Universidad Autónoma del Estado de Hidalgo

Instituto de Ciencias Básicas e Ingeniería

Área Académica de Ingeniería y Arquitectura

Implementación del algoritmo de Optimización del Búfalo Africano en el problema de corte en una dimensión para minimizar el desperdicio de material

T E S I S

PARA OBTENER EL TÍTULO DE
Doctor en Ciencias en Ingeniería, con Énfasis en Análisis y Modelación de Sistemas
P R E S E N T A :
M. en C. Leonardo Javier Montiel Arrieta

DIRECTOR DE TESIS:
Dr. Irving Barragán Vite

CODIRECTOR DE TESIS:
Dr. Norberto Hernández Romero

Febrero de 2024

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO
Instituto de Ciencias Básicas e Ingeniería
School of Engineering and Basic Sciences
Área Académica de Ingeniería y Arquitectura
Department of Engineering and Architecture

Número de control: ICBI-AAIyA/292/2024
Asunto: Autorización de impresión

MTRA. OJUKY DEL ROCÍO ISLAS MALDONADO
DIRECTORA DE ADMINISTRACIÓN ESCOLAR
PRESENTE.

El Comité Tutorial de **Tesis** del programa educativo de posgrado titulado **"Implementación del algoritmo de Optimización del Búfalo Africano en el problema de corte en una dimensión para minimizar el desperdicio de material"**, realizado por el sustentante **Leonardo Javier Montiel Arrieta** con **número de cuenta 450519** perteneciente al programa de **Doctorado en Ciencias en Ingeniería, con Énfasis en Análisis y Modelación de Sistemas**; una vez que ha revisado, analizado y evaluado el documento recepcional de acuerdo a lo estipulado en el Artículo 110 del Reglamento de Estudios de Posgrado, tiene a bien extender la presente:

AUTORIZACIÓN DE IMPRESIÓN

Por lo que el sustentante deberá cumplir los requisitos del Reglamento de Estudios de Posgrado y con lo establecido en el proceso de grado vigente.

Atentamente
"Amor, Orden y Progreso"
Pachuca, Hidalgo a 23 de febrero de 2024

El Comité Tutorial

Dr. Irving Barragán Vite
Director

Dr. Norberto Hernández Romero
Codirector

Dr. Manuel González Hernández
Miembro del comité

Dr. Juan Carlos Seck Tuoh Mora
Miembro del comité

Ciudad del Conocimiento
Carretera Pachuca-Tulancingo km 4.5 Colonia Carboneras, Mineral de la Reforma, Hidalgo, México. C.P. 42184
Teléfono: +52 (771) 71 720 00 ext. 4000, 4001
Fax 2109
aai_icbi@uaeh.edu.mx

Agradecimientos

Quiero expresar mi gratitud a mi familia que me ha acompañado durante la realización de mi doctorado, sobre todo en los momentos más difíciles que tuve tanto personales como académicos, ayudándome a centrarme para cumplir con mis metas. Asimismo, el agradecer a mi asesor el Dr. Irving Barragán Vite el que me haya orientado en el desarrollo de esta investigación. De igual forma, destacando el apoyo brindado por parte de mis catedráticos y asesor en relación con la enfermedad de mi padre. También quiero dar las gracias a mi comité tutorial por haberme brindado su tiempo cada que lo necesite, así como sus comentarios realizados durante el desarrollo de esta investigación.

Resumen

La problemática de corte unidimensional consta de la obtención de un conjunto de ítems con diferentes longitudes de un stock que puede contemplar un solo o varios tipos de objetos largos. Pertenece al conjunto de problemas de optimización combinatoria, así como a los problemas de corte y empaque. El principal objetivo a alcanzar es el minimizar el desperdicio que se genera derivado de la aplicación de los cortes sobre el stock. De igual forma, ésta problemática de corte se puede encontrar en diferentes industrias como el vidrio, acero, madera, entre otras más. A lo largo del tiempo se han desarrollado diversos enfoques, algunos métodos emplearon procedimientos exactos, mientras que otros se han centrado en desarrollar algoritmos basados en técnicas metaheurísticas como los algoritmos genéticos o métodos inspirados en inteligencia de enjambre como la optimización colonia de hormigas. El algoritmo de optimización del búfalo africano es una técnica metaheurística que pertenece a los métodos de inteligencia de enjambre, el cual ha sido empleado para resolver problemáticas de optimización combinatoria como el problema del agente viajero y el problema de embalaje de contenedores, éste último pertenece a los problemas de corte y empaque. En este trabajo se ha propuesto un enfoque basado en el algoritmo de optimización del búfalo africano para resolver el problema de corte unidimensional, buscando minimizar el desperdicio bajo la consideración de un solo stock, asimismo el minimizar el desperdicio y la cantidad de stocks con desperdicio bajo el escenario de varios stocks. Implementando un método para generar un nuevo líder a partir de la manada actual después de cierto número de iteraciones. Los resultados muestran que el método desarrollado es lo suficientemente competente en la minimización del desperdicio, sobresaliendo en la minimización del stock en el caso de emplear varios tipos de stocks.

Abstract

The one-dimensional cutting problem consists of obtaining a set of items with different lengths from a stock that can include a single or several types of long objects. It belongs to the set of combinatorial optimization problems, as well as the cutting and packing problems. The main objective is minimize the waste generated from the application of cuts to the stock. Likewise, this cutting problem can be found in different industries such as glass, steel, wood, among others. Various approaches have been developed over time, some methods used exact procedures, while others have focused on developing algorithms based on metaheuristic techniques such as genetic algorithms or methods inspired by swarm intelligence such as ant colony optimization. The African buffalo optimization algorithm is a metaheuristic technique that belongs to swarm intelligence methods, which has been used to solve combinatorial optimization problems such as the traveling agent problem and the container packing problem, where the last one belonging to cutting and packaging problems. In this work, an approach based on the African buffalo optimization algorithm has been proposed to solve the one-dimensional cutting problem, seeking to minimize waste considering a single stock, as well as minimizing waste and the number of stocks with low waste for the multi-stock scenario. Develovping a method to generate a new leader from the current pack after a certain number of iterations. The results show that the presented method is sufficiently competent in minimizing waste. While in the case of multiple stocks the algorithm based on African buffalo optimization proves to be the most efficient method in using the stock.

Contenido

Lista de Figuras

Lista de Tablas

Introducción

El One-Dimensional Cutting Stock Problem (1D-CSP) es una variante de los problemas de corte y empaque que fue abordada por Kantorovich en [5] en el año de 1939. Posteriormente, fue tratada por Gilmore y Gomory en [6] y [7]. El 1D-CSP es catalogado como un problema NP-hard como se menciona en [8] y [9], debido a que es computacionalmente intratable encontrar una solución adecuada en un tiempo razonable.

De acuerdo a [10] la versión clásica del 1D-CSP se clasifica como un problema unidimensional con un suministro de objetos, también llamados como stocks, del mismo o diferente tamaño, junto con un número determinado de órdenes de ítems regulares de diferentes longitudes que deben satisfacerse.

Asimismo, el 1D-CSP con un solo stock se presenta en diversas industrias, por ejemplo, en la construcción naval [11], en la industria del aluminio [12] y [13], en el moldeado de goma [14], la relacionada con la madera [15], entre otras más. De igual manera, se han encontrado trabajos del 1D-CSP que han considerado emplear múltiples stocks como en los cortes de las vigas estructurales [16] y en el molino de barras [17].

El principal objetivo que han buscado alcanzar diversos trabajos de investigación para resolver el 1D-CSP ha sido la minimización del desperdicio como en [18] y [15]. Mientras que otros investigadores se han centrado en minimizar el stock, el cual es la cantidad de la materia prima empleada para obtener los objetos pequeños o ítems, como en [6], [7], [19], entre otros más. También se han presentado trabajos que han buscado minimizar tanto el desperdicio como el stock de manera simultánea como en [20] y [1]. Mientras que al considerar un suministro de stocks con diferentes longitudes se han realizado trabajos de investigación centrados en minimizar el desperdicio como en [21], [22] y [23]. Por otro lado, algunos enfoques han tratado de minimizar el número de stocks necesario para cumplir con la demanda de ítems, como en [16], [24] y [17].

A lo largo de los años se han desarrollado diversos métodos para resolver el 1D-CSP, algunos se han basado en la programación lineal como en [5] y [6]. Otros han realizado enfoques basados en algoritmos metaheurísticos como el genetic algorithm (GA) tal como se describe en [2] y [25]. Mientras que algunos más se han basado en el ant colony optimization (ACO) como en [26] y [19]. Asimismo, se han desarrollado métodos híbridos como en [26] donde combinan el ACO con un método de búsqueda local.

En relación con los enfoques que se han desarrollado para resolver el 1D-CSP con múltiples stocks, algunos han usado el ACO como en [21] y [17]. Otros más han empleado el GA como en [2], [16] y [24]. De igual forma, se han presentado enfoques basados en procedimientos heurísticos como en [27], [28] y en [29].

El african buffalo optimization (ABO), es un algoritmo metaheurístico perteneciente

a los algoritmos de enjambre como el particle swarm optimization (PSO). Ambos algoritmos contemplan guardar la mejor partícula en el caso del PSO y el mejor búfalo en el caso del ABO; así como la mejor ubicación de cada partícula en el PSO y la mejor ubicación de cada búfalo en el ABO, para emplearlas en el proceso de actualización de la velocidad de cada partícula en el caso del PSO y la evaluación del fitness de cada búfalo en el ABO.

Ahora bien, en relación con el algoritmo del ABO, éste se presentó en [30] tratando de resolver la problemática del agente viajero, así como en [31] y [32]. Mientras que en [33] se empleó para resolver la problemática de one dimensional bin packing problem, la cual es una de las problemáticas de corte y empaque. Los problemas del agente viajero y del one dimensional bin packing problem son del tipo de optimización combinatoria, en las cuales el ABO obtuvo resultados favorables. Destacando el hecho que el 1D-CSP es una problemática de optimización combinatoria, esto nos motivó a desarrollar la presente investigación para comprobar si el ABO puede ser una alternativa metaheurística viable para minimizar el desperdicio en el 1D-CSP con un suministro de un solo stock, así como el minimizar el desperdicio y el número de stocks con desperdicio en el 1D-CSP con múltiples stocks.

Planteamiento del Problema

En los últimos años, el 1D-CSP ha ganado la atención alrededor del mundo en industrias como la del acero, textil y del aluminio [8]. Así también, éste es uno de los problemas de optimización combinatoria más estudiados en la literatura [34].

De igual forma, desde una perspectiva algorítmica son problemas bastante interesantes debido a su complejidad, por lo que el desarrollo de nuevos métodos exactos, heurísticos o inclusive metaheurísticos representa un área sumamente fértil para la investigación [34]. A consecuencia de lo anterior, es necesario encontrar enfoques que puedan reducir el desperdicio del material, beneficiando así a las industrias y al medio ambiente. El 1D-CSP tiene una mayor complejidad debido al número de variables que se pueden llegar a considerar como se menciona en [6], donde las variables más representativas son los propios ítems a obtener del stock, pudiendo obtener una cantidad grande de diferentes combinaciones de los ítems para formar patrones de corte; añadiendo ciertas restricciones como lo puede ser el número limitado de stocks o las diferentes longitudes que puedan tener los mismos.

Aunado a lo anterior, el incremento de la población y un creciente estándar del estilo de vida para las actuales y futuras generaciones solamente pueden ser alcanzados a través de un crecimiento económico sustentable [35]. Lo anterior motiva a las empresas que busquen a implementar nuevos métodos para reducir el uso de materia prima, con lo cual se puede obtener un beneficio económico y sostenible a la vez.

Se han propuesto diversos enfoques heurísticos, metaheurísticos e híbridos para obtener soluciones óptimas relacionadas con el 1D-CSP. Algunos de estos enfoques

han empleado programación lineal como en [5], el GA como en [2] o realizando hibridaciones como se aprecia en [19].

Destacando el hecho que los algoritmos metaheurísticos buscan encontrar una solución cercana a la solución óptima, empleando de manera razonable recursos computacionales como el tiempo y la memoria, como se menciona en [36].

A consecuencia de lo antes mencionado, se propuso desarrollar dos algoritmos basados en el ABO, el cual es un algoritmo metaheurístico que ha sido recientemente introducido para abordar problemas del tipo combinatorio como el agente viajero en [30], [31] y [32]; y el bin packing problem en [33], obteniendo mejores resultados en comparación con otros enfoques. Además de ser el primer trabajo de investigación en el cual se busque el proponer al ABO como una alternativa para buscar minimizar el desperdicio en el 1D-CSP con un solo stock, así como el desperdicio y el número de stocks con desperdicio en el 1D-CSP con múltiples stocks.

Preguntas de Investigación

- ¿El algoritmo ABO es capaz de obtener soluciones adecuadas para el 1D-CSP con diferentes instancias de prueba?
- ¿Contra cuáles enfoques el ABO obtiene mejores soluciones para resolver el 1D-CSP?
- ¿Cuál es la mejor combinación de parámetros para obtener las mejores soluciones?

- ¿Cuáles son las instancias de prueba con las que el ABO brinda mejores soluciones?

Objetivos

Objetivo General

Implementar, parametrizar y evaluar el desempeño del algoritmo de optimización del búfalo africano en el problema de corte en una dimensión para minimizar el desperdicio total en instancias de prueba de diferente complejidad.

Objetivos Específicos

- Determinar una forma de representar las soluciones del 1D-CSP en el ABO.
- Definir la forma en que se evaluará cada solución con la función objetivo.
- Determinar el banco de problemas para realizar las pruebas experimentales.
- Diseñar el algoritmo basado en el ABO para definir los procedimientos requeridos en el proceso de traslado a código fuente.
- Implementar el ABO en un lenguaje de programación.
- Establecer la mejor combinación de valores de los parámetros del algoritmo para obtener las mejores soluciones en las diferentes instancias de prueba.

- Realizar una comparación del ABO contra otros enfoques para contrastar la calidad de las soluciones obtenidas por el ABO.

Justificación

En los últimos años, el 1D-CSP ha tomado mayor importancia alrededor del mundo, como se hace mención en [8]. Con lo cual se han propuesto diversos métodos para generar soluciones que maximicen la reducción de desperdicio.

Así mismo, en [35] se alude que actualmente se han establecido acciones de prevención de residuos, predominando más aquellos del tipo unidimensional. Lo cual impulsa a que se planteen mejores métodos para resolver el 1D-CSP.

De igual forma, en [37] se menciona que las nuevas soluciones relacionadas con el 1D-CSP no solamente ahorran costos a las compañías, sino que también pueden reducir la emisión de gases.

El 1DCSP es un problema de tipo NP-hard como se establece en [7] y [10], lo que quiere decir que hallar una solución óptima requiere de un gran esfuerzo computacional, por lo que se necesita investigar nuevos métodos que ayuden a minimizar dicho esfuerzo.

En lo que se refiere al ABO, éste ha sido utilizado para resolver diversos problemas de optimización combinatoria, como el problema del agente viajero en [30], [31] y [32]. De igual manera se empleó en la problemática del bin packing problem en [33], el cual es una problemática de corte y empaque, donde obtuvo buenos resultados.

Dentro de las problemáticas del agente viajero y del bin packing problem el ABO requirió de una menor cantidad de tiempo en comparación contra otros enfoques metaheurísticos como el GA o el ACO. Esto es a consecuencia de los pocos procedimientos y parámetros necesarios para poder implementarlo, sin perder la robustez y efectividad. Asimismo, el ABO busca explotar y explorar el espacio, así como el evitar la prematura convergencia o estancamiento en una solución local [30].

Dentro de [33] se muestra que la implementación del ABO ha generado soluciones óptimas con un número inferior de compartimientos que otros enfoques como el best fit decreasing (BFD), first fit decreasing (FFD), ant system (AS), entre otros más.

Por todo lo antes mencionado, se considera que el ABO podría ser una opción igual o mejor que otros enfoques que han tratado de resolver el problema del 1D-CSP con un solo stock o múltiples stocks.

Hipótesis

El ABO es un algoritmo metaheurístico capaz de generar soluciones con el menor desperdicio en el problema del 1D-CSP en instancias de prueba de diferente complejidad.

Organización de la Tesis

El presente trabajo de tesis se ha organizado de la siguiente manera:

Capítulo 1. Estado del Arte y Descripción del 1D-CSP, se presenta la

descripción del 1D-CSP y algunos de los trabajos más importantes que han buscado resolver esta problemática de corte considerando un solo stock y múltiples stocks.

Capítulo 2. Algoritmo de Optimización del Búfalo Africano, se describirá el ABO y la forma en que se representan las soluciones en el 1D-CSP con un solo tipo de stock y con varios stocks. De igual forma se presentará el método Ranked Order Value (ROV) empleado para discretizar las soluciones.

Capítulo 3. Algoritmo ABO-1DCSP, se estará presentando el algoritmo ABO-1DCSP, propuesto para resolver el 1D-CSP considerando un solo stock. Empezando por la función objetivo utilizada para evaluar el desperdicio que genera cada búfalo. Posteriormente se describirá el algoritmo ABO-1DCSP. Por último se presentarán los resultados obtenidos del ABO-1DCSP con las instancias.

Capítulo 4. Algoritmo ABO-1DCSPMS, se expondrá el enfoque ABO-1DCSPMS, el cual busca resolver el 1D-CSP con múltiples stocks. Primero se presentará la función objetivo que considera el desperdicio total y la cantidad de stocks con desperdicio en cada búfalo. Posteriormente se mostrará el algoritmo realizado, así como los resultados obtenidos.

Capítulo 5. Conclusiones, se presentarán las deducciones a las que se llegó para resolver el 1D-CSP con un solo y múltiples stocks mediante métodos basados en el algoritmos ABO.

Capítulo 1

Estado del Arte y Descripción del 1D-CSP

El 1D-CSP es abordado por vez primera en [5], donde se planteaba el problema del desperdicio generado de los cortes aplicados sobre la materia prima en las fábricas como algo importante a tratar derivado de las normas a cumplir por las empresas. En dicho trabajo se presentaban una de las características importantes a considerar en el 1D-CSP, la cual es el tipo de stock con la que se disponía, pudiendo ser de un solo o varios tipos. Asimismo, no solo se consideraban longitudes meramente enteras tanto para los ítems, que son los objetos pequeños a obtener de los stocks, sino que podían ser fraccionarias. Con relación a los ítems, estos se presentaban de diversas longitudes.

De igual forma, el 1D-CSP se le puede catalogar como un problema de optimización combinatoria de acuerdo a [38] debido a que se busca obtener un set, que

es una solución, de elementos que pueden ser enteros o no. Así mismo, el 1D-CSP es clasificado como una problemática NP-hard de acuerdo a [5] y [7] debido a que no existe un algoritmo de tiempo polinomial que pueda resolverla [39]. Con base en lo anterior, al emplear algunos métodos pueden llegar a requerir un tiempo de cálculo exponencial. Debido a lo anterior, el utilizar métodos aproximados para resolver las problemáticas de optimización combinatoria ha tenido mayor interés, donde se sacrifica la garantía de encontrar soluciones óptimas por soluciones adecuadas en una cantidad de tiempo menor [40].

