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. 10 No. Especial 2 (2022) 1–8

Algoritmo del Búfalo Africano para resolver el problema de corte unidimensional
African Buffalo algorithm to solve the one dimensional cutting stock problem

Leonardo J. Montiel-Arrieta [a,*], Irving Barragán-Vite [a], Norberto Hernández-Romero [a], Manuel González-Hernández [a]

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

Resumen

El problema de corte unidimensional consiste en cortar un objeto o stock, cuya longitud puede ser finita o infinita, para producir objetos más pequeños o ítems con longitud finita. Este problema aparece en una gran cantidad de industrias alrededor del mundo. En este trabajo se propone adaptar el algoritmo de optimización del búfalo Africano con el objetivo de reducir los stocks necesarios para satisfacer la demanda de ítems. El algoritmo se probó en un conjunto de instancias que son referencia para este problema. Los resultados indican que el algoritmo consigue soluciones competitivas de acuerdo a la minimización del stock.

Palabras Clave: Corte y Empaque, Metaheurístico, Inteligencia Artificial, Minimización de Desperdicio, Algoritmo de Inteligencia de Enjambre.

Abstract

The one dimensional cutting stock problem consists in cutting an object or stock, whose length can be finite or infinite, to produce smaller objects or items with finite length. This problem appears in several industries around the world. In this work, it is proposed to implement the African buffalo optimization algorithm to reduce the stocks necessary to satisfy the demand for items. The algorithm was tested in a set of instances that are reference for this problem. The results indicate the algorithm developed here is competitive in minimizing the stock.

Keywords: Cutting and Packing, Metaheuristic, Artificial Intelligence, Waste Minimization, Swarm Intelligence Algorithm.

1. Introducción

El problema de corte unidimensional (1D-CSP, one-dimensional cutting stock problem) es una variante de los problemas de corte y empaque, que fue abordado por primera vez por Kantorovich (1960). La versión clásica del 1D-CSP de acuerdo a (Dyckhoff, 1990) se clasifica como un problema con un suministro de objetos grandes, los cuales pueden ser de diferente o de una misma longitud, incluyendo un número determinado de objetos pequeños o ítems con diferentes longitudes pero de forma regular que se obtienen de los objetos grandes cuya demanda debe satisfacerse. De igual forma, se cataloga como un problema NP-hard como se menciona en (Kantorovich, 1960) y (Gilmore and Gomory, 1961).

El 1D-CSP surge en industrias alrededor del mundo como lo es en la construcción naval (Dikili and Barlas, 2011), de la madera (Ogunranti and Oluleye, 2016), del moldeado de goma (Zanarini, 2017), en la producción de puertas de aluminio (Machado et al., 2020), entre otras más.

El principal objetivo del 1D-CSP es minimizar el desperdicio como en (Alfares and Alsawafy, 2019) y (Evtimov and Fidanova, 2018). Sin embargo, otros han buscado minimizar el número de stocks como en (Peng and Chu, 2010a) y (Peng and Chu, 2010b). Y otros más se han enfocado en reutilizar los sobrantes como en (Machado et al., 2020) y (Ravelo et al., 2020).

A lo largo de los años se han desarrollado diversos enfoques para tratar de obtener soluciones que generen un mínimo desperdicio, en algunos trabajos han empleado programación lineal (Gilmore and Gomory, 1961) y (Gilmore and Gomory, 1963), algoritmos genéticos (Chen et al., 2019) y (Hinterding and Khan, 1993), programación evolutiva (Liang et al., 2002), entre otros métodos más.

Por otro lado, el algoritmo de optimización del búfalo africano (ABO, African Buffalo Optimization) tiene pocos años que se presentó en (Odili et al., 2015a), para resolver problemáticas del tipo de optimización combinatoria como el Travelling Salesman Problem en (Odili et al., 2015a,b) y (Odili and

*Autor para correspondencia: mo450519@uaeh.edu.mx
Correo electrónico: mo450519@uaeh.edu.mx (Leonardo J. Montiel-Arrieta), irvingb@uaeh.edu.mx (Irving Barragán-Vite), nhromero@uaeh.edu.mx (Norberto Hernández-Romero), mghdez@uaeh.edu.mx (Manuel González-Hernández)

Fecha de recepción: 01/02/2022 Fecha de aceptación: 28/03/2022 Fecha de publicación: 24/06/2022
https://doi.org/10.29057/icbi.v10iEspecial2.8624

Mohmad Kahar, 2016); y el One Dimensional Bin Packing Problem en (Gherboudj, 2019). En estos trabajos el ABO demostró obtener soluciones óptimas y en un tiempo menor contra otros enfoques.

De igual forma hay que mencionar que el ABO pertenece a los algoritmos de enjambre como el Particle Swarm Optimization (PSO), el cual ha sido usado para resolver el 1D-CSP de una manera híbrida de acuerdo a (Li et al., 2007).

Con base en lo anterior, el ABO es un candidato idóneo para ser considerado como un algoritmo capaz de obtener soluciones óptimas en el 1D-CSP. Además de poder generarlas en un menor tiempo, debido a que el ABO requiere de menos parámetros que otros algoritmos como el Genetic Algorithm (GA), PSO, Ant Colony Optimization (ACO) o el Evolutionary Programming (EP).

