MOCHILA BIDIMENSIONAL RESTRICTA GUILLOTINADA CON Y SIN ROTACIÓN DE PIEZAS USANDO VECINDARIO VARIABLE Y CÚMULO DE PARTÍCULAS

Two-dimensional restricted guillotine knapsack problem with and without rotatable items using variable neighborhood search and particle swarm optimization

RESUMEN

En este documento se presenta un algoritmo que combina conceptos tomados de las técnicas de optimización cúmulo de partículas y búsqueda en vecindario variable para resolver el problema de la mochila bidimensional, donde se tiene por objetivo ubicar las piezas que generen mayor rentabilidad, a las cuales se les asocia un beneficio, además se deben respetar las restricciones de un empaquetamiento sin traslapes y con patrones de corte tipo guillotina. Esta propuesta fue confrontada con casos de prueba de la literatura especializada obteniéndose resultados de buena calidad.

PALABRAS CLAVES: cúmulo de partículas, mochila bidimensional, optimización combinatoria.

ABSTRACT

This document presents an algorithm that combines concepts taken from the optimization techniques with particle swarm optimization and variable neighborhood search to solve the two-dimensional knapsack problem. The aim of this problem is to locate the items that generate the most profitability, in which every item has a benefit different to his area and the patterns of cutting of the items have to be guillotined. To evaluate the effectiveness of the algorithm it was used case studies of the specialized literature. Finally, it was obtained excellent results.

KEYWORDS: *combinatorial optimization, knapsack problem, particle swarm optimization.*

DAVID ÁLVAREZ MARTÍNEZ
Ingeniero de Sistemas y Computación, *M.Sc.*
Joven Investigador
COLCIENCIAS-Universidad Tecnológica de Pereira
davidalv@utp.edu.co

ELIANA TORO OCAMPO
Ingeniera Industrial, *M.Sc.*
Docente Asistente
Facultad de Ingeniería Industrial
Universidad Tecnológica de Pereira
elianam@utp.edu.co

RAMÓN GALLEGO RENDÓN
Ingeniero Electricista, *Ph.D.*
Docente Titular
Programa Ingeniería Eléctrica
Universidad Tecnológica de Pereira
ragr@utp.edu.co

1. INTRODUCCIÓN

El problema de la mochila bidimensional, denominado en inglés como *two-dimensional knapsack problem* (2KP), consiste en un conjunto de ítems de pequeños rectángulos cada uno con un ancho w_i, un alto h_i y un costo de beneficio c_i asociado. Los ítems deben ubicarse en un rectángulo de mayor tamaño denominado mochila (*knapsack*) con un alto H y un ancho W. El objetivo es empacar el conjunto de rectángulos que maximice el beneficio de la mochila y cumpliendo las restricciones de los problemas de empaquetamiento.

El problema 2KP es una generalización del famoso problema binario unidimensional de la mochila (*knapsack problem*) considerado como NP-Duro debido a que el espacio de soluciones crece de forma exponencial Gallego et al. [1]. Para el problema de 2KP el espacio de soluciones es mucho más grande debido a que se pueden ubicar varias piezas del mismo tipo y por tanto también es considerado como un problema NP-Duro [2].

Este problema ha sido ampliamente estudiado por la comunidad académica por autores como Gilmore y Gomory [3], Dyckhoff et al [4]. En estos se presentan variaciones del problema así como su posible abordaje a través de técnicas metaheurísticas. Beasley [5], Hadjiconstantinou y Christofides [6], Boschetti et al [7] formulan el problema como uno de programación lineal entera, en [5] y [6] se resuelve el problema por una técnica exacta. En este trabajo se propone un algoritmo que combina la técnica cúmulo de partículas (PSO) del inglés *particle swarm optimization* y la técnica de vecindario variable del inglés (VNS) *variable neigborhood search*.

Este documento tiene la siguiente presentación:
En la siguiente sección se presenta la descripción del problema, en la sección 3 el modelo matemático del problema, en la sección 4 se habla de la técnica de solución, en la sección 5 se muestra la implementación de la técnica al problema propuesto, en la sección 6 se realiza el análisis de resultados y finalmente en la sección 7 se presentan las conclusiones y recomendaciones.