De acuerdo a [41] el 1D-CSP contempla dos elementos básicos, uno son stocks y el segundo son los objetos pequeños o ítems que se van a obtener del stock. Además, define las siguientes características: La primera está relacionada con la dimensión, la cual es de una sola, debido a que solo se consideran realizar cortes en una sola dimensión. La segunda característica va enfocada hacia el tipo de tarea, ésta puede estar orientada hacia una maximización del stock, debido a que no se cuenta con una suficiente cantidad de stock necesario para cumplir con la demanda de ítems, o bien se puede buscar la minimización del stock donde se tiene la cantidad suficiente de stocks para obtener los ítems. En las instancias empleadas en este trabajo de investigación se considera una cantidad ilimitada de stock. La tercera característica se refiere a la variedad de ítems que se tengan considerados, pudiendo tener ítems con poca o amplia heterogeneidad, o en el último caso tener un mismo tipo de ítems. En las instancias empleadas dentro de este trabajo se considera una amplia variedad heterogénea de ítems. La cuarta característica va enfocada hacia el tipo de stocks usados para cada instancia, pudiendo tener un solo objeto largo o varios objetos largos con una misma,

poca o amplia variedad de longitudes. En nuestras instancias se consideran varios objetos largos de una o diferentes longitudes.

De manera resumida podemos establecer que la tipología del 1D-CSP que se va a tratar de resolver en esta investigación es aquella que es unidimensional, con una cantidad de stocks ilimitados de una sola longitud y múltiples longitudes, de los cuales se van a obtener una amplia variedad heterogénea de ítems.

De acuerdo a lo mencionado en [41] la tarea dentro del 1D-CSP se puede centrar en la maximización o minimización del stock; sin embargo, en este trabajo en el caso del 1D-CSP con un solo stock nos centramos en minimizar el desperdicio así como en [36] y [15]. La búsqueda de la minimización del desperdicio es uno de los principales objetivos en el 1D-CSP, el cual se genera de la aplicación de diferentes patrones de corte, los cuales son conjuntos de ítems obtenidos del stock, al término de la aplicación de los patrones de corte se genera un desperdicio total derivado de la acumulación de las diferentes cantidades de desperdicio que se generen de los patrones de corte. Mientras que en el caso del 1D-CSP con múltiples stocks nos enfocamos en minimizar el desperdicio total y el número de stocks con desperdicio como se aprecia en [1].

Para explicar con mayor detalle el 1D-CSP se utilizará la Figura 1.1, donde se puede observar una instancia de ejemplo que considera un solo tipo de stock con una cantidad ilimitada y cuatro tipos de ítems con diferentes longitudes pero con una misma frecuencia.

Lo que se busca en el 1D-CSP es encontrar patrones de corte, que son en sí arreglos de ítems, de los cuales se obtenga la menor cantidad de desperdicio posible. Tal como se aprecia en la Figura 1.1 se presentan tres diferentes soluciones que están formadas

por distintos patrones de corte. Para determinar la cantidad de desperdicio total que se obtiene de cada solución hay que sumar de izquierda a derecha la longitud de cada ítem en cada patrón de corte para conseguir la longitud total del patrón de corte, la cual será restada a la longitud del stock, consiguiendo así el desperdicio de cada patrón de corte. Finalmente, solo hay que sumar todos los desperdicios de los patrones de corte para así medir el desperdicio total de cada solución.

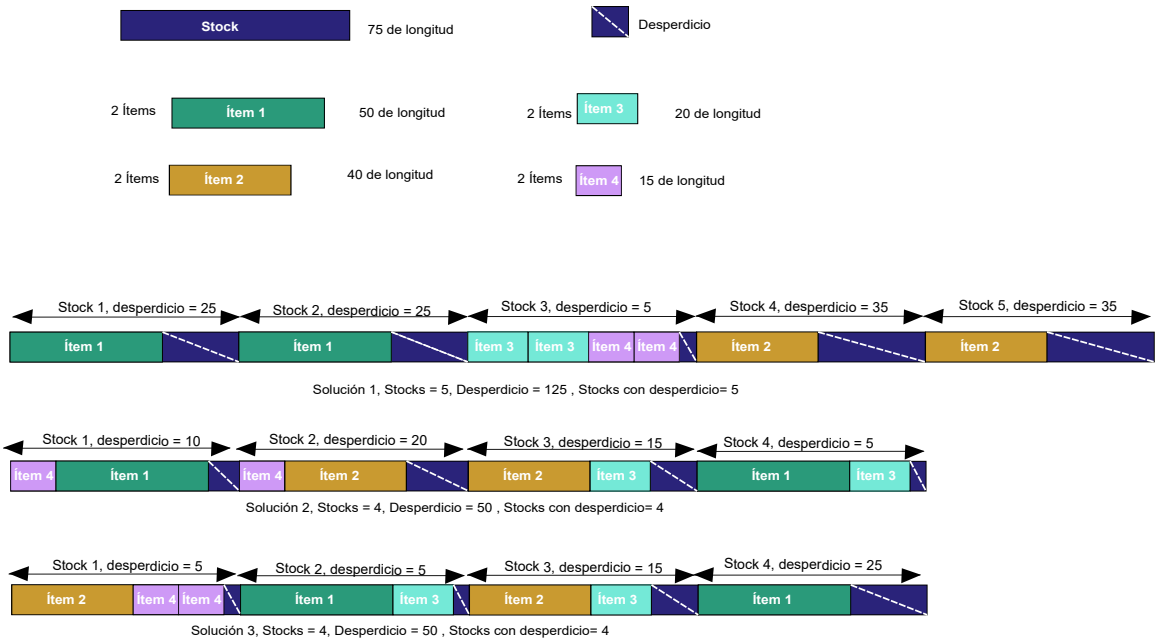


Figura 1.1: Explicación de la formación de patrones de corte en el 1D-CSP.

1.0.1 Modelo matemático del 1D-CSP

El 1D-CSP fue propuesto en 1939 por Kantorovich [5], para después ser abordado por Gilmore y Gomory en [6, 7]. Esta variante de corte se puede describir como un

problema estándar de cortes longitudinales en donde el material es cortado en varios tamaños específicos [8].

Uno de los primeros modelos propuestos del 1D-CSP fue desarrollado Gilmore y Gomory como se aprecia en [42], donde se enfoca a la minimización del stock. Sin embargo, en este modelo es permitida la sobreproducción.

Algunos otros trabajos como en [19] y [25] han propuesto un modelo que busca la minimización del costo del desperdicio.

De igual forma, se han desarrollado modelos que buscan minimizar el desperdicio del material, el cual es análogo a la minimización de la manufacturación del costo del material. Aunado a lo anterior aparece en muchos dominios industriales como el del metal, papel, textil y la madera [43].

En [36] el 1D-CSP es modelado de la siguiente manera buscando minimizar el desperdicio:

$$min \sum_{i=1}^{n} \sum_{j=1}^{m} (d_j - x_{ij} s_i) \tag{1.1}$$

$$\sum_{j=1}^{m} x_{ij} = n_i, \ i = 1, ..., n \tag{1.2}$$

Donde:

- $n =$ es el total de ítems
- $m =$ es el total del stock
- $i =$ es él i-ésimo ítem

- j = es él j-ésimo stock
- d_j = es la longitud del stock j
- s_i = es la longitud del ítem i
- n_i = es el número total de órdenes
- x_{ij} = es el número de ítems con longitud s_i que se obtendrán del stock j

La función objetivo en la Ecuación 1.1 contabiliza el desperdicio total obtenido de todos los stocks m necesarios para satisfacer la demanda total de los ítems de acuerdo la restricción en la Ecuación 1.2.

A continuación se presentan algunos de los trabajos más relevantes que han buscado resolver el 1D-CSP con los casos de un solo stock y múltiples stocks, empleando métodos heurísticos como la técnica de generación de columnas, técnicas metaheurísticas como los algoritmos genéticos o desarrollando un modelo híbrido con la combinación de un algún enfoque metaheurístico con un algoritmo de búsqueda local.

1.0.2 Heurísticos

Las técnicas heurísticas son métodos de búsqueda directa, las cuales emplean diversas reglas y procedimientos con el fin de encontrar soluciones adecuadas [44].

Los métodos heurísticos están diseñados para buscar soluciones cercanas a problemas del tipo combinatorio complejos, como lo es el 1D-CSP, debido a que no se pueden resolver con el uso de algoritmos de optimización disponibles, como lo puede ser una búsqueda exhaustiva [44].

Una de las ventajas de las técnicas o métodos heurísticos es la capacidad de conseguir soluciones de manera rápida, logrando llegar a obtener soluciones adecuadas frente a la problemática que se busca resolver. No obstante, en algunas ocasiones la calidad de las soluciones en relación con la óptima puede no conocerse. De igual manera que puedan llegar a caer en óptimos locales [44].

El primer método que buscó resolver el 1D-CSP considerando tanto el caso de un stock como el de múltiples stocks fue el desarrollado por Kantorovich en [5], implementando un algoritmo exacto basado en programación lineal (LP por sus siglas en inglés). En dicho método se proponía la generación de diversos patrones de corte con el uso de la LP para posteriormente seleccionar aquella solución que produjera una menor cantidad. Permitiendo la posibilidad de producir ítems de más.

Gilmore y Gomory en [6] y [7] emplearon un método de programación lineal para afrontar el 1D-CSP considerando un solo stock y múltiples stocks, en donde introdujeron una ingeniosa técnica de generación de columnas para crear los patrones de corte y resolver el problema de optimización de corte [45].

Gradisăr en [46] presentó un procedimiento heurístico para optimizar el 1D-CSP cuando todos los stocks tienen una longitud diferente.

Dentro de [47] emplean el método de exact cutting plane junto con el de generación de columna para minimizar el costo total de los stocks requeridos para cumplir con la demanda de ítems.

Dikili en [48] desarrolló un método basado en un "nesting algorithm", el cual reduce el número de patrones de corte, lo que contribuye al ahorro de tiempo y material. Con lo anterior, la reducción de desperdicio y el uso del stock son maximizados.

Tres versiones del algoritmo heurístico greedy rounding residual fueron presentadas en [49] para el caso del 1D-CSP con múltiples stocks. Emplean diferentes formas de ordenar los ítems. En la primera versión dan una mayor preferencia orientada a los arreglos de ítems más requeridos. En la segunda versión se enfocan en aquellos arreglos de ítems con menor desperdicio. Para la tercera versión consideran aquellos arreglos de ítems que tendrán una mayor relevancia.

En [50] proponen un procedimiento secuencial heurístico para generar planes de corte. Su principal objetivo es la reducción del coste de las barras de material, mientras que como objetivos secundarios se establecieron la reducción del patrón y del stock. Su punto de partida es la técnica de generación de columna de Gilmore y Gomory.

En [27] consideran un flujo de stocks ilimitado para el caso del 1D-CSP con múltiples stocks. Su función objetivo busca minimizar la frecuencia de los patrones de corte. Establecieron tres tipos de recursos. En el primero, denominado público, contiene los ítems, las longitudes de los ítems, los tipos de stocks, y otros datos. En el segundo tipo lo llamaron privado, el cual está relacionado con la frecuencia de los patrones de corte, los ítems usados, la demanda de ítems y otra información. El tercer tipo está asociado con la instancia de ítems y la mejor solución obtenida.

Cui en [51] buscan resolver la variante del 1D-CSP orientada a la reducción de patrones de corte implementando una heurística de dos fases, en la primera se busca obtener el conjunto de set patrones de corte, mientras que en la segunda fase se utiliza el optimizador CPLEX para resolver el modelo de programación lineal entera. En este trabajo intentan minimizar el total del material y los costos de maquinación

para satisfacer la demanda de ítems.

Dentro de [28] desarrollaron un algoritmo heurístico que implementa dos fases para el 1D-CSP con múltiples stocks. En la primera fase se crean los patrones de corte y en la segunda fase usan programación lineal entera para obtener una eficiente solución. Buscan minimizar el número de stocks.

En [52] emplean un enfoque heurístico basado en la generación de patrones. Dándole importancia a la secuencia en la que se disponen los ítems, debido a que entre cada corte de cada ítem existe una perdida de corte. De igual forma, consideran el hecho de obtener sobrantes más largos con el propósito de emplearlos a futuro. Su modelo matemático se basa en la minimización de los patrones usados y la maximización de los "squared leftovers".

Ogunranti en [15] utilizan LP para seleccionar los mejores patrones de corte que permitan minimizar el desperdicio basándose en la frecuencia total de los propios patrones de corte. En sus pruebas consideran un solo tipo de placas de madera.

Un modelo matemático fue diseñado en [22] junto con la aplicación de un modelo arc flow para minimizar el desperdicio. Dentro de su propuesta igualmente incluyeron un procedimiento heurístico para instancias complejas, así como el mejorar el tiempo computacional.

Dentro del trabajo de [14] implementaron programación lineal entera mixta y un procedimiento de programación restringida para generar los patrones de corte en la industria del moldeado de goma. Su objetivo principal fue minimizar el desperdicio, el número de cortes necesario para satisfacer la demanda de ítems y la sobreproducción.

Dentro de [53] intentan minimizar el desperdicio total generado en los cortes de

los lingotes dentro de la industria del acero, proponen un modelo del tipo mixed-integer linear programming. En su enfoque buscan primero el despachar o cumplir con aquellas órdenes más complejas.

En [54] emplean goal programming (GP) para poder cumplir diversos objetivos o metas, debido a que GP permite abarcar múltiples metas considerando restricciones un poco más suaves. Presentan el caso de una fábrica que se dedica a producir los dispositivos llamados "frogs" que se emplean en las vías de los trenes. Buscan minimizar la escasez y los excedentes en el 1D-CSP con una demanda mixta.

Dentro de [20] desarrollaron un algoritmo heurístico de dos etapas llamado least-loss algorithm, (LLA). En donde buscan alcanzar dos objetivos, el primero es minimizar la cantidad de total de pérdida de corte y el segundo es disminuir el número de stocks parcialmente cortados. La causa del segundo objetivo es debido a que se menciona el hecho de que el stock que ha sido parcialmente cortado no se le puede obtener una ganancia al revenderse como objetos sin cortar.

Para la primera etapa los ítems son ordenados de forma descendente, posteriormente se selecciona el primer ítem, el cual es el más largo y busca aquellos ítems pequeños que añadidos al más largo generen un desperdicio menor, es decir, encuentra el mejor arreglo de ítems pequeños que produzcan la máxima ocupación sobre el objeto, lo anterior se repite hasta que todos los pequeños ítems hayan sido asignados a un ítem largo [20].

Para la segunda etapa se hace un ordenamiento como en la primera etapa y después aquel ítem que esté colocado en el medio de la secuencia es colocado al principio de la secuencia, finalizado lo anterior se lleva a cabo un reordenamiento de los ítems

pequeños de alguna de las opciones consideradas. Al final de la segunda etapa se selecciona la mejor de las dos soluciones emitidas de la etapa 1 y 2 [20].

En [29] presentan dos enfoques, el primero basado en la técnica de generación de columna y el segundo en programación dinámica. En ambas propuestas buscan minimizar el costo total considerando la configuración del patrón, el costo del tiempo de inventario y la cantidad de stocks.

En [13] realizaron un método basado en programación lineal entera, estableciendo tres funciones objetivo que contemplan la reducción de los sobrantes obtenidos de los patrones de corte aplicados sobre las barras de aluminio. Su primera función objetivo está encaminada hacia la minimización del desperdicio. Mientras que la segunda se enfoca en la minimización del stock y la tercera combina tanto la primera como la segunda.

En [55] proponen un algoritmo que contempla dos métodos, el primero denominado como "minimization by special length" considera aquel stock con longitud irregular. Entre tanto, para el segundo método nombrado como "minimization by stock length" considera un stock con longitudes regulares. El algoritmo desarrollado busca minimizar el desperdicio relacionado con las barras de refuerzo en la industria de la construcción.

Un enfoque heurístico que busca la recombinación de los patrones de corte con desperdicio fue presentado en [56] para minimizar el 1D-CSP considerando un solo stock y múltiples stocks. Su objetivo se centró en la minimización de los patrones de corte necesarios para satisfacer la demanda de ítems.

Un método basado en la heurística greedy fue desarrollado en [57], donde los ítems

son ordenados en forma decremental en conjunto con la condición de que si hay más ítems con longitud par que impares, el primero en colocarse será par, en caso contrario ocurrirá lo mismo.

Dentro de [58] diseñaron un modelo basado en programación lineal entera mixta para un caso de estudio de la industria del acero. Su objetivo fue minimizar el desperdicio, tomando en cuenta el reutilizar los sobrantes y las restricciones del inventario.

En [59] presentan un modelo matemático, donde establece el término "sustainable trim" para definir cuando un desperdicio tiene un impacto económico menos negativo en la compañía. El "sustainable trim" se emplea como limite superior para los patrones de corte. En caso de que algún patrón de corte supere el límite establecido, el patrón de corte se eliminará.

1.0.3 Metaheurísticos

El término metaheurístico fue introducido en [60], el cual está conformado por las palabras heurístico, que significa encontrar, y la palabra meta, que implica ir más allá [40].

Los enfoques metaheurísticos son un cúmulo de estrategias que guían un proceso de búsqueda. Teniendo como fin común el explorar de forma eficiente el espacio de búsqueda con el objetivo de encontrar soluciones adecuadas. Llegan a incorporar mecanismos para evitar estancarse en un área del espacio de búsqueda [40]. De igual forma, al emplear los métodos metaheurísticos es posible contrarrestar la explosión

combinatoria de las posibles soluciones que se puedan llegar a tener en un problema de optimización combinatorio complejo, tal como es el caso del 1D-CSP. Las técnicas metaheurísticas son inspiradas en analogías basadas en la física, como el simulado recocido, en la biología, como aquellos algoritmos evolutivos, o en el comportamiento de animales, como es el caso del ABO [61].

Una de las desventajas de los métodos metaheurísticos es la cantidad de parámetros que son necesarios a considerar a configurar para obtener soluciones eficientes. Cuando se emplea algún método metaheurístico con una cantidad considerable de parámetros, el proceso de ajuste se volverá más complejo. De igual forma, el emplear varios parámetros dificulta el comprender el espacio de los propios parámetros. Otro aspecto es el hecho de la interacción que se tiene entre los parámetros, provocando que se puedan tener o no soluciones de calidad [62].

En [2] desarrollaron un método basado en GA, donde cada cromosoma representa un número de grupos, mientras que cada grupo es un gen que está relacionado con un stock. Su principal objetivo fue el minimizar el desperdicio que se generaba de los patrones de corte.

En [1] proponen un algoritmo evolutivo basado en evolutionary programming (EP), enfocándose en minimizar el desperdicio y el uso del stock. Consideraron emplear no solo una longitud para los stocks, sino de múltiples longitudes, así también el aspecto de la contigüidad, implicando un espacio para almacenar los ítems que se vayan cortando hasta que se complete la lista de ítems. Asimismo, establecieron que la lista de ítems a satisfacer fuera todo un vector, donde éste a su vez sería un cromosoma, dentro del mismo se irían formando los arreglos de los ítems de acuerdo

a la longitud del stock. En el proceso de mutación proponen cambiar el orden de los ítems dentro del vector para poder generar nuevos arreglos de ítems.

Dentro de [26] emplean el ACO en su propuesta, implementando una matriz de feromonas, la cual va a contener los diferentes ítems que estarán formando los patrones de corte, buscando detectar aquellos que destaquen para ser empleados en varias ocasiones.

Dentro de [63] emplean un método basado en GA, estudiaron tres casos de un entorno real en la industria del acero, donde sus soluciones son comparadas contra aquellas que eran comúnmente empleadas.

