasa ri de pulsos de emisión durante las iteraciones. Dado que el volumen decrece continuamente una vez el Bat ha encontrado su presa. Puede elegir cualquier valor a conveniencia entre $Amin$ y $Amax$ suponiendo $Amin$ igual a 0 significa que un Bat acaba de encontrar a su presa y se detendrá temporalmente sin emitir ningún pulso. Según esto tenemos:

$$At + 1i \; = \; \alpha At + 1i \, , \;\; rt + 1i \; = \; r0 \, [\, 1 - \, exp(-\gamma t)], \qquad (4)$$

Donde γ y α son constantes, en esencia aquí α es similar a un factor alineador, catalogado como enfriador en simulación fortalecida. Para cualquier $0 \; < \; \alpha \; < \; 1$ y $\gamma \; > 0$ tenemos:

$$Ati \; \rightarrow \;\; 0, \; rti \; \rightarrow \; r0, as \; t \; \rightarrow \; \infty, \qquad (5)$$

5.5 Pseudocódigo

Los pasos más importantes de la metaheurística Bat Algorithm [14] son:

1: Fitness.
2: Inicializa la población de murciélagos xi y vi , $i = 1,2,\ldots,m$.
3: Define el pulso de frecuencia para xi , $Vi = 1,2,\ldots,m$.
4: Inicializa los pulsos de emisión y el volumen Ai, $i = 1,2,\ldots,m$.
Iterar hasta el máximo de iteraciones T
Por cada bat bi , do
5: Genera nuevas soluciones a través de las ecuaciones (1), (2) y (3).
Si un valor randomico > ri, entonces
6: Selecciona una nueva solución entre las mejores soluciones.
7: Genera una solución local cerca de la solución global.
Si un valor randomico < Ai y $f\ (xi) < f\ (x*)$, entonces
8: Acepto nuevas soluciones
9: Incrementa ri y reduce A_i
10:Término de T iteraciones
11: Clasifica los murciélagos y encuentra el mejor $x*$

Figura 5.1 Pseudocódigo Bat Algorithm.

También en el paso 5 podemos aceptar la siguiente ecuación

$$Xnew = xold + \epsilon\, Ap(t), \qquad (6)$$

Donde ϵ es un valor aleatorio entre $[-1,1]$ y $Ap(t)$, es el promedio del volumen hasta la iteración t.

6 Implementación

El desarrollo de la implementación consistió en 2 etapas, la primera fue en generar un desarrollo para la técnica utilizada como Fitness, Lower Bound, mientras que, la segunda etapa consistió en desarrollar la metaheurística adaptada para el problema del Pre-Marshalling Problem.

6.1 Representación de soluciones

Una de las consideraciones, a tomar en cuenta al comenzar la implementación, es que el Pre-Marshalling Problem es representado por medio de vectores, es decir, una recolocación consiste en par de valores, por consiguiente, al generar una solución con varias recolocaciones se crea una matriz bidimensional. Esta visualización del problema contrasta con la metaheurística, debido a que la manipulación de variables se genera en un plano real. Debido a lo anterior mencionado, uno de los trabajos más costos sería esta conversación entre ambas partes, para lo cual, se logra idear una forma en la cual las soluciones no fuesen una matriz sino un simple vector de valores enteros, este es el motivo que hace surgir la Matriz de recolocaciones.

La Matriz de Recolocaciones consiste en generar un array bidimensional, donde se enumeran de manera selectiva y ordenada, la mayor parte de las casilla, las que a su vez, representan una recolocación. Para cada uno de estos espacios dentro de la matriz, existen dos identificadores de posición que simulan la pila inicial y la pila final.

La matriz posee una dimensión igual a la del número de pilas de la bahía, en su ancho y largo. Por ejemplo si se posee una bahía con 5 pilas o ancho de la bahía, esto se puede apreciar en la figura 6.2.

	0	1	2	3	4
0	0	1	2	3	4
1	5	0	6	7	8
2	9	10	0	11	12
3	13	14	15	0	16
4	17	18	19	20	0

Figura 6.1 Matriz de Recolocaciones

El cupo dentro de la matriz es de S x S, pero no todos las recolocaciones son admisibles, por ello se deben descartar las recolocaciones que poseen una pila de origen y destino igual. En la figura 6.1 se pueden visualizar por su color oscuro, que a su vez coordinan con la diagonal de la matriz.

