DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI
ISSN: 2007-6363

Publicación Semestral Pädi Vol. 11 No. Especial 3 (2023) 123–132

Algoritmo discreto del búfalo africano para resolver el problema de corte de material de una dimensión
Discrete african buffalo algorithm to solve the one-dimensional cutting stock problem

I. Barragan-Vite [a,*], L. J. Montiel-Arrieta [a], J. C. Seck-Tuoh-Mora [a], N. Hernandez-Romero [a], J. Medina-Marin [a]

[a] *Área Académica de Ingeniería y Arquitectura, Universidad Autónoma del Estado de Hidalgo, 42184, Pachuca, Hidalgo, México.*

Resumen

El problema de corte abordado en este documento consiste en minimizar el desperdicio total producido al cortar un conjunto de piezas pequeñas en una secuencia determinada a partir de piezas de material más grandes. El algoritmo del búfalo Africano ha sido empleado exitosamente para resolver problemas de tipo combinatorio. Una de las dificultades de este algoritmo es generar soluciones discretas para esta clase de problemas discretos. En este trabajo se emplea una variante discreta del algoritmo del búfalo Africano en la que se compara una técnica de cruza así como la técnica del valor del orden clasificado para obtener soluciones discretas. Se usa un conjunto de diez instancias de diferente complejidad para realizar la comparación de estas técnicas. Los resultados muestran que la técnica de cruza supera a las otras en cuanto a la calidad de las soluciones. Luego, esta técnica de cruza se compara contra la heurística *least-loss-algorithm* y la metaheurística ABO-CSP para evaluar su desempeño.

Palabras Clave: problema de corte, metaheurística, discretización, problema combinatorio.

Abstract

The cutting stock problem addressed in this document consists of minimizing the total waste produced by cutting a set of small parts in a given sequence from larger pieces of material. The African buffalo algorithm has successfully solved combinatorial problems. One of the difficulties of this algorithm is generating discrete solutions for this class of discrete problems. In this work, a discrete variant of the African buffalo algorithm is used, in which crossover technique and the ranked order value technique are compared to obtain discrete solutions. A set of ten instances of different complexity is used to compare these techniques. The results show that the crossover technique surpasses the others in terms of the quality of the solutions. Then this crossover technique is compared with the *least-loss-algorithm* heuristic and the ABO-CSP metaheuristic to evaluate its performance.

Keywords: Cutting stock problem, metaheuristics, discretization, combinatorial problem.

1. Introducción

El problema de corte de material es uno de los problemas de optimización más abordados en la literatura, así como uno de los problemas que enfrentan diferentes industrias como la del papel, el vidrio, la madera, el acero y otras, que requieren obtener piezas a partir del corte de grandes piezas de materia prima. El problema consiste principalmente en minimizar el desperdicio total producido al cortar la cantidad requerida de piezas pequeñas. De acuerdo con (Fang *et al.*, 2023) las compañías relacionadas con este problema pueden reducir costos a la vez que contribuyen a un entorno sustentable al usar adecuadamente los recursos para realizar su actividad. En (Ogunranti y Oluleye, 2016) se desarrolla una aplicación computacional para resolver el 1D-CSP mediante la solución de un modelo de PL con el método simplex revisado y un algoritmo para generar patrones de corte factibles. La aplicación computacional se emplea para resolver problemas de la industria de la madera obteniendo ahorros económicos aproximados del 30.7 %. En (Lee *et al.*, 2020) se aborda el problema del corte de barras de refuerzo usadas en la industria de la construcción mediante un algoritmo para resolver el problema de minimización del desperdicio debida al corte, que es similar al problema de corte de material. La finalidad es reducir las emisiones de CO_2 que resultan del uso de barras de refuerzo y del uso de concreto. Los experimentos

*Autor para correspondencia: irvingb@uaeh.edu.mx
Correo electrónico: irvingb@uaeh.edu.mx (Irving Barragan-Vite), mo450519@uaeh.edu.mx (Leonardo J. Montiel-Arrieta), jseck@uaeh.edu.mx (Juan Carlos Seck-Tuoh-Mora), nhromero@uaeh.edu.mx (Norberto Hernandez-Romero), jmedina@uaeh.edu.mx (Joselito Medina-Marin).

Historial del manuscrito: recibido el 14/08/2023, última versión-revisada recibida el 2/10/2023, aceptado el 04/10/2023, publicado el 20/11/2023. **DOI:** https://doi.org/10.29057/icbi.v11iEspecial3.11489

muestran una reducción de emisiones de alrededor del 3.93 % al minimizar el uso de materia prima. Asimismo, (Vishwakarma y Powar, 2021) desarrolla un modelo matemático del 1D-CSP en el que se considera el desperdicio sustentable, esto es, el modelo toma en cuenta solamente como desperdicio aquel material sobrante que no sea útil para la empresa.

En este trabajo se trata la variante de una dimensión del problema de corte (1D-CSP, *One-dimensional cutting stock problem*) y en particular en la que se usa una sola pieza de material o stock de acuerdo a la clasificación de (Wäscher *et al.*, 2007). En esta variante todas las piezas a cortar o ítems se obtienen de un solo tipo de stock, es decir, las piezas de stock a usar son de las mismas dimensiones (largo L y ancho W). Además, tanto el stock y los ítems tienen el mismo ancho, pero estos últimos pueden variar en longitud. De esta manera el problema se reduce a encontrar un acomodo de los ítems sobre las piezas de stock en una sola dimensión, es decir, el largo L. La Figura 1 muestra un ejemplo de un conjunto de ítems que deben ser obtenidos de piezas de stock de longitud L = 12. El plan de corte consiste en usar cinco stocks para cortar todas los ítems requeridos, donde las partes sombreadas de los stocks indican desperdicio.

Puesto que pueden existir diferentes formas de acomodar los ítems o planes de corte, el 1D-CSP se clasifica como un problema NP-hard, (Martinovic *et al.*, 2018). Debido a esto se han desarrollado diferentes métodos basados en programación lineal así como heurísticas, metaheurísticas y métodos híbridos para resolver el problema.

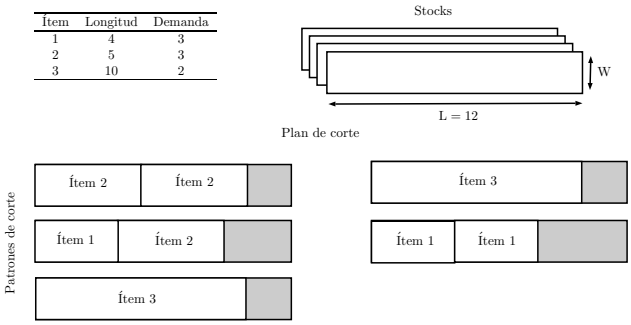

Figura 1: Ejemplo del 1D-CSP con tres ítems y stocks de una sola longitud.

Las principales contribuciones de este trabajo son:

- Emplear una variante discreta del algoritmo del búfalo Africano para resolver el 1D-CSP.
- Adaptar al 1D-CSP técnicas para discretizar las soluciones obtenidas con el algoritmo del búfalo Africano.
- Comparar la efectividad y eficiencia de los procedimientos de discretización.