En [64] emplean ACO, en donde cada hormiga selecciona una regla estocástica de probabilidad para elegir el tipo de corte y el patrón de corte denotado por otra regla probabilística. Con lo anterior se van reduciendo el número de cortes necesarios con cada iteración, por lo cual se irán seleccionando nuevas hormigas que contengan nuevos cortes y patrones hasta que no exista la necesidad de realizar corte alguno. Hay que mencionar que después de cierto número de iteraciones, la evaporación de feromonas se realiza para evitar llegar a respuestas similares, así como escapar del óptimo local que se pueda generar.

En el trabajo [65] se diseñó una metodología con base en tabu search (TS), introduciendo una función objetivo mixta y variante que permita obtener una solución óptima para resolver el 1D-CSP.

En [66] proponen una nueva mutación selectiva basada en la programación evolutiva para resolver el 1D-CSP con un solo stock como el de múltiples stocks, dentro de su metodología cada cromosoma padre es generado de manera aleatoria, estable-

ciendo puntos de separación entre los ítems para agruparlos mientras no se rebase la longitud del stock, estos grupos se les denomina como genes. Cada gen busca alcanzar la misma longitud del stock, en caso de lograrlo será almacenado. Ahora bien, si un padre cromosoma no tiene algún gen que se ajuste a la longitud del stock, entonces es movido para generar otro cromosoma. Para el caso de algunos padres que contengan por lo menos algún gen que se ajuste a la longitud del stock, se lleva a cabo un reordenamiento de ítems de manera aleatoria con los genes que no se ajustaron con la longitud del stock, buscando alcanzar un porcentaje de desperdicios del 30% con respecto a la longitud del stock. Solo los mejores genes se almacenan al cromosoma hijo, para después hacer una comparación entre todos los hijos y obtener al mejor de todos, guardándolo para compararlo con otras futuras generaciones de cromosomas.

En [21] proponen un algoritmo basado en ACO para minimizar el desperdicio en el caso del 1D-CSP con múltiples stocks. Implementaron una estrategia de mutación para evitar el estancamiento y la precocidad junto con un proceso de actualización con base en los resultados del proceso de mutación.

Dentro del artículo de Jahromi [9] hacen una comparativa de la eficiencia de los algoritmos metaheurísticos simulated annealing (SA) y TS. Los resultados muestran que el SA que propusieron es mejor que el TS en la búsqueda de un plan de corte junto con una reducción de desperdicio inferior, mientras que el TS tiene mejores tiempos de cómputo que el SA.

En [67, 18] proponen métodos basados en GA que usan dos genes en cada solución, el primer gen está asociado con la frecuencia del patrón de corte y el segundo con el patrón de corte. En el trabajo de [67] emplean una técnica heurística para crear

los patrones de corte. Mientras en [18] emplean dos poblaciones para obtener los individuos que conformarán la población solución, en la primera solución escogen a la mejor solución y en la segunda se seleccionan individuos de manera aleatoria, posteriormente se combinan ambas poblaciones para obtener la población solución.

En el trabajo [68] emplean el discrete particle swarm optimization (DPSO), se comparan contra los enfoques como el GA, PSO y Cuckoo. Donde obtuvieron mejores resultados en el aspecto de la efectividad de la utilización del material, reducción del desperdicio y en el tiempo de cómputo contra los enfoques antes mencionados.

En [36] hacen uso del algoritmo ACO buscando minimizar el desperdicio de corte, aplicándolo en la construcción de una estructura de acero, compuesta por perfiles y placas. Consideran que los enfoques metaheurísticos son más aptos para aplicaciones reales relacionadas con el 1D-CSP.

En [69] utilizan de igual forma el GA para resolver el 1D-CSP con objetos de múltiples longitudes, observaron que el porcentaje de material de desperdicio se mantiene por debajo del 3% mientras la demanda de ítems no incremente a más de 100.

En [37] desarrollaron dos algoritmos, uno con el greedy randomized adaptive search procedure y otro con un algoritmo evolutivo buscando reutilizar los residuos con el objetivo de volverlos a considerar como stock para obtener más ítems.

En [70] se muestra la primera implementación del ABO en el 1D-CSP. Obteniendo resultados cercanamente a los óptimos en las instancias con una menor cantidad de ítems. Mientras que en [71] se emplea una variante discreta del ABO comparando una técnica de cruza junto con la técnica del valor del orden clasificado para generar

soluciones discretas. En relación a los resultados se aprecia que la técnica de cruza es superior a las demás en relación a la calidad de las soluciones.

El trabajo [72] es el producto de esta tesis donde se implementa el ABO, proponiendo un nuevo método para generar un nuevo líder en la manada a partir de la manada actual, seleccionando los mejores patrones de corte para así conformar un nuevo mejor líder entre los búfalos. Obteniendo resultados bastante favorables en el parámetro de la minimización del desperdicio. De igual forma se pueden observar resultados favorables en relación con el parámetro de minimización del stock usado.

1.0.4 Híbridos

Los enfoques híbridos están conformados por dos o más algoritmos que pueden ser metaheurísticos, heurísticos o de búsqueda local. Como se menciona en [40], los metaheurísticos permiten que se puedan crear nuevos enfoques a partir de diferentes procedimientos de dos o más algoritmos, pudiendo llegar a ser esta integración de tres formas diferentes: 1) incluir algunos componentes de un algoritmo en otro, 2) establecer un intercambio de información entre los métodos involucrados y 3) la anexión completa de los métodos.

En [46] se propuso un enfoque combinando dos métodos: el primer método basado en LP orientado a patrones y el segundo procedimiento es un heurístico secuencial orientado a ítems. El propósito es cortar las órdenes de los ítems en exactamente el número de piezas requeridas y acumular el residuo generado secuencialmente en una pieza que posteriormente sea utilizada.

Dentro del trabajo de [73] se presenta un enfoque de solución exacta, el cual está basado en la combinación del método de "cutting plane" y la técnica de generación de columna de Gilmore y Gomory.

Dentro de [26] desarrollaron una versión híbrida del ACO con la inclusión de un algoritmo de búsqueda local, esto con el objetivo de obtener mejores soluciones. Cada solución creada por cada hormiga es tomada y llevada a una fase local de optimización. Dentro esta fase aquellos arreglos de ítems que menos abarquen la longitud del stock son eliminados, debido a esto sus ítems se catalogan como disponibles o libres. Lo siguiente es verificar si los ítems libres pueden sustituir algún ítem de aquellos arreglos de ítems que tengan desperdicio para obtener un mejor aprovechamiento del stock.

En [25] presentan el hybrid-multicromosome genetic algorithm (HMCGA), el cual tiene una variación en comparación con otros enfoques que emplean el GA para resolver el 1D-CSP. La diferencia radica en el hecho de implementar dos cromosomas, uno se relaciona con el patrón de corte y el segundo con la frecuencia que el patrón de corte es efectuado. Los padres con los mejores fitness son solo aquellos que tendrán descendencia, para el procedimiento se selecciona un patrón de corte y dos padres, acarreando la frecuencia que tiene cada uno de los patrones de corte seleccionados. El proceso de mutación se puede realizar de dos maneras, en la primera se reemplaza el patrón de corte con otro generado aleatoriamente, recalculando la frecuencia del nuevo patrón añadido; para la segunda forma se mantiene el primer cromosoma que es el patrón de corte y se recalcula el segundo cromosoma que es la frecuencia.

En el trabajo de [19] desarrollaron un método llamado modified ant colony algorithm (MACO), le dan mayor relevancia a los patrones de corte y las frecuencias de

estos, evitando considerar el orden en que los patrones de corte son seleccionados. Emplean el algoritmo de árbol de búsqueda junto con el ACO, buscando minimizar el desperdicio.

Dentro de [24] presentan un enfoque basado en los algoritmos de GA y branch and bound para el caso del 1D-CSP con múltiples stocks. Consideran que los precios de los stocks son proporcionales a sus respectivas longitudes. Tienen como objetivo el minimizar la cantidad de stocks. Implementan la operación de cruzamiento considerando un punto de selección aleatoria. Mientras que en el proceso de mutación emplean solo un padre.

Un enfoque basado en GA y clustering strategies fue desarrollado en [16] para minimizar los stocks en el caso de múltiples stocks del 1D-CSP, considerando la sobreproducción y la reutilización de remanentes. Su método consiste de tres pasos, en el primero se emplea un algoritmo GA para generar las soluciones, posteriormente se aplica otro GA para mejorar las soluciones previamente obtenidas y finalmente en el tercer paso usan cuatro métodos para refinar las soluciones del paso anterior.

Un enfoque basado en GA que busca minimizar el desperdicio, donde los individuos son creados por un método que combina la aleatoriedad y un esquema heurístico, fue propuesto en [23]. Establecen los cromosomas dando una mayor prioridad hacia aquellos stocks que no han sido empleados y posteriormente seleccionando los ítems de forma aleatoria.

En [74] proponen usar los enfoques de “simplex method” y “column generation” propuesto por Gilmore y Gomory. De igual forma, añaden un algoritmo de control para la reutilización del material de desperdicio del acero.

En el trabajo [75] desarrollaron un algoritmo matheuristic basado en la estrategia de fix-and-optimize hibrizada, incluyendo una búsqueda local aleatoria. La función objetivo de su modelo es minimizar el número de stocks y tiempo necesarios para satisfacer los pedidos de los clientes.

En nuestros métodos ABO-1DCSP y ABO-1DCSPMS basados en el ABO, se propone que cada búfalo sea una solución, la cual contendrá todos los ítems de la instancia de prueba, buscando aquella que tenga la menor cantidad de desperdicio generada de los patrones de corte obtenidos de acuerdo al acomodo de los ítems. Asimismo, se plantea el dar una mayor oportunidad al mejor búfalo de la manada de encontrar mejores soluciones, evitando la posibilidad de reiniciar la manada al término de cada iteración en caso de no actualizar al mejor búfalo de la manada. De igual manera, se presenta una forma de obtener al mejor búfalo de la manada antes de reiniciar la manada. Para lo cual se escogen los mejores patrones de corte de la manada actual que conformarán al nuevo mejor búfalo que guiará a la nueva manada.

Capítulo 2

Algoritmo de Optimización del Búfalo Africano

2.1 Conceptos Básicos del ABO

El African Buffalo Optimization (ABO) es un algoritmo propuesto por Julius Beneoluchi Odili, Mohd Nizam Mohmad Kahar y Shahid Anwar en [30] en el año 2015, el cual se basa en el comportamiento de los búfalos africanos para la búsqueda de pastizales.

El ABO pertenece a la clase de algoritmos denominada como "Swarm Intelligence", los cuales se basan en el comportamiento social de algunos animales, buscando adquirir una mejor explotación y exploración del espacio de búsqueda disponible en las problemáticas a las que se someta. De igual forma, es un algoritmo que es fácil de implementar y es rápido en la obtención de óptimos resultados, lo anterior debido

a que requiere de pocos parámetros [32].

Las tres principales características del ABO son [30]:

1. Su extensa capacidad de memoria: permite que los búfalos rastreen sus rutas a través de cientos de kilómetros dentro del paisaje africano.

2. La habilidad cooperativa de comunicarse tanto para los buenos y malos tiempos. Inclusive son de las pocas especies que son capaces de poner su vida en riesgo si alguno de su especie estuviera en peligro. Hacen uso de dos sonidos:

   (a) El sonido de "waaa", solicita a la manada que se desplace de la presente locación porque ya no es favorable, los pastizales son pobres o es peligrosa. Y en otras ocasiones es usado para solicitar a otros búfalos que ayuden a un animal en peligro.

   (b) El sonido "maaa" es indicativo de que el lugar es bueno para permanecer, ya que promete buenos pastizales y es seguro.

3. El último atributo es su naturaleza democrática de extrema inteligencia. Cuando algunos miembros de la manada se oponen a lo propuesto por la mayoría, ellos optan por la decisión que la mayoría desea, y así realizan el siguiente movimiento en conjunto.

El ABO consta de las siguientes fases [30]:

1. Se establece la función Objetivo $f(x)x = (x1, x2, .., xn)T$.

2. Se generan los búfalos aleatoriamente.

3. Se realiza la actualización del fitness de cada búfalo k con la Ecuación 2.1.

$$m.k + 1 = m.k + lp1(bgmax - w.k) + lp2(bpmax.k - w.k) \tag{2.1}$$

Donde $w.k$ y $m.k$ representan los movimientos de exploración y explotación respectivamente del búfalo k-ésimo $(k = 1, 2, ..., N); lp1$ y $lp2$ son factores de aprendizaje que pueden tener valores entre 0.1 a 0.6; $bgmax$ es el mejor fitness de la manada y él $bpmax.k$ es el mejor de cada búfalo.

4. Se actualiza la ubicación del búfalo k, considerando él $bpmax.k$ y $bgmax$ como se observa en la Ecuación 2.2.

$$w.k + 1 = \frac{w.k + m.k}{\pm\lambda} \tag{2.2}$$

El valor de λ de acuerdo a [76] puede estar entre valores de 0.1 y 2, pudiendo ser positivos o negativos. Entre más pequeño sea el valor de λ se realizará una mayor exploración y en caso contrario se realizará una mayor explotación.

5. Si él $bgmax$ se actualiza, entonces se procede ir al paso 6. Si no es así, se deriva al paso 2.

6. Si aún no se alcanza el criterio de parada, se procede regresar al paso 3, en caso contrario se procede dirigirse al paso 7.

7. Se emite la mejor solución

Como se puede apreciar, el ABO tiende a evitar el estancamiento al estar actualizando regularmente el mejor búfalo de toda la manada de manera regular [32].

Aunado a lo antes mencionado, el ABO lleva a cabo una exploración más adecuada al considerar al mejor búfalo y la mejor ubicación de cada búfalo con él $bpmax.k$ [32].

En el aspecto de la explotación se realiza considerando el aprovechamiento que tenga el mejor búfalo de toda la manada, junto con el mejor histórico de cada búfalo durante la ejecución del algoritmo [32].

2.2 Representación de las soluciones

Dentro de los métodos de ABO-1DCSP y ABO-1DCSPMS se establece que cada búfalo es una solución, la cual está conformada por todos los ítems a obtener de acuerdo a la instancia. Asimismo, en cada búfalo se pueden obtener diversos patrones de corte, estos a su vez son conjuntos de ítems a obtener de un stock.

Para poder ejemplificar de una mejor manera el cómo se representan las soluciones en el ABO-1DCSP y el ABO-1DCSPMS se empleará la Figura 2.1. Se puede observar una instancia con cuatro ítems con diferentes longitudes pero con una misma cantidad a obtener. De igual forma, se considera un stock ilimitado de una sola longitud. Como se puede apreciar en cada búfalo se consideran los 8 ítems a obtener, donde en cada búfalo se visualiza un acomodo diferente de los ítems.

Lo anterior tiene como consecuencia el generar diferentes patrones de corte aplicados a un stock en particular. La manera en la que se determinan los patrones de corte es de la siguiente manera, se estarán sumando las longitudes de los ítems de

izquierda a derecha mientras no se rebase la longitud del stock, por ejemplo en el primer patrón de corte del primer búfalo solamente se pueden considerar un ítem 2, un ítem 3 y un ítem 4, no obstante si quisiéramos agregar el primer ítem 1 se estaría rebasando la longitud del stock que es de 75, por lo que el primer ítem 1 sería el primer ítem a considerar en el segundo patrón de corte dentro del primer búfalo.

Una vez que se hayan formado los patrones de corte en cada búfalo, se procede a contabilizar el desperdicio que se genera en cada uno para irlo sumando y posteriormente obtener el total en cada búfalo, logrando así determinar cuál búfalo es el que genera una menor cantidad de desperdicio. De acuerdo a la Figura 2.1 el búfalo con la menor cantidad de desperdicio es el primero con un total de 50, también se puede observar que es el búfalo con la menor cantidad de stocks empleados y con desperdicio de acuerdo al acomodo de los ítems que tiene.

2.3 Método ROV

En los algoritmos ABO-1DCSP y ABO-1DCSPMS fue necesario el implementar el método ROV para discretizar las soluciones continuas emitidas por la Ecuación 2.2, la cual pertenece al ABO empleada para obtener la nueva ubicación de cada búfalo. Aunado al hecho que las instancias utilizadas para probar la eficiencia de los algoritmos ABO-1DCSP y ABO-1DCSPMS son del tipo discretas.

Para explicar como funciona el ROV tomaremos la instancia de la Figura 2.1, la cual contempla cuatro tipos de ítems y un solo stock. Primero lo que hay que realizar es ordenar los ítems de manera ascendente, como se aprecia en la Figura 2.2(a). Ahora

Figura 2.1: Representación de los búfalos en los algoritmos ABO-1DCSP y ABO-1DCSPMS

bien, supongamos que se obtienen los valores en *wb* derivados de la Ecuación 2.2 como se observa en la Figura 2.2(b). El siguiente paso es ordenar de manera ascendente los valores del *wb* como se contempla en la Figura 2.2(c) pero incluyendo los índices nuevo y anterior asignados a j y i respectivamente. Ahora se procede a discretizar cada valor del *wb* obtenido de la Figura 2.2(b) de la siguiente forma, donde la i-ésima posición del *wb* discretizado mostrado en la Figura 2.2(d) será llenado con la longitud j-ésimo ítem de la lista mostrada en la Figura 2.2(a). Tomando como ejemplo la posición del nuevo j = 1 y su correspondiente viejo i = 3 de la Figura 2.2(c), ahora en la posición i = 3 de la Figura 2.2(d) será llenada con la longitud del j = 1 del ítem de la Figura 2.2(a). Este proceso se repetirá con el resto de los valores de la Figura 2.2(c) para generar el nuevo arreglo de ítems presentados en la Figura 2.2(d).

Posición	Longitud
1	15
2	15
3	20
4	20
5	40
6	40
7	50
8	50

(a) Lista de ítems

Posición	Valor
1	41.5
2	21.3
3	14.9
4	51.2
5	39.2
6	51.3
7	21.8
8	15.2

(b) Nuevo Wb obtenido

Anterior Posición (j)	Nueva Posición (j)	Valor
3	1	14.9
8	2	15.2
2	3	21.3
7	4	21.8
5	5	39.2
1	6	41.5
4	7	51.2
6	8	51.3

(c) Nuevo Wb Ordenado

Posición	Longitud
1	40
2	20
3	15
4	50
5	40
6	50
7	20
8	15

(d) Nuevo Wb Discretizado

Figura 2.2: Explicación del ROV. (a) Los ítems de la instancia son ordenados de manera ascendente. (b) Son los valores obtenidos del Nuevo Wb. (c) Los valores del Nuevo Wb son ordenados de manera ascendente (nueva posición), colocando la anterior posición igualmente. (d) Son los valores discretizados del Nuevo Wb.

Capítulo 3

Algoritmo ABO-1DCSP

Dentro de este apartado se estará presentando el algoritmo ABO-1DCSP. Empezando por la función objetivo utilizada para evaluar el desperdicio que genera cada búfalo. Posteriormente, se describirá el algoritmo ABO-1DCSP. Por último se presentarán los resultados obtenidos del ABO-1DCSP con las instancias.

3.1 Función objetivo para el caso de 1D-CSP con un solo stock

En el algoritmo ABO-1DCSP empleamos la siguiente función objetivo, con la cual se contabiliza el desperdicio total generado Z de la aplicación de los patrones de corte en cada búfalo, considerando únicamente un solo stock con longitud Ls.

$$minimizar Z = \sum_{c=1}^{S} Ls_c - lp_c \tag{3.1}$$

Donde

- Z es el desperdicio total
- l_p es la longitud de un patrón de corte
- S es el total de los stocks
- L_s es la longitud del stock

Como puede observarse en la Ecuación 3.1 se estará sumando la diferencia entre la longitud del stock Ls menos la longitud de cada patrón de corte lp que se genere en cada uno de los búfalos de la manada. Al término se obtendrá el desperdicio total Z que tiene cada búfalo de acuerdo al ordenamiento de sus ítems.