En resumen una recolocación se puede representar por valor numérico entero, logrando obtener una cantidad de S x (S-1) recolocaciones representables, que en el caso de la figura 6.1 son 5 x (5 - 1) = 20 recolocaciones.

Para definir una solución, esta se compone de una serie de recolocaciones en una lista o array, que permite ser representados por un listado de números enteros, debido a la fácil representatividad que estos poseen con la matriz de recolocaciones.

Un ejemplo de una solución puede ser:

4	8	19	1	4	11

Figura 6.2 Lista de Recolocaciones

Cada una de estas casillas hace referencia a un recolocación.

6.2 Implementación de Lower Bound

Lower Bound define cuales son las variables numéricas enteras necesarias para poder generar un resultado, estas son: H, correspondiente a la altura de la bahía; S, que es el número de pilas de la bahía y la distribución de los contenedores a través de las pilas.

Un ejemplo claro de esto, es la siguiente configuración:

Label : LI_2
Width : 4
Height : 3
Containers : 4
Stack 1 : 1
Stack 2 :
Stack 3 : 2 11 5
Stack 4 :

En resumen, esta etapa es terminada una vez que el prototipo logra leer un archivo con extensión bay, identifica los elementos de la configuración, los representa en forma de estructuras de datos, y por último, generar recolocaciones dentro de la bahía ya leída. La implementación responda sin problemas, de acuerdo a las consideraciones mencionadas, y permita la resolución del Lower Bound con una nueva configuración.

6.3 Implementación de Bat Algorithm

En primer lugar, se diseña una clase Bat que albergará como atributos los parámetros que se manejan dinámicamente en el procedimiento del Bat algorithm, los vectores de velocidad y posición del bat, volumen y pulsaciones. También es de considerar agregar las variables de frecuencia mínima, frecuencia máxima, la cantidad de individuos a utilizar y las iteraciones, que a diferencia de las variables mencionadas anteriormente, no son modificadas durante la ejecución del programa.

Se crea una matriz de recolocaciones con las dimensiones según el número de pilas del caso de prueba.Se genera una instanciación de cada elemento del Bat asignándole valores iniciales, que permitan dar el primer paso a la búsqueda de soluciones, para cada ejemplar, enlista recolecciones posibles de generar, pero que no buscan una reducción del límite inferior. Estas recolocaciones son números enteros que se extienden desde uno hasta el número de pilas multiplicado por el número de pilas menos uno (matriz de recolocaciones). Luego se inicializa los valores, punto flotante, de los vectores de velocidad y frecuencia.

En el proceso de desplazamiento del Bat, se hace uso de las fórmulas principales de la metaheurística, para ir mejorando los resultados que se obtienen anteriormente. Uno de los valores que es importante en la selección de soluciones, es la frecuencia, ya que, es un valor racional que se utiliza en los condicionales de pseudocódigo para seleccionar el mejor resultado del ciclo.

Tras pruebas y errores, se experimenta que los valores no siempre son permisibles en el contexto de la bahía, es decir, tras aplicar la fórmula de posición, el valor obtenido en una recolocación se extendía en una pila fuera de la configuración o generaban valores negativos, lo cual es un problema de gran importancia. Esto genera la necesidad incrementar más de variabilidad de resultados obtenidos por el algoritmo, es por esto que, como solución es creado un reparador que intenta suplir las recolocaciones no admisibles generadas por la metaheurística y que creaban el conflicto anterior mencionado.

El reparador consiste en un método que busca la siguiente recolocación posible con menor límite inferior. Además existe otra secuencia de pasos que ayuda a la mejora de soluciones, la que consiste en aumentar en una recolocación la solución del Bat, la cual consiste en que cada cierta cantidad de iteraciones, se genera una nueva recolocación con la posibilidad de ir aumentando el largo del vector de solución pero buscando a la vez la recolocación que genere el menor límite inferior. De esta manera es posible dar un un poco más de agilidad a la implementación permitiendo así un mejor desempeño.

7 Experimentación

La experimentación consiste en mostrar las tipos de bahías con las cuales se prueba la implementación, para ello, es imperante establecer valores iniciales para la metaheurística, y delimitar y describir las especificaciones definidas en la implementación. Por último, se muestran los resultados obtenidos de la implementación.