Las siguientes secciones se organizan como sigue: En la sección 2 se realiza la revisión de la literatura sobre los métodos que se han empleado para resolver el 1D-CSP. En la sección 3 se describe el modelo matemático del 1D-CSP empleado en este trabajo y se presentan los términos relacionados a este problema. Asimismo se presentan los conceptos sobre el algoritmo del búfalo Africano así como de la variante discreta que se emplea en este trabajo. La sección 4 presenta el algoritmo que proponemos para resolver el 1D-CSP con base en la variante discreta del algoritmo del búfalo Africano. En la sección 5 se describe la metodología para resolver el 1D-CSP con el algoritmo propuesto y se presentan los resultados obtenidos y los análisis correspondientes.

2. Revisión de la literatura

Se considera que en (Kantorovich, 1960) abordaron el problema por primera vez formulando el modelo de programación lineal (PL) del problema y usando el método de multiplicadores para resolverlo. Posteriormente (Gilmore y Gomory, 1961, 1963) también emplearon PL con el método de generación de columnas para resolver el problema, la cual ha sido una de las aportaciones más novedosas en la solución del 1D-CSP. Desde entonces se aplicaron diferentes métodos basados en PL para encontrar la mejor solución del 1D-CSP. En (Valério de Carvalho, 2002) y (Delorme *et al.*, 2016) se hacen estudios extensos sobre las diferentes formulaciones y propuestas de solución basadas en métodos de PL durante los años subsecuentes a los primeros trabajos presentados en la década de 1960 dado el interés que se mostró por la problemática tanto desde la perspectiva de investigación como práctica.

Una de las dificultades al usar métodos de PL para resolver el 1D-CSP es la escalabilidad y la necesidad de soluciones enteras. Debido a esto, se han usado heurísticas y metaheurísticas como métodos de aproximación de soluciones óptimas. Dentro de los métodos heurísticos que se han empleado se encuentran los procedimientos secuenciales (Haessler, 1992; Gradišar *et al.*, 1999) y de reducción del número patrones de corte (Foerster y Wascher, 2000; Yanasse y Limeira, 2006; Renildo *et al.*, 2009), además de otros como los desarrollados en (Berberler y Nuriyev, 2010; Alfares y Alsawafy, 2019). Los algoritmos genéticos son uno de los algoritmos evolutivos que primeramente fueron empleados para tratar el 1D-CSP, (Reeves, 1996; Falkenauer, 1996). Sin embargo, como menciona (Hinterding y Khan, 1995) el tipo de representación de las soluciones impacta en su calidad y el consecuente desempeño del algoritmo como se muestra en los trabajos de (Peng y Chu, 2010b; Araujo *et al.*, 2014; Parmar *et al.*, 2015; Chen *et al.*, 2019). Además, uno de los principales problemas de los algoritmos genéticos es su pronta convergencia pudiendo quedar atrapados en soluciones óptimas locales. Otro de los algoritmos evolutivos empleados para resolver el 1D-CSP es la programación evolutiva (Liang *et al.*, 1998, 2002) que, a diferencia de los algoritmos genéticos solamente emplea el operador de mutación. (Chiong y Beng, 2007) realizan una comparación entre los algoritmos genéticos y la programación evolutiva para resolver el 1D-CSP, concluyendo que la programación evolutiva supera a los algoritmos genéticos. No obstante, el uso de la programación evolutiva no ha sido amplia para resolver problemas de tipo combinatorio. El interés por resolver el 1D-CSP y sus variantes ha resultado en el uso de otras metaheurísticas como se muestra en (Jahromi *et al.*, 2012), donde usan recocido simulado y búsqueda tabú, y en (Umetani *et al.*, 2003, 2006) empleando búsqueda local iterativa.

Las metaheurísticas basadas en inteligencia de enjambre también han sido consideradas para resolver el 1D-CSP. De esta forma, en (Levine y Ducatelle, 2004; Peng y Chu, 2010a;

Evtimov y Fidanova, 2018) emplean algoritmos basados en colonia de hormigas los cuales han sido probados para resolver problemas de tipo combinatorio, pero se requiere encontrar valores adecuados en sus parámetros para generar soluciones de buena calidad que compitan con los algoritmos evolutivos con los que generalmente se comparan. Una de las metaheurísticas de inteligencia de enjambre más conocidas para resolver problemas combinatorios es el algoritmo de cúmulo de partículas que se ha empleado tanto en su versión básica como de manera híbrida para mejorar su desempeño, (Shen *et al.*, 2007; Li *et al.*, 2007; Ben Lagha *et al.*, 2014). Aunque el algoritmo de enjambre o cúmulo de partículas ha mostrado ser idóneo para resolver problemas de optimización combinatoria dada la rapidez con la que puede encontrar la mejor solución, al igual que en los algoritmos basados en colonia de hormigas es necesario hacer un ajuste adecuado de sus parámetros para que no converja de manera prematura y quedar atrapado en un óptimo local. Además, el algoritmo de cúmulo de partículas es más adecuado para tratar con problemas continuos, mientras que para los problemas de tipo discreto se requieren procedimientos que ayuden a obtener soluciones de este tipo. Sin embargo y dada la amplia popularidad de este algoritmo para resolver diferentes problemas de optimización combinatoria, también se han desarrollado variantes para manejar problemas que requieren la representación de soluciones discretas como se muestra en (Strasser *et al.*, 1995; Strąk *et al.*, 2019), pero la variante basada en operaciones de conjuntos es la que ha sido más aceptada como se indica en (Chen y Tan, 2018; van Zyl y Engelbrecht, 2023). No obstante, no hay un consenso en cuanto qué variante es más idónea y para qué tipo de problema. En cuanto a resolver el 1D-CSP con variantes discretas basadas en algoritmos de enjambre, en (Asvany *et al.*, 2017) se emplea un algoritmo discreto de cúmulo de partículas en el que se aplica la técnica de intercambio de dos posiciones. Asimismo, en (Montiel-Arrieta *et al.*, 2022) se emplea el algoritmo del búfalo Africano para resolver el 1D-CSP y se propone el uso de la técnica del valor del orden clasificado para que algoritmo pueda manejar soluciones discretas.

3. Conceptos básicos

3.1. Modelo matemático y definiciones del 1D-CSP

Formalmente el 1D-CSP puede definirse como un conjunto de n objetos o ítems, cada uno con una demanda d_i ($i = 1, 2, \cdots, n$) que serán obtenidos de objetos grandes o stocks de una longitud dada L. Aunque existen diferentes formulaciones matemáticas para el 1D-CSP, en este trabajo adoptamos el modelo de programación lineal entera mixta (1)-(3), que emplea (Jahromi *et al.*, 2012).

$$\min \quad W = \sum_{j=1}^{m} w_j \tag{1}$$

s. a.

$$\sum_{j=1}^{m} x_{ij} = d_i, \quad i = \{1, 2, \ldots, n\} \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} l_i + w_j = \mathrm{L} y_j, \quad j = \{1, 2, \ldots, m\} \tag{3}$$

$$y_j \in \{0, 1\}$$

$$x_{ij} \geq 0 \text{ y entero.}$$

La ecuación (1) o función objetivo expresa la finalidad del problema que es minimizar el desperdicio total W generado por el plan de corte de tal forma que

$$w_j = \mathrm{L} - \sum_{i=1}^{n} x_{ij} l_i, \tag{4}$$

donde x_{ij} es el número de piezas del ítem i de longitud l_i cortadas del stock j.

La ecuación (2) indica que debe satisfacerse estrictamente toda la demanda de los ítems, es decir, no se permite sobreproducción. La ecuación (3) indica que el patrón de corte, incluyendo el desperdicio, no debe exceder la longitud del stock.