3.2 Algoritmo ABO-1DCSP

Dentro del ABO-1DCSP se contemplan las variables propias del ABO, como lo son los factores de aprendizaje $lp1$ y $lp2$, así como λ, con la cual se puede orientar al algoritmo a realizar una mayor exploración o explotación. De igual modo, se consideran la cantidad de búfalos (nb), iteraciones (k) e iteraciones para reiniciar la manada (q). Contemplando de igual forma la *instancia* que contiene los ítems a obtener del stock, cada uno con su respectiva longitud ($longitud\_stock$).

Como primer paso se procede a obtener los *búfalos* de manera aleatoria con el método *CrearBúfalosAleatoriamente* como se aprecia en la línea 1 del Algoritmo 1. Después es necesario el encontrar al mejor búfalo de la manada (*bgmax*) mediante la función *Buscarbgmax* junto con la Ecuación 3.1 para ir contabilizando el desperdicio que se vaya obteniendo de acuerdo a los patrones de corte que se generen en cada búfalo.

Algoritmo 1: ABO-1DCSP

Input: instancia, longitud_stock, número_búfalos(nb), λ, iteraciones (k), lp1, lp2, número_iteraciones_reiniciar(q)
Output: mejor_bgmax

```
1  búfalos = CrearBúfalosAleatoriamente(instancia, longitud_stock, nb);
2  bgmax = Buscarbgmax(búfalos, longitud_stock);
3  j = 1;
4  i = 1;
5  while i ≤ k do
6  |  búfalos = ActualizarBúfalos(búfalos, longitud_stock, nb, λ, lp1, lp2,
   |   instancia, bgmax);
7  |  bgmaxactualizado, bgmax = VerificarActualizaciónbgmax(búfalos,
   |   bgmax, longitud_stock);
8  |  if bgmaxactualizado == Falso Y i % q == 0 then
9  |  |  lista_bgmax[j] = bgmax;
10 |  |  j = j + 1;
11 |  |  bgmax_B = Generarbgmax(búfalos, instancia, longitud_stock);
12 |  |  búfalos =CrearBúfalosAleatoriamente(instancia, longitud_stock, nb);
13 |  |  bgmax = bgmax_B;
14 |  |  i = i + 1;
15 |  else
16 |  |  i = i + 1;
17 lista_bgmax[j] = bgmax;
18 mejor_bgmax = BuscarMejorbgmax(lista_bgmax);
```

El siguiente paso es entrar a un ciclo donde se comienza con el método *ActualizarBúfalos* como se observa en la línea 6 del Algoritmo 1, se emplean las Ecuaciones 2.1 y 2.2 para calcular el fitness y ubicación nueva de cada búfalo. En el proceso de actualización de la manada se implementó el método ROV para convertir los valores continuos que entregan las Ecuaciones 2.1 y 2.2 del ABO a valores a discretos. Debido a que las longitudes de los ítems en cada una de las instancias empleadas se consideran longitudes enteras en los ítems, tal como se aprecia en el Anexo 5.0.1. El proceso de conversión del ROV se puede apreciar con más detalle en la subsección 2.3.

Cuando se termine el proceso de actualización de los *búfalos* se procede a comprobar si él *bgmax* se actualizó con el método *VerificarActualizaciónbgmax* el cual retorna un valor de Falso o Verdadero hacia la variable *bgmaxactualizado* y el propio *bgmax*.

En dado caso que se alcancen las q iteraciones y no se haya actualizado el *bgmax*, éste se deberá guardar en *lista_bgmax* para hacer la búsqueda del mejor *bgmax* al término del algoritmo. El siguiente proceso implica el generar un nuevo *bgmax* denominado $bgmax_B$, el cual estará formado a partir de los mejores patrones de corte de los *búfalos* actuales. Lo anterior se realizará mediante la función *Generarbgmax*, éste proceso se puede observar con mayor detalle en la Figura 3.1.

Para explicar como es el proceso de la generación del $bgmax_B$ hay que tomar la instancia de ejemplo de la Figura 2.1. Suponiendo que se tiene una manada de *búfalos* como la mostrada en la Figura 3.1(a) en la cual su líder no se actualizó después de q iteraciones, se realiza un ordenamiento ascendentemente, primero con base en el

desperdicio total de los *búfalos* para luego llevar a cabo otro ordenamiento ascendente de acuerdo al desperdicio que tienen los patrones de corte en cada búfalo como se aprecia en la Figura 3.1(b). Terminado el proceso de ordenamiento de los *búfalos*, se procede a seleccionar los mejores patrones de corte de la manada, para lo cual se selecciona el primer patrón de corte del primer búfalo (de izquierda a derecha) debido a que este es aquel que tiene la menor cantidad de desperdicio de toda la manada. El primer búfalo será la referencia para la búsqueda y su primer patrón de corte será el primer patrón de corte del $bgmax_B$ cómo se aprecia en la Figura 3.1(c). La demanda para cada uno de los ítems en cada patrón de corte seleccionado será restado de la lista de ítems de la instancia y cada patrón de corte seleccionado se eliminará del búfalo al que pertenezca. El siguiente paso es verificar si el próximo patrón de corte del búfalo de referencia tiene un desperdicio de cero, de lo contrario se saltará al siguiente búfalo. Lo antes mencionado se puede observar con el primer búfalo la Figura 3.1(c) donde el primer patrón de corte es seleccionado, pero el siguiente patrón de corte al no tener un desperdicio de cero motiva ir al siguiente búfalo. Si el primer patrón de corte del siguiente búfalo puede ser seleccionado, éste se convertirá en el nuevo búfalo de referencia para continuar con la búsqueda de los mejores patrones de corte. En caso contrario, el patrón de corte se descartará del búfalo y se procederá a ir con el siguiente búfalo hasta que se pueda seleccionar un patrón de corte. Este proceso continúa hasta que todos los ítems de la instancia estén integrados en él $bgmax_B$. Sin embargo, se puede dar el caso que al recorrer todos los patrones de corte de los *búfalos* no se hayan integrado todos los ítems de la instancia, por lo que los ítems faltantes de la instancia tendrán que ser agregados en orden ascendente al

final del $bgmax_B$ formado por los patrones de corte seleccionados de la manada de *búfalos*.

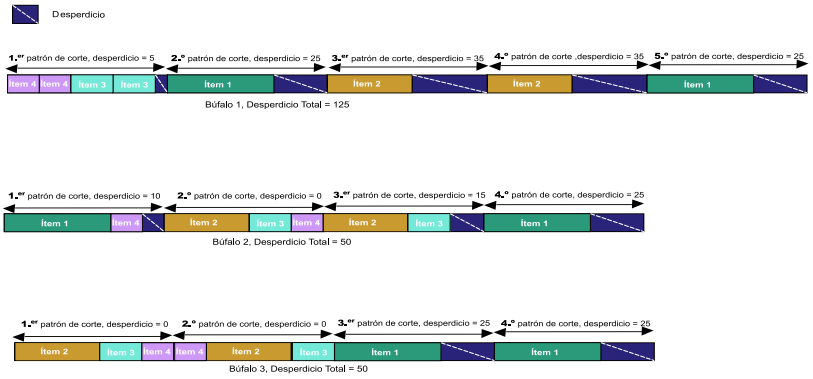
Con la generación de $bgmax_{\mathrm{B}}$ se puede reemplazar la manada actual con una nueva y asignar a $bgmax_{\mathrm{B}}$ como el nuevo $bgmax$ de la nueva manada que los estará guiando. Una vez que se terminen de realizar las k iteraciones hay que buscar el mejor $bgmax$ encontrado con el método *BuscarMejorbgmax* y guadarlo en la variable *mejor_bgmax* para enviarla como la mejor solución que logró obtener el ABO-1DCSP.

3.3 Resultados

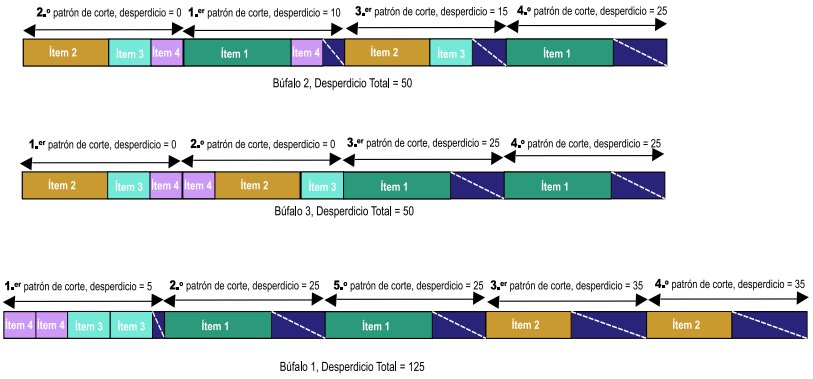
El algoritmo ABO-1DCSP para los casos de un solo stock del 1D-CSP se llevó a cabo en el lenguaje de programación Python en su versión 3.7. siendo ejecutada en un equipo de cómputo que cuenta con un procesador Intel® Core™ i7-6700HQ CPU 2.60GHz con 8 GB de RAM y Windows 10 Home Single Language. Se ejecutó cada instancia 50 veces debido a que fue el mayor número realizado por los trabajos [26] y [1].

En la Tabla 3.1 se pueden apreciar los algoritmos contra los cuales se comparó al ABO-1DCSP, detallando qué medidas de rendimiento fueron empleadas en cada método a efectos de comparación contra el ABO-1DCSP. Como se puede observar, el método SMBEP y EP son los únicos algoritmos que obtuvieron valores del desperdicio promedio, stock usado promedio y el stock usado con desperdicio promedio.

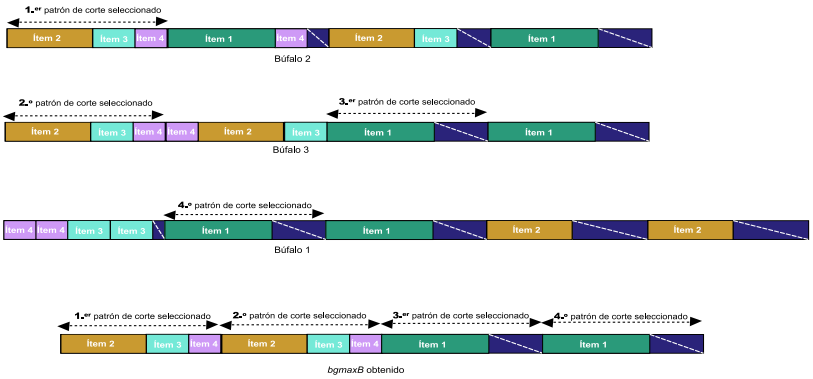
Asimismo, se consideró el calcular el relative percetage deviation (RPD) como se observa en [77] y [78] junto con la aplicación de la prueba no paramétrica de Friedman



(a) Manada Actual.



(b) Ordenamiento de los búfalos y sus patrones de corte.



(c) Selección de los mejores patrones de corte.

Figura 3.1: El proceso para obtener el $bgmax_B$. (a) Es un ejemplo de una manada de la cual se obtendrá el $bgmax_B$. (b) Los búfalos son ordenados ascendentemente con base en su desperdicio para luego ordenar los patrones de corte cada búfalo de manera ascendente. (c) Se seleccionan los mejores patrones de corte de la manada actual para formar el $bgmax_B$.

Tabla 3.1: Algoritmos relacionados con medidas de rendimiento para el 1D-CSP con un solo stock.

Algoritmo	Referencia	Medidas de Rendimiento		
		Desperdicio Promedio	Stock usado Promedio	Stock usado con desperdicio Promedio
LLA	[20]	✓	✓	
SMBEP	[66]	✓	✓	✓
HACO	[26]		✓	
Pure ACO	[26]		✓	
EP	[1]	✓	✓	✓
MACO	[19]	✓	✓	
HMCGA	[25]	✓	✓	
GA	[18]	✓		

con el propósito de corroborar que las diferencias entre los métodos son significativas. Para calcular el RPD se emplea la Ecuación 3.2 donde se considera el best value obtained (BOV), el cual es el mejor valor obtenido de cada método, y el best value known (BKV) que es el mejor valor conocido en cada instancia.

$$RPD = \frac{BOV - BKV}{BOV} \times 100 \tag{3.2}$$

En lo que se refiere a la prueba no paramétrica de Friedman, se aplicó en el lenguaje R con la versión 4.2.0. empleando los valores promedio RPD tal como se aprecia en [79] y [80].

En la Tabla 3.2 se puede apreciar la comparación del ABO-1DCSP contra otras enfoques bajo el desperdicio promedio. El símbolo * indica que el método alcanzo el BKV. Donde el ABO-1DCSP fue capaz de generar soluciones con una cantidad de desperdicio mínima o igual a la mejor conocida en las instancias 1a, 2a, 3a, 4a, 7a y 9a. Destacando que en la instancia 9a fue el único método en generar la cantidad de 142. A pesar de que el ABO-1DCSP en las demás instancias no alcanzó el BKV, se

logró posicionar en las primeras tres posiciones. Obteniendo el segundo lugar en la instancia 6a y el tercer lugar en las instancias 5a, 8a y 10a. Basado en lo anterior, se puede inferir que el ABO-1DCSP es bastante competente en la generación mínima de desperdicio.

Tabla 3.2: Comparación del desperdicio promedio del ABO-1DCSP contra otros enfoques.

Instancia	BKV	ABO-1DCSP	LLA	SMBEP	EP	GA	HMCGA	MACO
1a	3	3*	3*	3*	3*	3*	3*	3*
2a	13	13*	13*	13*	13*	14.5	13*	13*
3a	0	0*	0*	0*	0*	2.5	0*	0*
4a	11	11*	11*	11*	11*	11*	11*	11*
5a	10850	11450	11450	11370	11966	10850*	11966	11966
6a	103	109.88	275	240.6	309.4	330.9	103*	103*
7a	84	84*	84*	84*	189.6	327.6	264	264
8a	212	320	332	308	788	547.95	212*	212*
9a	142	142*	382	250	730	673.8	334	334
10a	130	274	130*	190	1037.2	662.5	490	490

La prueba no paramétrica de Friedman y el RPD promedio del parámetro del desperdicio promedio se puede observar en la Tabla 3.3. Tal como se aprecia, el ABO-1DCSP obtuvo la primera posición entre los demás métodos, mientras que el valor p calculado con la aplicación de la prueba de Friedman fue menor a 0.05, lo cual nos indica que sí hay diferencias significativas entre los métodos.

Tabla 3.3: Promedio del RPD y prueba no paramétrica de Friedman para el desperdicio promedio.

Algoritmo	ABO-1DCSP	LLA	SMBEP	EP	GA	HMCGA	MACO	Valor p
Promedio RPD:	9.78	16.67	16.77	37.28	47.41	20.84	20.84	0.0047
Lugar	1	2	3	5	6	4	4	

Dentro de la Tabla 3.4 se puede observar la comparación del stock usado promedio

obtenido por el ABO-1CSP contra otros métodos. En este parámetro el ABO-1DCSP igualo la cantidad del BKV de la instancia 1a a la 5a. Aunque en las demás instancias no consiguió el BKV si se mantuvo muy cerca, manteniéndose entre los primeros tres lugares en las instancias 6a, 7a, 8a y 9a. En el caso de la instancia 10a se posicionó en el cuarto mejor método de los 8 algoritmos

Tabla 3.4: Comparativa del ABO-1DCSP bajo el stock usado promedio.

Instancia	BKV	ABO-1DCSP	LLA	SMBEP	EP	PureACO	HACO	HMCGA	MACO
1a	9	9*	9*	9*	9*	-	-	9*	9*
2a	23	23*	23*	23*	23*	-	-	23*	23*
3a	15	15*	15*	15*	15*	-	-	15*	15*
4a	19	19*	19*	19*	19*	-	-	19*	19*
5a	53	53*	53*	53*	53.12	-	-	53*	53*
6a	79	79.08	81	80.6	81.4	79*	79*	79.1	79.1
7a	67.3	68	68	68	68.88	69	68	67.3*	67.3*
8a	143	144.9	145	144.8	148.8	146	143*	144.8	144.8
9a	149	150	152	150.9	154.9	151	149*	149.4	149.4
10a	215	217.2	216	216.5	223.56	218.9	215*	219.8	219.8

En la Tabla 3.5 se aprecia el cálculo del promedio RPD y la aplicación de la prueba de Friedman sobre el stock usado promedio, donde se muestra que el ABO-1DCSP se posicionó en el segundo lugar, estando muy cerca de los métodos que obtuvieron el primer lugar. En lo que se refiere al valor p de la prueba de Friedman, se consiguió un valor inferior al 0.05, lo cual nos indica que las diferencias entre los cinco métodos son relevantes.

Tabla 3.5: Promedio del RPD y prueba no paramétrica de Friedman para el stock usado promedio.

Algoritmo	ABO-1DCSP	LLA	SMBEP	EP	HMCGA	MACO	Valor p
Promedio del RPD:	0.41	0.73	0.62	1.7	0.38	0.38	0.0034
Lugar	2	4	3	5	1	1	

En la Tabla 3.6 se aprecia la comparación del óptimo teórico (N_{min}) relacionado con el stock usado promedio obtenido de [20] para las instancias de [2] y [1]. Algo importante a mencionar es el hecho que los métodos HMCGA y MACO son los únicos métodos que llegaron a reportar cantidades de stock inferiores al N_{min}, esto solo sucedió en la instancia 7a. Se observa que el ABO-1DCSP es capaz de igualar al N_{min} en 5 de las 10 instancias, mientras que requiere de entre 0.67 % y 1.38 % más stock que el N_{min} en las instancias 6a, 8a, 9a y 10a. Solo en la instancia 5a el ABO-1DCSP emplea 3.92 % más stock que el N_{min}. Destacando el hecho que la función objetivo del ABO-1DCSP se centra en minimizar el desperdicio, por el contrario, en trabajos como en [1] proponen una función objetivo que considera minimizar el desperdicio y el número de stocks con desperdicio. En tanto que en [20] implementan dos funciones objetivo que tienen como meta el reducir el desperdicio y el número de stocks.

Tabla 3.6: Comparación del N_{min} stock usado de las instancias [1] y [2].

Instancia	$\mathbf{N_{min}}$	**% ABO** $> \mathbf{N_{min}}$	**% LLA** $> \mathbf{N_{min}}$	**% SMBEP** $> \mathbf{N_{min}}$	**% EP** $> \mathbf{N_{min}}$	**% Pure ACO** $> \mathbf{N_{min}}$	**% HACO** $> \mathbf{N_{min}}$	**% HMCGA** $> \mathbf{N_{min}}$	**% MACO** $> \mathbf{N_{min}}$
1a	9	0	0	0	0	-	-	0	0
2a	23	0	0	0	0	-	-	0	0
3a	15	0	0	0	0	-	-	0	0
4a	19	0	0	0	0	-	-	0	0
5a	51	3.92	3.92	3.92	4.15	-	-	3.92	3.92
6a	78	1.38	3.84	3.33	4.35	1.28	1.28	1.41	1.41
7a	68	0	0	0	1.29	1.47	0	-1.02	-1.02
8a	143	1.32	1.39	1.25	4.05	2.09	0	1.25	1.25
9a	149	0.67	2.01	1.27	3.95	1.34	0	0.26	0.26
10a	215	1.02	0.46	0.69	3.98	1.81	0	2.23	2.23

En la Tabla 3.7 se muestra la comparación del RPD del stock usado con desperdicio entre el ABO-1DCSP y los métodos SMBEP y EP. Como se puede apreciar, el ABO-1DCSP obtiene una menor o igual cantidad de stock con desperdicio que el enfoque

SMBEP de la instancia 1a a 6a. En la comparación del ABO-1DCSP contra el método EP, el ABO-1DCSP obtiene una menor o igual cantidad de stocks usados con desperdicio en las instancias 3a, 8a, 9a y 10a.