7.1 Bats

Los siguientes valores numéricos son empleados para los Murciélagos:

$$fmin = 1.15$$
$$fmax = 0.75$$
$$ri(0)\ \epsilon\ [0,1]$$
$$Ai(0)\ \epsilon\ [1,2]$$

7.2 Parámetros de Implementación

Se debe considerar las configuraciones o parámetros que se utilizan en la implementación, lo cual está representado en la Tabla 7.1.

Parámetros	Bats	Iteraciones	Recolocaciones adheridas *	Ejecuciones	cada cuantos ciclos agrega una recolocación
valores	25	5000	40%	5	100

Tabla 7.1 Parámetros de Implementación.

*Las recolocaciones adheridas se refieren a la cantidad de recolocaciones máximas que puede poseer un bat, con respecto al óptimo conocido de la carpeta.

7.3 Casos de prueba

Los casos de prueba fueron otorgados por parte del profesor guía, los cuales consisten en carpetas con un enunciado que clasifica los tipos de casos de prueba dentro de la misma que se acompañan con un número que hace referencia a las diferentes dimensiones de los casos de prueba. Por ejemplo, existen 4 carpetas con el enunciado CV [15] y se acompañan con la enumeración de 1 a 4, correspondiente a las diferentes dimensiones del tipo de caso de prueba. Es por esto que debe tomar en cuenta que carpetas de casos de prueba se están testeando.

En este trabajo es fue posible abarcar todas las carpetas con el enunciado CV, donde las pruebas tienen una bahía que posee una altura mayor a la cantidad de pilas en esta.

Abreviatura	Significado
LI	Limite Inferior
RME	Recolocaciones de la mejor ejecución
LME	LI de la mejore ejecución
TE	Tiempo de Ejecución
PR5	Promedio de 5 ejecuciónes
PLI	Promerio de Limite inferior de 5 las ejecuciones
OC	Optimo conocido de la Carpeta
PO	Promedio Obtenido

Tabla 7.2 Abreviaturas.

7.4 Resultados obtenidos

En la siguiente tabla 7.2 y 7.3 es posible ver el promedio de recolocaciones en cada caso de prueba por carpeta y el óptimo anteriormente conocido.

Carp.	Caso	LI	RME	LME	TE	PR5	PLI	OC	PO
CV1	**CV1_1.bay**	8	13	0	0' 54"	14,6	0	**10,1**	**12,5**
	CV1_10.bay	7	8	0	0' 37"	10,6	0		
	CV1_2.bay	8	15	0	0' 47"	15	0,2		
	CV1_3.bay	8	14	0	0' 46"	14,6	0		
	CV1_5.bay	8	15	0	0' 49"	15	0,8		
	CV1_4.bay	7	11	0	0' 54"	13	0		
	CV1_6.bay	6	12	0	0' 50"	14,2	0		
	CV1_7.bay	8	15	0	0' 53"	15	0		
	CV1_8.bay	7	13	0	0' 52"	14,4	0		
	CV1_9.bay	7	9	0	1' 0"	11,4	0		
CV2	**CV2_1.bay**	14	23	0	2' 52"	29	0	**19,1**	**22,9**
	CV2_2.bay	13	22	0	2' 20"	23,2	0		
	CV2_3.bay	10	12	0	1' 14"	13,4	0		
	CV2_4.bay	14	29	0	3' 14"	28	1,4		
	CV2_5.bay	14	22	0	3' 12"	26,8	1		
	CV2_6.bay	13	20	0	2' 24"	22,2	0		
	CV2_7.bay	15	27	0	2' 56"	30,6	0		
	CV2_8.bay	13	21	0	2' 3"	24	0		
	CV2_9.bay	15	29	0	3' 35"	28,6	1,4		
	CV2_10.bay	15	24	0	2' 58"	25,4	0		

Tabla 7.3 Resultados obtenidos en las carpetas CV1 y CV2.

Es posible observar las tabla 7.2 es posible ver resultados que informan cercanía con los óptimos o al menos, con soluciones (lista de recolocaciones) que entregan un límite inferior igual 0. Además muestran una cercanía con los valores de los óptimos conocidos para las carpetas CV1 y CV2, generando expectativa y proximidad a producir buenos resultados en futuros trabajos.