Fecha de Recepción: Enero 26 de 2010
Fecha de Aceptación: Marzo 25 de 2010

2. DESCRIPCIÓN DEL PROBLEMA

En [4], Dyckhoff propone una tipología para los problemas de corte y empaquetamiento basada en las características básicas comunes de todo problema de corte y empaquetamiento, para comprender su tipología Dyckhoff explica de forma sencilla en que consiste un problema de corte o empaquetamiento de la siguiente manera: existen dos elementos principales, un conjunto de ítems grandes y una lista de ítems pequeños, los ítems grandes también son llamados objetos (mochila), siendo el objetivo ubicar los ítems en el conjunto de objetos. Las características básicas de los problemas de corte y empaquetamiento son enumeradas a continuación:

a. Dimensionalidad (la característica más importante, define el número mínimo de dimensiones necesarias para describir la geometría de la disposición de los ítems en los objetos).
(1) Unidimensional, (2) Bidimensional, (3) Tridimensional, (N) Multidimensional con N>3

b. Clase de asignación (disponibilidad tanto de los ítems como de los objetos).
(B) Todos los objetos y una selección de ítems, (V) Una selección de objetos y todos los ítems

c. Surtido de los objetos (número de diferentes formas de los objetos).
(O) Un objeto, (I) Figuras idénticas, (D) Diferentes figuras

d. Surtido de los ítems (número de diferentes formas de los ítems).
(F) Pocos ítems (de diferentes figuras), (M) Muchos ítems de gran variedad de figuras, (R) Muchos ítems de relativamente pocas diferentes figuras (no-congruentes), (C) Figuras congruentes.

Además de estas características existen otras no tan generales que dependen particularmente del problema, Lodi et al. en [8] proponen las siguientes:

e. Restricciones inherentes al patrón de corte (tipos de corte, separación entre los cortes).
(G) Exclusivamente cortes tipo guillotina, (U) No requiere cortes tipo no guillotina

f. Restricciones inherentes a la orientación de las piezas (posibilidad de que las piezas puedan rotar 90° o no).
(T) Las piezas pueden rotar 90°, (A) Las piezas no pueden rotar 90°

g. Valores de las piezas (beneficio que ofrece empacar una determinada pieza).
(W) Ítems con valores de beneficio diferente a su área
(Z) Ítems con valores de beneficio igual a su área

h. Demanda de las piezas.
(E) Piezas con límite máximo de corte, (I) Piezas sin límite de corte (infinito)

i. Forma de las figuras.
(L) Ítems con forma regular, (K) Ítems con forma irregular.

Se definen las características de un problema por la nueve-tupla *(a, b, c, d, e, f, g, h, i)*. La nueve-tupla que identifica el problema propuesto en este artículo es la siguiente (*2, B, O, M, G, -, W, E, L*). La ausencia de un identificador de una característica representada por el símbolo (-) expresa que todas las opciones de esa característica son tomadas en cuenta.

Existen diferentes objetivos en los problemas de empaquetamiento, algunos de estos son con respecto al valor de los ítems a ser empacados, como la minimización de las perdidas de material, otros objetivos con relación a los precios de las items, como maximizar el costo del material embalado, mientras otros objetivos dependen del proceso de corte. En este trabajo se considera como único objetivo maximizar el beneficio del material empacado.

3. MODELO MATEMÁTICO

Se presenta un modelo de programación entera mixta para el problema de empaquetamiento óptimo en placas con piezas de valor basado en la caracterización de los patrones guillotina y el uso de las coordenadas donde pueden ser ubicadas las piezas.
Se asume que n piezas tienen que ser embaladas en una placa con ancho W y una altura H. Para un patrón donde la esquina inferior izquierda de cada pieza k (donde $k = 1, 2,\ldots, n$) es ubicada en las coordenadas (α_k, β_k), se puede obtener siempre dos series ordenadas $(x_1, x_2,\ldots,x_n)$ y $(y_1, y_2,\ldots,y_n)$ por un ordenamiento decreciente de las series $(\alpha_1, \alpha_2,\ldots, \alpha_n)$ y $(\beta_1, \beta_2,\ldots, \beta_n)$ respectivamente. Como consecuencia, se obtiene $0 \le x_1 \le x_2 \le \ldots \le x_n \le W$ y $0 \le y_1 \le y_2 \le \ldots \le y_n \le H$.
Para obtener un modelo lineal, se usaran las siguientes variables binarias para caracterizar la ubicación de las piezas en la placa:

$$z_{i,j,k} = \begin{cases} 1 & \text{si la pieza } k \text{ es empacada en } (i,\ j) \\ 0 & \text{de lo contrario} \end{cases}$$