Tabla 3.7: Promedio Stock usado con desperdicio

Instancia	BKV	ABO-1DCSP	SMBEP	EP
1a	2	2.3	2.8	2*
2a	4	4.48	4.7	4*
3a	0	0*	0*	0*
4a	1.02	2.1	3.2	1.02*
5a	22.8	23.48	27.1	22.8*
6a	24.8	24.8*	26.5	29.96
7a	6.6	8.34	6.6*	7.48
8a	27.4	33.34	27.4*	56.24
9a	17.6	23.22	17.6*	48.54
10a	11.4	33.52	11.4*	73.06

De igual manera, se calculó el RPD promedio del stock usado con desperdicio, generando los valores mostrados en la Tabla 3.8, donde se aprecia que el ABO-1DCSP fue capaz de situarse en la segunda posición.

Tabla 3.8: RPD promedio para el parámetro del stock usado con desperdicio

Algoritmo	ABO-1DCSP	SMBEP	EP
RPD promedio:	20.69	13.37	22.82
Ranking	2	1	3

Se realizaron más experimentos con las instancias u120, u250, u500 y u1000 de [3] para comparar el ABO-1DCSP contra el N_{min} reportado en [33]. En el apartado del Anexo 5.0.1, en las Tablas 5.11, 5.12, 5.13 y 5.14 se puede visualizar con más detalle información de las instancias previamente mencionadas.

En la Tabla 3.9 se aprecia que el ABO-1DCSP es capaz de alcanzar el N_{min} en 14 de las 20 instancias. Mientras que en las instancias u120_03 y u120_17 ocupa entre 0.20 % y 0.26 % más stock que el N_{min}. En tanto que en las instancias u120_09 y u120_12 el ABO-1DCSP requiere entre 1.17 % y 1.79 % más stock que el N_{min}. Solo en las instancias u120_08 y u120_19 requiere más del 1.99 % stock que el N_{min} para cumplir con la demanda de ítems.

Tabla 3.9: Comparación del N_{min} stock usado de las instancias u120 de [3].

Instancia	$\mathbf{N_{min}}$	**ABO-1DCSP**	**% ABO-1DCSP** $>$ $\mathbf{N_{min}}$
u120_00	48	48	0
u120_01	49	49	0
u120_02	46	46	0
u120_03	49	49.1	0.20
u120_04	50	50	0
u120_05	48	48	0
u120_06	48	48	0
u120_07	49	49	0
u120_08	50	51	2
u120_09	46	46.54	1.17
u120_10	52	52	0
u120_11	49	49	0
u120_12	48	48.86	1.79
u120_13	49	49	0
u120_14	50	50	0
u120_15	48	48	0
u120_16	52	52	0
u120_17	52	52.14	0.26
u120_18	49	49	0
u120_19	49	50	2.04

En la Tabla 3.10 se observa que el ABO-1DCSP en 8 de las 20 instancias logra alcanzar el N_{min}. Por otro lado, en las instancias u250_00, u250_02, u250_04, u250_05,

u250_08, u250_11, u250_12, u250_15 y u250_16 el ABO-1DCSP requiere de menos del 1 % más stock que el N_{min} para cumplir con la demanda de ítems. Mientras que en las instancias u250_07, u250_13 y u250_18 el ABO-1DCSP requiere de más del 0.99 % stock que el N_{min}.

Tabla 3.10: Comparación del N_{min} stock usado de las instancias u250 de [3].

Instancia	$\mathbf{N_{min}}$	**ABO-1DCSP**	**% ABO-1DCSP** $>$ $\mathbf{N_{min}}$
u250_00	99	99.52	0.52
u250_01	100	100	0
u250_02	102	102.26	0.25
u250_03	100	100	0
u250_04	101	101.1	0.09
u250_05	101	102	0.99
u250_06	102	102	0
u250_07	103	104.1	1.06
u250_08	105	106	0.95
u250_09	101	101	0
u250_10	105	105	0
u250_11	101	102	0.99
u250_12	105	106	0.95
u250_13	102	103.02	1
u250_14	100	100	0
u250_15	105	106	0.95
u250_16	97	97.06	0.06
u250_17	100	100	0
u250_18	100	101	1
u250_19	102	102	0

En la Tabla 3.11 se visualiza que el ABO-1DCSP en 7 de las 20 instancias iguala la cantidad de stock usado al N_{min}. Por otra parte, el ABO-1DCSP en las instancias u500_02, u500_11, u500_13, u500_14 y u500_17 el ABO-1DCSP requiere entre 0.03 % y 0.22% más stock que el N_{min}. Para el resto de las instancias el ABO-1DCSP emplea

entre 0.43 % y 0.51 % más stock que el N_{min} para cumplir con la demanda de ítems.

Tabla 3.11: Comparación del N_{min} stock usado de las instancias u500 de [3].

Instancia	$\mathbf{N_{min}}$	**ABO-1DCSP**	**% ABO-1DCSP** > $\mathbf{N_{min}}$
u500_00	198	198.92	0.46
u500_01	201	202	0.49
u500_02	202	202.46	0.22
u500_03	204	205	0.49
u500_04	206	206	0
u500_05	206	206	0
u500_06	207	208	0.48
u500_07	204	205	0.49
u500_08	196	196.86	0.43
u500_09	202	202	0
u500_10	200	200	0
u500_11	200	200.3	0.15
u500_12	199	200	0.50
u500_13	196	196.06	0.03
u500_14	204	204.26	0.12
u500_15	201	201	0
u500_16	202	202	0
u500_17	198	198.18	0.09
u500_18	202	202	0
u500_19	196	197	0.51

En la Tabla 3.12 se observa que el ABO-1DCSP es capaz de igualar en 6 de las 20 instancias al N_{min}. A diferencia que en las instancias u1000_01, u1000_08, u1000_10, u1000_18 y u1000_19 el ABO-1DCSP requiere entre 0.005 % y 0.05 % más stock que el N_{min}. Mientras que para las instancias u1000_00, u1000_02, u1000_04, u1000_05, u1000_09, u1000_11, u1000_14 y u1000_15 el ABO-1DCSP ocupa entre 0.11 % y 0.25 % más stock que el N_{min}. Solo en la instancia u1000_03 requiere de 0.43 % más stock que el N_{min}.

Tabla 3.12: Comparación del N_{min} stock usado de las instancias u1000 de [3].

Instancia	$\mathbf{N_{min}}$	**ABO-1DCSP**	**% ABO-1DCSP** $>$ $\mathbf{N_{min}}$
u1000_00	399	399.6	0.15
u1000_01	406	406.24	0.05
u1000_02	411	411.48	0.11
u1000_03	411	412.74	0.42
u1000_04	397	398	0.25
u1000_05	399	399.78	0.19
u1000_06	395	395	0
u1000_07	404	404	0
u1000_08	399	399.08	0.02
u1000_09	397	398	0.25
u1000_10	400	400.02	0.005
u1000_11	401	401.96	0.23
u1000_12	393	393	0
u1000_13	396	396	0
u1000_14	394	395	0.25
u1000_15	402	403	0.24
u1000_16	404	404	0
u1000_17	404	405	0
u1000_18	399	399.08	0.02
u1000_19	400	400.02	0.005

Como puede observarse, el ABO-1DCSP en 35 de las 80 instancias relacionadas con [3] alcanza el N_{min}, es decir, en 43.75 % de todas las instancias. Mientras que en 38 de las 80 instancias, es decir, el 47.5 %, requiere de menos de 1 % más stock que el N_{min} para cumplir con la demanda de ítems. Solo en 7 de las 80 instancias, es decir, el 8.75 %, ocupa más de 0.99 % stock que el N_{min} en cumplir con la demanda de ítems. Demostrando que el ABO-1DCSP es eficiente en cuanto al parámetro del stock usado, aunque el ABO-1DCSP tenga como función objetivo el minimizar el desperdicio.

De igual forma, se emplearon algunas instancias del Set 1 y 2 de [4] para comprobar la eficiencia del ABO-1DCSP bajo el parámetro del stock usado contra el N_{min} reportado en [33]. En el Anexo 5.0.1, dentro de las Tablas 5.15 y 5.16 se observan con detalle las características de las instancias.

En la Tabla 3.13 se puede visualizar que el ABO-1DCSP en 11 de las 20 instancias alcanza el N_{min}. Mientras que en las instancias N2C1W2_P, N3C1W1_A, N3C2W2_D, N3C2W4_B, N4C1W2_T, N4C1W4_A, N4C1W4_B, N4C1W4_C y N4C1W4_D el ABO-1DCSP requiere de menos del 0.95 % más stock que el N_{min}. Únicamente en las instancias N2C1W1_Q y N2C1W2_O el ABO-1DCSP necesita 1 % más stock que el N_{min}.

En la Tabla 3.14 se observa que el ABO-1DCSP en 15 de las 20 instancias logra igualar la cantidad de stock usado al N_{min}. Por otra parte, en las instancias N3W2B2R3, N4W2B1R0, N4W2B1R3 y N4W3B3R7 necesita entre 0.99 % y 1.35 % más stock que el N_{min}. Solo en la instancia N3W4B2R1 necesita de más del 3 % stock que el N_{min} para cumplir con la demanda de ítems.

De tal forma que el ABO-1DCSP en 26 de las 40 instancias, es decir, el 65 %, iguala

Tabla 3.13: Comparación del N_{min} stock usado de las instancias del Set 1 de [4].

Instancia	$\mathbf{N_{min}}$	**ABO-1DCSP**	**% ABO-1DCSP** $>$ $\mathbf{N_{min}}$
N1C1W1_A	25	25	0
N1C1W1_B	31	31	0
N1C1W1_D	28	28	0
N1C1W1_E	26	26	0
N1C1W1_F	27	27	0
N1C1W1_G	25	25	0
N1C1W1_I	25	25	0
N2C1W1_Q	46	46.72	1.56
N2C1W2_N	64	64	0
N2C1W2_O	64	65.14	1.78
N2C1W2_P	68	68.04	0.05
N2C1W2_R	67	67	0
N3C1W1_A	105	106	0.95
N3C2W2_D	107	107.58	0.54
N3C2W4_B	112	112.8	0.71
N4C1W2_T	323	323.44	0.13
N4C1W4_A	368	368	0
N4C1W4_B	349	349.76	0.21
N4C1W4_C	365	365	0
N4C1W4_D	359	360.8	0.50

el número de stocks al N_{min}. Entre tanto que en 8 de las 40 instancias, es decir, el 20 %, requiere menos del 1% más stock que el N_{min}. Y tan solo en 6 de las 40 instancias, es decir el 15 %, necesita más del 0.99% stock que el N_{min} para poder cumplir con la demanda de ítems. Con base en lo anterior, se puede ver que nuevamente el ABO-1DCSP en otro conjunto de instancias es eficiente bajo el parámetro del stock usado, considerando que el ABO-1DCSP tiene como objetivo el minimizar el desperdicio.

Tabla 3.14: Comparación del N_{min} stock usado de las instancias del Set 2 de [4].

Instancia	$\mathbf{N_{min}}$	**ABO-1DCSP**	**% ABO-1DCSP > $\mathbf{N_{min}}$**
N1W1B1R2	19	19	0
N1W1B1R9	17	17	0
N1W1B2R0	17	17	0
N1W1B2R1	17	17	0
N1W1B2R3	16	16	0
N2W1B1R0	34	34	0
N2W1B1R1	34	34	0
N2W1B1R3	34	34	0
N2W1B1R4	34	34	0
N2W3B3R7	13	13	0
N2W4B1R0	12	12	0
N3W2B2R3	39	39.42	1.07
N3W3B1R3	29	29	0
N3W4B1R1	23	23	0
N3W4B2R1	22	22.78	3.54
N4W2B1R0	101	102	0.99
N4W2B1R3	100	101	1
N4W3B3R7	74	75	1.35
N4W4B1R0	56	56	0
N4W4B1R1	56	56	0

Capítulo 4

Algoritmo ABO-1DCSPMS

En este capítulo se estará presentando el enfoque ABO-1DCSPMS. Primero se presentará la función objetivo que considera el desperdicio total y la cantidad de stocks con desperdicio en cada búfalo. Posteriormente, se mostrará el algoritmo realizado, así como los resultados obtenidos.

4.1 Función objetivo

$$C = \frac{1}{m+1}(\sum_{j=1}^{m}\sqrt{\frac{w_j}{L_j}} + \sum_{j=1}^{m}\frac{V_j}{m}) \tag{4.1}$$

donde w_j y V_j son:

$$w_j = L_j - \sum_{i=1}^{n} x_{ij}, l_i, \quad j = 1, ..., m, \tag{4.2}$$

$$V_j = \begin{cases} 1 & \text{si } w_j > 0, \\ 0 & \text{de otra manera,} \end{cases} \qquad j = 1, ..., m \tag{4.3}$$

restringido a

$$\sum_{j=1}^{m} x_{ij} = N_i, i = 1, ..., n, \tag{4.4}$$

Donde:

- n = es el total de los diferentes ítems obtenidos del j-ésimo stock
- m = es el total de los stocks usados para completar la demanda total de ítems
- w_j = es el desperdicio del j-ésimo stock
- V_j = está relacionado con él j-ésimo stock con desperdicio
- L_j = es la longitud del j-ésimo stock
- x_ij = es la cantidad del i-ésimo ítem obtenido del j-ésimo stock
- l_i = es la longitud de cada ítem
- N_i = es el total de los ítems necesarios del i-ésimo ítem

La función objetivo a implementar en el ABO-1DCSPMS es la mostrada en el trabajo de [1], representada en la Ecuación 4.1, la cual considera la suma de la raíz cuadrada de los cocientes del desperdicio de cada stock w_j calculada con la Ecuación

4.2 entre la longitud del propio stock j junto con el cociente del número de stocks con desperdicio V_j obtenido con la Ecuación 4.3 entre el número total de stocks m, para así obtener el costo de cada búfalo en el ABO-1DCSPMS. Considerando el no obtener ítems de más de acuerdo a la Ecuación 4.4

4.2 Algoritmo ABO-1DCSPMS

El algoritmo del ABO-1DCSPMS, al igual que el ABO-1DCSP, considera las variables del ABO, las cuales son los factores de aprendizaje *lp1* y *lp2* relacionadas con la Ecuación 2.1 empleada para calcular el fitness de cada búfalo. Mientras que λ se asocia con la Ecuación 2.2 para obtener la nueva ubicación de cada búfalo. Asimismo, se consideran las variables de entrada *nb*, *k*, *q* e *instancia* del algoritmo ABO-1DCSP. La única variable que se tuvo que cambiar fue la de *longitud_objeto* por *lista_stocks*, debido a que las instancias empleadas consideran varios tipos de stocks.

El algoritmo ABO-1DCSPMS parte de la generación de búfalos mediante la función *CrearBúfalosAleatoriamente*, en la cual se obtendrán los diferentes acomodos de los ítems en cada búfalo de manera aleatoria. Una vez finalizado la generación de búfalos se procede a asignar el stock en cada búfalo mediante la función *AsingarStocksBúfalos*, ésta busca el minimizar el desperdicio y maximizar el número de ítems en cada patrón de corte.

Para explicar como se lleva la función *AsingarStocksBúfalos* se utilizarán las Figuras 4.1 y 4.2, en la cual se usará una instancia con tres tipos de stocks ilimitados y cuatro tipos de ítems con diferentes longitudes con una misma frecuencia de dos, los

cuales podrían generar un búfalo como el denominado *Búfalo Ejemplo* que se presenta en la Figura 4.2(a). El procedimiento de la asignación de los stocks en cada búfalo comienza ordenando los stocks de manera ascendente para generar los patrones candidatos, de la siguiente manera, se toma al primer ítem de izquierda a derecha del búfalo y se verifica si este se puede añadir al stock más pequeño de los que se tenga disponibles, como podemos observar al comienzo de la Figura 4.2(a) el Ítem 4 se puede añadir al Stock 1 formando al Patrón Candidato 1. Posteriormente, se verifica si se pueden agregar más ítems del búfalo al patrón candidato formado, en caso de que no sea así hay que calcular el desperdicio que se genera del patrón candidato y contar los ítems que están dentro de él, para después formar el siguiente patrón candidato considerando los ítems del patrón candidato previamente constituido en caso de que aún falten stocks por considerar, de lo contrario se deberá escoger aquel patrón candidato con la menor cantidad de desperdicio y mayor número de ítems. Retomando el ejemplo del Patrón Candidato 1, éste genera una cantidad de desperdicio de 15 y contempla un solo ítem. Después hay que buscar generar otro patrón candidato, para lo cual se tomarán los ítems del Patrón Candidato 1 junto con el siguiente ítem del búfalo, siendo el Ítem 2, buscando que éste nuevo arreglo se pueda obtener en alguno de los siguientes stocks disponibles, logrando obtenerlo del Stock 2 dando origen el Patrón Candidato 2. De nueva cuenta se verifica si se pueden añadir más ítems a este nuevo patrón candidato, pero como no es posible, entonces se calcula el desperdicio y se cuentan los ítems que hay dentro de este. Como todavía falta comprobar si se puede generar otro patrón candidato con el Stock 3 se toma el arreglo de ítems del Patrón Candidato 2 y se suma el siguiente ítem del búfalo a considerar, el cual es el Ítem 1;

sin embargo, al calcular la longitud de este nuevo arreglo de ítems nos percatamos de que no es posible obtenerse del Stock 3, por lo que detenemos la generación de patrones candidatos para seleccionar aquel patrón candidato con la menor cantidad de desperdicio y mayor número de ítems de los formados hasta el momento, siendo seleccionado el Patrón Candidato 1, éste patrón pasará a formar parte del arreglo denominado Asignación Stocks Búfalo Ejemplo y los ítems de este patrón escogido se eliminarán del arreglo *Búfalo Ejemplo*. El proceso anterior se estará realizando con los ítems restantes del *Búfalo Ejemplo* , formando así el arreglo de patrones de corte final mostrado en la Figura 4.2(b).