En este trabajo se presenta la primera implementación del ABO sin una hibridación como usualmente se hace con algunas metaheurísticas para resolver el 1D-CSP con la finalidad de determinar que tan eficiente es el algoritmo. De igual forma se usó la variable de exploración del ABO para evaluar el fitness de cada búfalo debido a la similitud con el PSO donde la variable de posición es la que se evalúa.

Los resultados obtenidos muestran que el enfoque desarrollado basado en el ABO al que denominamos como African Buffalo Optimization Cutting Stock Problem (ABO-CSP) es capaz de generar soluciones con un número de stocks similar en comparación contra otros enfoques en diversas instancias de prueba.

Este trabajo está organizado de la siguiente manera. En la sección 2 se realiza la revisión de la literatura. Dentro de la sección 3 se presenta el marco teórico que contiene la definición del 1D-CSP y la descripción del ABO. En la sección 4 se presenta el algoritmo ABO-CSP. En la sección 5 se exponen los resultados obtenidos de la experimentación realizada. Finalmente en la sección 6 se muestran las conclusiones obtenidas del trabajo realizado en esta investigación.

2. Revisión de la Literatura

En (Kantorovich, 1960) se abordó el 1D-CSP mediante un algoritmo exacto aplicando programación lineal.

Gilmore y Gomory (1961 y 1963) desarrollaron una técnica de generación de columnas empleando la programación lineal para crear los patrones de corte.

En (Liang et al., 2002) utilizaron EP para minimizar el desperdicio y el uso del stock . En su propuesta establecieron que la lista de ítems se empleara como un vector, y al mismo tiempo representa un cromosoma con el que se estarían formando los arreglos de ítems de acuerdo a la longitud del objeto, mientras en el proceso de mutación proponen intercambiar de lugar los ítems para poder ir obteniendo nuevos arreglos de ítems. De igual forma, se consideró emplear un stock de una sola longitud y múltiples longitudes.

En (Levine and Ducatelle, 2004) implementaron el ACO en su forma pura y una versión híbrida. En la versión pura emplean la mejor hormiga o la mejor solución global para actualizar al resto de población de hormigas. Para el caso de la versión híbrida implementan una búsqueda local donde aquellos arreglos de ítems que menos abarquen el stock son desagrupados, con lo cual quedan libres para que puedan integrarse con alguno de los arreglos de ítems que tengan aún remanentes. De igual forma hacen uso de una menor cantidad de hormigas y emplean el método first fit decreasing para generar las soluciones iniciales.

En (Li et al., 2007) desarrollaron un enfoque híbrido empleando el General Particle Swarm Optimization y Simulated Annealing (SA) considerando un nuevo operador de actualización basado en los operadores de cruzamiento y mutación del GA.

En el trabajo (Chiong et al., 2008) emplean Evolutionary Computation (EC) donde cada cromosoma padre es generado de manera aleatoria, con la lista de ítems. Después se establecen los puntos de separación entre los ítems para agruparlos evitando que rebasen la longitud del stock en turno. Llevan a cabo una búsqueda de patrones óptimos similares en el cromosoma padre para agregarlos al cromosoma hijo.

En relación con el trabajo de (Peng and Chu, 2010a) se desarrolló el Modified Ant Colony Algortihm (MACO), donde le dan mayor importancia a los patrones de corte y las frecuencias de los mismos, evitando considerar el orden de selección de los patrones de corte. También emplean un procedimiento basado en un algoritmo de árbol de búsqueda para obtener los patrones de corte dominantes.

En (Peng and Chu, 2010b) presentan el Hybrid-multicromosome Genetic Algorithm, donde emplean dos cromosomas en su solución, un cromosoma está relacionado al patrón de corte y el otro se encuentra enlazado con la frecuencia del patrón de corte.

En (Jahromi et al., 2012) implementan el SA y Tabu Search (TS), buscando realizar una comparación de eficiencia bajo los parámetros de desperdicio de corte y el tiempo de CPU al resolver el 1D-CSP.

En (Cui et al., 2015) implementan un procedimiento de agrupación secuencial para minimizar la suma total del material y los costos de maquinación. El procedimiento propuesto consta de dos fases, en la primera se realiza una agrupación secuencial para generar el conjunto de patrones de corte y en la segunda fase se emplea el optimizador CPLEX para resolver un modelo de programación lineal entera.

En (Garraffa et al., 2016) se enfocaron en minimizar los patrones usados y maximizar los sobrantes mediante un enfoque heurístico basado en patrones.

Dentro de (Ogunranti and Oluleye, 2016) se enfocaron en minimizar el desperdicio generado derivado del total de veces que se ejecutan los patrones de corte. Utilizaron programación lineal para seleccionar los patrones de corte óptimos.

En (Evtimov and Fidanova, 2018) implementan el ACO donde, en cada iteración, cada hormiga selecciona de manera aleatoria un stock y un ítem. Después aplica una regla de probabilidad de transición para efectuar cortes, es decir, se van agregando más ítems al arreglo inicial mientras que no se rebase la longitud del stock. Logrando así obtener un patrón de corte.

