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. 13 No. 26 (2026) 1-12

Evaluación de Funciones Objetivo para Resolver el Problema de Corte

Objective Functions Evaluation for Solving the One-Dimensional Cutting Stock Problem

E. F. Acevedo-Mendoza[a], I. Barragan-Vite[a,*], J. Medina-Marín[a], G. E. Anaya-Fuentes[a]

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

Resumen

El documento aborda el problema de corte unidimensional que consiste en cortar objetos pequeños o ítems de piezas grandes de material o stocks. A lo largo del tiempo, se han desarrollado diversos modelos basados en programación lineal y estrategias para abordar este problema, que van desde métodos exactos hasta algoritmos heurísticos, metaheurísticos e híbridos. En este estudio se evalúa la efectividad del Algoritmo Discreto del Búfalo Africano con tres diferentes funciones objetivo. La primera función considera minimizar el desperdicio; la segunda, el número de stocks usados y la tercera es una combinación de las dos anteriores. Se realizaron experimentos bajo dos escenarios con instancias de diferente complejidad y características. Los resultados muestran que las funciones con una sola medida de desempeño son adecuadas para una complejidad moderada y alta, mientras que la función con dos medidas de desempeño es idónea para cualquier complejidad, pero es sensible a la heterogeneidad de los ítems.

Palabras Clave: Problema de corte, Algoritmo del Búfalo Africano, Programación Lineal, Algoritmos Heurísticos.

Abstract

The document addresses the one-dimensional cutting stock problem which consists in cutting small objects or items from large pieces of material or stocks. Over time, various linear programming models have been developed and strategies to tackle this problem, ranging from exact methods, to heuristic algorithms, metaheuristics, and hybrid methods. In this study the effectiveness of the Discrete African Buffalo Algorithm is evaluated with three objective functions. The first function considers the minimization of waste; the second one, the number of stocks used, and the third one is a combination of the first two functions. Experiments were conducted under two scenarios with instances of different complexity and features. Results show that the functions with one performance measure are suitable for moderate and high complexity while the function with two performance measures is suitable for any type of complexity but it is sensible to the heterogeneity of the items.

Keywords: Cutting Stock Problem, African Buffalo Algorithm, Linear Programming, Heuristic Algorithms.

1. Introducción

El problema de corte es una clase de problemas de optimización vinculados a la producción industrial y a la gestión de materiales. Actualmente, es uno de los problemas de optimización más estudiados, y las empresas se enfrentan al reto de cortar materiales de gran tamaño, como papel, vidrio, madera o acero, en piezas más pequeñas para responder a una demanda específica. Su meta principal es satisfacer esta demanda empleando la menor cantidad de material posible o reduciendo el desperdicio al mínimo. En ciertos problemas del mundo real, no resulta adecuado enfocarse únicamente en la minimización del desperdicio de material como objetivo principal, ya que este objetivo impacta solo una parte de los costos totales involucrados en el proceso de corte, como el costo de la compra de material, el costo de almacenamiento, la gestión de inventario, el uso de maquinaria y de mano de obra, que pueden ser impactados por objetivos como el número de rollos o placas de material comprados, o bien la reutilización de material.

Este trabajo aborda una variante unidimensional del problema de corte (1D-CSP, *One-dimensional Cutting Stock Problem*), específicamente aquella en la que se utiliza una única pieza de material o stock, según la clasificación de

*Autor para la correspondencia: irvingb@uaeh.edu.mx
Correo electrónico: ac354160@uaeh.edu.mx (Eloísa Fernanda Acevedo-Mendoza), irvingb@uaeh.edu.mx (Irving Barragan-Vite), jmedina@uaeh.edu.mx (Joselito Medina-Marín), ganaya@uaeh.edu.mx (Gustavo Erick Anaya Fuentes).

Historial del manuscrito: recibido el 30/01/2025, última versión-revisada recibida el 05/09/2025, aceptado el 10/09/2025, en línea (postprint) desde el 26/09/2025, publicado el DD/MM/AAAA. **DOI:** https://doi.org/10.29057/icbi.v13i26.14491

(Wäscher, Hausner, & Schumann, 2007). En esta variante, todas las piezas a cortar, o ítems, se obtienen de un único tipo de stock, lo que implica que las piezas de stock empleadas tienen las mismas dimensiones (largo L y ancho W). Además, tanto el stock como los ítems comparten el mismo ancho, aunque estos últimos pueden variar en longitud. Así, el problema se simplifica al encontrar una disposición adecuada de los ítems sobre las piezas de stock en una única dimensión, es decir, a lo largo del stock (L).

Se considera que (Kantorovich, 1960) fue el primero en abordar este problema, al plantear su modelo mediante programación lineal (PL) y emplear el método de multiplicadores para encontrar una solución. Después se desarrolló una heurística en (Hassler, 1975) para generar patrones que incorporan de manera secuencial nuevos patrones de corte a la solución existente hasta cubrir completamente la demanda. En cada iteración, el método elige un patrón de corte que combina una baja pérdida de material con una alta frecuencia de uso (número de veces que se aplica dicho patrón). Precisamente para los problemas de patrones se formuló un problema de minimización de patrones de corte en (Farley & Richardson, 1984) como un problema de carga fija, aplicando el método simplex para sustituir las variables básicas, correspondientes a los patrones de corte, por variables excedentes, con el objetivo de disminuir la cantidad de patrones utilizados. En (Foerster & Wäscher, 2000) se presentó el método de solución KOMBI, que amplía este procedimiento al basarse en la premisa de que, al combinar patrones de corte, la suma de las frecuencias de los nuevos patrones debe ser igual a la suma de las frecuencias de los patrones originales, garantizando así que se mantenga constante la cantidad de material de entrada. Para el mismo año en el trabajo de (Vanderbeck, 2000) se propuso un método exacto para resolver el problema de minimización de patrones, formulando como un modelo de programación entera cuadrática. En el trabajo de (Kolen & Spieksma, 2000) se desarrolló un algoritmo de ramificación y acotación diseñado para obtener las soluciones óptimas de Pareto en un conjunto de instancias pequeñas del problema. Tres años más tarde, en (Umetani, Yagiura, & Ibaraki, 2003) se propuso una formulación considerando un enfoque de búsqueda local iterada para minimizar la cantidad de stocks, limitando el uso a una cantidad máxima predefinida de patrones de corte distintos. Este algoritmo de búsqueda local explora la vecindad generada al modificar un patrón de corte dentro del conjunto actual, aplicando perturbaciones basadas en la solución dual del problema auxiliar de programación lineal.

En (Lee, 2007) se presentó una heurística de búsqueda local denominada CRAWLA, basada en programación lineal entera. Esta heurística aborda de manera integral tanto el problema principal como el subproblema del enfoque tradicional de generación de patrones, logrando un modelo unificado que crea directamente nuevos patrones de corte. Aunque el método destina tiempo a la generación de nuevas columnas, asegura que estas contribuyan a una mejora en el modelo de programación lineal entera, en lugar de limitarse a la mejora continua que suele buscar el enfoque tradicional de generación de columnas basado en precios. Un año más tarde se propuso un algoritmo de ramificación (branch-price-and-cut) para resolver de manera exacta el problema de minimización de patrones en (Alves & de Carvalho, 2008) y en (Alves, Macedo, & de Carvalho, 2009) abordaron este problema utilizando la generación de columnas y presentaron diversas estrategias para reforzar la formulación planteada en su estudio. Se obtuvieron límites inferiores de alta calidad tanto a partir del nuevo modelo de programación entera como de un modelo basado en programación por restricciones. De manera más reciente algunos estudios han abordado métodos clásicos para resolver casos particulares del 1D-CSP como se muestra en (Sarper & Jaksic, 2019), (Sa Santos & Nepomuceno, 2022).