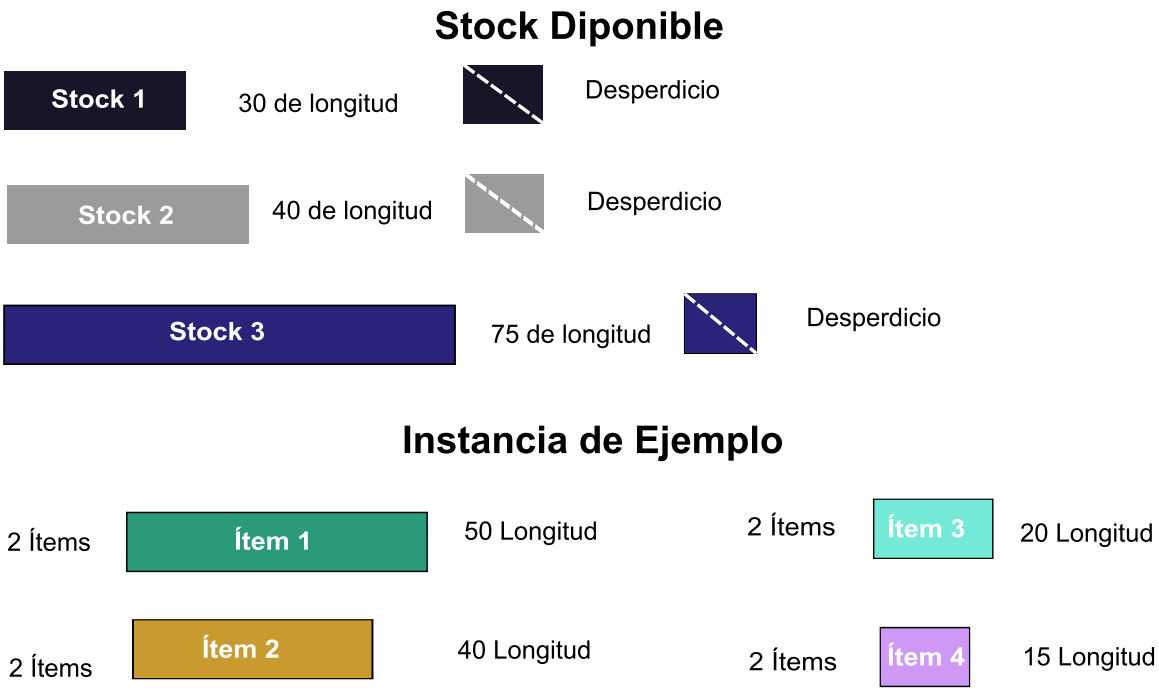
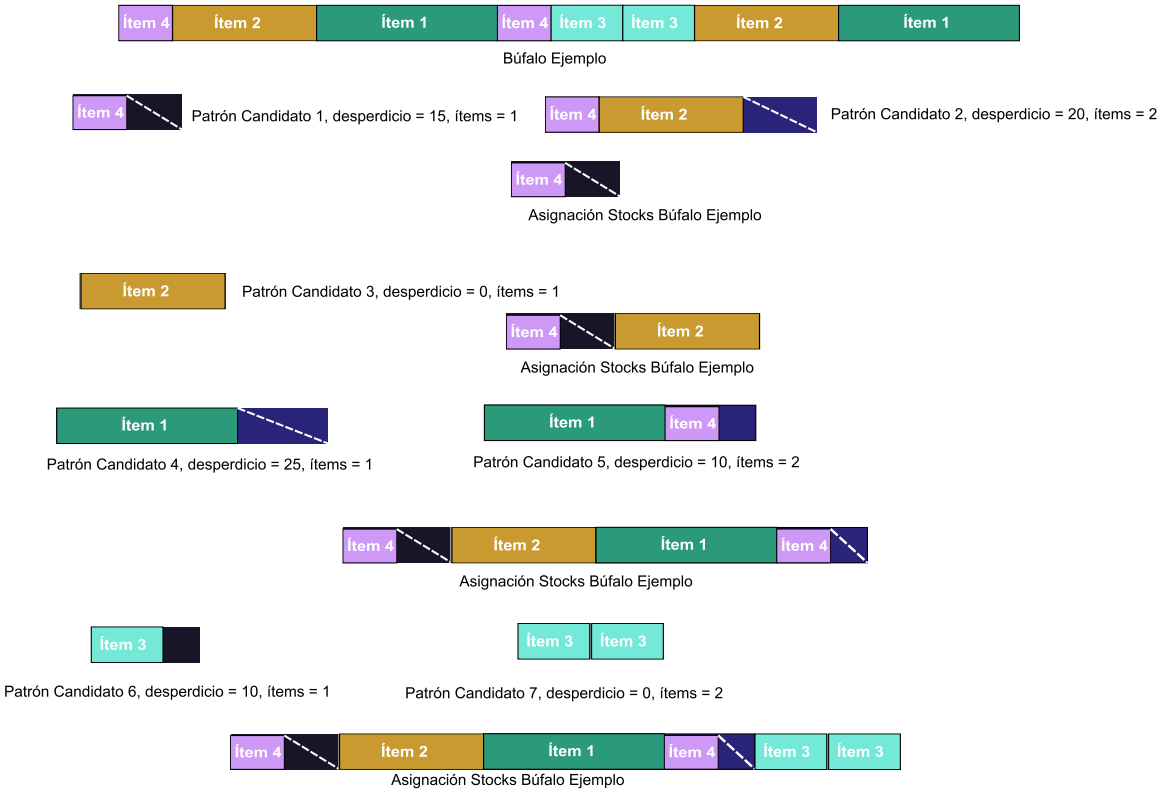


Figura 4.1: Instancia de ejemplo para el proceso de asignación de stocks.



(a) Proceso de Selección de Patrones Candidatos.

(b) Asignación Completa Búfalo Ejemplo.

Figura 4.2: El proceso para asignar el stock en cada búfalo. (a) Es el proceso de asignación stocks con un búfalo de ejemplo de acuerdo a la disposición de los ítems. (b) Asignación del stock al búfalo de ejemplo.

Una vez llevada a cabo la asignación de stocks se deberá calcular el desperdicio total y el número de stocks con desperdicio con la función *CalcularDesperdicioStocks-Desperdicio* como se aprecia en la línea 3 del Algoritmo 2. Posteriormente, se buscará el *bgmax* con la función *Buscarbgmax*, en la cual se empleará la Ecuación 4.1 buscando aquel búfalo con el menor costo.

Después se someten los búfalos al proceso llamado *ActualizarBúfalos* en el cual se emplearán las Ecuaciones 2.1 y 2.2 propias del ABO junto con la aplicación del método ROV para realizar la discretización de los valores emitidos por las Ecuaciones del ABO debido a que las instancias empleadas emplean longitudes enteras en los ítems, la forma en la que se emplea el ROV se puede ver con más detalle en la sección 2.3. De nueva cuenta se aplican los procesos *AsignarStocksBúfalos* y *CalcularDesperdicioStocksDesperdicio* para realizar la asignación de los stocks y el cálculo del desperdicio y número de stocks con desperdicio respectivamente. El siguiente paso es comprobar si se actualizó él *bgmax* con la función *VerificarActualizaciónbgmax*, la cual nos regresará la variable *bgmaxactualizado* y el *bgmax* que encontró.

En caso de que la variable *bgmaxactualizado* retorne un valor Falso y se hayan recorrido las q iteraciones es necesario el guardar el *bgmax* en $lista\_bgmax$ y proceder a obtener un nuevo *bgmax* a partir de la manada actual con la función *Generarbgmax* para obtener el $bgmax_B$, este procedimiento es el mismo que se aplicó en el ABO-1DCSP tal como se aprecia en la Figura 3.1. Una vez conformado el $bgmax_B$ hay que generar una nueva manada y asignar como nuevo *bgmax* al $bgmax_B$.

Cuando se hayan realizado las iteraciones k hay que guardar el actual *bgmax* en la variable $lista\_bgmax$ y proceder a buscar el mejor *bgmax* encontrado a lo largo de las iteraciones con la Función *BuscarMejorbgmax*

4.3 Resultados

El algoritmo ABO-1DCSPMS se realizó en el lenguaje de programación Python en su versión 3.7. siendo ejecutada en un equipo de cómputo que cuenta con un procesador Intel® Core™ i7-6700HQ CPU 2.60GHz con 16 GB de RAM y Windows 10 Home Single Language. Se ejecutó cada instancia 50 veces debido a que fue el mayor número realizado por el trabajo de [1].

Para comprobar la eficiencia del ABO-1DCSPMS al minimizar el desperdicio y el número de stocks con desperdicio se emplearon las instancias 1 a 5 de [2] y 6 a 10 de [1], los cuales se pueden apreciar en el Apéndice 5.0.1. Los trabajos empleados para realizar la comparación son los mostrados en la Tabla 4.1, donde se consideran los parámetros de desperdicio, stock usado y stock usado con desperdicio. De igual manera se calculó el RPD con la Ecuación 3.2 y se efectuó la prueba no paramétrica de Friedman, como en el ABO-1DCSP, para comprobar si existen diferencias significativas entre los algoritmos bajo los parámetros de desperdicio, stock usado y stock usado con desperdicio.

La Tabla 4.2 muestra la comparativa entre el ABO-1DCSPMS y otros enfoques bajo el parámetro de desperdicio promedio. La columna BKV está relacionada con el mejor valor conocido en cada instancia. Mientras que el símbolo * indica qué enfoques obtuvieron el BKV. Como se puede apreciar, el ABO-1DCSPMS alcanzó el BKV en las instancias 1, 3, 8 y 10. Mientras que en las demás instancias obtuvo el segundo mejor valor, estando muy cerca del primer lugar en las instancias 2, 3 y 7. En el caso de la instancia 5 donde los ítems y stocks están en el orden de los millares, la

diferencia que hay entre el ABO-1DCSPMS y el enfoque EP es menor si se compara al ABO-1DCSPMS con el tercer mejor enfoque que fue el de GA-Parmar, donde la diferencia es de 448. Asimismo, en la instancia 9 hay una diferencia menor entre el ABO-1DCSPMS y el método EP, siendo de 5.5, en cambio, la diferencia entre el ABO-1DCSPMS y el algoritmo SMBEP es de 9.8. La única instancia en la que el tercer mejor enfoque se mantiene más cerca del ABO-1DCSPMS es en la 6, donde la diferencia es de 2.86.

En la Tabla 4.7 se muestran los valores RPD promedio y el valor p de la prueba de Friedman para el desperdicio promedio del ABO-1DCSPMS y los demás enfoques. Los promedios que se emplearon para calcular el RPD promedio y el valor p de la prueba de Friedman fueron los de la instancia 1 a 8 debido a que los métodos GA-Parmar y LP no registron promedios en las instancias 9 y 10. Mientras que los métodos que se consideraron fueron el ABO-1DCSPMS, EP, SMBEP, GA-Parmar y LP debido a que son los métodos reportan promedios en las instancias 1 a 8. Como se puede observar, el ABO-1DCSPMS se colocó en la segunda posición, demostrando que es un método eficiente en la minimización del desperdicio promedio. En tanto que el valor p nos indica que las diferencias entre los métodos son significativas debido a que el valor es menor a 0.05.

En la Tabla 4.4 se muestra la comparativa entre el ABO-1DCSPMS contra otros enfoques bajo el parámetro del stock usado. Como se puede apreciar, el ABO-1DCSPMS fue el único que obtuvo el BKV en cada instancia. Partiendo de un orden ascendente, la menor diferencia entre el ABO-1DCSPMS y el segundo mejor enfoque en todas las instancias, se puede constatar en la instancia 1 donde fue de

0.22 entre el ABO-1DCSPMS y los métodos de EP y ACO-MCSP. La segunda menor diferencia está en la instancia 5, siendo de 0.95 entre el ABO-1DCSPMS y el enfoque EP. La tercera menor diferencia se encuentra en la instancia 2, siendo de 1 entre el ABO-1DCSPMS y el método EP. La cuarta menor diferencia está en la instancia 7, siendo de 1.02 entre el ABO-1DCSPMS y el algoritmo EP. La quinta menor diferencia fue en la instancia 4, siendo de 1.14 entre el ABO-1DCSPMS y los métodos de EP, GA-Lu y ACO-MCSP. La sexta menor diferencia está en la instancia 6, con 1.8 entre el ABO-1DCSPMS y el algoritmo EP. La séptima menor diferencia está en la instancia 8, con 2.12 entre el ABO-1DCSPMS y el método SMBEP. La octava menor diferencia está en la instancia 9, con 2.5 entre el ABO-1DCSPMS y el enfoque EP. La novena menor diferencia está en la instancia 10, con 3.8 entre el ABO-1DCSPMS y el algoritmo SMBEP. La última menor diferencia está en la instancia 3, entre el ABO-1DCSPMS y el método SMBEP.

El RPD promedio y el test de Friedman para el parámetro del stock usado se encuentran en la Tabla 4.5. Casi todos los métodos se tomaron en cuenta para calcular el RPD y la prueba de Friedman, con excepción del método Two-swap debido a que no tiene resultados de la instancia 1 a 5. Asimismo, no se consideraron los promedios de todos los métodos en las instancias 6 y 9 debido a que los métodos GA-Lu, Pure-ACO y ACO-MCSP no registraron promedios en dichas instancias. Por lo que solamente se consideraron los promedios de las instancias 1, 2, 3, 4, 5, 7, 8 y 10 de los métodos ABO-1DCSPMS, EP, SMBEP, GA-Lu, Pure-ACO y ACO-MCSP para calcular el RPD promedio y la prueba de Friedman. Como se puede observar, el ABO-1DCSPMS alcanza la primera posición, demostrando que es un algoritmo eficiente en

cuanto al uso del stock para cumplir con la demanda de ítems. A pesar del hecho que la función objetivo empleada busca minimizar el desperdicio y el número de stocks con desperdicio. Y el valor p confirma que hay diferencias significativas entre los algoritmos, en relación con el parámetro del stock usado.

En la Tabla 4.6 se aprecia la comparación del ABO-1DCSPMS bajo el parámetro del promedio de stock usado con desperdicio contra otros enfoques. Se puede observar que en las instancias 1, 3 y 10 el ABO-1DCSPMS alcanza el BKV. Para el resto de las instancias se posiciona como el segundo mejor método en el parámetro de stock usado con desperdicio. Manteniendo una diferencia menor a 1 con el método EP en las instancias 2, 4, 7 y 8. Mientras que en las instancias 5 y 9 el ABO-1DCSPMS tiene una diferencia entre 1 y 3 con respecto del método EP. La única instancia en la que el ABO-1DCSPMS alcanza una diferencia mayor a 3 contra el enfoque EP es en la sexta instancia. Lo cual nos indica que se mantiene muy cercanamente en la mayoría de las instancias en las que no alcanza el BKV. De igual manera, hay que destacar el hecho que en la instancia 10 la diferencia entre el ABO-1DCSPMS y el método SMBEP es de casi 15, siendo este último el segundo mejor enfoque en minimizar el número de stocks con desperdicio.

La Tabla 4.7 contiene los RPD promedio y el valor p de la prueba de Friedman en lo que se refiere al stock usado con desperdicio. Hay que mencionar que solamente se tomaron los valores de los métodos EP y SMBEP de la Tabla 4.6, debido a que el método Two-swap no registro resultado de las instancias 1 a 5. Como se puede observar, el ABO-1DCSPMS alcanzó la segunda posición, mientras que el valor p de la prueba de Friedman nos confirma que las diferencias son significativas al tener un

valor menor a 0.05.

Algoritmo 2: ABO-1DCSPMS

Input: instancia, lista_stocks, número_búfalos(nb), λ, iteraciones (k), lp1, lp2, número_iteraciones_reiniciar(q)

Output: mejor_bgmax

```
1  búfalos = CrearBúfalosAleatoriamente(instancia, nb);
2  búfalos = AsignarStocksBúfalos(búfalos, lista_stocks);
3  búfalos = CalcularDesperdicioStocksDesperdicio(búfalos);
4  bgmax = Buscarbgmax(búfalos);
5  j = 1;
6  i = 1;
7  while i ≤ k do
8      búfalos = ActualizarBúfalos(búfalos, nb, λ, lp1, lp2, instancia, bgmax);
9      búfalos = AsignarStocksBúfalos(búfalos, lista_stocks);
10     búfalos = CalcularDesperdicioStocksDesperdicio(búfalos);
11     bgmaxactualizado, bgmax = VerificarActualizaciónbgmax(búfalos,
        bgmax);
12     if bgmaxactualizado == Falso Y i % q == 0 then
13         lista_bgmax[j] = bgmax;
14         j = j + 1;
15         bgmax_B = Generarbgmax(búfalos, instancia);
16         búfalos =CrearBúfalosAleatoriamente(instancia, longitud_objeto, nb);
17         búfalos = AsignarStocksBúfalos(búfalos, lista_stocks);
18         búfalos = CalcularDesperdicioStocksDesperdicio(búfalos);
19         bgmax = bgmax_B;
20         i = i + 1;
21     else
22         i = i + 1;
23 lista_bgmax[j] = bgmax;
24 mejor_bgmax = BuscarMejorbgmax(lista_bgmax);
```

Tabla 4.1: Algoritmos empleados contra los que se comparó al ABO-1DCSPMS.

Algoritmo	Referencia	Medidas de Desempeño		
		Desperdicio Promedio	Stock Usado Promedio	Stock usado con desperdicio Promedio
EP	[1]	✓	✓	✓
Two-swap	[1]	✓	✓	✓
SMBEP	[66]	✓	✓	✓
GA-Lu	[21]		✓	
Pure-ACO	[21]		✓	
ACO-MCSP	[21]		✓	
GA-Parmar	[18]	✓		
LP	[18]	✓		

Tabla 4.2: Comparativa del ABO-1DCSPMS bajo el desperdicio promedio.

Instancia	BKV	ABO-1DCSPMS	EP	Two-swap	SMBEP	GA-Parmar	LP
1	0	0*	0*	-	1.9	0*	4
2	0	0.02	0*	-	5.8	0*	12
3	0	0*	0*	-	2	1	6
4	0	0.36	0*	-	8.1	4	6
5	0	252	0*	-	850	700	766
6	0.16	18.22	0.16*	21.08	32.6	61	79
7	4	4.7	4*	10.4	35.5	64	119
8	13.1	13.1*	17.8	215.7	73	181	210
9	3.7	9.2	3.7*	26.6	19	-	-
10	29.9	29.9*	208.2	260.9	206.5	-	-

Tabla 4.3: RPD Promedio y Friedman Test para el parámetro del desperdicio promedio del ABO-1DCSPMS y otros enfoques.

Algoritmo	ABO-1DCSPMS	EP	SMBEP	GA-Parmar	LP	Valor p
RPD Promedio:	51.7519	2.6404	96.2870	73.2812	98.7747	0.0003369
Ranking	2	1	4	3	5	

Tabla 4.4: Comparativa del ABO-1DCSPMS bajo variable de stock usado promedio.

Instancia	BKV	ABO-1DCSPMS	EP	Two-swap	SMBEP	GA-Lu	Pure-ACO	ACO-MCSP
1	9.84	9.84*	10.06	-	10.1	10.06	11.2	10.06
2	26.04	26.04*	27.04	-	28.1	27.35	29.14	27.18
3	19.02	19.02*	25.06	-	23.7	26.01	27.81	25.73
4	21.96	21.96*	23.1	-	24.5	23.1	24.51	23.1
5	54.7	54.7*	55.65	-	56.4	56.01	57.94	55.97
6	84.92	84.92*	86.72	90.2	87.1	-	-	-
7	72.92	72.92*	73.94	76.7	74.8	74.57	82.19	75.82
8	148.88	148.88*	151.1	155.74	151	151.92	179.07	153.57
9	162.94	162.94*	165.44	171	166.1	-	-	-
10	223.9	223.9*	228.92	234.88	227.7	230.08	245.79	230.45

Tabla 4.5: Comparativa del RPD promedio y prueba de Friedman para el stock usado promedio del ABO-1DCSPMS.