CV3	**CV3_1.bay**	22	36	2	6' 27"	38	4	**30,4**	**36,1**
	CV3_10.bay	20	36	0	5' 17"	38	1,4		
	CV3_2.bay	21	32	0	4' 20"	34,6	2,4		
	CV3_3.bay	20	37	4	5' 18"	34,6	6,4		
	CV3_4.bay	20	38	6	6' 0"	33,4	7		
	CV3_5.bay	21	40	5	5' 50"	36,6	6		
	CV3_6.bay	22	38	6	6' 24"	37,4	7		
	CV3_7.bay	24	33	4	5' 29"	38,8	5,8		
	CV3_8.bay	22	41	4	7' 18"	36,4	5,4		
	CV3_9.bay	23	30	4	5' 0"	37,8	3,6		
CV4	**CV4_1.bay**	26	37	9	19' 53"	41,6	7,8	**44,4**	**43,2**
	CV4_2.bay	30	42	11	23' 36"	42,4	11		
	CV4_3.bay	33	44	13	25' 49"	46	12		
	CV4_4.bay	32	48	11	25' 52"	43,8	13,4		
	CV4_5.bay	32	49	6	20' 44"	39,6	11,6		
	CV4_6.bay	33	44	10	22' 53"	41	14		
	CV4_7.bay	27	43	7	21' 39"	34	11,6		
	CV4_8.bay	28	42	8	20' 56"	34,6	11		
	CV4_9.bay	29	42	12	22' 21"	42	11,8		
	CV4_10.bay	28	41	12	20' 52"	40,4	11,8		

Tabla 7.4 Resultados obtenidos en carpetas CV3 y CV4.

En la Tabla 7.4, los resultados obtenidos no muestran una cercanía con los valores de los óptimos conocidos para las carpetas CV3 y CV4, generando la idea de generar nuevas aplicaciones e implementaciones que puedan ayudar a obtener buenos resultados.

Carp.	Caso	LI	RME	LME	TE	PR5	PLI	OC	PO
BF1	cpmp_16_5_48_10_29_1.bay	29	30	0	70' 8"	30,4	0	29	30,78
	cpmp_16_5_48_10_29_2.bay	29	30	0	67' 38"	30	0		
	cpmp_16_5_48_10_29_3.bay	29	32	0	28' 42"	32	0		
	cpmp_16_5_48_10_29_4.bay	29	30	0	98' 36"	30,6	0		
	cpmp_16_5_48_10_29_5.bay	29	30	0	87' 8"	30,4	0		
	cpmp_16_5_48_10_29_6.bay	29	31	0	75' 46"	31	0		
	cpmp_16_5_48_10_29_7.bay	29	30	0	70' 14"	30	0		
	cpmp_16_5_48_10_29_8.bay	29	30	0	88' 44"	30	0		
	cpmp_16_5_48_10_29_9.bay	29	32	0	92' 34"	31,8	0		
	cpmp_16_5_48_10_29_10.bay	29	30	0	78' 18"	30	0		
	cpmp_16_5_48_10_29_11.bay	30	33	0	85' 5"	33,4	0		
	cpmp_16_5_48_10_29_12.bay	29	31	0	71' 35"	31	0		
	cpmp_16_5_48_10_29_13.bay	29	29	0	52' 41"	29	0		
	cpmp_16_5_48_10_29_14.bay	29	30	0	43' 17"	30,8	0		
	cpmp_16_5_48_10_29_15.bay	29	32	0	60' 54"	32	0		
	cpmp_16_5_48_10_29_16.bay	29	30	0	37' 45"	30,4	0		
	cpmp_16_5_48_10_29_17.bay	29	29	0	51' 25"	29,4	0		
	cpmp_16_5_48_10_29_18.bay	29	30	0	43' 38"	30,8	0		
	cpmp_16_5_48_10_29_19.bay	29	31	0	52' 13"	31	0		
	cpmp_16_5_48_10_29_20.bay	29	31	0	56' 56"	31,6	0		

Tabla 7.5 Resultados obtenidos en carpeta BF1.

En la Tabla 7.5 es posible ver una cercanía del promedio de la carpeta [17] con el óptimo conocido.

Carp.	Caso	LI	RME	LME	TE	PR5	PLI	OC	PO
LC1	lc1.bay	14	17	0	10' 41"	18,2	0	15	18,2

Tabla 7.6 Resultados obtenidos en carpeta LC1.