El Algoritmo del Búfalo Africano (ABO) ha ganado atención en la literatura reciente como una técnica innovadora dentro de los algoritmos de optimización inspirados en la naturaleza. Este algoritmo, que emula el comportamiento colectivo de los búfalos africanos en su búsqueda de recursos, se ha actualizado y mejorado en diversos artículos, reflejando su creciente importancia en la solución de problemas complejos en diferentes áreas. Entre los artículos recientes, las contribuciones más destacadas han explorado la adaptación del Algoritmo Discreto del Búfalo Africano (DABO) a problemas de optimización combinatoria cuyas soluciones requieren ser discretas, (Zhou, Jiang, & Wang, 2020), (Jiang, Zhu, & Deng, 2020), (Gherboudj, 2018). Estas aplicaciones han mostrado resultados prometedores al comparar su desempeño con otros algoritmos bio-inspirados, como el enjambre de partículas o los algoritmos genéticos, (Montiel-Arrieta, Barragan-Vite, Seck-Tuoh-Mora, Hernandez-Romero, & Gonzalez-Hernandez, 2023). Nuevas variantes del ABO continúan siendo objeto de investigaciones que expanden sus aplicaciones y refinan su funcionamiento como en (Singh, Meena, Slowik, & Bishnoi, 2020), consolidándose como una herramienta valiosa en el ámbito de la optimización. En particular, en la variante DABO se han realizado ajustes para abordar problemas con soluciones discretas. No obstante, en la solución del 1D-CSP con esta variante del ABO aún no se han explorado modificaciones que mejoren su desempeño como el uso de funciones *fitness* que permitan evaluar de manera adecuada las soluciones, hasta el empleo de métodos de cruza o mutación que mejoren la eficiencia y efectividad del algoritmo.

En este sentido el presente trabajo busca mejorar la efectividad del DABO para resolver el 1D-CSP a través de una función objetivo o *fitness* con dos criterios para guiar la búsqueda de la mejor solución ya que, usualmente los estudios sobre el 1D-CSP se enfocan en el uso de un solo criterio. En este trabajo se realizó un análisis de tres funciones objetivo para resolver el 1D-CSP. La primera considera solamente la minimización de desperdicio total como se propuso en (Montiel Arrieta, Barragán Vite, Hernández Romero, & González Hernández, 2023); la segunda, considera solamente la minimización del número de stocks usados, y la tercera función toma en cuenta la minimización tanto el desperdicio como el número de stocks con desperdicio. Esto se ha planteado así debido a que existen soluciones en el 1D-CSP que pueden tener el mínimo desperdicio total, pero un gran número de stocks usados, y viceversa. Para determinar la efectividad de las tres funciones, se emplea un conjunto de 70 instancias de la literatura con diferente complejidad. Estas instancias se seleccionaron debido a que, en un primer escenario 10 de las instancias poseen una complejidad variable con stocks de diferente longitud, mientras que en un segundo escenario, el grupo restante de instancias consideran el mismo tamaño de stock lo que permite eliminar la influencia de este

factor en los resultados y en la comparación de las funciones objetivo.

Se determinó que la función que combina tanto el desperdicio total como el número de stocks usados ofrece los mejores resultados para instancias desde baja hasta alta complejidad, mientras que las funciones que consideran un solo criterio, como desperdicio o número de stocks usados en total, arrojan buenos resultados para instancias de moderada y alta complejidad. Los resultados obtenidos en este trabajo contribuyen a ampliar el conocimiento sobre la implementación del DABO en la solución del 1D-CSP, al comparar y mostrar su desempeño con funciones *fitness* que consideran un criterio o dos criterios de manera conjunta para evaluar las soluciones y cómo los resultados mejoran cuando se emplean dos criterios, lo que puede ampliarse a problemas similares como el *Binpacking Problem.*

El trabajo se limita a comparar los objetivos de reducción del desperdicio y del uso de stocks, de manera individual y de forma conjunta, con la implementación del DABO para resolver el 1D-CSP, sin explorar el uso de otros métodos de cruza, mutación o reinicio de la manada. Asimismo, solamente se realiza la comparación del DABO con otras implementaciones del ABO que han utilizado el mismo tipo de instancias seleccionadas en este trabajo.

Las siguientes secciones se organizan como sigue: en la Sección 2 se describen los conceptos básicos sobre el problema de corte, también sobre el algoritmo discreto del búfalo africano con el que se desarrolla el trabajo, en el que se implementa una función de costo que considera dos medidas de desempeño de manera conjunta. En la Sección 3 se describe la metodología para resolver el problema de corte con los tres modelos propuestos y se presentan los resultados obtenidos acompañado del análisis correspondiente.

2. Conceptos básicos

En esta sección se proporcionan los conceptos clave del 1D-CSP así como los modelos matemáticos que usualmente se han utilizado en la literatura en la implementación de los diferentes algoritmos desarrollados para resolverlo. Además, se ofrecen las definiciones del DABO y los pasos para su implementación.

2.1. Problema de corte de una dimensión

De acuerdo con la clasificación de (Wäscher et al., 2007), el 1D-CSP corresponde al problema de un solo stock, es decir, todos los stocks disponibles tienen las mismas dimensiones, ancho (W) y largo (L). Los ítems que se obtienen de los stocks varían en el largo, pero el ancho es fijo e igual al del stock. Se considera que el número de stocks disponibles es ilimitado. La Figura 1 muestra un ejemplo de un conjunto de ítems que deben ser obtenidos de piezas de stock de longitud $L = 10$. El plan de corte consiste en usar seis stocks para cortar todos los ítems requeridos, donde las partes de color verde de los stocks indican desperdicio.

En general y de manera formal, una instancia del 1D-CSP puede definirse por la longitud del stock L, un número m de ítems con longitudes l_i y demanda d_i, para $i = 1,2,\dots,m$. Aunque en la literatura existen diferentes modelos matemáticos para abordar la solución del 1D-CSP, aquí mostramos tres modelos cuyas funciones objetivo son objeto de estudio en este trabajo.

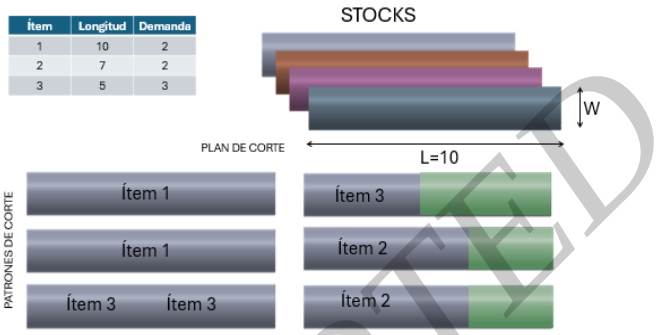


Figura 1. Ejemplo del 1D-CSP con tres ítems y el plan de corte.