Las siguientes variables de decisión intermedias también son necesarias:

$$u_{i,j,i'} = \begin{cases} 1 & \text{si la pieza en } (i,\ j),\ \text{no excede (horizontalmente) } x_{i'} \text{ con } i' > i \\ 0 & \text{de lo contrario} \end{cases}$$

$$v_{i,j,j'} = \begin{cases} 1 & \text{si la pieza en } (i,\ j),\ \text{no excede (verticalmente) } y_{j'} \text{ con } j' > j \\ 0 & \text{de lo contrario} \end{cases}$$

El siguiente conjunto de tres variables binarias son necesarias para garantizar las restricciones guillotina:

$$p_{i_1,i',j_1,j_2} = \begin{Bmatrix} 1 \quad \text{si no hay pieza entre } (i_1, j_1) \text{ y } (i'-1, j_2) \\ \text{que exceda } x_{i'} \text{ (consecuentemente, un corte} \\ \text{vertical en } x_{i'} \text{ no cruza ninguna pieza} \\ \text{empacada entre } (i_1, j_1) \text{ y } (i'-1, j_2)), i' > i_1 \\ 0 \quad \text{de lo contrario} \end{Bmatrix}$$

$$q_{i_1,i_2,j_1,j'} = \begin{Bmatrix} 1 \quad \text{si no hay pieza entre } (i_1, j_1) \text{ y } (i_2, j'-1) \\ \text{que exceda } y_{j'} \text{ (consecuentemente, un corte} \\ \text{horizontal en } y_{j'} \text{ no cruza ninguna pieza} \\ \text{empacada entre } (i_1, j_1) \text{ y } (i_2, j'-1)), j' > j_1 \\ 0 \quad \text{de lo contrario} \end{Bmatrix}$$

$$d_{i_1,i_2,j_1,j_2} = \begin{Bmatrix} 1 \quad \text{si existe al máximo una pieza} \\ \text{empacada entre } (i_1, j_1) \text{ y } (i_2, j_2) \\ 0 \quad \text{de lo contrario} \end{Bmatrix}$$

En la formulación matemática presentada en [(1)-(24)], las restricciones (4), (5) y (6) garantizan que cada posición horizontal o vertical sea ocupada por exactamente una pieza y cada pieza es empacada exactamente una vez.

Las restricciones (7) y (8) garantizan la coherencia de la definición de $u_{i,j,i'}$. Si $x_{i'} > x_i + \sum_{k=1}^{n} w_k z_{i,j,k}$ (es decir si la pieza empacada en la posición (i, j) no excede $x_{i'}$), entonces $u_{i,j,i'} = 1$. La restricción (8) requiere que $u_{i,j,i'} = 1$ en caso de que la restricción (7) sea satisfecha. Caso contrario, si $x_{i'} < x_i + \sum_{k=1}^{n} w_k z_{i,j,k}$ (es decir, si una pieza empacada en la posición (i, j) excede $x_{i'}$), entonces la restricción (7) requiere $u_{i,j,i'} = 0$. Note que $u_{i,j,i'}$ puede tomar el valor *0* o el valor *1*, si $x_{i'} - x_i = \sum_{k=1}^{n} w_k z_{i,j,k}$. En tal caso, en las restricciones (14) y (18) $u_{i,j,i'}$ sea hace igual a *1*.

Por simetría, las restricciones (9) y (10) aseguran la coherencia de la definición de $v_{i,j,j'}$. La restricción (11) garantiza la coherencia de la definición de d_{i_1,i_2,j_1,j_2}. En efecto, si $\sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} \sum_{i=1}^{n} z_{i,j,k} \geq 2$ (al menos dos piezas son empacadas entre (i_1, j_1) y (i_2, j_2)), la restricción (11) requiere que $d_{i_1,i_2,j_1,j_2} = 0$. De otra forma, d_{i_1,i_2,j_1,j_2} puede tomar un valor de *0* o *1*. En la restricción (18) d_{i_1,i_2,j_1,j_2} sea hace igual a *1*, lo cual es consistente con la definición.