Los resultados obtenidos en la tabla 7.6 muestran que los resultados se acercan al óptimo por 3,2 recolocaciones.

Carp.	Caso	LI	RME	LME	TE	PR5	PLI	OC	PO
LC2b	LC2b_1.bay	37	48	0	48' 4"	51	0,4	**38,4**	**48,34**
	LC2b_2.bay	36	45	0	48' 3"	47,2	0		
	LC2b_3.bay	39	49	0	58' 13"	50,8	0		
	LC2b_4.bay	35	48	0	51' 44"	50	0		
	LC2b_5.bay	39	52	0	60' 36"	53	0		
	LC2b_6.bay	40	46	0	40' 30"	48,6	0		
	LC2b_7.bay	37	43	0	37' 9"	44,6	0		
	LC2b_8.bay	35	41	0	35' 12"	43	0		
	LC2b_9.bay	35	42	0	40' 54"	47,4	0		
	LC2b_10.bay	39	47	0	41' 32"	47,8	0		

Tabla 7.7 Resultados obtenidos en carpeta LC2b.

En la Tabla 7.7 es posible apreciar que existe una lejanía entre el óptimo conocido y el promedio de ejecuciones generadas.

Carp.	Caso	LI	RME	LME	TE	PR5	PLI	OC	PO
LC2a	lc2a_1.bay	21	29	0	21' 53"	34	0,4	22,6	30,64
	lc2a_2.bay	21	29	0	21' 32"	32,4	0,2		
	lc2a_3.bay	21	27	0	22' 45"	29,4	0		
	lc2a_4.bay	22	32	0	35' 2"	34	0		
	lc2a_5.bay	24	30	0	30' 12"	32	0		
	lc2a_6.bay	22	33	0	31' 59"	34,8	0		
	lc2a_7.bay	23	29	0	26' 1"	30,8	0		
	lc2a_8.bay	20	24	0	19' 55"	24,8	0		
	lc2a_9.bay	19	25	0	26' 30"	26,6	0		
	lc2a_10.bay	23	27	0	23' 31"	27,6	0		

Tabla 7.7 Resultados obtenidos en carpeta LC2a.

En la Tabla 7.7 es posible ver una diferencia entre el óptimo conocido y el promedio obtenido de 8,64 recolocaciones.

8 Conclusiones

El Pre-Marchalling Problem es un problema de optimización que ha sido tema de investigación en la literatura por varios autores, que hoy en día siguen retroalimentando con más información en el tema, generando nuevos métodos de búsqueda de soluciones que varían en el cómo se realiza la búsqueda de soluciones. Tras una línea investigativa sobre Bat Alorithm, se observa que tiene buenas propiedades de búsqueda de soluciones con un domino real, con variables que modelan posicionamientos y variabilidad de soluciones que permiten, a través de la modulación de frecuencia, mejoras de las soluciones para enfrentar un problema.

Otro punto importante a mencionar es que en la generación de soluciones, que a su vez es base importante que se reproduzca una solución factible, es el uso de las herramientas reparadoras de soluciones, que permiten mantener que los valores obtenidos por la metaheurística no salgan de los márgenes del problema, es decir, en el caso de los valores de las recolocaciones se debe verifica que sean enteros positivos, según la definición de estos. Otra consideración a tomar, es no permitir recolocaciones desde una pila que no tiene contenedores o dirigida a una pila llena, ya que, extienden y no siguen las normas planteadas en el problema. Las consideraciones mencionadas son aglomeradas junto con varias otras, identificadas en el proceso de implementación, que asimilan una recolocación posible. Desde este punto, la metaheurística, puede ayudar o sugerir las siguientes recolecciones, que son igualmente evaluadas con estas normas, formando un proceso de generación y descarte. De este modo se arma la lista de recolocaciones, la cual, se compara con otras, para saber si es la mejor solución.

Sin embargo el Pre-Marshalling Problem no posee un dominio real, es por esto que el autor realiza modificaciones, como es el uso de la matriz de movimientos, que en la implementación ayudan, acercan e intentan ejercer la mejor comunicación posible entre el problema y la metaheurística. A pesar de ello, las soluciones solo se acercan a los óptimos conocidos.