Algoritmo	ABO-1DCSPMS	EP	SMBEP	GA-Lu	Pure- ACO	ACO-MCSP	p value
RPD Promedio:	0	5.2088	6.0774	6.0030	13.4285	6.1731	6.396e-06
Ranking	1	2	4	3	6	5	

Tabla 4.6: Comparación del ABO-1DCSPMS bajo el stock usado con desperdicio promedio para el 1D-CSP con multistock

Instancia	BKV	ABO-1DCSPMS	EP	Two-swap	SMBEP
1	0	0*	0*	-	1.1
2	0	0.02	0*	-	2.4
3	0	0*	0*	-	1
4	0	0.26	0*	-	3
5	0	2.9	0*	-	7.3
6	0.16	6.6	0.16*	12.02	8.2
7	1	1.54	1*	3.16	3.4
8	1.96	2.56	1.96*	12.74	6.8
9	1.96	3.14	1.96*	6.72	5.8
10	5.66	5.66*	24.62	31.08	20.5

Tabla 4.7: RPD Promedio y Friedman test del ABO-1DCSPMS bajo la variable del stock usado con desperdicio

Algoritmo	ABO-1DCSPMS	EP	SMBEP	Valor p
RPD Promedio:	49.3657	7.7010	87.84	0.002409
Ranking	2	1	3	

Capítulo 5

Conclusiones

En este trabajo de investigación enfocado en la problemática del 1D-CSP con un solo stock y múltiples stocks. Se desarrollaron dos algoritmos basados en el enfoque metaheurístico de tipo enjambre ABO inspirado en el comportamiento del búfalo africano. El ABO ha sido previamente implementado en problemáticas de optimización combinatoria como lo son el TSP y el 1BPP, donde el 1BPP pertenece a al conjunto de problemas de corte y empaque, así como el 1D-CSP.

Los algoritmos realizados en este trabajo, aparte de implementar el ABO, hacen uso de la técnica de conversión ROV, debido a que las instancias empleadas son de tipo discreto y el ABO solo emite soluciones continuas. Asimismo, se propone el no reinicializar la manada de búfalos en cada iteración, aunque no se haya actualizado al bgmax (el mejor búfalo de la manada), sino que se da un cierto número de iteraciones, para que cada bgmax tenga la oportunidad de encontrar mejores soluciones. De igual forma se diseñó un método para generar un nuevo bgmax cuando se reinicie la manada,

con el fin de aprovechar el avance logrado por la propia manada, el método consiste en seleccionar los mejores patrones de corte que se tengan en la manada para construir un nuevo bgmax que guíe a la nueva manada que se va a generar.

Para poder comparar los algoritmos ABO-1DCSP y ABO-1DCSPMS se consideraron tres parámetros: stock usado, desperdicio y stock usado con desperdicio. Ambos métodos se contrastaron contra enfoques heurísticos, metaheurísticos e híbridos.

El ABO-1DCSP tiene como función objetivo el minimizar el desperdicio, bajo este parámetro el ABO-1DCSP demostró ser eficiente al posicionarse en primer lugar al calcular le RPD promedio, asimismo en 60 % de las instancias de [1] alcanzó la menor cantidad de desperdicio.

En relación con el parámetro del stock usado, el ABO-1DCSP en las instancias 1a a 10a de [1] logró en el 50 % de las instancias alcanzar al BKV y calculando el RPD promedio se posicionó en la segunda posición estando muy cerca del primer lugar, a pesar de que tiene como meta el minimizar el desperdicio. Asimismo, se comparó al ABO-1DCSP contra el N_{min} de las instancias 1a a 10a, en dicha comparación el ABO-1DCSP en el 50 % de estas alcanzó el N_{min}, mientras que en el 40 % requirió 1.39 % más stock que el N_{min} para cumplir con la demanda de ítems. Solo en una instancia empleo 3.92 % más stock que el N_{min}. Mientras que en las instancias de [3] el ABO-1DCSP en 35 de las 80 de las instancias empleo la misma cantidad de stock que el N_{min}, es decir, en el 43.75 % de las instancias. Entre tanto que en el 47.5 % de las instancias ocupó menos del 1 % más stock que el N_{min}. Y en el 8.75 % de las instancias requiere más del 0.99 % stock que el N_{min}, con lo cual el ABO-1DCSP demuestra nuevamente que es competente en el uso del stock para satisfacer

la demanda de ítems en otro conjunto de instancias. Para el caso de las instancias de [4] el ABO-1DCSP fue capaz de alcanzar en 26 de las 40 instancias el N_{min}, es decir, en el 65 % de las instancias. Por otra parte, en un 20 % de las instancias ocupo menos del 1 % más stock que el N_{min} y en el 15 % de las instancias requiere de 0.99 % más stock que el N_{min}. Demostrando que en un tercer set de instancias el ABO-1DCSP es capaz de obtener soluciones competentes en el aprovechamiento del stock para cumplir con la demanda de ítems. Tomando todo lo antes mencionado, se puede establecer que el ABO-1DCSP es capaz de generar arreglos de ítems con una eficiencia del stock adecuada bajo diferentes instancias de ítems, desde las más pequeñas que impliquen una cantidad de ítems de 20 hasta 1000 ítems. Recalcando que el ABO-1DCSP no considero el minimizar el stock en su función objetivo, sino el reducir el desperdicio.

Pasando el parámetro del stock con desperdicio, el ABO-1DCSP demostró que en las instancias 1a a 10a alcanzó el BKV en 2 de las 10 instancias. Entre tanto que en 6 instancias fue el segundo método en obtener una menor cantidad de stock con desperdicio. Al calcular el RPD promedio se posicionó en el segundo lugar. Recordando que el ABO-1DCSP no incluye en su función objetivo el minimizar el stock usado con desperdicio, caso contrario que en el enfoque de [1] si lo toma en cuenta.

En cuanto al algoritmo ABO-1DCSPMS, aparte de incorporar el método ROV y el proceso de generación del bgmax como se hace en el ABO-1DCSP, fue necesario el establecer un método para asignar el stock de acuerdo al arreglo de ítems, debido a que se consideró el resolver el 1D-CSP con múltiples stocks. Este proceso de asignación de

stocks básicamente consta de ir generando los diferentes patrones de corte de acuerdo al arreglo de ítems que se obtenía junto con los diferentes stocks con los que se cuenta para así poder seleccionar aquel patrón de corte que implicará una menor cantidad de desperdicio y un mayor número de ítems. Para la función objetivo se cambió por la empleada en [1] debido a que consideraba dos parámetros, desperdicio y stock con desperdicio.

El ABO-1DCSPMS bajo el parámetro de desperdicio en las instancias 1 a 10 de [1] en 40 % de las instancias alcanzó el BKV, destacando que en la instancia 10 obtuvo un desperdicio promedio bastante bajo, casi siete veces más bajo que el más cercano al obtenido por el ABO-1DCPMS. Mientras que en el resto de las instancias se posicionó como el segundo mejor enfoque en minimizar el desperdicio. Al calcular el RPD promedio, el ABO-1DCSPMS obtuvo la segunda posición y junto con la prueba de Friedman se reafirmó que las diferencias entre los métodos son relevantes.

Para el caso del stock usado, el ABO-1DCSPMS en todas las instancias de 1 a 10 de [1] consiguió la menor cantidad de stock usado, es decir, se obtuvieron nuevos BKV para las antes mencionadas instancias. Al calcular el RPD promedio se observa que el ABO-1DCSPMS alcanza la primera posición y junto con la prueba de Friedman se afirma que las diferencias entre los enfoques es relevante.

Con relación al parámetro de stock con desperdicio, el ABO-1DCSPMS en 3 de las 10 instancias alcanza el BKV, sobresaliendo en la instancia 10 de [1] donde consiguió 3.6 veces menos stock desperdicio que el segundo mejor enfoque. Al calcular el RPD promedio, el ABO-1DCSPMS se colocó en la segunda posición y con la prueba de Friedman se verificó que las diferencias entre los métodos comparados son significati-

vas.

Tomando todo lo antes mencionado con el ABO-1DCSPMS, se demuestra que es un enfoque eficiente en los parámetros de desperdicio y stock usado, destacando el hecho de ser el único método que en todas las instancias alcanza el BKV relacionado con el stock usado. Asimismo, no requiere de otro enfoque heurístico o metaheurístico para conseguir soluciones eficientes.

Con base en lo antes expuesto de los métodos ABO-1DCSP y ABO-1DCSPMS, se puede inferir que el ABO es un algoritmo metaheurístico que es fácil de implementar en el 1D-CSP en los casos de un solo y múltiples stocks, asimismo es capaz de generar soluciones competentes que generan una cantidad de desperdicio mínima en comparación de otros métodos heurísticos, metaheurísticos e híbridos, de igual forma emplean una cantidad de stock adecuada en el 1D-CSP con un solo tipo de stock, pero en el caso de múltiples stocks es todavía mucho menor la cantidad de stock necesario para cumplir con la demanda de ítems. Sobre los enfoques ABO-1DCSP y ABO-1DCSPMS se puede constatar que al implementar el ABO en la problemática del 1D-CSP no requieren de otro algoritmo para poder conseguir soluciones eficientes.

5.0.1 Trabajos Futuros

Sería interesante comprobar si el ABO mantiene esa eficiencia en otras variantes de los problemas de corte o empaque. También podrían probarse los enfoques del ABO-1DCSP y ABO-1DCSPMS con instancias de la industria o considerando otros parámetros como los tiempos de inventario o el desgaste de las cuchillas.

Apéndice Banco de Pruebas

En las Tablas 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9 y 5.10 se presentan las instancias de prueba obtenidas de [2] y [1], las cuales fungirán como banco de pruebas. Como se puede apreciar en cada tabla se encuentran dos problemas, en uno se podrá contar con diferentes tipos de stocks, mientras en el otro solo se podrá emplear un solo tipo de stock. Asimismo, en ambos problemas se deberán considerar los mismos ítems con sus mismas frecuencias.

Problema 1. Longitud de los objetos 10, 13 y 15.								
Problema 1a. Longitud de los objetos 14.								
Total de ítems: 20 ítems requeridos por problema								
Longitud de cada ítem	3	4	5	6	7	8	9	10
Numero requerido por ítem	5	2	1	2	4	2	1	3

Tabla 5.1: Problemas 1 y 1a obtenidos de [2]

Problema 2. Longitud de los objetos 10, 13 y 15.								
Problema 2a. Longitud de los objetos 15.								
Total de ítems: 50 ítems requeridos por problema								
Longitud de cada ítem	3	4	5	6	7	8	9	10
Numero requerido por ítem	4	8	5	7	8	5	5	8

Tabla 5.2: Problemas 2 y 2a obtenidos de [2]

Problema 3. Longitud de los objetos 10, 13, 15, 20, 22 y 25.								
Problema 3a. Longitud de los objetos 25.								
Total de ítems: 60 ítems requeridos por problema								
Longitud de cada ítem	3	4	5	6	7	8	9	10
Numero requerido por ítem	6	12	6	5	15	6	4	6

Tabla 5.3: Problemas 3 y 3a obtenidos de [2]

Problema 4. Longitud de los objetos 13, 20 y 25.								
Problema 4a. Longitud de los objetos 25.								
Total de ítems: 60 ítems requeridos por problema								
Longitud de cada ítem	5	6	7	8	9	10	11	12
Numero requerido por ítem	7	12	15	7	4	6	8	1

Tabla 5.4: Problemas 4 y 4a obtenidos de [2]

Problema 5. Longitud de los objetos 4300, 4250, 4150, 3950, 3800, 3700, 3550 y 3500.									
Problema 5a. Longitud de los objetos 4300.									
Total de ítems: 126 ítems requeridos por problema									
Longitud de cada ítem	2350	2250	2220	2100	2050	2000	1950	1900	1850
Numero requerido por ítem	2	4	4	15	6	11	6	15	13
Longitud de cada ítem	1700	1650	1350	1300	1250	1200	1150	1100	1050
Numero requerido por ítem	5	2	9	3	6	10	4	8	3

Tabla 5.5: Problemas 5 y 5a obtenidos de [2]

Problema 6. Longitud de los objetos 86, 83, 79, 76 y 72.									
Problema 6a. Longitud de los objetos 86.									
Total de ítems: 200 ítems requeridos por problema									
Longitud de cada ítem	21	23	24	25	26	27	28	29	31
Numero requerido por ítem	10	14	10	7	14	4	13	9	5
Longitud de cada ítem	33	34	35	37	38	41	42	44	47
Numero requerido por ítem	10	13	10	11	15	12	15	15	13

Tabla 5.6: Problemas 6 y 6a obtenidos de [1]

Problema 7. Longitud de los objetos 120, 115, 110, 105 y 100.												
Problema 7a. Longitud de los objetos 120.												
Total de ítems: 200 ítems requeridos por problema												
Longitud de cada ítem	22	26	27	28	29	30	31	32	34	36	37	38
Numero requerido por ítem	6	3	14	12	9	15	11	10	11	13	4	3
Longitud de cada ítem	39	46	47	48	52	53	54	56	58	60	63	64
Numero requerido por ítem	6	14	7	3	14	9	7	3	5	14	4	3

Tabla 5.7: Problemas 7 y 7a obtenidos de [1]

Problema 8. Longitud de los objetos 120, 115 y 110.												
Problema 8a. Longitud de los objetos 120.												
Total de ítems: 400 ítems requeridos por problema												
Longitud de cada ítem	22	23	24	26	27	28	29	30	31	36	39	41
Numero requerido por ítem	12	8	27	15	25	7	10	22	5	16	19	21
Longitud de cada ítem	42	48	49	50	51	54	55	56	59	60	66	67
Numero requerido por ítem	26	16	12	26	20	25	9	17	22	14	17	9

Tabla 5.8: Problemas 8 y 8a obtenidos de [1]

Problema 9. Longitud de los objetos 120, 115, 110, 105 y 100.												
Problema 9a. Longitud de los objetos 120.												
Total de ítems: 400 ítems requeridos por problema												
Longitud de cada ítem	21	22	24	25	27	29	30	31	32	33	34	35
Numero requerido por ítem	13	15	7	5	9	9	3	15	18	17	4	17
Longitud de cada ítem	38	39	42	44	45	46	47	48	49	50	51	52
Numero requerido por ítem	20	9	4	19	9	12	15	3	20	14	15	6
Longitud de cada ítem	53	54	55	56	57	59	60	61	63	65	66	67
Numero requerido por ítem	4	7	5	19	19	6	3	7	20	5	10	17

Tabla 5.9: Problemas 9 y 9a obtenidos de [1]

En las Tablas 5.11, 5.12, 5.13 y 5.14 se exponen las instancias de prueba obtenidas de [3]. Se puede observar que en cada tabla se presenta la longitud del stock implicado en cada set. De igual forma se menciona la cantidad de ítems que está entre 120 y 1000. Asimismo, puede llegar a variar el número necesario de ítems. Cada ítem puede tener una longitud entre 20 y 100.

En las Tablas 5.15 y 5.16 se muestran las instancias de [4] empleadas para probar la eficiencia del método ABO-1DCSP. Como se puede apreciar las instancias relacionadas con el Set 1 de [4] consideran dos tipos de stock, así como una cantidad de ítems entre 50 y 500 ítems, para el número de necesario por ítems oscila entre 1 y 17 y en relación con las longitudes de los ítems, estas varían entre 1 y 100. Mientras que en las instancias del Set 2 de [4] contemplan un solo tipo de stock de 1000, manteniendo una cantidad de ítems entre 50 y 500, en cuanto al número de ítems requerido puede ser entre 1 y 22, pero cambiando notablemente en las longitudes de los ítems, las cuales pueden ser desde 11 hasta 490.

Problema 10. Longitud de los objetos 120, 115 y 110.												
Problema 10a. Longitud de los objetos 120.												
Total de ítems: 600 ítems requeridos por problema												
Longitud de cada ítem	21	22	23	24	25	27	28	29	30	31	33	35
Numero requerido por ítem	13	19	24	20	23	24	15	5	24	16	12	24
Longitud de cada ítem	36	39	40	41	42	43	44	45	46	47	48	50
Numero requerido por ítem	16	4	20	24	6	14	21	20	24	2	11	26
Longitud de cada ítem	51	54	56	57	58	61	62	63	64	65	66	67
Numero requerido por ítem	23	25	8	16	10	14	6	19	18	11	27	16

Tabla 5.10: Problemas 10 y 10a obtenidos de [1]

Sets de u120_00 − u120_19	
Longitud del stock u objetos:	150
Total de ítems:	120
Número mínimo - máximo requerido por ítems:	1- 13
Instancia	Longitud Mínima - Máxima de Ítems
u120_00	20 - 98
u120_01	20 - 100
u120_02	20 - 100
u120_03	20 - 100
u120_04	20 - 99
u120_05	20 - 100
u120_06	20 - 100
u120_07	20 - 100
u120_08	20 - 100
u120_09	20 - 100
u120_10	20 - 100
u120_11	21 - 100
u120_12	20 - 99
u120_13	20 - 100
u120_14	21 - 100
u120_15	20 - 100
u120_16	21 - 100
u120_17	22 - 100
u120_18	21 - 100
u120_19	21 - 100

Tabla 5.11: Instancias u120_00 - u120_19 [3]

Sets de u250_00 − u250_19	
Longitud del stock u objetos:	150
Total de ítems:	250
Número mínimo - máximo requerido por ítems:	1- 13
Instancia	Longitud Mínima - Máxima de Ítems
u250_00	20 - 100
u250_01	20 - 100
u250_02	20 - 100
u250_03	20 - 100
u250_04	20 - 100
u250_05	20 - 100
u250_06	20 - 100
u250_07	20 - 100
u250_08	21 - 100
u250_09	21 - 100
u250_10	20 - 100
u250_11	20 - 100
u250_12	20 - 100
u250_13	20 - 100
u250_14	20 - 100
u250_15	20 - 100
u250_16	20 - 100
u250_17	20 - 100
u250_18	20 - 100
u250_19	20 - 100

Tabla 5.12: Instancias u250_00 - u250_19 [3]

Sets de u500_00 − u500_19	
Longitud del stock u objetos:	150
Total de ítems:	500
Número mínimo - máximo requerido por ítems:	1- 13
Instancia	Longitud Mínima - Máxima de Ítems
u500_00	20 - 100
u500_01	20 - 100
u500_02	20 - 100
u500_03	20 - 100
u500_04	21 - 100
u500_05	20 - 100
u500_06	20 - 100
u500_07	20 - 100
u500_08	20 - 100
u500_09	20 - 100
u500_10	20 - 100
u500_11	20 - 100
u500_12	20 - 100
u500_13	20 - 100
u500_14	20 - 100
u500_15	20 - 100
u500_16	20 - 100
u500_17	20 - 100
u500_18	20 - 100
u500_19	20 - 100

Tabla 5.13: Instancias u500_00 - u500_19 [3]

Sets de u1000_00 − u1000_19	
Longitud del stock u objetos:	150
Total de ítems:	1000
Número mínimo - máximo requerido por ítems:	1- 13
Instancia	Longitud Mínima - Máxima de Ítems
u1000_00	20 - 100
u1000_01	20 - 100
u1000_02	20 - 100
u1000_03	20 - 100
u1000_04	20 - 100
u1000_05	20 - 100
u1000_06	20 - 100
u1000_07	20 - 100
u1000_08	20 - 100
u1000_09	20 - 100
u1000_10	20 - 100
u1000_11	20 - 100
u1000_12	20 - 100
u1000_13	20 - 100
u1000_14	20 - 100
u1000_15	20 - 100
u1000_16	20 - 100
u1000_17	20 - 100
u1000_18	20 - 100
u1000_19	20 - 100

Tabla 5.14: Instancias u1000_00 - u1000_19 [3]

Instancias del Set 1	
Longitud del stock u objetos:	100 - 120
Total de ítems:	50 - 500
Número mínimo - máximo requerido por ítems:	1- 17
Instancia	Longitud Mínima - Máxima de Ítems
N1C1W1_A	3 - 99
N1C1W1_D	2 - 99
N1C1W1_G	4 - 100
N1C1W1_B	8 - 100
N1C1W1_E	2 - 91
N1C1W1_F	2 - 99
N1C1W1_I	2 - 95
N2C1W1_Q	1 - 99
N2C1W2_P	20 - 99
N2C1W2_N	20 - 100
N2C1W2_O	20 - 100
N2C1W2_R	20 - 100
N3C1W1_A	1 - 100
N3C2W2_D	20 - 100
N3C2W4_B	30 - 100
N4C1W2_T	20 - 100
N4C1W4_C	30 - 100
N4C1W4_A	30 -100
N4C1W4_D	30 - 100
N4C1W4_B	30 - 100

Tabla 5.15: Instancias del Set 1 [4]

Instancias del Set 2	
Longitud del stock u objetos:	1000
Total de ítems:	50 - 500
Número mínimo - máximo requerido por ítems:	1- 22
Instancia	Longitud Mínima - Máxima de Ítems
N1W1B2R1	169 - 483
N1W1B1R9	269 - 394
N1W1B1R2	271 - 396
N1W1B2R0	173 - 494
N1W1B2R3	168 - 490
N2W1B1R0	267 - 393
N2W1B1R3	266 - 396
N2W1B1R1	266 - 393
N2W1B1R4	266 - 394
N2W3B3R7	16 - 260
N2W4B1R0	90 - 132
N3W2B2R3	102 - 300
N3W3B1R3	114 - 168
N3W4B1R1	90 - 132
N3W4B2R1	57 - 165
N4W2B1R0	162 - 240
N4W2B1R3	162 - 240
N4W3B3R7	16 - 265
N4W4B1R0	90 - 132
N4W4B1R1	11 - 132

Tabla 5.16: Instancias del Set 2 [4]

Referencias

[1] K.-H. Liang, X. Yao, C. Newton, and D. Hoffman, "A new evolutionary approach to cutting stock problems with and without contiguity," *Computers & Operations Research*, vol. 29, no. 12, pp. 1641–1659, 2002.