Las restricciones [(12)-(18)] son restricciones guillotina para cada área rectangular. Si las restricciones guillotina son satisfechas para cada área rectangular, entonces también se cumple para todo el patrón de corte.

Las restricciones [(12)-(14)] aseguran la coherencia de la definición de p_{i_1,i',j_1,j_2}. En particular, si $\sum_{j=j_1}^{j_2} \sum_{i=1}^{n} z_{i',j,k} = 0$; es decir, ninguna pieza es empacada entre (i', j_1) y (i', j_2), la restricción (12) requiere $p_{i_1,i',j_1,j_2} = 0$. Similarmente, si

$$\text{Maximizar} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_k z_{i,j,k} \quad \forall k, \tag{1}$$

sujeto a:

$$0 \leq x_1 \leq x_2 \leq \ldots \leq x_n, \tag{2}$$

$$0 \leq y_1 \leq y_2 \leq \ldots \leq y_n, \tag{3}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} z_{i,j,k} \leq 1 \quad \forall k, \tag{4}$$

$$\sum_{i=1}^{n} \sum_{k=1}^{n} z_{i,j,k} \leq 1 \quad \forall j, \tag{5}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{n} z_{i,j,k} \leq 1 \quad \forall i, \tag{6}$$

$$x_{i'} - x_i - \sum_{k=1}^{n} w_k z_{i,j,k} \geq (u_{i,j,i'} - 1) W \quad \forall i, \forall j, \forall i' > i, \tag{7}$$

$$x_{i'} - x_i - \sum_{k=1}^{n} w_k z_{i,j,k} \geq u_{i,j,i'} W \quad \forall i, \forall j, \forall i' > i, \tag{8}$$

$$y_{j'} - y_j - \sum_{k=1}^{n} h_k z_{i,j,k} \geq (v_{i,j,i'} - 1) H \quad \forall i, \forall j, \forall j' > j, \tag{9}$$

$$y_{j'} - y_j - \sum_{k=1}^{n} h_k z_{i,j,k} \geq u_{i,j,j'} H \quad \forall i, \forall j, \forall j' > j, \tag{10}$$

$$(1 - d_{i_1,i_2,j_1,j_2}) n \geq \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j_2} \sum_{k=1}^{n} z_{i,j,k} - 1 \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j_2 > j_1, \tag{11}$$

$$p_{i_1,i',j_1,j_2} \leq \sum_{j=j_1}^{j_2} \sum_{k=1}^{n} z_{i',j,k} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \forall i' > i_1, \tag{12}$$

$$p_{i_1,i',j_1,j_2} \leq \sum_{i=i_1}^{i'-1} \sum_{j=j_1}^{j_2} \sum_{k=1}^{n} z_{i,j,k} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \forall i' > i_1, \tag{13}$$

$$(i'-i_1)(j_2 - j_1 + 1) p_{i_1,i',j_1,j_2} \leq \sum_{i=i_1}^{i'-1} \sum_{j=j_1}^{j_2} u_{i,j,i'} \quad \forall i_1, \forall j_1, \forall j_2 > j_1, \forall i' > i_1, \tag{14}$$

$$q_{i_1,i_2,j_1,j'} \leq \sum_{i=i_1}^{i_2} \sum_{k=1}^{n} z_{i,j',k} \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j' > j_1, \tag{15}$$

$$q_{i_1,i_2,j_1,j'} \leq \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j'-1} \sum_{k=1}^{n} z_{i,j,k} \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j' > j_1, \tag{16}$$

$$(j'-j_1)(i_2 - i_1 + 1) q_{i_1,i_2,j_1,j'} \leq \sum_{i=i_1}^{i_2} \sum_{j=j_1}^{j'-1} v_{i,j,i'} \quad \forall i_1, \forall j_1, \forall j_2 > i_1, \forall j' > j_1, \tag{17}$$