2.1.1. Modelo enfocado en la minimización del desperdicio

El modelo (2)-(6) considera la minimización del desperdicio total como función objetivo (*FO1*), donde:

$$w_j = L - \sum_{i=1}^{n} x_{ij} l_i, \qquad j = \{1, 2, \dots, m\} \quad (1)$$

y x_{ij} es la cantidad de órdenes del ítem i obtenidas del stock j. La restricción (2) indica que la cantidad total de piezas del ítem i obtenidas de los stocks j no debe exceder su demanda, mientras que la restricción (3) muestra que la longitud de cada patrón de corte no debe exceder la longitud del stock, considerando el desperdicio del patrón.

$$\min\ FO1 = \sum_{j=1}^{m} w_j \quad (2)$$

s. a.

$$\sum_{j=1}^{m} x_{ij} = d_i, \qquad i = \{1, 2, \dots, n\} \quad (3)$$

$$\sum_{i=1}^{n} x_{ij} l_i + w_j = L y_j, \qquad j = \{1, 2, \dots, m\} \quad (4)$$

$$y_j \in \{0,1\} \quad (5)$$

$$x_{ij} > 0 \text{ y entero} \quad (6)$$

2.1.2 Modelo enfocado en la minimización del número de stocks usados

Un ligero cambio en el modelo enfocado en el desperdicio produce el modelo (7) – (11) donde la función objetivo (*FO2*) considera el número total de stocks usados, el cual debe ser minimizado.

$$\min FO2 = \sum_{j=1}^{m} y_j \quad (7)$$

$s.a.$

$$\sum_{j=1}^{m} x_{ij} = d_i, \qquad i = \{1, 2, \dots, n\} \quad (8)$$

$$\sum_{i=1}^{n} x_{ij} l_i = L y_j, \qquad j = \{1, 2, \dots, m\} \quad (9)$$

$$y_j \in \{0,1\} \quad (10)$$

$$x_{ij} \geq 0 \text{ y entero} \quad (11)$$

2.1.3 Modelo enfocado en la minimización conjunta del número de stocks con desperdicio y el desperdicio total

El modelo (13) – (15) considera la minimización tanto del desperdicio total como del número de stocks con desperdicio como se muestra en su función objetivo (*FO3*).

$$\min FO3 = \frac{1}{m+1}\left(\sum_{j=1}^{m} \sqrt{\frac{w_j}{L}} + \sum_{j=1}^{m} \frac{V_j}{m}\right) \quad (13)$$

$s.a.$

$$\sum_{j=1}^{m} x_{ij} = d_i, \qquad i = \{1, 2, \dots, n\} \quad (14)$$

$$x_{ij} \geq 0 \text{ y entero} \quad (15)$$

donde:

$$V_j = \begin{cases} 1, & \text{si } w_j > 0 \quad j = \{1, 2, \dots m\} \\ 0, & \text{en otro caso} \end{cases} \quad (16)$$

2.2 Algoritmo Discreto del Búfalo Africano

En este estudio se retoma el trabajo de (Barragan-Vite, Montiel-Arrieta, Seck-Tuoh-Mora, Hernandez-Romero, & Medina-Marin, 2023) en el que se introdujo el Algoritmo Discreto del Búfalo Africano para resolver el 1D-CSP (DABO-1DCSP) en un espacio de búsqueda discreto. El algoritmo está adaptado para problemas en los que las soluciones posibles están definidas en un conjunto finito de valores, esto lleva a que la adaptación del comportamiento colectivo conlleve a realizar movimientos en los que los búfalos cambian de posición entre diferentes soluciones discretas según las mejores posiciones observadas. Esto permite que el algoritmo explore y explote el espacio de soluciones factibles de manera eficiente para la naturaleza discreta del problema.

De cierta forma su funcionamiento inicia al igual que en el ABO original, (Odili, Mohd Nizam, & Shahid, 2015). Se genera un conjunto inicial de soluciones, después se evalúa el desempeño de cada solución con base en su calidad con respecto al objetivo del problema. En cada iteración las soluciones se mueven en el espacio de búsqueda, tomando en cuenta la calidad de las soluciones colindantes y, así mismo, la comunicación entre las soluciones les permite compartir información sobre los puntos prometedores en el espacio discreto, lo cual les ayuda a moverse hacia soluciones potencialmente mejores, el algoritmo se detiene cuando se alcanza un número máximo de iteraciones o cuando se logra una solución satisfactoria que cumple con los requisitos del problema. La explotación del espacio de soluciones se realiza mediante la Ecuación con la que se actualiza el *fitness* de cada búfalo, mientras que con la Ecuación se realiza la exploración. Con los valores de fitness de cada nueva posición se determina el nuevo mejor búfalo o solución para la siguiente iteración.

$$m_{k+1} = m_k + lp1(bgmax - w_k) + lp2(bpmax_k - w_k) + lp3(br - w_k) \quad (17)$$

$$w_{k+1} = \frac{w_k + m_{k+1}}{\lambda} \quad (18)$$

En la Ecuación (17), $lp1$, $lp2$ y $lp3$ son factores de aprendizaje. A diferencia del ABO original, el término $lp3(br - w_k)$ se introduce para representar el mecanismo de aprendizaje aleatorio, donde br es un búfalo o solución seleccionada de manera aleatoria de la manada actual, (Barragan-Vite, et al., 2023), (Jiang, Zhu, & Deng, 2020).

Para discretizar las soluciones se emplean las Ecuaciones (19) y (20), en las que se realizan operaciones de cruza y mutación de acuerdo a las Ecuaciones (21) y (22), donde CR indica una operación de cruza y MU una operación de mutación. Asimismo, $lp1 + lp2 + lp3 = 1$ debe satisfacerse, de tal forma que estos valores representan las tasas de diferentes tipos de cruza, y λ es la tasa de mutación. El símbolo $\otimes$ indica que si se satisface la condición mostrada 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. Por otro lado, **r** es un número aleatorio en el intervalo [0,1].

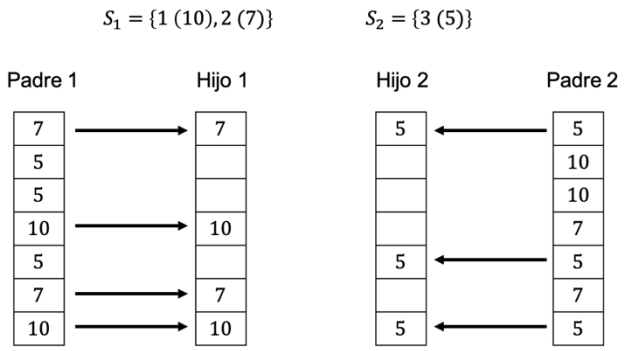
$$m_{k+1} = lp1 \otimes CR\,(bgmax, w_k) \oplus lp2 \otimes CR(bpmax_k, w_k) \oplus lp3 \otimes CR(br, w_k) \quad (19)$$