En (Santos et al., 2018) se minimizan los diferentes tipos de desperdicios que se generan dentro de la industria del acero. Proponen un modelo basado en mixed integer lineal programming, tratando de satisfacer en primera instancia las ordenes más complejas.

Dentro de (Sarper and Jaksic, 2018) emplean Goal Pro-

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

En (Alfares and Alsawafy, 2019) implementan un modelo heurístico de dos etapas denominado como least-loss algorithm. Dentro de la primera etapa realizan un ordenamiento descendente de los ítems de acuerdo a su longitud. Una vez encontrado el ítem más grande se buscan todas las posibles combinaciones que éste pueda tener con otros ítems, buscando aquellas combinaciones que tengan un desperdicio de 0. Si después de realizar lo antes mencionado con los demás ítems no se logra satisfacer la demanda de ítems, entonces se procede a considerar un desperdicio de 1 en las combinaciones. En caso de que no se produzca una solución satisfactoria, se conducirá a la segunda etapa donde se aplicará un ordenamiento decremental diferente y se efectuará una búsqueda de vecindario.

En el trabajo (Pitombeira-Neto and de Athayde Prata, 2019) buscan resolver el 1D-CSP que considera ordenes de ítems heterogéneas con un algoritmo metaheurístico basado en la estrategia fix-and-optimize hibrizada con una búsqueda local aleatoria. Ponen su atención en minimizar el número de stocks y tiempo para satisfacer las diversas órdenes de los clientes.

En (Ravelo et al., 2020) se enfocan en la reutilización de remanentes, donde se busca determinar cuándo un residuo es aceptado para obtener más ítems. Desarrollaron dos enfoques metaheurísticos, el primero esta basado en Greedy Randomized Adaptative Search Procedure (GRASP), mientras que el segundo fue mediante un enfoque evolutivo.

En (Machado et al., 2020) emplean un modelo basado en programación lineal entera para minimizar los sobrantes de las barras de aluminio en la producción de puertas. En su enfoque implementan tres funciones objetivo, en la primera consideran la minimización de la pérdida de corte, la segunda es reducir las barras necesarias y la tercera es la combinación de las dos primeras.

De los trabajos antes mencionados puede notarse que los métodos metaheurísticos para resolver el 1D-CSP, como el que se propone en este trabajo, tienden a ser métodos híbridos y requieren procedimientos elaborados para representar las soluciones del problema; además de que se precisa del uso de un mayor número de parámetros y de pasos para encontrar las mejores soluciones. En nuestra propuesta son necesarios menos pasos y parámetros lo que permite una fácil implementación y representación de las soluciones, además de requerir menos pruebas para determinar la mejor combinación de valores de los parámetros.

3. Marco Teórico

A continuación, se presentan los conceptos básicos del 1D-CSP y del ABO.

3.1. Definición del 1D-CSP

El 1D-CSP de acuerdo a (Dyckhoff, 1990) en su versión clásica consiste en un suministro de objetos largos o también llamado stock de forma regular con una misma longitud o varias longitudes, del cual se van a obtener los objetos pequeños o ítems de forma igualmente regular pero con longitudes diferentes.

Gilmore y Gomory propusieron el siguiente modelo relacionado al 1D-CSP, en la cual se aborda minimizar la frecuencia de cada patrón que ha de ser cortado (Scheithauer, 2017):

$$z^* = z^{CSP} = min \sum_{j \in J} x_j \tag{1}$$

Sujeto a

$$\sum_{j \in J} a_{ij} x_j \geq b_i, i \in I, \tag{2}$$

$$x_j \in Z_+, j \in J \tag{3}$$

Donde:

- z^*= es el total de patrones de corte llevados a cabo.
- i= es un ítem.
- j= es un patrón de corte.
- I= es el conjunto de ítems.
- J= es el conjunto de patrones.
- x_j= es la frecuencia de un patrón de corte.
- a_{ij}= es un patrón de corte fáctible.
- b_i= es la demanda de cada ítem.

En la Ecuación (1) cuenta el total de número de patrones de corte necesarios para cubrir la demanda de ítems. En la Ecuación (2) se asegura que se satisfaga la demanda de cada ítem $i \in I$, en (3) se establece que el número de patrones x_j es un valor no negativo. En este modelo la sobreproducción es permitida (Scheithauer, 2017).

3.2. African Buffalo Optmization

El ABO es un algoritmo propuesto por Julius Beneoluchi Odili, Mohd Nizam Mohmad Kahar y Shahid Anwar en (Odili et al., 2015a), el cual se basa en el comportamiento de los búfalos africanos para la búsqueda de pastizales.

Este algoritmo metaheurístico pertenece a la clase de algoritmos denominada como "Swarm Intelligence", los cuales se basan en el comportamiento social de algunos animales, tratando de adquirir una mejor explotación y exploración del espacio de búsqueda disponible en las problemáticas a las que se someta.