3.2. Algoritmo de optimización del búfalo Africano

El algoritmo de optimización del búfalo Africano (ABO) fue presentado como una nueva metaheurística en (Odili y Kahar, 2015a,b; Odili *et al.*, 2015). A partir de estos trabajos el algoritmo ha sido empleado para resolver algunos de los problemas de tipo combinatorio como el problema del agente viajero, el problema de flujo de trabajos en taller, el problema de empaque y el 1D-CSP, (Man *et al.*, 2016; Panella *et al.*, 2021; Jiang *et al.*, 2020; Krisnawati *et al.*, 2020; Gherboudj, 2019; Montiel-Arrieta *et al.*, 2022).

El ABO es un algoritmo inspirado en el comportamiento de los búfalos Africanos los cuales cooperan para buscar las mejores zonas de pastizales o bien, para defenderse de algún peligro. De acuerdo a (Odili *et al.*, 2015), las manadas de búfalos emplean los sonidos "maaa" y "waaa" para indicar si deben permanecer en el sitio actual por ser seguro y con suficiente alimento, o bien para trasladarse a otro sitio, respectivamente. Las ecuaciones (5) y (6) representan los sonidos "maaa" y "waaa", correspondientemente, donde $bgmax$ es el mejor búfalo de la manada, $bpmax_k$ es la mejor posición del búfalo k para $k = 1, 2, \ldots, N$, y $lp1$ y $lp2$ son factores de aprendizaje con valores en el intervalo $[0.1, 0.6]$.

$$m_{k+1} = m_k + lp1(bgmax - w_k) + lp2(bpmax_k - w_k) \tag{5}$$

$$w_{k+1} = \frac{w_k + m_k}{\pm 0.5} \tag{6}$$

En la dinámica de los búfalos (5) representa la *explotación* de la ubicación actual y (6) es el movimiento de *exploración*. En (5) m_k es la *memoria* de los búfalos con lo que recuerdan su condición anterior; $lp1(bgmax - w_k)$ representa la cualidad cooperativa de los búfalos y $lp2(bpmax_k - w_k)$, es la parte *inteligente* de estos animales para transmitir su mejor ubicación

en relación a la actual. El pseudocódigo del ABO de acuerdo a (Odili *et al.*, 2015) se muestra en el Algoritmo 1.

1. Establecer la función objetivo $f(x) = (x_1, x_2, \ldots, x_n)^T$.
2. Inicialización: Colocar aleatoriamente a los búfalos en el espacio de solución.
3. Actualizar con (5) el *fitness* de los búfalos.
4. Actualizar con (6) la posición del búfalo k ($bpmax_k$ y $bgmax$).
5. ¿Se actualiza $bgmax$? Sí, ir al paso 6. No, ir al paso 2.
6. Si no se satisface el criterio de parada, volver al paso 3. En caso contrario, ir al paso 7.
7. Obtener la mejor solución.

Algoritmo 1: Algoritmo de optimización del búfalo Africano (adaptado de (Odili *et al.*, 2015)).

De las ecuaciones (6) y (5) puede observarse que la actualización de las soluciones serán valores continuos, por lo que si el problema que se está resolviendo requiere valores discretos deberá emplearse algún procedimiento para obtenerlos y evaluar adecuadamente la función *fitness* que guía la búsqueda de la mejor solución.

En (Jiang *et al.*, 2020) se propone una variante del ABO para solventar el problema de discretizar las soluciones en el problema de programación de tareas en taller. De igual forma, en (Gherboudj, 2019) y (Montiel-Arrieta *et al.*, 2022) se emplea una técnica basada en claves aleatorias (*random keys*) para el mismo propósito, pero en los problemas de empaque y el 1D-CSP, correspondientemente. Hasta donde tenemos conocimiento, son las únicas propuestas para discretizar soluciones al emplear el ABO. En este trabajo empleamos y adaptamos la propuesta de (Jiang *et al.*, 2020) para resolver el 1D-CSP.

3.3. Algoritmo discreto de optimización del búfalo Africano

El algoritmo discreto de optimización del búfalo Africano (DABO) propuesto en (Jiang *et al.*, 2020) usa (7) y (8) para la actualización del *fitness* y la posición de los búfalos, respectivamente. En (7) el término $lp3(br - w_k)$ se introduce para representar *el mecanismo de aprendizaje aleatorio*, donde br es un búfalo seleccionado de manera aleatoria de la manada actual. (Jiang *et al.*, 2020) introducen este término bajo el criterio de que en el ABO original se pierde diversidad en la población a lo largo de las iteraciones debido a la aplicación de (5) y (6), lo que ocasiona una convergencia prematura.

$$m_{k+1} = m_k + lp1(bgmax - w_k) + lp2(bpmax_k - w_k) + lp3(br - w_k) \quad (7)$$

$$w_{k+1} = \frac{w_k + m_{k+1}}{\lambda} \quad (8)$$

Sin embargo, para discretizar las soluciones en el DABO se emplean (9) y (10) para la explotación y la exploración, respectivamente, donde CR indica un operación de cruza y MU una operación de mutación. Asimismo, $lp1 + lp2 + p3 = 1$ debe satisfacerse, de tal forma que estos valores representan las tasas de diferentes tipos de cruza, y λ es la tasa de mutación. Por otra parte, el símbolo $\otimes$ indica que si se satisface la condición mostrada en (11) o (12) , se realizará la operación de cruza o de mutación del elemento a la izquierda de CR o MU; mientras que el símbolo $\oplus$ indica que la operación de cruza CR a su izquierda tiene mayor prioridad, pero si su condición no se satisface se inicia el elemento a su derecha. En (11) y (12), **r** es un número aleatorio en el intervalo [0, 1].

$$m_{k+1} = lp1 \otimes CR(bgmax, w_k) \oplus lp2 \otimes CR(bpmax_k, w_k) \oplus lp3 \otimes CR(br, w_k) \quad (9)$$

$$w_{k+1} = \lambda \otimes MU(m_k) \quad (10)$$

$$m_{k+1} = \begin{cases} CR(bgmax, w_k), & \text{si} \quad \mathbf{r} < lp1 \\ CR(bpmax_k, w_k), & \text{si} \quad lp1 \leq \mathbf{r} < lp1 + lp2 \\ CR(br, w_k), & \text{si} \quad lp1 + lp2 \leq \mathbf{r} < lp1 + lp2 + lp3 \end{cases} \quad (11)$$

$$w_{k+1} = \begin{cases} MU(m_k), & \text{si} \quad \mathbf{r} < \lambda \\ m_k, & \text{si} \quad \mathbf{r} \geq \lambda \end{cases} \quad (12)$$

En (Jiang *et al.*, 2020) se comparan diferentes mecanismos de reinicialización de la manada de búfalos, pero el mecanismo de re-inicialización individual es el que mejores resultados produce. A diferencia del ABO original donde se reinicializa toda la población de búfalos si el $bgmax$ no se actualiza en la iteración actual, la reinicialización individual consiste en reinicializar solamente el búfalo que ha alcanzado un límite de esperanza de vida o *life span* (ls).

Asimismo, se emplean diferentes operaciones de cruza dada la representación de las soluciones del problema que abordan. La operación de cruza basada en tareas (JBX) y la operación de cruza basada en la preservación del orden de precedencia (POX) se aplican aleatoriamente a la parte de las soluciones que representan la asignación de operaciones; mientras que para la parte de las soluciones que representan la asignación de máquinas se utilizan de manera aleatoria las operaciones de cruza de dos puntos (TPX) y multipunto (MPX). En el caso de la mutación se emplean el intercambio de dos posiciones (*swap*) y de una sola posición.