$$d_{i_1,i_2,j_1,j_2} + \sum_{i'=i_1+1}^{i_2} p_{i_1,i',j_1,j_2} + \sum_{j'=j_1+1}^{j_2} q_{i_1,i_2,j_1,j'} \geq 1 \quad \forall i_1, \forall j_1, \forall i_2 > i_1, \forall j_2 > j_1, \tag{18}$$

$$W \geq x_i + \sum_{j=1}^{n} \sum_{k=1}^{n} w_k z_{i,j,k} \quad \forall i, \tag{19}$$

$$H \geq y_j + \sum_{i=1}^{n} \sum_{k=1}^{n} h_k z_{i,j,k} \quad \forall j, \tag{20}$$

$$z_{i,j,k} \in \{0,1\} \quad \forall i, \forall j, \forall k, \tag{21}$$

$$u_{i,j,i'} \in \{0,1\} \quad \forall i, \forall j, \forall i' > i, \tag{22}$$

$$v_{i,j,j'} \in \{0,1\} \quad \forall i, \forall j, \forall j' > j, \tag{23}$$

$$d_{i_1,i_2,j_1,j_2}, p_{i_1,i_2,j_1,j_2}, q_{i_1,i_2,j_1,j_2} \in \{0,1\} \quad \forall i, \forall j, \forall i_2 > i_1, \forall j_2 > j_1, \tag{24}$$

$\sum_{i=i_1}^{i'-1}\sum_{j=j_1}^{j_2}\sum_{i=1}^{n} z_{i,j,k} = 0$; es decir, ninguna pieza es empacada entre (i_1, j_1) e (i'-1, j_2), la restricción (12) requiere $p_{i_1,i',j_1,j_2} = 0$. En cualquier caso, $x_{i'}$ no es el límite inferior de un intervalo disjunto con piezas empacadas entre (i_1, j_1) y (i'-1, j_2). Además, si $p_{i_1,i',j_1,j_2} = 1$ en la restricción (14) entonces $u_{i,j,i'} = 1$ para todo i tal que $i_1 \le i \le i'$ y para todo j tal que $j_1 \le i \le j_2$. Esto significa que ninguna pieza empacada entre (i_1, j_1) e (i'-1, j_2) excedería $x_{i'}$. La fusión de intervalos corresponde a la proyección de estas piezas en el eje-x, formando al menos dos intervalos disjuntos. En algunos casos, p_{i_1,i',j_1,j_2} puede ser indiferentemente igual a *0* o *1*. En estos casos, la restricción (18) establece que p_{i_1,i',j_1,j_2} sea igual a *1*, lo cual es consistente con la definición.

Por simetría, las restricciones [(15)-(17)] garantizan la coherencia de la definición de $q_{i_1,i_2,j_1,j'}$.

La restricción (18) requiere que al menos una de las siguientes condiciones se cumpla:

1. $d_{i_1,i_2,j_1,j_2} = 1$.
2. $p_{i_1,i',j_1,j_2} = 1$ para algún i' tal que $i_1 \le i' \le i_2$.
3. $q_{i_1,i_2,j_1,j'} = 1$ para algún j' tal que $j_1 \le j' \le j_2$.

Las restricciones (19) y (20) garantizan que ninguna pieza exceda horizontalmente el ancho W y que ninguna pieza exceda verticalmente la altura H. La función objetivo es maximizar el lucro de piezas empacadas (ver ecuación (1)).

4. TÉCNICAS DE SOLUCIÓN

4.1. Optimización con cúmulo de partículas

El algoritmo PSO está basado en una técnica de optimización estocástica desarrollada por el Dr. Eberhart y el Dr. Kennedy en 1995 inspirada en el comportamiento social de las aves migratorias y los cardúmenes de peces [9].

Este algoritmo puede ser computacionalmente ineficiente por que puede quedar atrapado fácilmente en óptimos locales cuando resuelve problemas cuyo espacio de solución es multimodal. Sin embargo, acelerar la velocidad de convergencia y evitar los óptimos locales se han convertido en los dos más importantes objetivos en la investigación de PSO [9], [10].