Las tres principales características del comportamiento de los búfalos africanos son (Odili et al., 2015a):

1. Poseen una increíble capacidad de memoria, debido a que pueden rastrear sus rutas a través de cientos de kilómetros.
2. Mantienen una habilidad cooperativa todo el tiempo, incluso ante situaciones que pongan en riesgo su vida o la de un miembro de la manada. Emplean los siguientes dos sonidos para comunicarse:
   a) El sonido de "waaa" se usa para indicar a la manada que la ubicación actual no cuenta con adecuados pastizales o es peligrosa, por lo que es necesario cambiar de lugar.

b) El sonido de "maaa" se aplica para establecer que el emplazamiento es bueno porque contiene pastizales suficientes y es lo bastante seguro para quedarse.

3. El último atributo esta relacionado a su naturaleza democrática, la cual emplean para decidir el siguiente movimiento cuando no exista una totalidad en el consenso, por lo que se decidirá con base en lo que la mayoría de la manada decida.

El ABO consta de las siguientes fases (Odili et al., 2015a):

1. Se establece la función objetivo.
2. Una población de búfalos es generada de manera aleatoria.
3. Se realiza la actualización del fitness de cada búfalo k con la Ecuación 4.

$$m_{k+1} = m_k + lp1(bgmax - w_k) + lp2(bpmax_k - w_k) \quad (4)$$

Donde w_k y m_k representan los movimientos de exploración y explotación respectivamente del búfalo k ésimo ($k = 1, 2, ..., N$). Mientras que $lp1$ y $lp2$ son los factores de aprendizaje. En relación al $bgmax$, éste es el mejor búfalo de toda la manada. El último elemento es $bpmax_k$, el cual representa la mejor ubicación de cada búfalo.

4. Posteriormente se actualiza la ubicación de cada búfalo usando la Ecuación 5.

$$w_{k+1} = \frac{w_k + m_k}{\lambda} \quad (5)$$

De acuerdo a (Odili et al., 2017) λ puede tomar valores entre 0.1 y 2. Entre menor sea el valor asignado a λ se realizará una mayor exploración y en caso contrario se hará una mayor explotación.

5. Una vez que se actualizó el fitness y ubicación de cada búfalo se verifica si el $bgmax$ se ha actualizado, si es así se procede a ir paso 6. En caso contrario hay que volver al paso 2 para reiniciar la manada.
6. Se comprueba si se alcanzó el criterio de parada, en caso de lograrlo hay que proceder al paso 7. Pero de no ser así, se tiene que volver al paso 3.
7. Finalmente se envía la mejor solución, es decir, el $bgmax$.

Como se puede apreciar el ABO tiende a evitar el estancamiento al estar actualizando periódicamente el mejor búfalo de toda la manada. Además, el ABO requiere pocos parámetros para su implementación (Odili and Mohmad Kahar, 2016).

Aunado a lo mencionado, el ABO lleva a cabo una exploración más adecuada al considerar la mejor ubicación del mejor búfalo ($bgmax$) y la mejor de cada búfalo ($bpmax_k$).

En el aspecto de la explotación se realiza considerando el aprovechamiento que tenga el mejor búfalo de la manada junto con el mejor histórico de cada uno de los búfalos a lo largo de la ejecución del algoritmo.

4. Método

El ABO-CSP busca obtener aquel arreglo de ítems o patrón de corte que genere el menor número de stocks, de tal manera que se satisfaga la demanda de ítems de acuerdo a la Ecuación 1. Sin embargo, no se considera la sobreproducción establecida en la Ecuación 2.

La Figura 1 representa una instancia de ejemplo, la cual esta conformada por tres tipos de ítems con diferentes longitudes y una demanda de 2 unidades en todos los casos. De igual forma se establece una cantidad ilimitada de stocks disponible, cada uno con una longitud de 50.

Ahora bien, en el ABO-CSP cada búfalo es una solución, la cual contendrá todos los ítems de la instancia en cuestión. Cada búfalo tendrá los ítems de la instancia de prueba dispuestos de una manera diferente como se aprecia en la Figura 2.

Para determinar el acomodo de los ítems en cada stock que se obtienen del arreglo de ítems en cada búfalo se suma de izquierda a derecha la longitud de cada ítem dentro del stock mientras no se rebase la longitud del stock. En caso contrario, si con el nuevo ítem se rebasa la longitud del stock actual, entonces el nuevo ítem será el primero del nuevo stock tal como se aprecia en la Figura 3.

En la Figura 3 se observa que en cada búfalo con base en su arreglo de ítems se conforman dos patrones de corte relacionados a un stock cada uno. Así también, se visualiza que la suma del desperdicio de los dos patrones de corte da un total de diez. Sin embargo, en los búfalo 2 y 3 solo el Stock 1 se aprovecha de manera completa.

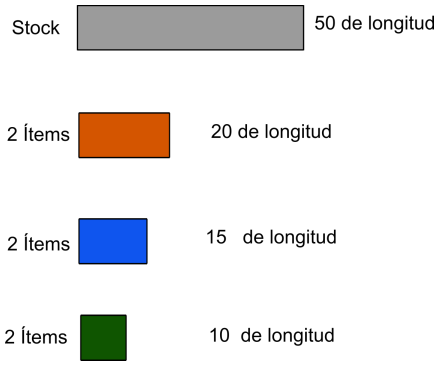

Figura 1: Ejemplo de instancia.