En este trabajo realizamos una adaptación de la operación de cruza JBX empleada en (Jiang *et al.*, 2019; Lu y Jiang, 2019) para usarla en el 1D-CSP. A esta adaptación la denominamos ITX y se describe en la siguiente sección. Asimismo, utilizamos la operación de mutación *swap*. Adicionalmente, y para fines de comparación con la operación de cruza, empleamos la técnica del valor del orden clasificado (ROV, *ranked order value*) propuesta en (Gherboudj, 2019; Montiel-Arrieta *et al.*, 2022).

4. Algoritmo para resolver el 1D-CSP basado en el DABO

En esta sección se describe la adaptación del DABO para resolver el 1D-CSP, a la cual denominaremos DABO-1DCSP en adelante. Para los ejemplos y las descripciones pertinentes se empleará la instancia mostrada en la Figura 1.

A continuación se describen los pasos del DABO-1DCSP que se muestran en el Algoritmo 2.

1. Establecer la función objetivo.
2. Crear aleatoriamente una población de búfalos de tamaño N.
3. Actualizar los búfalos con (9) y (10).
4. Determinar $bpmax_k$ y $bgmax$.
5. Evaluar si se ha alcanzado el *life span* para algún búfalo k. Si es así, ir al paso 6. Si no, ir al paso 7.
6. Realizar el procedimiento de reinicialización individual.
7. Determinar si se ha alcanzado el criterio de terminación. Si es así, ir al paso 8. En otro caso, ir al paso 3.
8. Obtener la mejor solución.

Algoritmo 2: Algoritmo DABO-1DCSP.

El paso 1 consiste en establecer la función objetivo que, para el 1D-CSP abordado en este trabajo, será como se indica en (1) del modelo (1)-(3).

El paso 2 consiste en crear de manera aleatoria una población de búfalos o de soluciones de tamaño N. En este sentido, una solución consiste en un vector $\mathbf{s}$ de longitud D en el que cada elemento s_i representa la longitud del ítem i, $i = 1, 2, \ldots, n$. Nótese que D es el resultado de la suma de las demandas de los ítems, por ejemplo para la instancia de la Figura 1, $D = 8$ mientras que la solución mostrada para el plan de corte sería el vector $\mathbf{s}_k$ = [5 5 4 5 10 10 4 4]. En la generación aleatoria de la población se verifica que no existan búfalos idénticos.

$S_1 = \{1, 3\}$ $S_2 = \{2\}$

Padre 1	[5 5 **4** 5 **10** **10** **4** **4**]
Padre 2	[10 10 4 **5** 4 **5** **5** 4]
Hijo 1	[*5* *5* **4** *5* **10** **10** **4** **4**]
Hijo 2	[*4* *10* *10* **5** *4* **5** **5** *4*]

Figura 2: Operación de cruza ITX.

En el paso 3 se realiza la actualización de la población empleando (9) y (10). Para discretizar m_{k+1} se emplean ya sea la técnica ROV o la operación de cruza ITX. La operación de cruza ITX se describe en el Algoritmo 3 y se ilustra en la Figura 2. Puesto que esta operación produce dos soluciones, ambas se evalúan con la función *fitness* y se descarta aquella que obtenga el peor valor. En la técnica ROV, una vez calculado m_{k+1} con (7) se ordenan sus valores en orden ascendente como se muestra en la Figura 3 con $\mathbf{s}_{ordenada}$, donde también el vector $\mathbf{j}$ representa la posición reordenada de los elementos de m_{k+1}. De esta manera y mediante un vector $\mathbf{l}$ con las longitudes de los ítems ordenadas de forma ascendente, la solución discreta $\mathbf{s}_{discreta}$ de m_{k+1} será un vector tal que la posición $\mathbf{j}(i)$ será la longitud del ítem i en $\mathbf{l}$, para $i = 1, 2, \ldots, n$.

1. Formar los conjuntos de ítems S_1 y S_2 con un número aleatorio de elementos, tal que $S_1 \cap S_2 = \emptyset$
2. Localizar en el *Padre 1* las longitudes de los ítems de S_1 y copiarlas al *Hijo 1* en las mismas posiciones del padre.
3. Localizar en el *Padre 2* las longitudes de los ítems de S_2 y copiarlas al *Hijo 2* en las mismas posiciones del padre.
4. Copiar las longitudes de los ítems de S_2, de acuerdo a su secuencia en el *Padre 2*, al *Hijo 1* en las posiciones faltantes.
5. Copiar las longitudes de los ítems de S_1, de acuerdo a su secuencia en el *Padre 1*, al *Hijo 2* en las posiciones faltantes.
6. Terminar el procedimiento de cruza.

Algoritmo 3: Algoritmo para realizar la operación de cruza ITX

Longitudes ordenadas de los ítems:

$\mathbf{l} = [l_i] = [4 \quad 4 \quad 4 \quad 5 \quad 5 \quad 5 \quad 10 \quad 10]$

$i = 1, 2, 3, 4, 5, 6, 7, 8$

Solución de m_{k+1}:

$\mathbf{s} = [s_i] = [2.2 \quad 5.7 \quad 4.3 \quad 5 \quad 10.8 \quad 10 \quad 4 \quad 5]$

Ordenamiento:

$\mathbf{s}_{ordenada} = [x_i = \mathbf{s}(\mathbf{j}(i))] = [2.2 \quad 4 \quad 4.3 \quad 5 \quad 5 \quad 5.7 \quad 10 \quad 10.8]$

$\mathbf{j} = [1 \quad 7 \quad 3 \quad 4 \quad 8 \quad 2 \quad 6 \quad 5]$ Índices de $\mathbf{s}$ reordenados de acuerdo a $\mathbf{s}_{ordenada}$

Solución discreta de m_{k+1}:

$\mathbf{s}_{discreta} = [y_{\mathbf{j}(i)} = \mathbf{l}(i)] = [4 \quad 5 \quad 4 \quad 5 \quad 10 \quad 10 \quad 4 \quad 5]$

Figura 3: Obtención de soluciones discretas con la técnica ROV.

Por otra parte, para obtener w_{k+1} se emplea la operación de mutación *swap*, la cual se ilustra en la Figura 4. Esta operación consiste en seleccionar de manera aleatoria dos elementos de la solución e intercambiarlos para obtener una nueva.

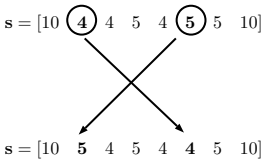

Figura 4: Operación de mutación *swap*.

En el paso 4 y una vez actualizada la población, se determina la mejor posición $bpmax_k$ de cada búfalo k así como el mejor búfalo $bgmax$ de la manada. En ambos casos, a partir de w_{k+1} se evalúa el *fitness* de cada búfalo de acuerdo a (1). Para una solución dada $\mathbf{s}$, el procedimiento para calcular w_j se basa en la heurística *next fit*. Dada una solución, la heurística consiste en asignar de manera secuencial las longitudes en la solución (de izquierda a derecha) al primer stock. Si una longitud en la secuencia no puede asignarse por exceder la longitud del stock, este stock se *cierra* y la longitud se asigna a uno nuevo. El proceso continua hasta asignar todas las longitudes de la solución. Un ejemplo de su aplicación se muestra en la Figura 5 que arroja el plan de corte de la Figura 1 con un *fitness* de $W = 13$. Cada stock constituye un patrón de corte.