[2] R. Hinterding and L. Khan, "Genetic algorithms for cutting stock problems: with and without contiguity," in *Progress in evolutionary computation*, pp. 166–186, Springer, 1993.

[3] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of heuristics*, vol. 2, pp. 5–30, 1996.

[4] A. Scholl, R. Klein, and C. Jürgens, "Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem," *Computers & Operations Research*, vol. 24, no. 7, pp. 627–645, 1997.

[5] L. V. Kantorovich, "Mathematical methods of organizing and planning production," *Management science*, vol. 6, no. 4, pp. 366–422, 1960.

[6] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Operations research*, vol. 9, no. 6, pp. 849–859, 1961.

[7] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting stock problem—part ii," *Operations research*, vol. 11, no. 6, pp. 863–888, 1963.

[8] D. Tanir, O. Ugurlu, A. Guler, and U. Nuriyev, "One-dimensional cutting stock problem with divisible items," *arXiv preprint arXiv:1606.01419*, 2016.

[9] M. H. Jahromi, R. Tavakkoli-Moghaddam, A. Makui, and A. Shamsi, "Solving an one-dimensional cutting stock problem by simulated annealing and tabu search," *Journal of Industrial Engineering International*, vol. 8, no. 1, pp. 1–8, 2012.

[10] H. Dyckhoff, "A typology of cutting and packing problems," *European Journal of Operational Research*, vol. 44, no. 2, pp. 145–159, 1990.

[11] A. C. Dikili and B. Barlas, "A generalized approach to the solution of one-dimensional stock-cutting problem for small shipyards," *Journal of marine science and technology*, vol. 19, no. 4, pp. 368–376, 2011.

[12] H. Stadtler, "A one-dimensional cutting stock problem in the aluminium industry and its solution," *European Journal of Operational Research*, vol. 44, no. 2, pp. 209–223, 1990.

[13] A. de Araujo Machado, J. C. Zayatz, G. M. Neto, G. C. L. Leal, and R. H. P. Lima, "Aluminum bar cutting optimization for door and window manufactur-

ing," *DYNA: revista de la Facultad de Minas. Universidad Nacional de Colombia. Sede Medellín*, vol. 87, no. 212, pp. 155–162, 2020.

[14] A. Zanarini, "Optimal stock sizing in a cutting stock problem with stochastic demands," in *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pp. 293–301, Springer, 2017.

[15] G. A. Ogunranti and A. E. Oluleye, "Minimizing waste (off-cuts) using cutting stock model: The case of one dimensional cutting stock problem in wood working industry," *Journal of Industrial Engineering and Management*, vol. 9, no. 3, pp. 834–859, 2016.

[16] C. Gracia, C. Andrés, and L. Gracia, "A hybrid approach based on genetic algorithms to solve the problem of cutting structural beams in a metalwork company," *Journal of Heuristics*, vol. 19, no. 2, pp. 253–273, 2013.

[17] D. Díaz, P. Valledor, P. Areces, J. Rodil, and M. Suárez, "An aco algorithm to solve an extended cutting stock problem for scrap minimization in a bar mill," in *Swarm Intelligence* (M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. Montes de Oca, C. Solnon, and T. Stützle, eds.), (Cham), pp. 13–24, Springer International Publishing, 2014.

[18] K. B. Parmar, H. B. Prajapati, and V. K. Dabhi, "Cutting stock problem: A solution based on novel pattern based chromosome representation using modified ga," in *2015 International Conference on Cir-*

cuits, Power and Computing Technologies [ICCPCT-2015], pp. 1–7, 2015. https://doi.org/10.1109/ICCPCT.2015.7159318.

[19] Jin Peng and Zhang Shu Chu, "A hybrid ant colony algorithm for the cutting stock problem," in *2010 International Conference on Future Information Technology and Management Engineering*, vol. 2, pp. 32–35, 2010.

[20] H. K. Alfares and O. G. Alsawafy, "A least-loss algorithm for a bi-objective one-dimensional cutting-stock problem," *International Journal of Applied Industrial Engineering (IJAIE)*, vol. 6, no. 2, pp. 1–19, 2019.

[21] Q. Lu, Z. Wang, and M. Chen, "An ant colony optimization algorithm for the one-dimensional cutting stock problem with multiple stock lengths," in *2008 Fourth International Conference on Natural Computation*, vol. 7, pp. 475–479, 2008.

[22] K. C. Poldi and S. A. de Araujo, "Mathematical models and a heuristic method for the multiperiod one-dimensional cutting stock problem," *Annals of Operations Research*, vol. 238, no. 1, pp. 497–520, 2016.

[23] Y.-H. Chen, H.-C. Huang, H.-Y. Cai, and P.-F. Chen, "A genetic algorithm approach for the multiple length cutting stock problem," in *2019 IEEE 1st Global Conference on Life Sciences and Technologies (LifeTech)*, pp. 162–165, 2019.

[24] P. T. H. Phuong, "Hybridization of genetic algorithm and branch-and-price framework for solving the one dimensional cutting stock problem with multiple stock sizes," in *2012 IEEE RIVF International Conference on Computing &*

Communication Technologies, Research, Innovation, and Vision for the Future, pp. 1–5, 2012.

[25] J. Peng and Z. S. Chu, "A hybrid multi-chromosome genetic algorithm for the cutting stock problem," in *2010 3rd International Conference on Information Management, Innovation Management and Industrial Engineering*, vol. 1, pp. 508–511, 2010.

[26] J. Levine and F. Ducatelle, "Ant colony optimization and local search for bin packing and cutting stock problems," *Journal of the Operational Research Society*, vol. 55, no. 7, pp. 705–716, 2004.

[27] Y.-P. Cui and T.-B. Tang, "Parallelized sequential value correction procedure for the one-dimensional cutting stock problem with multiple stock lengths," *Engineering Optimization*, vol. 46, no. 10, pp. 1352–1368, 2014.

[28] Y. Cui, Y.-P. Cui, and Z. Zhao, "Pattern-set generation algorithm for the one-dimensional multiple stock sizes cutting stock problem," *Engineering Optimization*, vol. 47, no. 9, pp. 1289–1301, 2015.

[29] N. Ma, Y. Liu, and Z. Zhou, "Two heuristics for the capacitated multi-period cutting stock problem with pattern setup cost," *Computers & Operations Research*, vol. 109, pp. 218–229, 2019.

[30] J. B. Odili, M. N. M. Kahar, and S. Anwar, "African buffalo optimization: a swarm-intelligence technique," *Procedia Computer Science*, vol. 76, pp. 443–448, 2015.

[31] J. B. Odili, M. N. M. Kahar, S. Anwar, and M. A. K. Azrag, "A comparative study of african buffalo optimization and randomized insertion algorithm for asymmetric travelling salesman's problem," in *2015 4th International Conference on Software Engineering and Computer Systems (ICSECS)*, pp. 90–95, IEEE, 2015.

[32] J. B. Odili and M. N. Mohmad Kahar, "Solving the traveling salesman's problem using the african buffalo optimization," *Computational intelligence and neuroscience*, vol. 2016, 2016.

[33] A. Gherboudj, "African buffalo optimization for one dimensional bin packing problem," *International Journal of Swarm Intelligence Research (IJSIR)*, vol. 10, no. 4, pp. 38–52, 2019.

[34] G. María D, M.-O. Julio, and L.-C. Oscar, "Análisis computacional de los problemas del vendedor viajero y patrones de corte," *Ingeniería, Investigación y Tecnología*, vol. 16, no. 1, pp. 59–70, 2015.

[35] E. Sheehan, A.-T. Braun, T. Kuhlmann, and A. Sauer, "Improving material efficiency for ultra-efficient factories in closed-loop value networks," *Procedia CIRP*, vol. 40, pp. 455–462, 2016.

[36] G. Evtimov and S. Fidanova, "Ant colony optimization algorithm for 1d cutting stock problem," in *Advanced Computing in Industrial Mathematics*, pp. 25–31, Springer, 2018.

[37] S. V. Ravelo, C. N. Meneses, and M. O. Santos, "Meta-heuristics for the one-dimensional cutting stock problem with usable leftover," *Journal of Heuristics*, vol. 26, no. 4, pp. 585–618, 2020.

[38] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Dover Publications, Inc, 1982.

[39] D. S. Johnson and M. R. Garey, *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman, 1979.

[40] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM computing surveys (CSUR)*, vol. 35, p. 268–308, sep 2003.

[41] G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1109–1130, 2007.

[42] G. Scheithauer, *Introduction to cutting and packing optimization: Problems, modeling approaches, solution methods*, vol. 263. Springer, 2017.

[43] H. Sarper and N. I. Jaksic, "Simulation of the stochastic one-dimensional cutting stock problem to minimize the total inventory cost," *Procedia Manufacturing*, vol. 38, pp. 916–923, 2019.

[44] H. A. Taha, *Investigación de Operaciones*. Pearson Educación, 2012.

[45] C. Zheng and M. Lu, "Optimized reinforcement detailing design for sustainable construction: Slab case study," *Procedia Engineering*, vol. 145, pp. 1478–1485, 2016.

[46] M. Gradišar, G. Resinovič, and M. Kljajić, "A hybrid approach for optimization of one-dimensional cutting," *European Journal of Operational Research*, vol. 119, no. 3, pp. 719–728, 1999.

[47] G. Belov and G. Scheithauer, "A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths," *European Journal of Operational Research*, vol. 141, no. 2, pp. 274–294, 2002.

[48] A. C. Dikili, E. Sarıöz, and N. A. Pek, "A successive elimination method for one-dimensional stock cutting problems in ship production," *Ocean engineering*, vol. 34, no. 13, pp. 1841–1849, 2007.

[49] K. C. Poldi and M. N. Arenales, "Heuristics for the one-dimensional cutting stock problem with limited multiple stock lengths," *Computers & Operations Research*, vol. 36, no. 6, pp. 2074–2081, 2009.

[50] Y. Cui, "A cam system for one-dimensional stock cutting," *Advances in Engineering Software*, vol. 47, no. 1, pp. 7–16, 2012.

[51] Y. Cui, C. Zhong, and Y. Yao, "Pattern-set generation algorithm for the one-dimensional cutting stock problem with setup cost," *European Journal of Operational Research*, vol. 243, no. 2, pp. 540–546, 2015.

[52] M. Garraffa, F. Salassa, W. Vancroonenburg, G. Vanden Berghe, and T. Wauters, "The one-dimensional cutting stock problem with sequence-dependent cut losses," *International Transactions in Operational Research*, vol. 23, no. 1-2, pp. 5–24, 2016.

[53] J. L. Santos, J. Santos, M. J. Ferreira, N. Alves, and M. Guevara, "Application of the two-stage one-dimensional cutting stock problem in the steel industry," in *2018 IEEE 27th International Symposium on Industrial Electronics (ISIE)*, pp. 683–690, 2018.

[54] H. Sarper and N. I. Jaksic, "Evaluation of procurement scenarios in one-dimensional cutting stock problem with a random demand mix," *Procedia Manufacturing*, vol. 17, pp. 827–834, 2018.

[55] D. Lee, S. Son, D. Kim, and S. Kim, "Special-length-priority algorithm to minimize reinforcing bar-cutting waste for sustainable construction," *Sustainability*, vol. 12, no. 15, 2020. https://doi.org/10.3390/su12155950.

[56] B. Campello, C. Ghidini, A. Ayres, and W. Oliveira, "A residual recombination heuristic for one-dimensional cutting stock problems," *TOP*, pp. 1–27, 2021. https://doi.org/10.1007/s11750-021-00611-3.

[57] G. R. Cerqueira, S. S. Aguiar, and M. Marques, "Modified greedy heuristic for the one-dimensional cutting stock problem," *Journal of Combinatorial Optimization*, pp. 1–18, 2021. https://doi.org/10.1007/s10878-021-00695-4.

[58] D. Morillo-Torres, M. T. Baena, J. W. Escobar, A. R. Romero-Conrado, J. R. Coronado-Hernández, and G. Gatica, "A mixed-integer linear programming model for the cutting stock problem in the steel industry," in *Applied Computer Sciences in Engineering* (J. C. Figueroa-García, Y. Díaz-Gutierrez, E. E. Gaona-García, and A. D. Orjuela-Cañón, eds.), pp. 315–326, Springer International Publishing, 2021. https://doi.org/10.1007/978-3-030-86702-7_27.

[59] R. Vishwakarma and P. Powar, "An efficient mathematical model for solving one-dimensional cutting stock problem using sustainable trim," *Advances in Industrial and Manufacturing Engineering*, vol. 3, p. 100046, 2021. https://doi.org/10.1016/j.aime.2021.100046.

[60] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986. Applications of Integer Programming.

[61] J. D. A. Petrowski and P. S. E. Taillard, "Metaheuristics for hard optimization," *Springer*, 2006.

[62] M. Gendreau, J.-Y. Potvin, *et al.*, *Handbook of metaheuristics*, vol. 2. Springer, 2010.

[63] A. A. Shahin and O. M. Salem, "Using genetic algorithms in solving the one-dimensional cutting stock problem in the construction industry," *Canadian journal of civil engineering*, vol. 31, no. 2, pp. 321–332, 2004.

[64] K. Eshghi and H. Javanshir, "An aco algorithm for one-dimensional cutting stock problem," *Journal of Industrial Engineering, International*, vol. 1, no. 1, pp. 10–19, 2005.

[65] C.-T. Yang, T.-C. Sung, and W.-C. Weng, "An improved tabu search approach with mixed objective function for one-dimensional cutting stock problems," *Advances in Engineering Software*, vol. 37, no. 8, pp. 502–513, 2006.

[66] R. Chiong, Yang Yaw Chang, Pui Chang Chai, and Ai Leong Wong, "A selective mutation based evolutionary programming for solving cutting stock problem without contiguity," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 1671–1677, 2008.

[67] K. B. Parmar, H. B. Prajapati, and V. K. Dabhi, "Cutting stock problem: A survey of evolutionary computing based solution," in *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, pp. 1–6, 2014.

[68] T. Asvany, J. Amudhavel, and P. Sujatha, "One-dimensional cutting stock problem with single and multiple stock lengths using dpso," *Advances and Applications in Mathematical Sciences*, vol. 17, no. 1, pp. 147–163, 2017.

[69] Y.-H. Chen, H.-C. Huang, H.-Y. Cai, and P.-F. Chen, "A genetic algorithm approach for the multiple length cutting stock problem," in *2019 IEEE 1st Global Conference on Life Sciences and Technologies (LifeTech)*, pp. 162–165, IEEE, 2019.

[70] L. J. Montiel-Arrieta, I. Barragán-Vite, N. Hernández-Romero, M. González-Hernández, *et al.*, "Algoritmo del búfalo africano para resolver el problema de corte unidimensional," *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, vol. 10, no. Especial2, pp. 1–8, 2022.

[71] I. Barragan-Vite, L. J. Montiel-Arrieta, J. C. Seck-Tuoh-Mora, N. Hernández-Romero, J. Medina-Marin, *et al.*, "Algoritmo discreto del búfalo africano para resolver el problema de corte de material," *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, vol. 11, no. Especial3, pp. 123–132, 2023.

[72] L. J. Montiel-Arrieta, I. Barragan-Vite, J. C. Seck-Tuoh-Mora, N. Hernandez-Romero, M. González-Hernández, and J. Medina-Marin, "Minimizing the total waste in the one-dimensional cutting stock problem with the african buffalo optimization algorithm," *PeerJ Computer Science*, vol. 9, p. e1728, 2023.

[73] G. Scheithauer, J. Terno, A. Müller, and G. Belov, "Solving one-dimensional cutting stock problems exactly with a cutting plane algorithm," *Journal of the Operational Research Society*, vol. 52, no. 12, pp. 1390–1401, 2001.

[74] R. A. Maher, N. N. Melhem, and M. Almutlaq, "Developing a control and management system for reinforcement steel-leftover in industrial factories," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 625–629, 2019.

[75] A. R. Pitombeira-Neto and B. de Athayde Prata, "A matheuristic algorithm for the one-dimensional cutting stock and scheduling problem with heterogeneous orders," *Top*, pp. 1–15, 2019.

[76] J. B. Odili, M. N. Mohmad Kahar, and A. Noraziah, "Parameters-tuning of pid controller for automatic voltage regulators using the african buffalo optimization," *PLOS ONE*, vol. 12, pp. 1–17, 04 2017. https://doi.org/10.1371/journal.pone.0175901.

[77] F. S. Madenoğlu, *Solving Optimization Problem with Particle Swarm Optimization: Solving Hybrid Flow Shop Scheduling Problem with Particle Swarm Optimization Algorithm*, pp. 263–277. Springer International Publishing, 2021. https://doi.org/10.1007/978-3-030-70281-6_14.

[78] R. Ruiz, E. Vallada, and C. Fernández-Martínez, *Scheduling in Flowshops with No-Idle Machines*, pp. 21–51. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. https://doi.org/10.1007/978-3-642-02836-6_2".

[79] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011. https://doi.org/10.1016/j.swevo.2011.02.002.

[80] N. J. E. Serna, J. C. Seck-Tuoh-Mora, J. Medina-Marin, N. Hernandez-Romero, I. Barragan-Vite, and J. R. C. Armenta, "A global-local neighborhood search algorithm and tabu search for flexible job shop scheduling problem," *PeerJ Computer Science*, vol. 7, p. e574, 2021.