Por lo anteriormente mencionado, en los casos específicos, las soluciones generadas en las instancias de las pruebas pertenecientes a las carpetas CV1 y CV2, han podido ofrecer resultados prometedores, cercanos a un 2.4 y 3.8 de recolocaciones al óptimo conocido respectivamente. Se espera poder trabajar más aun en los casos de prueba correspondientes a las carpetas CV para mejorar los resultados.

9 Referencias:

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

[2] D. Steenken, S. VoB, and R. Stahlbock, "Container terminal operations and operations research - a classification and literature review," OR spectrum,vol. 26, pp., 3-49, 2004.

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

[4] Y. Lee, and N. Hsu, "An optimization model for the container pre-marshalling problem," Computer & Operations Research, vol. 34,pp. 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, pp. 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, pp. 89-101, 2000.

[7] M. Caseta, and S. VoB, "A corridor method-based algorithm for the pre-marshalling problem", EvoWorkshops, vol. 5484 LNCS, pages 788-797, 2009.

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

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

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

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

[12] A. Bortfeldt, Florian Forster, "A tree search procedure for the container pre-marshalling problem", Journal European of Operational Research, October 2011.

[13] Xin-She Yang, "Bat Algorithm: Literature Review and Applications", School of Science and Technology, Middlesex University, The Burroughs, London NW4 4BT, United Kingdom, 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 Graphics, Patterns and Images (SIBGRAPI), 2012.

[15] Caserta, M., Voß, S.: A corridor method-based algorithm for the pre-marshalling problem. In: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G.A., Ek´art, A., Esparcia-Alc´azar, A.I., Farooq, M., Fink, A., Machado, P. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 788–797.

[16] Y. Lee and S.Chao. A neighborhood search heuristic for premarshalling export containers. European Journal of Operational Research, 196(2):468-475, 2009.

[17] Bortfeldt, A., Forster, F.: A tree search procedure for the container pre-marshalling problem. Eur. J. Oper. Res. 217(3), 531–540 (2012) 2.

10 Anexos

En primera instancia se mostraran las abreviaturas utilizadas en las tablas para referirse al detalle de la información obtenida.

10.1 Detalle de resultados obtenidos

Las columnas EX, donde X es un número, corresponden a las ejecuciones, y LX, donde X es un número, corresponden a el límite inferior final una vez aplicado todas las recolocaciones de la ejecución. PE es el promedio de recolocaciones de las 5 ejecuciones y PL es el Promedio del Límite inferior de las 5 ejecuciones

carpeta	Caso	LI	E1	L1	E2	L2	E3	L3	E4	L4	E5	L5	PE	PL
CV1	**CV1_1.bay**	8	13	0	15	0	15	0	15	0	15	0	**14,6**	**0**
	CV1_2.bay	7	13	0	8	0	10	0	12	0	10	0	**10,6**	**0**
	CV1_3.bay	8	15	1	15	0	15	0	15	0	15	0	**15**	**0,2**
	CV1_4.bay	8	14	0	14	0	15	0	15	0	15	0	**14,6**	**0**
	CV1_5.bay	8	15	1	15	0	15	1	15	0	15	2	**15**	**0,8**
	CV1_6.bay	7	13	0	14	0	11	0	12	0	15	0	**13**	**0**
	CV1_7.bay	6	15	0	15	0	14	0	15	0	12	0	**14,2**	**0**
	CV1_8.bay	8	15	0	15	0	15	0	15	0	15	0	**15**	**0**
	CV1_9.bay	7	13	0	14	0	15	0	15	0	15	0	**14,4**	**0**
	CV1_10.bay	7	12	0	11	0	9	0	12	0	13	0	**11,4**	**0**
CV2	**CV2_1.bay**	14	33	0	25	0	23	0	32	0	32	0	**29**	**0**
	CV2_2.bay	13	25	0	24	0	24	0	22	0	21	0	**23,2**	**0**
	CV2_3.bay	10	12	0	13	0	15	0	14	0	13	0	**13,4**	**0**
	CV2_4.bay	14	24	2	29	0	24	2	33	1	30	2	**28**	**1,4**
	CV2_5.bay	14	27	2	32	0	26	1	27	2	22	0	**26,8**	**1**
	CV2_6.bay	13	23	0	20	0	20	0	23	0	25	0	**22,2**	**0**
	CV2_7.bay	15	27	0	31	0	33	0	31	0	31	0	**30,6**	**0**
	CV2_8.bay	13	24	0	20	0	31	0	24	0	21	0	**24**	**0**
	CV2_9.bay	15	29	0	31	1	23	4	29	1	31	1	**28,6**	**1,4**
	CV2_10.bay	15	28	0	27	0	24	0	24	0	24	0	**25,4**	**0**