$\mathbf{s} = [5 \quad 5 \quad 4 \quad 5 \quad 10 \quad 10 \quad 4 \quad 4]$

$L = 12$

Stock 1: {5 5}	$w_1 = 2$	
Stock 2: {4 5}	$w_2 = 3$	
Stock 3: {10}	$w_3 = 2$	$W = 13$
Stock 4: {10}	$w_4 = 2$	
Stock 5: {4 4}	$w_5 = 4$	

Figura 5: Obtención de los patrones de corte con la heurística *next fit*.

En el paso 5 se evalúa si alguno de los búfalos ha alcanzado

la esperanza de vida ls. En la inicialización de la población la edad de cada búfalo se asigna con un valor de 1. Cada vez que se actualiza la manada se determina si el $fitness$ de cada búfalo k mejoró o no en comparación con la iteración inmediata anterior. En caso de no haber mejoría, la edad se aumenta en una unidad.

En el paso 6 se realiza la reinicialización de los búfalos que hayan alcanzado el valor de *life span*. Esta reinicialización consiste en generar de manera aleatoria un nuevo búfalo que sustituye al búfalo que ha alcanzado su *life span* y la edad del nuevo búfalo se reinicia al valor 1. Al reiniciar el búfalo se verifica que no sea idéntico a algún otro existente en la manada.

En el paso 7 se verifica la condición de terminación del algoritmo. Este criterio estará basado en un número máximo de iteraciones $iter_{max}$.

En el paso 8 se obtiene el búfalo con el valor mínimo de desperdicio total, W.

5. Experimentos y resultados

El DABO-1DCSP se probó en un conjunto de diez instancias tomadas de (Liang *et al.*, 2002). La datos necesarios de estas instancias se muestran en la Tabla 1, de donde puede observarse que varían en complejidad (número de ítems, n).

Tabla 1: Conjunto de instancias de prueba.

Instancia	n	L
1a	20	14
2a	50	15
3a	60	25
4a	60	25
5a	126	4300
6a	200	86
7a	200	120
8a	400	120
9a	400	120
10a	600	120

Los experimentos consisten en tres escenarios que dan origen a los siguientes algoritmos basados en el DABO-1DCSP:

ITX-DABO. Se realiza el paso 3 del DABO-1DCSP empleando la operación de cruza ITX para obtener m_{k+1} y la operación de mutación *swap* para obtener w_{k+1}.

ROV-DABO. Se realiza el paso 3 del DABO-1DCSP empleando la técnica ROV para obtener m_{k+1} y la operación de mutación *swap* para obtener w_{k+1}.

SROV-DABO. Se realiza el paso 3 del DABO-1DCSP empleando la técnica ROV para obtener w_{k+1}, mientras que m_{k+1} se calcula de acuerdo a (7).

Los experimentos se realizaron en una computadora con procesador Quad-Core Intel$^{(R)}$ Core$^{(TM)}$ i5 de 3.4 Ghz y RAM de 8 GB. El DABO-1DCSP fue codificado en Python 3.11.1.

Los valores de los parámetros del tamaño de población (Pob), del máximo número de iteraciones ($Iter_{max}$), de $lp1$ y $lp2$, de la tasa de mutación λ y del *life span* (ls), se ajustaron mediante el método de Taguchi de acuerdo a (Jiang *et al.*, 2020). Se usó el diseño $L_{25}(5^6)$ para los seis parámetros con cinco niveles cada uno, los cuales se detallan en la Tabla 2. Los experimentos para el ajuste se realizaron con base en la instancia 6a como un punto medio en la complejidad de las instancias y el valor de respuesta usado fue el desperdicio promedio ($\overline{W}$) de diez corridas. El arreglo ortogonal se muestra en la Tabla 3 donde el valor de $\overline{W}$ corresponde a ITX-DABO.

Tabla 2: Diseño experimental para el ajuste de los parámetros del DABO-1DCSP.

Factor	Nivel 1	Nivel 2	Nivel 3	Nivel 4	Nivel 5
Pob	80	100	120	150	200
$Iter_{max}$	500	600	800	1000	1500
$lp1$	0.1	0.2	0.3	0.4	0.5
$lp2$	0.1	0.2	0.3	0.4	0.5
λ	0.05	0.1	0.15	0.2	0.25
ls	5	10	15	20	25

Tabla 3: Arreglo ortogonal para el ajuste de los parámetros del DABO-1DCSP.

Pob	$Iter_{max}$	$lp1$	$lp2$	λ	ls	$\overline{W}$
1	1	1	1	1	1	825.4
1	2	2	2	2	2	791
1	3	3	3	3	3	791
1	4	4	4	4	4	799.6
1	5	5	5	5	5	782.4
2	1	2	3	4	5	782.4
2	2	3	4	5	1	756.6
2	3	4	5	1	2	842.6
2	4	5	1	2	3	765.2
2	5	1	2	3	4	799.6
3	1	3	5	2	4	791
3	2	4	1	3	5	765.2
3	3	5	2	4	1	713.6
3	4	1	3	5	2	722.2
3	5	2	4	1	3	782.4
4	1	4	2	5	3	799.6
4	2	5	3	1	4	825.4
4	3	1	4	2	5	756.6
4	4	2	5	3	1	730.8
4	5	3	1	4	2	748
5	1	5	4	3	2	782.4
5	2	1	5	4	3	756.6
5	3	2	1	5	4	722.2
5	4	3	2	1	5	756.6
5	5	4	3	2	1	756.6

La Tabla 4 muestra los resultados de respuesta para medias del método de Taguchi. De acuerdo a estos resultados los valores de los parámetros se ajustaron como sigue: $Pob = 120$, $Iter_{max} = 1000$, $lp1 = 0.2$, $lp2 = 0.1$, $\lambda = 0.25$ y $ls = 5$. En relación al tamaño de la población y de acuerdo a los resultados de la Tabla 4, es posible elegir tanto una población 120 individuos como de 200. Sin embargo, un menor tamaño de población ahorra tiempo computacional en la ejecución del algoritmo.

Tabla 4: Respuesta para medias y clasificación (*Rank*) de los factores.

Nivel	*Pob*	$Iter_{max}$	$lp1$	$lp2$	λ	ls
1	797.9	796.2	772.1	765.2	806.5	756.6
2	789.3	779.0	761.8	772.1	772.1	777.2
3	754.9	765.2	768.6	775.5	773.8	779.0
4	772.1	754.9	792.7	775.5	760.0	787.6
5	754.9	773.8	773.8	780.7	756.6	768.6
Delta	43.0	41.3	31.0	15.5	49.9	31.0
Rank	2	3	5	6	1	4

Con los valores de los parámetros descritos arriba, se realizaron 50 ejecuciones con los algoritmos ITX-DABO, ROV-DABO y SROV-DABO. Los resultados del desperdicio total promedio $\overline{W}$ para cada uno de los algoritmos se muestran en la Tabla 5. En todas las instancias, excepto la 3a, el ITX-DABO obtiene los mejores valores de desperdicio total promedio. En la instancia 3a los tres algoritmos arrojan el mismo resultado. También se reporta el desperdicio total mínimo $\overline{W}_{min}$ obtenido de las 50 ejecuciones de cada algoritmo. El ITX-DABO obtiene los mejores resultados en las instancias 6a a 10a, mientras que para las instancias 1a a 3a los tres algoritmos obtienen los mismos resultados. En la instancia 4a el ITX-DABO y el SROV-DABO obtienen el mismo resultado de $\overline{W}_{min}$. De igual forma, en la instancia 5a el ITX-DABO y el ROV-DABO llegan al mismo resultado. La diferencia entre los tres algoritmos puede observarse en la diagrama de caja de la Figura 6 en relación a la desviación relativa promedio (*RPD*) que se calculó de acuerdo a (13) donde *Min* es el valor mínimo de los W_{min} de los algoritmos para cada instancia. Los valores de *RPD* para cada algoritmo se muestran en la Tabla 5.