$$w_{k+1} = \lambda \otimes MU(m_k) \quad (20)$$

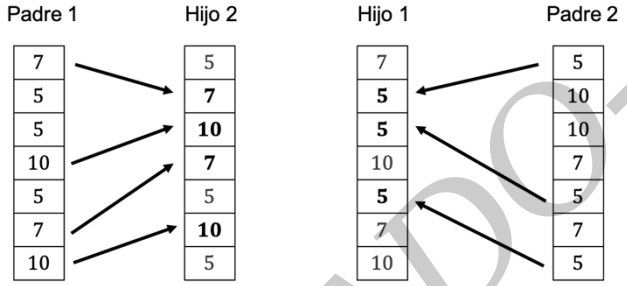
$$m_{k+1} = \begin{cases} CR\,(bgmax, w_k), & \text{si } \mathbf{r} < lp1 \\ CR(bpmax_k, w_k), & \text{si } lp1 \leq \mathbf{r} < lp1 + lp2 \\ CR(br, w_k), & \text{si } lp1 + lp2 \leq r < lp1 + lp2 + lp3 \end{cases} \quad (21)$$

$$w_{k+1} = \begin{cases} MU(m_k), & \text{si } \mathbf{r} < \lambda \\ m_k, & \text{si } \mathbf{r} \geq \lambda \end{cases} \quad (22)$$

En este trabajo se emplea la adaptación de la operación de cruza ITX empleada en (Barragan-Vite, et al., 2023), la cual se ilustra en la Figura 2 donde S_1 y S_2 son subconjuntos de ítems asignados aleatoriamente tal que $S_1 \cap S_2 = \emptyset$. Entre paréntesis se muestra la longitud correspondiente al ítem de acuerdo a la instancia de la Figura 1. Por otra parte, la operación de mutación se realiza con la operación swap, es decir, se intercambian entre sí de manera aleatoria dos posiciones de cada solución como se muestra en la Figura 3.



a) Copia de longitudes de S_1 (S_2, respectivamente) localizadas en Padre 1 (Padre 2) a Hijo 1 (Hijo 2).



b) Copia de longitudes de S_1 (S_2, respectivamente) localizadas en Padre 1 (Padre 2) a Hijo 2 (Hijo 1) en las posiciones faltantes.

Figura 2. Operación de cruza ITX.

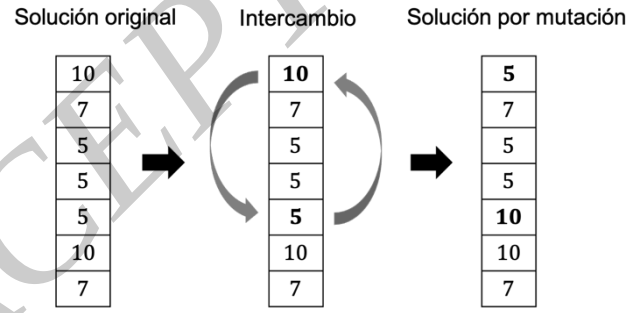


Figura 3. Operación de mutación swap o intercambio de dos longitudes de ítems.

3. Experimentos y Resultados

El propósito de este estudio es analizar la efectividad de las funciones objetivo descritas en la Sección 2 cuando se emplean como funciones *fitness* en el DABO-1DCSP, y determinar cuál o cuáles producen un mejor desempeño del algoritmo. Se plantean dos escenarios de experimentos. El primer escenario consiste en un grupo de diez instancias con complejidad variable (número de ítems, n), donde el largo de los stocks también es variable. En este escenario se empleó el método de Taguchi para ajustar los valores de los parámetros de ejecución del algoritmo como se mostró en (Jiang, Zhu, & Deng, 2020). El ajuste se realizó con cada una de las funciones. En el segundo escenario se realizaron experimentos con cuatro grupos de instancias denominadas *binpack1*, *binpack2*, *binpack3* y *binpack*4 con incremento de complejidad. Sin embargo, el largo de los stocks es el mismo en todos los grupos lo cual nos permite tener una mejor comprensión del desempeño de cada función *fitness*. En este escenario no se ajustaron los parámetros para la ejecución del algoritmo, sino que se tomaron los mejores valores ajustados de cada función *fitness* en el primer escenario con base en su clasificación (Rank). Todos los experimentos se realizaron en una computadora con procesador Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.20 GHz y RAM de 8GB. El DABO-1DCSP fue codificado en Python3.11.1. El código del DABO-1DCSP con el que se realizaron las pruebas se puede encontrar en https://github.com/viteib/articulo_AcevedoMendoza.git.

3.1 Experimentos y resultados del escenario uno.

En este escenario se probaron las tres funciones $FO1$, $FO2$ y $FO3$ descritas en la Sección 2 con el DABO-1DCSP en un conjunto de diez instancias tomadas de (Liang, Yao, Newton, & Hoffman, 2002). Esto es, cada función se empleó de manera independiente como la función *fitness* que guía la búsqueda de la mejor solución o la solución óptima. Los datos necesarios de las instancias se muestran en la Tabla 1, de donde puede observarse que varían en complejidad (número de ítems, n). Además, la columna cuatro muestra el número de ítems diferentes (n_{dif}) en todas las instancias, así como la variabilidad entre la instancia con el menor número de ítems y la instancia con el mayor número de ítems, expresada como desviación estándar (D. E.).

Tabla 1: Conjunto de instancias de prueba del primera fase.

Instancia	n	L	n_{dif} (D. E.)
1a	20	14	4 (10.55)
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 valores de los parámetros se ajustaron mediante el método de Taguchi con el diseño $L_{25}(5^6)$ para seis parámetros con cinco niveles cada uno, los cuales se muestran en la Tabla 2, donde *ls* representa el *life span* para el procedimiento de re-inicialización del búfalo que alcance este valor a lo largo de las iteraciones. Los experimentos para el ajuste se realizaron con base en la instancia 6a como un punto medio en la complejidad de las instancias de la Tabla 1 y el valor de respuesta usado fue

el promedio de diez corridas con cada una de las medidas de desempeño de las funciones objetivo $FO1$, $FO2$ y $FO3$, correspondientemente. El arreglo ortogonal para el diseño experimental se empleó tal como se describe en (Barragan-Vite, et al., 2023). Los resultados de respuesta para medias del método de Taguchi para cada una de las funciones $FO1$, $FO2$ y $FO3$, se muestran en las Tablas 3, 4 y 5, respectivamente. Los resultados de la Tabla 4 se han replicado del trabajo de (Barragan-Vite, et al., 2023).

Tabla 2: Diseño experimental para el ajuste de los parámetros del DABO-1D-CSP de la fase 1 de experimentos.

Factor (Parámetro)	Nivel 1	Nivel 2	Nivel 3	Nivel 4	Nivel 5
Población	80	100	120	150	200
Número de iteraciones	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: Respuesta para medias y clasificación (Rank) de los factores para $FO1$ (desperdicio total).

Nivel	Pob.	Núm. iter.	$lp1$	$lp2$	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

Tabla 4: Respuesta para medias y clasificación (Rank) de los factores para $FO2$ (stocks usados).