Tabla 10.1 Ejecuciones CV1 y CV2.

carpeta	Nombre bahía	LI	E1	L1	E2	L2	E3	L3	E4	L4	E5	L5	PE	PL
CV3	**CV3_1.bay**	22	37	4	36	2	38	4	38	5	41	5	**38**	**4**
	CV3_2.bay	21	36	0	34	1	35	3	33	1	35	2	**34,6**	**1,4**
	CV3_3.bay	20	33	5	32	2	39	4	37	1	32	0	**34,6**	**2,4**
	CV3_4.bay	20	34	7	37	4	28	8	37	7	31	6	**33,4**	**6,4**
	CV3_5.bay	21	30	8	39	7	40	6	36	8	38	6	**36,6**	**7**
	CV3_6.bay	22	39	6	40	5	37	6	37	7	34	6	**37,4**	**6**
	CV3_7.bay	24	41	6	33	8	38	6	40	8	42	7	**38,8**	**7**
	CV3_8.bay	22	32	7	40	5	41	7	33	4	36	6	**36,4**	**5,8**
	CV3_9.bay	23	42	6	31	5	36	7	39	5	41	4	**37,8**	**5,4**
	CV3_10.bay	20	37	4	37	5	39	3	30	4	39	2	**36,4**	**3,6**
CV4	**CV4_1.bay**	26	43	9	40	10	43	10	41	10	37	9	**40,8**	**9,6**
	CV4_2.bay	30	45	11	49	11	41	13	44	13	42	11	**44,2**	**11,8**
	CV4_3.bay	33	47	12	47	12	52	13	48	14	44	13	**47,6**	**12,8**
	CV4_4.bay	32	48	11	41	16	49	13	46	15	32	16	**43,2**	**14,2**
	CV4_5.bay	32	47	9	51	10	49	13	41	14	49	6	**47,4**	**10,4**
	CV4_6.bay	33	50	14	47	12	42	14	48	11	44	10	**46,2**	**12,2**
	CV4_7.bay	27	43	7	45	9	45	11	33	9	32	10	**39,6**	**9,2**
	CV4_8.bay	28	46	8	45	10	43	10	40	11	42	8	**43,2**	**9,4**
	CV4_9.bay	29	42	12	42	19	37	13	47	12	38	13	**41,2**	**13,8**
	CV4_10.bay	28	40	14	46	11	41	12	44	12	47	13	**43,6**	**12,4**

Tabla 10.2 Ejecuciones CV3 y CV4.

carpeta	Nombre bahía	LI	E1	L1	E2	L2	E3	L3	E4	L4	E5	L5	PE	PL
LC1	**lc1.bay**	14	19	0	18	0	18	0	19	0	17	0	**18,2**	**0**

Tabla 10.3 Ejecuciones LC1.

carpeta	Nombre bahía	LI	E1	L1	E2	L2	E3	L3	E4	L4	E5	L5	PE	PL
LC2a	lc2a_1.bay	21	36	0	38	0	36	0	31	2	29	0	**34**	**0,4**
	lc2a_2.bay	21	32	1	36	0	35	0	29	0	30	0	**32,4**	**0,2**
	lc2a_3.bay	21	30	0	27	0	30	0	30	0	30	0	**29,4**	**0**
	lc2a_4.bay	22	36	0	32	0	34	0	34	0	34	0	**34**	**0**
	lc2a_5.bay	24	34	0	32	0	30	0	32	0	32	0	**32**	**0**
	lc2a_6.bay	22	35	0	37	0	33	0	35	0	34	0	**34,8**	**0**
	lc2a_7.bay	23	31	0	29	0	31	0	32	0	31	0	**30,8**	**0**
	lc2a_8.bay	20	25	0	25	0	24	0	25	0	25	0	**24,8**	**0**
	lc2a_9.bay	19	25	0	28	0	27	0	26	0	27	0	**26,6**	**0**
	lc2a_10.bay	23	28	0	28	0	27	0	28	0	27	0	**27,6**	**0**