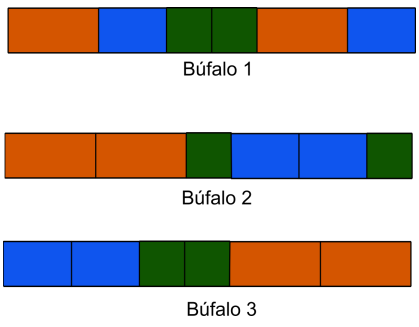

Figura 2: Ejemplos de formación de soluciones o patrones de corte (búfalos).

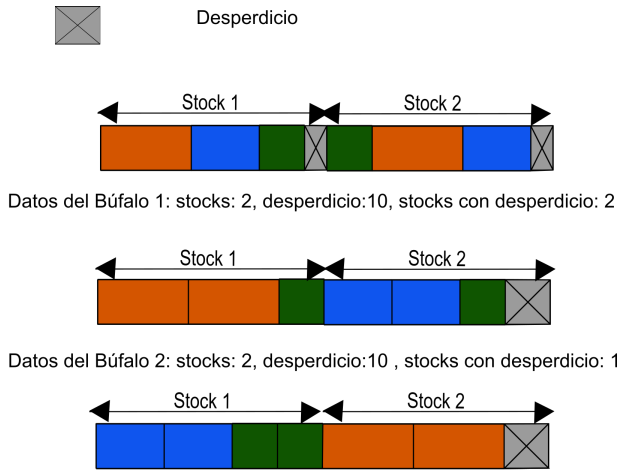

Figura 3: Agrupamiento de los ítems dentro de los búfalos.

Debido a que el ABO produce soluciones continuas se implementó el método Ranked Order Value (ROV) para realizar la discretización de las soluciones como se propuso en (Liu et al., 2008).

Los pasos del algoritmo ABO-CSP propuesto en este trabajo se describen en el Algoritmo 1, donde los datos de entrada son el set_de_items, siendo la instancia de prueba; la longitud_stock; el num_bufalos, es la población de soluciones; λ, es la variable relacionada con la exploración dentro del ABO; iteraciones, es el número máximo de veces que el algoritmo intentará encontrar la mejor solución en cada réplica o ejecución; y finalmente lp1 y lp2 son los factores de aprendizaje propios del ABO.

Dentro del ABO-CSP se utiliza la variable de exploración (wb) del ABO para evaluar el fitness de cada búfalo en términos del stock.

Así también, se definió la variable bgmaxparcial con el propósito de servir como punto de comparación entre cada búfalo y el bgmax como se observa en la línea 15 del Algoritmo 1.

Por otro lado, la variable cambiobgmax se emplea para establecer cuando se ha actualizado el bgmax como se aprecia en la línea 26 del Algoritmo 1.

El primer proceso es crear los búfalos de manera aleatoria, por lo que se usa la función CrearBufalosAleatoriamente. Después se hace la búsqueda del bgmax mediante la función Busqueda_bgmax.

Posteriormente se lleva a cabo la actualización de mb, es decir, el fitness de cada búfalo con la función CalcularNuevoMb, la cual se aprecia con más detalle en el Algoritmo 2, donde se emplea la Ecuación 4.

Después se deberá actualizar el wb, es decir, la posición de cada búfalo con la función CalcularNuevoWb, ésta se observa con mayor detalle en el Algoritmo 3. En esta función se hace uso de la Ecuación 5 y del ROV.

El siguiente paso es contar el número de stocks que se obtienen del wb_nuevo mediante la función ContarStock que se encuentra en la línea 9 del Algoritmo 1. Luego se procede a comparar el stock_wb_nuevo contra el bpmax del búfalo términos del número de stocks, y en caso de que sea menor el stock_wb_nuevo ahora se comparará contra el bgmaxparcial.

```
   Entrada: set_de_items, longitud_stock, num_bufalos, λ,
            iteraciones, lp1,lp2
   Salida: bgmax
1  bufalos = CrearBufalosAleatoriamente(set_de_items,
   longitud_stock, num_bufalos);
2  bgmax = Busqueda_bgmax(bufalos, longitud_stock);
3  bgmaxparcial = bgmax;
4  cambiobgmax = Falso;
5  mientras i ≤ iteraciones hacer
6      per j ← 1 Hasta ≤ num_bufalos fai
7          mb_nuevo = CalcularNuevoMb(bufalos[j][mb],
           bufalos[j][wb], bufalos[j][bpmax], bgmax, lp1, lp2);
8          wb_nuevo = CalcularNuevoWb(bufalos[j][mb],
           bufalos[j][wb], set_de_items, λ);
9          stock_wb_nuevo = ContarStock(wb_nuevo,
           longitud_stock);
10         bufalos[j][mb] = mb_nuevo;
11         bufalos[j][wb] = wb_nuevo;
12         si stock_wb_nuevo < bufalos[j][stock] entonces
13             bufalos[j][bpmax] = wb_nuevo;
14             bufalos[j][stock] = stock_wb_nuevo;
15             si stock_wb_nuevo < bgmaxparcial[stock]
               entonces
16                 bgmaxparcial[wb] = bufalos[j][wb];
                   bgmaxparcial[stock] = bufalos[j][stock]
17             fin
18         fin
19     fine
20     si bgmaxparcial[stock] < bgmax[stock] entonces
21         bgmax = bgmaxparcial;
22         i = i + 1;
23         si i > 10 Y i % 10 == 1 entonces
24             cambiobgmax = Falso;
25         en otro caso
26             cambiobgmax = Verdadero;
27         fin
28     en otro caso
29         si (cambiobgmax == Falso) Y (i % 10 == 0 ) entonces
30             bufalos =
               CrearBufalosAleatoriamente(set_de_items,
               longitud_stock, num_bufalos);
31             bgmax = Busqueda_bgmax(bufalos,
               longitud_stock);
32             bgmaxparcial = bgmax;
33             cambiobgmax = Falso;
34             i = i + 1;
35         en otro caso
36             i = i + 1;
37         fin
38     fin
39 fin
```