Nivel	Pob.	Núm. iter.	$lp1$	$lp2$	λ	ls
1	87.20	86.98	86.80	86.84	87.04	86.70
2	86.92	86.98	**86.70**	86.92	86.86	86.90
3	86.56	86.84	86.82	86.72	86.64	**86.54**
4	**86.50**	86.68	86.72	**86.54**	**86.62**	87.00
5	86.68	**86.38**	86.82	86.84	86.70	86.72
Delta	0.70	0.60	0.12	0.38	0.42	0.46
Rank	1	2	6	5	4	3

Tabla 5: Respuesta para medias y clasificación (Rank) de los factores para $FO3$ (stocks usados y desperdicio).

Nivel	Pob.	Núm. iter.	$lp1$	$lp2$	λ	ls
1	0.2059	0.2060	0.2275	**0.1863**	0.2035	**0.1902**
2	0.2007	0.2082	0.2139	0.1961	0.1934	0.1905
3	0.1884	0.2008	0.1903	0.1981	**0.1889**	0.2022
4	0.1956	**0.1766**	**0.1704**	0.2031	0.1956	0.1937
5	**0.1842**	0.1832	0.1728	0.1913	0.1934	0.1982
Delta	0.0217	0.0316	0.0572	0.0168	0.0146	0.0121
Rank	3	2	1	4	5	6

Con base en los resultados del método de Taguchi, se seleccionaron los valores de los parámetros que se muestran en la Tabla 6 para cada una de las funciones *fitness*. Los resultados de los experimentos con cada una de las funciones *fitness* se muestran en la Tabla 7. Se realizaron 50 ejecuciones con cada una de las funciones. Los valores en negrita indican el resultado mínimo entre las funciones comparadas.

Tabla 6: Valores seleccionados de los parámetros para cada función *fitness* para los experimentos del escenario uno.

Función	Pob.	Núm. iter.	$lp1$	$lp2$	λ	ls
$FO1$	120	1000	0.2	0.1	0.25	15
$FO2$	150	1500	0.2	0.2	0.2	15
$FO3$	200	1000	0.4	0.1	0.15	5

Tabla 7: Resultados de Stock Promedio ($\bar{S}$) con la ejecución del DABO-1DCSP con las diferentes funciones *fitness*.

Instancia	$FO1$	$FO2$	$FO3$
	$\bar{S}$	$\bar{S}$	$\bar{S}$
1a	**9**	**9**	**9**
2a	23.94	23.96	**23.4**
3a	16	16	**15.96**
4a	19.94	19.96	**19.64**
5a	55.76	55.68	**55.7**
6a	**87.52**	87.68	88.2
7a	74.82	**74.8**	**74.8**
8a	**161.08**	161.1	162.58
9a	**168.88**	169.08	170.52
10a	245.78	**245.54**	249.22

De la Tabla 7 se observa que la ejecución del DABO-1DCSP con la función $FO3$ obtiene el mejor stock promedio en seis de las diez instancias. En segundo lugar, la función $FO1$ que considera solamente el desperdicio, obtiene mejores resultados en cuatro de las diez instancias, mientras que la $FO2$ solamente obtiene mejores resultados en tres de las diez instancias. No obstante, aunque la función $FO3$ resuelve más instancias, la diferencia con los resultados con las otras dos funciones es muy pequeña. La función $FO1$ logra resolver mejor instancias de mayor complejidad, es decir, entre 200 y 400 ítems. En la Tabla 8 se muestran los tiempos computacionales obtenidos con cada una de las funciones con los que se lograron sus mejores resultados de stock promedio. La función $FO3$ obtiene tiempos ligeramente superiores a los de las otras dos funciones, particularmente donde las tres funciones logran resultados similares o iguales, como en las instancias 1a y 7a. La ligera superioridad en tiempo de la función $FO3$ puede atribuirse a su propia evaluación en cada iteración, ya que se consideran dos términos en la función.

Tabla 8: Resultados de tiempo promedio $\bar{T}$ con la ejecución del DABO-1DCSP con las diferentes funciones *fitness*.

Instancia	$FO1$	$FO2$	$FO3$
	$\bar{T}$	$\bar{T}$	$\bar{T}$
1a	210.6	216.95	213.9
2a	378.7	362.98	418.5
3a	342.9	337.4	383.4
4a	362.9	361.6	376.2
5a	724.6	711.8	716.5
6a	1112.9	1079.1	1105.8
7a	1023.7	996.2	991.9
8a	2208.0	2127.7	2142.9
9a	2216.3	2182.3	2216.1
10a	3347.6	3285.5	3221.6

La Figura A1 del apéndice A muestra las gráficas de convergencia de las funciones $FO1$, $FO2$ y $FO3$ para cada una de las instancias de la Tabla 1. La comparación se realizó para las primeras 1000 iteraciones en la ejecución de cada función. Se observa que la función $FO3$ converge de manera más rápida que las funciones $FO1$ y $FO2$ en intancias de baja y moderada complejidad (instancias 1a a 4a), pero en instancias de alta

complejidad la función $FO3$ tiene un marcado menor desempeño que las otras dos funciones. Las funciones $FO1$ y $FO2$ tienen un comportamiento muy similar a lo largo de las instancias, siendo la función las función $FO1$ ligeramente mejor que la función $FO2$. Debido a que la función $FO3$ tuvo mejores resultados de Stock Promedio, en la Tabla 9 se muestra su comparación con los algoritmos ABO-1DCSP y ABO-CSP, que son otras implementaciones recientes del ABO para resolver el 1D-CSP. Los resultados del algoritmo ABO-1DCSP se tomaron de (Montiel-Arrieta, et al., 2023) y los resultados del ABO-CSP, de (Montiel-Arrieta, Barragan-Vite, Hernandez-Romero, & Gonzalez-Hernandez, 2022). El algoritmo ABO-CSP emplea en su función objetivo únicamente el número de stocks usados, mientras que el algoritmo ABO-1DCSP emplea el desperdicio total. Ambos algoritmos usan el método de discretización *Ranked Order Value*. El algoritmo ABO-CSP reinicializa la población de búfalos (soluciones) cada diez iteraciones. El algoritmo ABO-1DCSP realiza un procedimiento adicional antes de reinicializar la población, que consiste en formar un nuevo $bgmax$ con base en las mejores soluciones de la última iteración antes de la reiniciar toda la población. La comparación de los resultados muestra que el ABO-1DCSP obtiene los mejores resultados en todas las instancias bajo prueba. Solamente en la instancia 1a el algoritmo DABO-1DCSP con la función $FO3$ iguala al resultado del ABO-1DCSP. Sin embargo, el DABO-1DCSP- $FO3$ supera ligeramente al ABO-CSP en todas las instancias, aunque las diferencias entre las soluciones no son muy marcadas.

Tabla 9: Comparación del Stock Promedio $\bar{S}$ con otras implementaciones del algoritmo ABO.

Instancia	DABO-1DCSP-$FO3$	ABO-CSP	ABO-1DCSP
	$\bar{S}$	$\bar{S}$	$\bar{S}$
1a	**9**	9.26	**9**
2a	23.4	24.52	**23**
3a	15.96	16	**15**
4a	19.64	20	**19**
5a	55.7	56.62	**53**
6a	88.2	89.46	**79.08**
7a	74.8	75.62	**68**
8a	162.58	162.76	**144.9**
9a	170.52	172.4	**150**
10a	249.22	250.4	**217.2**

3.2 Experimentos y resultados del escenario dos

Los experimentos del segundo escenario se realizaron con un conjunto de cuatro grupos de instancias denominadas *binpack1*, *binpack2*, *binpack3* y *binpack4*. Estas instancias usan una misma longitud del stock, pero la complejidad incrementa en los grupos de instancias, como se muestra en la Tabla 10.