$$RPD = \frac{\overline{W} - Min}{Min} \tag{13}$$

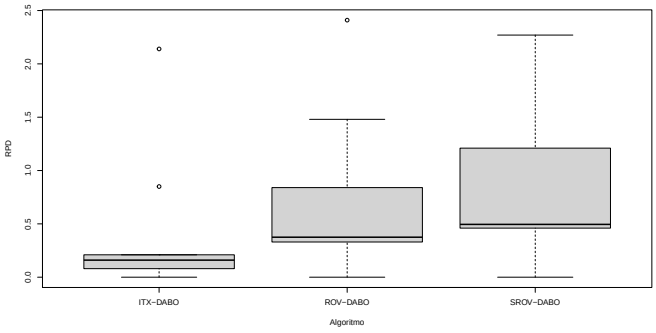

Figura 6: Diagrama de caja para los RPD de los tres algoritmos.

Con estos resultados puede observarse que el ITX-DABO es más efectivo para resolver el 1D-CSP al emplear la operación de cruza ITX y de mutación *swap* para actualizar m_k y w_k, respectivamente. Sin embargo, para confirmar si existe diferencia estadísticamente significativa se realizó la prueba de Friedman con base en el *RPD*. El resultado de la prueba de Friedman arroja el valor $p = 0.0003002$ por lo que para un nivel de significancia de 0.05, se confirma que existe diferencia significativa entre los tres algoritmos. No obstante, el ITX-DABO es el menos eficiente de los tres algoritmos comparados ya que requirió mayor tiempo en su ejecución como se muestra en la Tabla 6, mientras que el más eficiente fue el SROV-DABO. El mayor uso de tiempo por el ITX-DABO puede deberse a que debe realizar tanto una operación de cruza como de mutación en cada iteración, mientras que el SROV-DABO solamente realiza una operación de discretización. En la Figura 7 se muestran las gráficas de convergencia para el desperdicio promedio $\overline{W}$ en cada iteración de los tres algoritmos con las instancias 2a, 7a, 8a y 10a, las cuales cuentan con 50, 200, 400 y 600 ítems, respectivamente. Puede observarse que tanto el algoritmo ITX-DABO como el SROV-DABO convergen con rapidez hacia su valor $\overline{W}_{min}$, aunque el SROV-DABO lo hace ligeramente más rápido, particularmente en las instancias de 200, 400 y 600 ítems. El algoritmo ROV-DABO tiene una convergencia marcadamente más lenta.

Tabla 6: Resultado del tiempo promedio computacional (en segundos).

Instancia	ITX-DABO	ROV-DABO	SROV-DABO
1a	32.17	8.51	7.96
2a	44.39	15.61	15.16
3a	45.65	16.41	16.18
4a	45.80	16.92	16.50
5a	73.36	32.51	32.27
6a	99.37	50.80	50.04
7a	98.70	49.86	49.55
8a	176.43	102.04	99.49
9a	181.67	102.77	102.36
10a	250.38	154.29	152.95

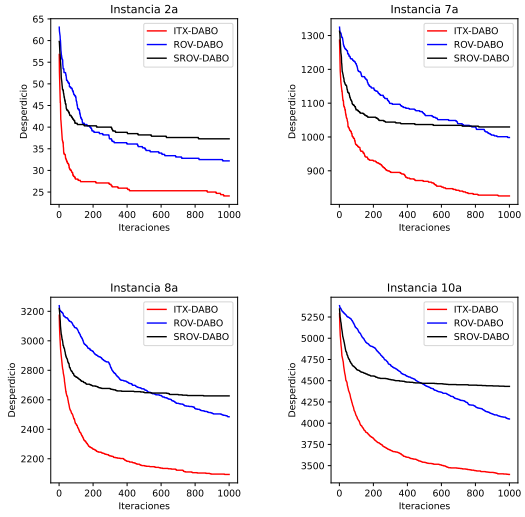

Figura 7: Gráficas de convergencia para el desperdicio promedio $\overline{W}$.

Para comparar la efectividad del ITX-DABO con otros algoritmos se usan los algoritmos LLA (*least-loss algorithm*) de (Alfares y Alsawafy, 2019) y el ABO-CSP de (Montiel-Arrieta *et al.*, 2022), por ser algunos de los algoritmos más recientes en abordar el 1D-CSP y que emplean las mismas diez instancias descritas en párrafos previos. En particular el algoritmo LLA es uno de los que mejores resultados han arrojado sobre estas instancias, mientras que el ABO-CSP es una primera implementación del ABO para resolver el 1D-CSP. La comparación se rea-

Tabla 5: Resultados del desperdicio promedio.

	ITX-DABO			ROV-DABO			SROV-DABO		
Instancia	W_{min}	$\overline{W}$	RPD	W_{min}	$\overline{W}$	RPD	W_{min}	$\overline{W}$	RPD
1a	3	3	0	3	5.52	0.84	3	6.64	1.21
2a	13	24.1	0.85	28	32.2	1.48	13	37.3	1.87
3a	25	25	0	25	25	0	25	25	0
4a	11	34.5	2.14	36	37.5	2.41	36	36	2.27
5a	20050	21598	0.08	20050	26930	0.34	20050	27704	0.38
6a	705	803.04	0.14	791	930.32	0.32	877	1030.08	0.46
7a	684	825.6	0.21	804	998.4	0.46	804	1029.6	0.51
8a	1772	2093.6	0.18	2132	2484.8	0.40	2252	2626.4	0.48
9a	1942	2292.4	0.18	2182	2580.4	0.33	2662	2971.6	0.53
10a	3010	3396.4	0.13	3490	4049.2	0.35	3850	4433.2	0.47

Tabla 7: Comparación del ITX-DABO con otros algoritmos con base en el número de stocks usados.

		ITX-DABO		LLA		ABO-CSP	
Instancia	lb	Stocks	$\% > lb$	Stocks	$\% > lb$	Stocks	$\% > lb$
1a	9	9	0	9	0	9.26	2.89
2a	23	23.74	3.22	23	0	24.52	6.60
3a	15	16	6.67	15	0	16	6.67
4a	19	19.94	4.95	19	0	20	5.26
5a	51	55.36	8.55	53	3.92	56.62	11.02
6a	78	87.14	11.72	81	3.85	89.46	14.69
7a	68	74.18	9.09	68	0	75.62	11.21
8a	143	159.68	11.66	145	1.40	162.76	13.82
9a	149	167.92	12.70	152	2.01	172.40	15.70
10a	215	243.22	13.13	216	0.47	250.40	16.47

liza con base en el número de stocks usados para el plan de corte de cada instancia, ya que los trabajos citados no emplearon el desperdicio total como su función objetivo; sin embargo, la minimización del desperdicio total es equivalente a minimizar el número de stocks usados siempre que no se permita sobreproducción y los stocks sean de una sola longitud (Scheithauer, 2018). El ABO-CSP se implementó con los mismos valores de los parámetros que el ITX-DABO en cuanto al tamaño de la población, el número máximo de iteraciones, los factores $lp1$, $lp2$ y λ.