Tabla 10.4 Ejecuciones LC2a.

carpeta	Nombre bahía	LI	E1	L1	E2	L2	E3	L3	E4	L4	E5	L5	PE	PL
BF1	cpmp_16_5_48_10_29_1.bay	29	30	0	30	0	31	0	31	0	30	0	**30,4**	**0**
	cpmp_16_5_48_10_29_2.bay	29	30	0	30	0	30	0	30	0	30	0	**30**	**0**
	cpmp_16_5_48_10_29_3.bay	29	32	0	32	0	32	0	32	0	32	0	**32**	**0**
	cpmp_16_5_48_10_29_4.bay	29	31	0	30	0	31	0	30	0	31	0	**30,6**	**0**
	cpmp_16_5_48_10_29_5.bay	29	30	0	31	0	31	0	30	0	30	0	**30,4**	**0**
	cpmp_16_5_48_10_29_6.bay	29	31	0	31	0	31	0	31	0	31	0	**31**	**0**
	cpmp_16_5_48_10_29_7.bay	29	30	0	30	0	30	0	30	0	30	0	**30**	**0**
	cpmp_16_5_48_10_29_8.bay	29	30	0	30	0	30	0	30	0	30	0	**30**	**0**
	cpmp_16_5_48_10_29_9.bay	29	32	0	31	0	32	0	32	0	32	0	**31,8**	**0**
	cpmp_16_5_48_10_29_10.bay	29	30	0	30	0	30	0	30	0	30	0	**30**	**0**
	cpmp_16_5_48_10_29_11.bay	30	33	0	33	0	34	0	33	0	34	0	**33,4**	**0**
	cpmp_16_5_48_10_29_12.bay	29	31	0	31	0	31	0	31	0	31	0	**31**	**0**
	cpmp_16_5_48_10_29_13.bay	29	29	0	29	0	29	0	29	0	29	0	**29**	**0**
	cpmp_16_5_48_10_29_14.bay	29	30	0	31	0	31	0	31	0	31	0	**30,8**	**0**
	cpmp_16_5_48_10_29_15.bay	29	32	0	31	0	32	0	33	0	32	0	**32**	**0**
	cpmp_16_5_48_10_29_16.bay	29	30	0	31	0	30	0	31	0	30	0	**30,4**	**0**
	cpmp_16_5_48_10_29_17.bay	29	29	0	30	0	29	0	30	0	29	0	**29,4**	**0**
	cpmp_16_5_48_10_29_18.bay	29	30	0	31	0	31	0	31	0	31	0	**30,8**	**0**
	cpmp_16_5_48_10_29_19.bay	29	31	0	31	0	31	0	31	0	31	0	**31**	**0**
	cpmp_16_5_48_10_29_20.bay	29	32	0	31	0	31	0	32	0	32	0	**31,6**	**0**

Tabla 10.5 Ejecuciones BF1.

carpeta	Nombre bahía	LI	E1	L1	E2	L2	E3	L3	E4	L4	E5	L5	PE	PL
LC2b	**LC2b_1.bay**	37	52	0	51	0	52	2	48	0	52	0	**51**	**0,4**
	LC2b_2.bay	36	46	0	46	0	50	0	45	0	49	0	**47,2**	**0**
	LC2b_3.bay	39	50	0	52	0	51	0	49	0	52	0	**50,8**	**0**
	LC2b_4.bay	35	48	0	51	0	50	0	51	0	50	0	**50**	**0**
	LC2b_5.bay	39	52	0	53	0	53	0	53	0	54	0	**53**	**0**
	LC2b_6.bay	40	49	0	49	0	46	0	49	0	50	0	**48,6**	**0**
	LC2b_7.bay	37	43	0	43	0	46	0	45	0	46	0	**44,6**	**0**
	LC2b_8.bay	35	42	0	41	0	45	0	45	0	42	0	**43**	**0**
	LC2b_9.bay	35	42	0	46	0	53	0	46	0	50	0	**47,4**	**0**
	LC2b_10.bay	39	48	0	47	0	48	0	49	0	47	0	**47,8**	**0**

Tabla 10.6 Ejecuciones LC2b.