Tabla 10: Detalles de las instancias de prueba del escenario dos.

Grupo de instancias	Núm. de intancias	n	L	n_{dif} (D. E.)
binpack1	20	120	150	11 (2.71)
binpack2		250		8 (2.14)
binpack3		500		2 (0.40)
binpack4		1000		1 (0)

Para la ejecución del DABO-1DCSP, los valores de los parámetros fueron seleccionados de los resultados del método de Taguchi obtenidos en el escenario uno para cada función objetivo de acuerdo a la clasificación (Rank) mínima de cada parámetro. Por ejemplo, el parámetro de Población tuvo un $Rank = 2$ en los resultados de $FO1$, un $Rank = 1$ para $FO2$ y un $Rank = 3$, para $FO3$, por lo que para los experimentos se seleccionó una Población de 150. De las misma manera se obtuvieron los valores del resto de los parámetros. Estos valores se muestran en la Tabla 11. Los resultados para el stock promedio se muestran en las Tablas B1 a B4 del Apéndice B. También se muestra el óptimo teórico de cada instancia en la segunda columna de cada tabla de resultados, donde este valor se calcula con la Ecuación (23).

$$Opt. = \left\lceil \frac{\sum_{i=1}^{m} l_i d_i}{L} \right\rceil \tag{23}$$

Los resultados del segundo escenario muestran que la función $FO3$ es adecuada para resolver todas las instancias desde *binpack1* hasta *binpack4*. La función $FO3$ sobresale de manera muy evidente para resolver las intancias de estos grupos, a pesar de que la complejidad incrementa de manera muy marcada. No obstante, ninguna de las funciones muestra tener resultados cercanos al valor óptimo teórico de las instancias bajo prueba. La función $FO3$ tiene los porcentajes más cercanos al valor óptimo teórico, apenas superando el 10% en las instancias de *binpack4*; aunque se desearía que fueran menores al 1% como otros algoritmos logran obtener.

En los resultados de los dos escenarios puede observarse que la función $FO3$ puede verse afectada por la variabilidad en el número de items diferentes, ya que a pesar de que existen instancias en ambos escenarios con una cantidad de items similar, la variabilidad (D. E.) de ítems diferentes en el escenario uno es mucho mayor que en las instancias del escenario dos.

Tabla 11: Valores seleccionados de los parámetros para las funciones *fitness* para los experimentos del escenario dos.

Pob.	Núm. iter.	$lp1$	$lp2$	λ	ls
150	1000	0.4	0.1	0.25	15

4. Conclusiones

Resolver problemas como el 1D-CSP de manera eficiente tiene beneficios importantes para las empresas, como la reducción de costos de material, disminución de residuos optimización del tiempo de producción y una mejor gestión de recursos, además desde una perspectiva ambiental, optimizar los problemas de corte contribuye a la sostenibilidad al minimizar el desperdicio y promover el uso eficiente de los materiales. En este trabajo se utilizó el DABO-1DCSP para resolver el 1D-CSP utilizando tres funciones objetivo que minimizan el desperdicio, el número de *stocks* usados o ambas métricas de manera conjunta. El propósito fue determinar con qué tipo de función es más efectivo el DABO-1DCSP. Los resultados de dos escenarios desarrollados muestran que las funciones con una sola medida de desempeño son más efectivas con instancias de moderada y alta complejidad, mientras que la función con dos medidas de desempeño es

idónea para instancias de baja, media y alta complejidad, en términos generales, aunque la diversidad en el tipo ítems puede afectar su desempeño.

En conclusión, el presente documento ofrece un análisis exhaustivo del problema de corte unidimensional, destacando al Algoritmo Discreto del Búfalo Africano como una herramienta innovadora en el campo de la optimización industrial. La evaluación de las funciones objetivo (minimización del desperdicio, reducción del uso de stocks y su combinación), permite concluir que la combinación de las dos funciones con un único objetivo es más efectiva en la solución del problema 1DCSP aunque su desempeño se aleja un poco más de los valores óptimos teóricos cuando la cantidad de los ítems se incrementa y su heterogeneidad disminuye. Por lo tanto, este estudio no solo profundiza en la comprensión teórica y la aplicabilidad práctica del DABO en problemas combinatorios, sino que también fortalece su utilidad al integrar ajustes y estrategias adaptativas orientadas a mejorar su desempeño. Los hallazgos obtenidos aportan contribuciones significativas tanto al ámbito teórico como a la implementación práctica en la resolución de problemas industriales de alta complejidad. En un trabajo futuro se planea explorar otros procedimientos de cruza y mutación, así como de reinicio de la manada que pudieran mejorar el desempeño del DABO-1DCSP con el uso de la función $FO3$, y al mismo tiempo ampliar las comparaciones mediante un conjunto más grande de instancias y con algoritmos diferentes al ABO o basados en otro tipo de dinámica de búsqueda.

Referencias

Barragan-Vite, I., Montiel-Arrieta, L. J., Seck-Tuoh-Mora, J. C., Hernandez-Romero, N., & Medina-Marin, J. (2023). Algoritmo discreto del búfalo africano para resolver el problema de corte de material de una dimensión. *Pädi: Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, 123-132.

Odili, J. B., Mohd Nizam, K. M., & Shahid, A. (2015). African Buffalo Optimization: A Swarm-Intelligence Technique. *2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS 2015)* (pp. 443-448). Malaysia: Elsevier B. V.

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

Liang, K.-H., Yao, X., Newton, C., & Hoffman, D. (2002). Anewevolutionaryapproachtocuttingstockproblems with and without contiguity . *Computers & Operations Research*, 1641-1659.

Montiel-Arrieta, L. J., Barragan-Vite, I., Seck-Tuoh-Mora, J. C., Hernandez-Romero, N., & Gonzalez-Hernandez, M. (2023). Minimizing the total waste in the one-dimensional cutting stock problem with the African buffalo optimization algorithm. *PeerJ Computer Science*, e1728.

Montiel-Arrieta, L. J., Barragan-Vite, I., Hernandez-Romero, N., & Gonzalez-Hernandez, M. (2022). Algoritmo del búfalo africano para resolver el problema de corte unidimensional. *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, 1-8.

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

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

Hassler, R. (1975). Control de cambios en el patron de corte en problemas de corte unidimensionales . *Investigación de operaciones*.

Farley, A., & Richardson, K. (1984). Problemas de carga fija con cargas fijas idénticas. *Revista Europea de Investigación Operativa* .

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

Umetani, S., Yagiura, M., & Ibaraki, T. (2003). Una búsqueda local basada en LP para el problema de corte unidimensional utilizando un número dado de patrones de corte. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*.

Lee, J. (2007). Generación de columnas in situ para un problema de material de corte. *Computer & Operations Research.*

Alves, C., & de Carvalho, J. (2008). A branch-and-price-and-cut algorithm for the pattern minimization problem. *ALIO/EURO V Conference on Combinatorial Optimization.* (pp. 435-453). Paris: EDP Sciences.