La Tabla 7 muestra los resultados del número de stocks usados así como el $\% > lb$, donde lb es el límite inferior óptimo para el 1D-CSP que se obtiene con (14). Puede notarse que el algoritmo LLA es superior a los otros dos algoritmos al tener los porcentajes más cercanos a lb, no obstante el ITX-DABO logra mejores resultados que la implementación del ABO-CSP en todas las instancias. Sin embargo, el LLA es un método heurístico diseñado para resolver el 1D-CSP, mientras que el ITX-DABO y el ABO-CSP son métodos metaheurísticos cuya solución es en esencia aproximada.

$$lb = \left\lceil \frac{\sum_i^n l_i d_i}{\mathrm{L}} \right\rceil \tag{14}$$

La Figura 8 muestra las gráficas de convergencia de los algoritmos ITX-DABO y ABO-CSP para el número promedio de stocks usados en cada iteración en las instancias 2a, 7a, 8a y 10a. Solamente se compara la convergencia de estos dos algoritmos ya que el algoritmo LLA no se basa en población ni en iteraciones para resolver las instancias. De esta forma, puede notarse que en la instancia 2a el algoritmo ITX-DABO tiene una convergencia ligeramente más rápida que el ABO-CSP. En las instancias 7a, 8a y 10a, el algoritmo ABO-CSP converge un poco más rápido hacia su valor promedio mínimo lo que sería deseable en instancias de mayor complejidad como la 7a, 8a y 10a. En contraste, el ITX-DABO tarda un poco más en converger hacia su valor mínimo en este tipo de instancias.

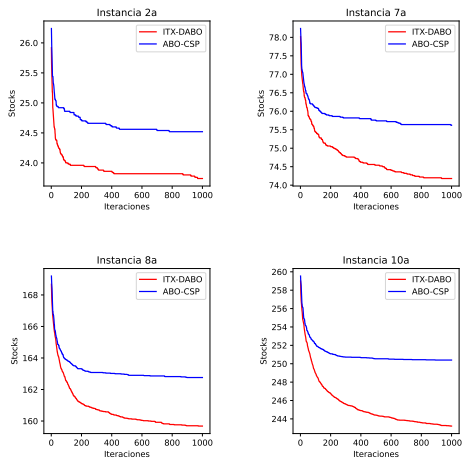

Figura 8: Gráficas de convergencia de los algoritmos ITX-DABO y ABO-CSP para el número promedio de stocks usados.

6. Conclusiones

En este trabajo se presentó una implementación del algoritmo discreto de optimización del búfalo Africano para resolver el 1D-CSP. El algoritmo desarrollado consistió en emplear un operación de cruza o la técnica de discretización ROV para actualizar m_k, mientras que para actualizar w_k se empleó una operación de mutación. Se desarrolló la operación de cruza ITX con base en las longitudes de los ítems. Tanto la operación ITX como la técnica ROV se compararon para determinar su eficiencia y efectividad. Asimismo, se comparó el algoritmo discreto empleando solamente la técnica ROV para discretizar únicamente w_k. Los resultados mostraron que el uso de la operación ITX en el algoritmo discreto es más efectiva que emplear en su lugar la técnica ROV, o que emplear únicamente esta última técnica en la solución del 1D-CSP. Sin embargo, la operación ITX es menos eficiente al utilizar un tiempo mayor a las otras implementaciones. Finalmente, el algoritmo propuesto con la operación ITX se comparó con otros dos algoritmos que recientemente han abordado el 1D-CSP con las instancias de prueba empleadas en este trabajo. Los resultados de esta comparación muestran que el algoritmo discreto desarrollado queda cerca del límite inferior óptimo de las instancias, pero no logra ser el mejor. No obstante, obtiene mejores resultados que una de las implementaciones del ABO para resolver el 1D-CSP. Los resultados del algoritmo propuesto en este trabajo pueden mejorarse al experimentar con otros esquemas de reinicialización de la manada o con la hibridación con alguna técnica de búsqueda local.

Referencias

Alfares, H. K. y Alsawafy, O. G. (2019). A least-loss algorithm for a bi-objective one-dimensional cutting-stock problem. *International Journal of Applied Industrial Engineering (IJAIE)*, 6(2):1–19.

Araujo, S. A. d., Poldi, K. C., y Smith, J. (2014). A genetic algorithm for the one-dimensional cutting stock problem with setups. *Pesquisa Operacional*, 34(2):165–187.

Asvany, T., Amudhavel, J., y Sujatha, P. (2017). One-dimensional cutting stock problem with single and multiple stock lengths using dpso. *Advnaces and Applications in Mahtematical Sciences*, 17(4):147–163.

Ben Lagha, G., Dahmani, N., y Krichen, S. (2014). Particle swarm optimization approach for resolving the cutting stock problem. En *2014 International Conference on Advanced Logistics and Transport (ICALT)*, pp. 259–263.

Berberler, M. y Nuriyev, U. (2010). A new heuristic algorithm for the one-dimensional cutting stock problem. *Applied and Computational Mathematics*, 9(1):19–30.

Chen, W.-N. y Tan, D.-Z. (2018). Set-based discrete particle swarm optimization and its applications: a survey. *Frontiers of Computer Science: Selected Publications from Chinese Universities*, 12(2).

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

Chiong, R. y Beng, O. K. (2007). A comparison between genetic algorithms and evolutionary programming based on cutting stock problem. *Engineering Letters*, 14(1):72–77.

Delorme, M., Iori, M., y Martello, S. (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20.

Evtimov, G. y Fidanova, S. (2018). Ant colony optimization algorithm for 1d cutting stock problem. En Georgiev, K., Todorov, M., y Georgiev, I., editores, *Advanced Computing in Industrial Mathematics: 11th Annual Meeting of the Bulgarian Section of SIAM December 20-22, 2016, Sofia, Bulgaria. Revised Selected Papers*, pp. 25–31. Springer International Publishing.

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2:5–30.

Fang, J., Rao, Y., Luo, Q., y Xu, J. (2023). Solving one-dimensional cutting stock problems with the deep reinforcement learning. *Mathematics*, 11(4):1028.

Foerster, H. y Wascher, G. (2000). Pattern reduction in one-dimensional cutting stock problems. *International Journal of Production Research*, 38(7):1657–1676.

Gherboudj, A. (2019). African buffalo optimization for one dimensional bin packing problem. *International Journal of Swarm Intelligence Research (IJSIR)*, 10(4):38–52.

Gilmore, P. C. y Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859.

Gilmore, P. C. y Gomory, R. E. (1963). A linear programming approach to the cutting stock problem—part ii. *Operations Research*, 11(6):863–888.

Gradišar, M., Kljajić, M., Resinovič, G., y Jesenko, J. (1999). A sequential heuristic procedure for one-dimensional cutting. *European Journal of Operational Research*, 114(3):557–568.

Haessler, R. W. (1992). One-dimensional cutting stock problem problems and solution procedures. *Mathl. Comput. Modelling*, 16(1):1–8.

Hinterding, R. y Khan, L. (1995). Genetic algorithms for cutting stock problems: With and without contiguity. En Yao, X., editor, *Progress in Evolutionary Computation*, pp. 166–186, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jahromi, M. H., Tavakkoli-Moghaddam, R., Makui, A., y Shamsi, A. (2012). Solving an one-dimensional cutting stock problem by simulated annealing and tabu search. *Journal of Industrial Engineering International*, 8(24).