Algoritmo 1: ABO-CSP.

Al terminar de actualizar todos los búfalos se realiza la comparación entre el bgmaxparcial y el bgmax con el propósito de verificar si se actualizó el bgmax, esto se aprecia en la línea 20 en el Algoritmo 1.

En el ABO-CSP se propone no reiniciar la población de búfalos al término de cada iteración en caso de que no se actualice el bgmax, debido a que se buscó dar mayor apertura a la actualización del bgmax, haciendo el reinicio de la manada cada diez iteraciones, para lo cual se uso la variable cambiobgmax

para llevar el control del reinicio de la manada.

De la misma forma, se estableció una condición que permitiera saber cuándo se iniciaba un nuevo ciclo de diez iteraciones, la cual se puede apreciar en la línea 23 en el Algoritmo 1. Así también, se colocó la condición necesaria para llevar a cabo el reinicio de la manada, ésta se puede apreciar en la línea 29 en el Algoritmo 1.

```
   Entrada: mb_actual, wb_actual, bpmax, bgmax, lp1, lp2
   Salida: mb_nuevo
1  per i ← 0 hasta < tamaño_mb_actual fai
2  |  mb_nuevo[i] = mb_actual[i] + lp1(bgmax[i] - wb_actual[i])
   |  + lp2(bpmax[i] - wb_actual[i]);
3  fine
```

Algoritmo 2: Función CalcularNuevoMb.

```
   Entrada: mb_actual, wb_actual, set_de_items, λ
   Salida: wb_nuevo
1  per i ← 0 hasta < tamaño_wb_actual fai
2  |  wb_nuevo[i] = (mb_actual[i] + wb_actual[i]) / λ;
3  fine
4  wb_nuevo_ordenado = OrdenarAscendente(wb_nuevo);
5  set_de_items = OrdenarAscendente(set_de_items);
6  per i ← 0 hasta < tamaño_wb_nuevo_ordenado fai
7  |  index = BuscarPosicion(wb_nuevo_ordenado[i],
   |  wb_nuevo);
8  |  wb_nuevo [index] = set_de_items[i];
9  fine
```

Algoritmo 3: Función CalcularNuevoWb.

5. Resultados

A pesar de que el problema de corte unidimensional ha sido tratado por varios autores desde diferentes perspectivas y aplicaciones, no existe un banco de pruebas estandarizado para poder probar nuevos métodos de solución.

Para probar la eficiencia del ABO-CSP se emplearon 10 instancias, donde las primeras cinco numeradas de 1a a 5a se consiguieron de (Hinterding and Khan, 1993) mientras que las últimas cinco, es decir, de 6a a 10a se obtuvieron de (Liang et al., 2002). En la Tabla 1 se aprecian el número de ítems y la longitud del stock de cada una de las instancias. El ABO-CSP se ejecutó 50 veces en cada instancia al igual que en (Liang et al., 2002) y (Levine and Ducatelle, 2004) siendo el número de veces que emplearon para realizar sus pruebas.

Tabla 1: Descripción de las Instancias

Instancia	Cantidad de Ítems	Longitud del Stock
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

El número de búfalos fue de 40 de acuerdo a (Odili et al., 2015a), mientras que los valores para los factores de aprendizaje $lp1$ y $lp2$ fueron de 0.3 y 0.6 respectivamente, de acuerdo a (Gherboudj, 2019). Se usó λ=1 y 40 iteraciones debido a que se realizaron pruebas con diferentes valores y se encontró que con estos valores se obtuvieron los mejores resultados.

El ABO-CSP se compara con el método LLA de (Alfares and Alsawafy, 2019), el SMBEP de (Chiong et al., 2008), el HACO y Pure ACO de (Levine and Ducatelle, 2004), el EP de (Liang et al., 2002), el MACO de (Peng and Chu, 2010a) y el HMCGA de (Peng and Chu, 2010b) ya que en ellos se emplearon las instancias de prueba utilizadas en este trabajo.