Alves, C., Macedo, R., & de Carvalho, J. V. (2009). New lower bounds based on column generation and constraint programming for the pattern minimization problem. *Computers & Operations Research, 36*(11), 2944-2954.

Gherboudj, A. (2018). Two discrete binary versions of African Buffalo Optimization metaheuristic. *ACSIT, ICITE, SIPM - 2018* (pp. 33-46). Academy & Industry Research Collaboration Center (AIRCC).

Zhou, H., Jiang, T., & Wang, Y. (2020). Discrete African Buffalo Optimization Algorithm for the Low-carbon Flexible Job Shop Scheduling Problem. *Journal of Advanced Manufacturing Systems, 9*(4), 837-854.

Kolen, A. W., & Spieksma, F. C. (2000). Solving a bi-criterion cutting stock problem with open-ended demand: a case study. *Journal of the Operational Research Society*, 1238-1247.

Singh, P., Meena, N. K., Slowik, A., & Bishnoi, S. K. (2020). Modified African Buffalo Optimization for Strategic Integration of Battery Energy Storage in Distribution Networks. *IEEE Access*, 14289-14301.

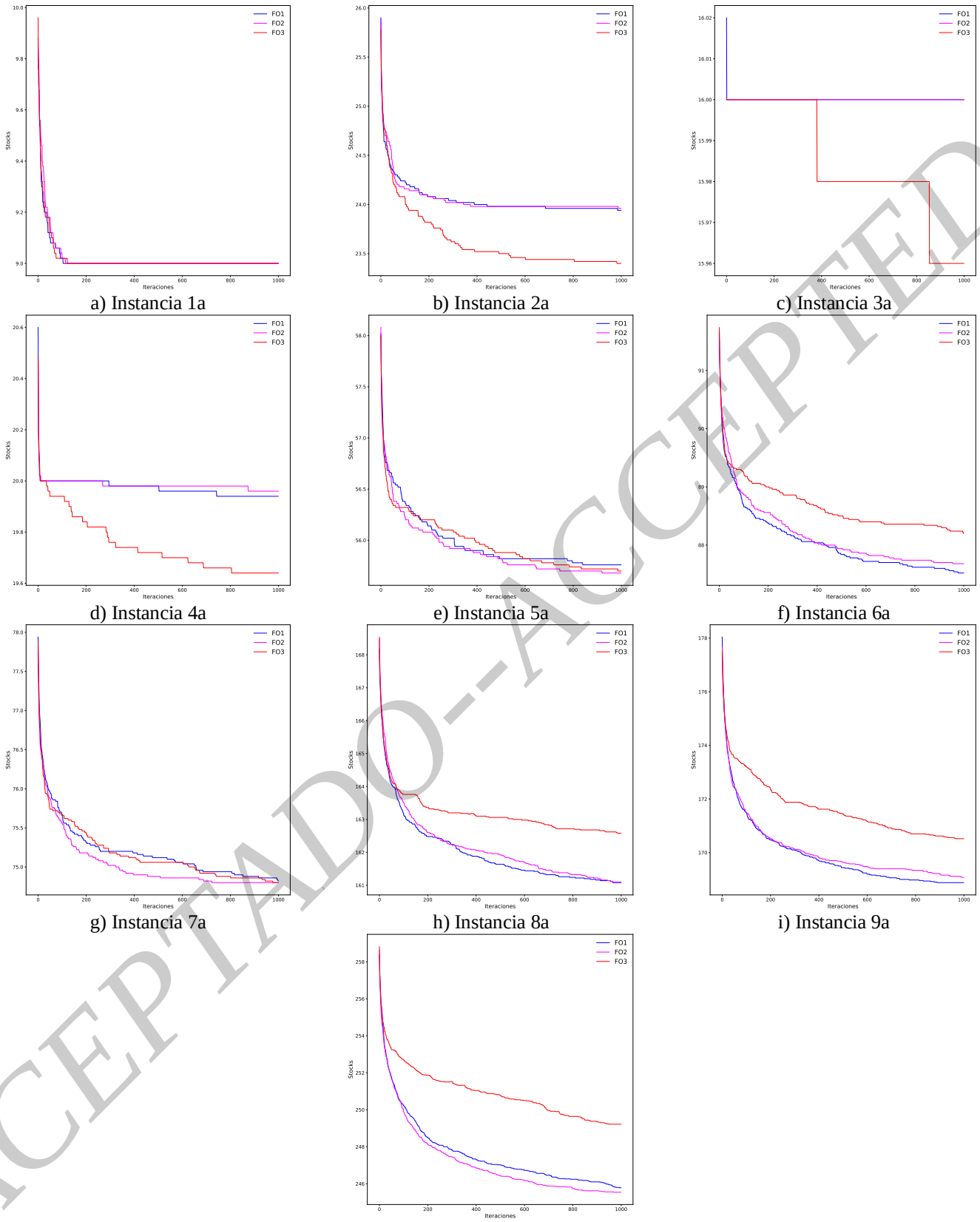
Sarper, H., & Jaksic, N. I. (2019). Simulation of the stochastic one-dimensional cutting stock problem Simulation of the stochastic one-dimensional cutting stock problem to minimize the total inventory cost. *29th International Conference on Flexible Automation and Intelligent Manufacturing* (pp. 24-28). Limerick: Elsevier.

Sa Santos, J. V., & Nepomuceno, N. (2022). Computational Performance Evaluation of Column Generation and Generate-and-Solve Techniques for the One-

Dimensional Cutting Stock Problem. *Algorithms, 15*, 394.

Vanderbeck, F. (2000). Exact Algorithm for Minimising the Number of Setups in the One-Dimensional Cutting Stock Problem. *Operations Research, 48*(6), 915-926.

Apéndice A. Gráficas de convergencia de los experimentos del escenario uno



a) Instancia 1a b) Instancia 2a c) Instancia 3a

d) Instancia 4a e) Instancia 5a f) Instancia 6a

g) Instancia 7a h) Instancia 8a i) Instancia 9a

j) Instancia 10a

Figura A1: Gráficas de convergencia de las funciones *fitness* para las instancias de los experimentos del escenario uno.

Apéndice B. Resultados de las instancias de los experimentos del escenario dos

Tabla B1: Resultados de Stock Promedio ($\bar{S}$) para las instancias *binpack1*.