Jiang, T., Zhang, C., y Sun, Q.-M. (2019). Green job shop scheduling problem with discrete whale optimization algorithm. *IEEE Access*, 7:43153–43166.

Jiang, T., Zhu, H., y Deng, G. (2020). Improved african buffalo optimization algorithm for the green flexible job shop scheduling problem considering energy consumption. *Journal of Intelligent & Fuzzy Systems*, 38(4):4573–4589.

Kantorovich, L. V. (1960). Mathematical methods of organizing and planning production. *Management science*, 6(4):366–422.

Krisnawati, M., Sibarani, A. A., Mustikasari, A., y Aulia, D. (2020). African buffalo optimization for solving flow shop scheduling problem to minimize makespan. *IOP Conference Series: Materials Science and Engineering*, 982(1):012061.

Lee, D., Son, S., Kim, D., y Kim, S. (2020). Special-length-priority algorithm to minimize reinforcing bar-cutting waste for sustainable construction. *Sustainability*, 12(15).

Levine, J. y Ducatelle, F. (2004). Ant colony optimization and local search for bin packing and cutting stock problems. *Journal of the Operational Research Society*, 55(7):705–716.

Li, Y., Zheng, B., y Dai, Z. (2007). General particle swarm optimization based on simulated annealing for multi-specification one-dimensional cutting stock problem. En Wang, Y., Cheung, Y.-m., y Liu, H., editores, *Computational Intelligence and Security*, pp. 67–76, Berlin, Heidelberg. Springer Berlin Heidelberg.

Liang, K.-H., Yao, X., Newton, C., y Hoffman, D. (1998). Solving cutting stock problems by evolutionary programming. En *7th Annual Conference on Evolutionary Programming, EP 1998*, volumen 1447, pp. 755–764.

Liang, K.-H., Yao, X., Newton, C., y Hoffman, D. (2002). A new evolutionary approach to cutting stock problems with and without contiguity. *Computers & Operations Research*, 29(12):1641–1659.

Lu, Y. y Jiang, T. (2019). Bi-population based discrete bat algorithm for the low-carbon job shop scheduling problem. *IEEE Access*, 7:14513–14522.

Man, H., Odili, J. B., y Mohmad Kahar, M. N. (2016). Solving the traveling salesman's problem using the african buffalo optimization. *Computational Intelligence and Neuroscience*, 2016:1510256.

Martinovic, J., Scheithauer, G., y Valério de Carvalho, J. (2018). A comparative study of the arcflow model and the one-cut model for one-dimensional cutting stock problems. *European Journal of Operational Research*, 266(2):458–471.

Montiel-Arrieta, L. J., Barragán-Vite, I., Hernández-Romero, N., y González-Hernández, M. (2022). Algoritmo del búfalo africano para resolver el problema de corte unidimensional. *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, 10(Especial2):1–8.

Odili, J. B. y Kahar, M. N. M. (2015a). African buffalo optimization (abo): A new meta-heuristic algorithm. En *National Conference for Postgraduate Research Universiti Malaysia PAHANG, Kuantan*, pp. 1–8, Kuantan, Malaysia. NCON-PGR.

Odili, J. B. y Kahar, M. N. M. (2015b). African buffalo optimization (abo): A new metaheuristic algorithm. *Journal of Advanced & Applied Sciences (JAAS)*, 03(03):101–106.

Odili, J. B., Kahar, M. N. M., y Anwar, S. (2015). African buffalo optimization: A swarm-intelligence technique. *Procedia Computer Science*, 76:443–448. 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IEEE IRIS2015).

Ogunranti, G. A. y Oluleye, A. E. (2016). 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*, 9(3):834–859.

Panella, M., Odili, J. B., Noraziah, A., y Zarina, M. (2021). A comparative performance analysis of computational intelligence techniques to solve the asymmetric travelling salesman problem. *Computational Intelligence and Neuroscience*, 2021:6625438.

Parmar, K. B., Prajapati, H. B., y Dabhi, V. K. (2015). Cutting stock problem: A solution based on novel pattern based chromosome representation using modified ga. En *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, pp. 1–7.

Peng, J. y Chu, Z. S. (2010a). A hybrid ant colony algorithm for the cutting stock problem. En *2010 International Conference on Future Information Technology and Management Engineering*, volumen 2, pp. 32–35.

Peng, J. y Chu, Z. S. (2010b). A hybrid multi-chromosome genetic algorithm for the cutting stock problem. En *2010 3rd International Conference on Information Management, Innovation Management and Industrial Engineering*, volumen 1, pp. 508–511.

Reeves, C. (1996). Hybrid genetic algorithms for bin-packing and related problems. *Annals of Operations Research*, 63:371–396.

Renildo, G., Cerqueira, L., y Yanasse, H. (2009). A pattern reduction procedure in a one-dimensional cutting stock problem by grouping items according to their demands. *Journal of Computational Interdisciplinary Sciences*, 1:159–164.

Scheithauer, G. (2018). One-dimensional cutting stock. En Price, C. C., editor, *Introduction to Cutting and Packing Optimization: Problems, Modeling Approaches, Solution Methods*, volumen 263, capítulo 4, pp. 73–122. Springer International Publishing AG, Cham, Switzerland.

Shen, X., Li, Y., Yang, J., y Yu, L. (2007). A heuristic particle swarm optimization for cutting stock problem based on cutting pattern. En Shi, Y., van Albada, G. D., Dongarra, J., y Sloot, P. M. A., editores, *Computational Science – ICCS 2007*, pp. 1175–1178, Berlin, Heidelberg. Springer Berlin Heidelberg.

Strąk, Ł., Skinderowicz, R., Boryczka, U., y Nowakowski, A. (2019). A self-adaptive discrete pso algorithm with heterogeneous parameter values for dynamic tsp. *Entropy*, 21(8).

Strasser, S., Goodman, R., Sheppard, J., y Butcher, S. (1995). A new discrete particle swarm optimization algorithm. En Friedrich, T., editor, *GECCO '16: Genetic and Evolutionary Computation Conference*, pp. 53–60, New York, United States. Association for Computing Machinery.

Umetani, S., Yagiura, M., e Ibaraki, T. (2003). One-dimensional cutting stock problem to minimize the number of different patterns. *European Journal of Operational Research*, 146(2):388–402.

Umetani, S., Yagiura, M., e Ibaraki, T. (2006). One-dimensional cutting stock problem with a given number of setups: A hybrid approach of metaheuristics and linear programming. *Journal of Mathematical Modelling and Algorithms*, 5:43–64.

Valério de Carvalho, J. (2002). Lp models for bin packing and cutting stock problems. *European Journal of Operational Research*, 141(2):253–273.

van Zyl, J.-P. y Engelbrecht, A. P. (2023). Set-based particle swarm optimisation: A review. *Mathematics*, 11(13).

Vishwakarma, R. y Powar, P. L. (2021). An efficient mathematical model for solving one-dimensional cutting stock problem using sustainable trim. *Advances in Industrial and Manufacturing Engineering*, 3:100046.

Wäscher, G., Hausner, H., y Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183:1109–1130.

Yanasse, H. H. y Limeira, M. S. (2006). A hybrid heuristic to reduce the number of different patterns in cutting stock problems. *Computers & Operations Research*, 33(9):2744–2756. Part Special Issue: Anniversary Focused Issue of Computers & Operations Research on Tabu Search.