Se utilizó un equipo de cómputo con las siguientes características: procesador Intel ® Core™ i7-6700HQ CPU 2.60GHz 2.60 GHz y 8GB de RAM. Se llevó a cabo la programación del ABO-CSP en el lenguaje Python versión 3.7.

En la columna dos de la Tabla 2 se muestra el Promedio de Stocks obtenido de las 50 ejecuciones con el ABO-CSP en cada una de las instancias. Se puede apreciar cómo en las instancias 1a a 4a el ABO-CSP emplea casi el mismo número de stocks para poder satisfacer la demanda de ítems en comparación contra otros enfoques, debido a que se requiere satisfacer un número de ítems menor en comparación contra el resto de las instancias. En la columna denominada " % ABO-CSP (Promedio) > Mejor solución"se representa cuanto mayor, en porcentaje, es el promedio de stocks del ABO-CSP comparada contra la mejor solución (negrita) en cada instancia. Se observa que el porcentaje del stock usado es aproximadamente 10 % mayor a la mejor solución en las instancias 1a y 2a. Mientras que en la instancia 3a es un 6.6 % mayor con respecto a la mejor solución. En cuanto a la instancia 4a el ABO-CSP emplea un 5.47 % más que el resto de los enfoques. Con base en lo anterior el ABO-CSP tiende a obtener soluciones con un porcentaje de stock no mayor a 10 % en instancias con una cantidad de ítems entre 20 y 126.

Sin embargo, a partir de la instancia 5a en adelante, donde el número de ítems está entre 126 y 600, el promedio de stocks del ABO-CSP se aleja considerablemente de los otros trabajos. Con base en lo anterior se ve aumentado la diferencia de porcentaje promedio de stocks con respecto a la mejor solución en las instancias 5a a 10a.

Debido a que el ABO es un algoritmo diseñado para emitir soluciones continuas, fue necesario implementar el método ROV en el ABO-CSP para evitar que las soluciones se salieran de los valores límites, es decir, las longitudes de los ítems. Sin embargo, las soluciones que se generaron no fueron competentes para instancias con un número mayor o igual a 126 ítems. Por lo cual se podría optar por implementar un método de búsqueda local como el First Fit para generar al búfalo guía (bgmax) después de cierto número de iteraciones tomando en cuenta el avance obtenido por el resto de los búfalos.

Otro aspecto a considerar en el ABO-CSP es el hecho que al reiniciar la manada se pierde todo el avance logrado en la búsqueda de una mejor solución, por lo que consideramos en un trabajo futuro guardar algunos de los mejores búfalos para que estos sean un referente hacia la nueva manada.

Los resultados muestran que el ABO sin ninguna hibridación y junto con el hecho de ser un algoritmo diseñado para emitir soluciones continuas es capaz de generar soluciones competentes para la problemática del 1D-CSP en instancias con pocos ítems. Asimismo, se muestra en este trabajo que el ABO

Tabla 2: Promedio stock usado (los mejores valores están en negrita)

Instancia	Promedio Stocks ABO-CSP	% ABO-CSP (Promedio) > Mejor solución	LLA	SMBEP	MACO	PureACO	HACO	HMCGA	EP
1a	9.84	9.33	**9**	**9**	**9**	-	-	**9**	**9**
2a	25.2	9.56	**23**	**23**	**23**	-	-	**23**	**23**
3a	16	6.6	**15**	**15**	**15**	-	-	**15**	**15**
4a	20.04	5.47	**19**	**19**	19	-	-	**19**	**19**
5a	57.32	8.15	**53**	**53**	**53**	-	-	**53**	53.12
6a	91.48	15.79	81	80.6	79.1	**79**	**79**	79.1	81.4
7a	77.06	14.5	68	68	**67.3**	69	68	**67.3**	68.88
8a	167.04	16.81	145	144.8	144.8	146	**143**	144.8	148.8
9a	175.66	17.89	152	150.9	149.4	151	**149**	149.4	154.9
10a	256.26	19.19	216	216.5	219.8	218.9	**215**	219.8	223.56

es un algoritmo adecuado para resolver el 1D-CSP, aunque se requieren hacer más pruebas con diferentes combinaciones de valores para lambda y los factores de aprendizaje. Sin embargo, para lograr mejores resultados en las instancias con un mayor número de ítems se considera necesario agregar un método de búsqueda local o un algoritmo metaheurístico en el ABO-CSP para poder obtener mejores resultados, tomando en cuenta que el ABO requiere de pocos parámetros.

6. Conclusiones

En este trabajo se aplicó el ABO para resolver la problemática del 1D-CSP buscando minimizar el número de stocks necesarios para satisfacer las diferentes instancias de ítems.

El ABO-CSP presentó en diversas instancias de prueba resultados similares o con una diferencia mínima en comparación contra otros enfoques heurísticos y metaheurísticos bajo el promedio del stock usado. Obteniendo soluciones con un número de stocks cercano a la mejor solución en las instancias con un menor número de ítems. Sin embargo, en instancias con una demanda de ítems mayor el ABO-CSP tiende a generar soluciones con un porcentaje mayor que la mejor solución en cada instancia.