Instancia	Opt.	$\frac{FO1}{\bar{S}}$	%>Opt.	$\frac{FO2}{\bar{S}}$	%>Opt.	$\frac{FO3}{\bar{S}}$	%>Opt.
u120_00	48	52.4	9.17	52.48	9.33	**49.56**	3.25
u120_01	49	53.06	8.29	53.08	8.33	**49.92**	1.88
u120_02	46	50.12	8.96	50.06	8.83	**47.38**	3.00
u120_03	49	54.04	10.29	53.88	9.96	**50.8**	3.67
u120_04	50	54.32	8.64	54.28	8.56	**51.4**	2.80
u120_05	48	52.64	9.67	52.4	9.17	**49.8**	3.75
u120_06	48	52.52	9.42	52.54	9.46	**49.52**	3.17
u120_07	49	53.72	9.63	53.9	10.00	**50.72**	3.51
u120_08	50	55.3	10.60	55.4	10.80	**52.02**	4.04
u120_09	46	50.68	10.17	50.44	9.65	**47.98**	4.30
u120_10	52	56.92	9.46	57.04	9.69	**53.8**	3.46
u120_11	49	53.6	9.39	53.54	9.27	**50.72**	3.51
u120_12	48	52.78	9.96	52.78	9.96	**50.04**	4.25
u120_13	49	53.12	8.41	53.08	8.33	**50.1**	2.24
u120_14	50	54.34	8.68	54.48	8.96	**51.36**	2.72
u120_15	48	52.66	9.71	52.6	9.58	**49.86**	3.88
u120_16	52	56.94	9.50	56.92	9.46	**53.84**	3.54
u120_17	52	56.96	9.54	57.02	9.65	**53.94**	3.73
u120_18	49	53.5	9.18	53.36	8.90	**50.38**	2.82
u120_19	49	54.16	10.53	53.96	10.12	**51.0**	4.08

El valor en negrita indica el mejor resultado obtenido entre las funciones comparadas.

Tabla B2: Resultados de Stock Promedio ($\bar{S}$) para las instancias *binpack2*.

Instancia	Opt.	$\frac{FO1}{\bar{S}}$	%>Opt.	$\frac{FO2}{\bar{S}}$	%>Opt.	$\frac{FO3}{\bar{S}}$	%>Opt.
u250_00	99	110.06	11.17	110.2	11.31	**104.94**	6.00
u250_01	100	110.98	10.98	110.9	10.90	**105.6**	5.60
u250_02	102	113.66	11.43	113.64	11.41	**108.22**	6.10
u250_03	100	110.74	10.74	110.82	10.82	**105.52**	5.52
u250_04	101	112.4	11.29	112.16	11.05	**106.92**	5.86
u250_05	101	113.0	11.88	112.9	11.78	**107.32**	6.26
u250_06	102	112.94	10.73	113.08	10.86	**107.42**	5.31
u250_07	103	115.7	12.33	115.66	12.29	**109.82**	6.62
u250_08	105	117.12	11.54	117.32	11.73	**111.7**	6.38
u250_09	101	112.46	11.35	112.0	10.89	**106.96**	5.90
u250_10	105	117.04	11.47	117.16	11.58	**111.28**	5.98
u250_11	101	112.96	11.84	112.74	11.62	**107.16**	6.10
u250_12	105	117.86	12.25	117.8	12.19	**112.06**	6.72
u250_13	102	114.1	11.86	114.26	12.02	**108.72**	6.59
u250_14	100	110.78	10.78	110.76	10.76	**105.84**	5.84
u250_15	105	117.7	12.10	117.36	11.77	**111.9**	6.57
u250_16	97	107.64	10.97	107.62	10.95	**102.82**	6.00
u250_17	100	110.94	10.94	110.68	10.68	**105.3**	5.30
u250_18	100	111.42	11.42	111.46	11.46	**106.08**	6.08
u250_19	102	113.3	11.08	113.36	11.14	**107.48**	5.37

El valor en negrita indica el mejor resultado obtenido entre las funciones comparadas.

Tabla B3: Resultados de Stock Promedio ($\bar{S}$) para las instancias *binpack3*.

Instancia	Opt.	$\frac{FO1}{\bar{S}}$	%>Opt.	$\frac{FO2}{\bar{S}}$	%>Opt.	$\frac{FO3}{\bar{S}}$	%>Opt.
u500_00	198	222.96	12.61	222.96	12.61	**214.88**	8.53
u500_01	201	227.04	12.96	226.48	12.68	**218.02**	8.47
u500_02	202	227.42	12.58	227.56	12.65	**218.94**	8.39
u500_03	204	231.28	13.37	231.4	13.43	**222.22**	8.93
u500_04	206	232.36	12.80	232.32	12.78	**223.84**	8.66
u500_05	206	232.58	12.90	232.8	13.01	**223.66**	8.57
u500_06	207	234.24	13.16	233.9	13.00	**225.5**	8.94
u500_07	204	230.68	13.08	231.08	13.27	**222.56**	9.10
u500_08	196	220.44	12.47	220.76	12.63	**212.04**	8.18
u500_09	202	227.08	12.42	227.24	12.50	**218.6**	8.22
u500_10	200	224.6	12.30	224.74	12.37	**216.62**	8.31
u500_11	200	226.14	13.07	226.2	13.10	**217.62**	8.81
u500_12	199	224.48	12.80	224.12	12.62	**215.82**	8.45
u500_13	196	220.48	12.49	220.54	12.52	**212.38**	8.36
u500_14	204	229.92	12.71	229.86	12.68	**221.34**	8.50
u500_15	201	225.44	12.16	225.72	12.30	**217.54**	8.23
u500_16	202	227.58	12.66	227.52	12.63	**218.56**	8.20
u500_17	198	223.2	12.73	222.94	12.60	**214.32**	8.24
u500_18	202	227.6	12.67	227.76	12.75	**219.18**	8.50
u500_19	196	220.94	12.72	220.6	12.55	**213.62**	8.99

El valor en negrita indica el mejor resultado obtenido entre las funciones comparadas.

Tabla B4: Resultados de Stock Promedio ($\bar{S}$) para las instancias *binpack4*.

Instancia	Opt.	$\frac{FO1}{\bar{S}}$	%>Opt.	$\frac{FO2}{\bar{S}}$	%>Opt.	$\frac{FO3}{\bar{S}}$	%>Opt.
u1000_00	399	455.5	14.16	455.74	14.22	**444.34**	11.36
u1000_01	406	464.24	14.34	464.42	14.39	**453.66**	11.74
u1000_02	411	470.92	14.58	470.16	14.39	**458.94**	11.66
u1000_03	411	470.78	14.55	470.9	14.57	**459.68**	11.84
u1000_04	397	452.94	14.09	452.66	14.02	**441.36**	11.17
u1000_05	399	456.38	14.38	456.18	14.33	**445.76**	11.72
u1000_06	395	450.36	14.02	450.4	14.03	**439.92**	11.37
u1000_07	404	461	14.11	460.82	14.06	**449.7**	11.31
u1000_08	399	456.16	14.33	456.46	14.40	**445.52**	11.66
u1000_09	397	453.98	14.35	454.06	14.37	**443.78**	11.78
u1000_10	400	456.82	14.21	456.86	14.22	**446.02**	11.51
u1000_11	401	457.42	14.07	457.58	14.11	**446.66**	11.39
u1000_12	393	448.08	14.02	447.12	13.77	**436.82**	11.15
u1000_13	396	450.66	13.80	450.64	13.80	**440.04**	11.12
u1000_14	394	449.08	13.98	448.96	13.95	**438.18**	11.21
u1000_15	402	459.98	14.42	460.24	14.49	**449.8**	11.89
u1000_16	404	462.38	14.45	462	14.36	**451.16**	11.67
u1000_17	404	463.5	14.73	463.36	14.69	**452.5**	12.00
u1000_18	399	454.84	13.99	455.78	14.23	**445.06**	11.54
u1000_19	400	456.48	14.12	457.02	14.26	**446**	11.50

El valor en negrita indica el mejor resultado obtenido entre las funciones comparadas.