Para un trabajo a futuro, se propone implementar un método de búsqueda local que permita explorar aún más el espacio de soluciones. Inclusive diseñar un método híbrido con un algoritmo metaheurístico como el GA, donde se puede conjuntar el proceso de mutación para obtener una mayor diversidad de soluciones o la aplicación del elitismo para seleccionar los mejores búfalos e incorporarlos en la siguiente manada. Con el propósito de generar soluciones con un menor número de stocks en instancias con un mayor número de ítems.

Referencias

Alfares, H. K. and 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.

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

Chiong, R., Chang, Y. Y., Chai, P. C., and Wong, A. L. (2008). A selective mutation based evolutionary programming for solving cutting stock problem without contiguity. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1671–1677.

Cui, Y., Zhong, C., and Yao, Y. (2015). Pattern-set generation algorithm for the one-dimensional cutting stock problem with setup cost. *European Journal of Operational Research*, 243(2):540–546.

Dikili, A. C. and Barlas, B. (2011). A generalized approach to the solution of one-dimensional stock-cutting problem for small shipyards. *Journal of marine science and technology*, 19(4):368–376.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159. Cutting and Packing.

Evtimov, G. and Fidanova, S. (2018). *Ant Colony Optimization Algorithm for 1D Cutting Stock Problem*, pages 25–31. Springer International Publishing, Cham.

Garraffa, M., Salassa, F., Vancroonenburg, W., Vanden Berghe, G., and Wauters, T. (2016). The one-dimensional cutting stock problem with sequence-dependent cut losses. *International Transactions in Operational Research*, 23(1-2):5–24.

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

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

Hinterding, R. and Khan, L. (1993). Genetic algorithms for cutting stock problems: with and without contiguity. In *Progress in evolutionary computation*, pages 166–186. Springer.

Jahromi, M. H., Tavakkoli-Moghaddam, R., Makui, A., and Shamsi, A. (2012). Solving an one-dimensional cutting stock problem by simulated annealing and tabu search. *Journal of Industrial Engineering International*, 8(1):1–8.

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

Levine, J. and 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., and Dai, Z. (2007). General particle swarm optimization based on simulated annealing for multi-specification one-dimensional cutting stock problem. In Wang, Y., Cheung, Y.-m., and Liu, H., editors, *Computational Intelligence and Security*, pages 67–76, Berlin, Heidelberg. Springer Berlin Heidelberg.

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

Liu, B., Wang, L., Qian, B., and Jin, Y. (2008). Hybrid particle swarm optimization for stochastic flow shop scheduling with no-wait constraint. *IFAC Proceedings Volumes*, 41(2):15855–15860. 17th IFAC World Congress.

Machado, A. A., Zayatz, J. C., da Silva, M. M., Melluzzi Neto, G., Leal, G. C. L., and Palma Lima, R. H. (2020). Aluminum bar cutting optimization for door and window manufacturing. *DYNA*, 87(212):155–162.

Odili, J. B., Kahar, M. N. M., and Anwar, S. (2015a). African buffalo optimization: a swarm-intelligence technique. *Procedia Computer Science*, 76:443–448.

Odili, J. B., Kahar, M. N. M., Anwar, S., and Azrag, M. A. K. (2015b). A comparative study of african buffalo optimization and randomized insertion

algorithm for asymmetric travelling salesman's problem. In *2015 4th International Conference on Software Engineering and Computer Systems (ICSECS)*, pages 90–95.

Odili, J. B. and Mohmad Kahar, M. N. (2016). Solving the traveling salesman's problem using the african buffalo optimization. *Computational intelligence and neuroscience*, 2016.

Odili, J. B., Mohmad Kahar, M. N., and Noraziah, A. (2017). Parameters-tuning of pid controller for automatic voltage regulators using the african buffalo optimization. *PLOS ONE*, 12(4):1–17.

Ogunranti, G. A. and 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.

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

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

Pitombeira-Neto, A. R. and de Athayde Prata, B. (2019). A matheuristic algorithm for the one-dimensional cutting stock and scheduling problem with heterogeneous orders. *Top*, pages 1–15.

Ravelo, S. V., Meneses, C. N., and Santos, M. O. (2020). Meta-heuristics for the one-dimensional cutting stock problem with usable leftover. *Journal of Heuristics*, 26(4):585–618.

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

Sarper, H. and Jaksic, N. I. (2018). Evaluation of procurement scenarios in one-dimensional cutting stock problem with a random demand mix. *Procedia Manufacturing*, 17:827–834.

Scheithauer, G. (2017). *Introduction to cutting and packing optimization: Problems, modeling approaches, solution methods*, volume 263. Springer.

Zanarini, A. (2017). Optimal stock sizing in a cutting stock problem with stochastic demands. In Salvagnin, D. and Lombardi, M., editors, *Integration of AI and OR Techniques in Constraint Programming*, pages 293–301, Cham. Springer International Publishing.