Para mejorar la eficiencia y acelerar el proceso de búsqueda es fundamental determinar el estado de evolución y los mejores valores para los parámetros, esto con el fin de evitar posibles óptimos locales en el estado de convergencia. Algunas técnicas se han planteado introduciendo operadores como la selección [11], cruzamiento [12], mutación [13], búsqueda local [14].

En este trabajo se está proponiendo un algoritmo que combina el PSO con la técnica VNS a fin de realizar el proceso de intensificación en el espacio de soluciones de cada iteración de una forma controlada evitando de está forma quedar atrapados en los óptimos locales.

4.2. Estructura del PSO

En PSO, un conjunto de partículas representan potenciales soluciones, donde cada partícula *i* está asociada a dos vectores, así:

El vector de velocidades $V_i = [V_i^1, V_i^2,\ldots, V_i^D]$

El vector de posiciones $X_i = [x_i^1, x_i^2,\ldots, x_i^D]$

donde *D* determina el tamaño del espacio de solución. La velocidad y la posición de cada partícula son inicializadas por vectores aleatorios con sus correspondientes rangos. Durante el proceso de evolución, la velocidad y la posición de la partícula *i* se actualizan mediante la siguiente expresión.

$$v_i^d = wv_{i-1}^d + \cdots + c_1 rand_1^d (pbest_i^d - x_i^d) + \cdots + c_2 rand_2^d (nbest^d - x_i^d) \quad (25)$$

$$x_i^d = x_{i-1}^d + v_i^d \quad (26)$$

Donde *w* es el peso de inercia, c_1 y c_2 son los coeficientes de aceleración y $rand_1^d$ y $rand_1^d$ son números aleatorios uniformemente distribuidos y generados en el intervalo [0-1] para la *d-esima* dimensión.

Los pasos del algoritmo son los siguientes:

i) Se inicia con una población de partículas, con posiciones y velocidades en el espacio del problema *d* dimensional generado de forma aleatoria y los registros *pbest* y *gbest* son inicializados con las posiciones iniciales de cada partícula.

ii) A cada partícula se le evalúa la función objetivo.

iii) Se compara el valor de la función de adaptación (*fitness*) de la partícula con su *pbest* (la mejor posición obtenida por la partícula). Si su valor actual es mejor, el *pbest* pasa a ser igual al valor de la *fitness* de la partícula y la localización de la *pbest* pasa a ser igual a la localización actual del espacio d dimensional.

iv) Se compara el valor de la función *fitness* con el mejor valor de adaptación de la población. Si el valor actual es mejor que el *gbest* (la mejor posición alcanzada por el cúmulo), actualizar el valor del *gbest*.

v) Modificar la velocidad y la posición de acuerdo a las ecuaciones (25) y (26), respectivamente.

vi) Volver al paso *ii* hasta que el criterio de parada sea satisfecho. Uno de los más utilizados es el número máximo de ciclos generacionales.

4.3. Búsqueda en vecindario variable (VNS)

La búsqueda en vecindario variable (Variable Neighborhood Search, VNS) es una metaheurística para resolver problemas de optimización cuya idea básica es el cambio sistemático de vecindad dentro de una búsqueda local [15].

Los procesos de búsqueda heurística están generalmente basados en transformaciones de las alternativas que determinan una estructura de entornos en el espacio de soluciones. La búsqueda en vecindario variable es una metaheurística propuesta recientemente [16], [17]. Su desarrollo ha sido rápido, con muchos artículos ya publicados o pendientes de aparecer. Se han realizado muchas extensiones, principalmente para permitir la solución de problemas de gran tamaño [18].

Un problema de optimización consiste en encontrar, dentro de un conjunto *X* de soluciones factibles, la que optimiza una función *f(x)*. Si el problema es de minimización se formula como:

$$\min\{ f(x) / x \in X \} \quad (27)$$

Donde *x* representa una solución alternativa, *f* es la función objetivo y *X* es el espacio de soluciones factibles del problema. Una solución optima x^* (o mínimo global) del problema es una solución factible donde se alcanza el mínimo de la ecuación (27).

Una estructura de entornos en el espacio de soluciones *X* es una aplicación $N : X \rightarrow 2^x$ que asocia a cada solución $x \in X$ un entorno de soluciones $N(x) \subset X$, que se dicen vecinas de *x*. Las metaheurísticas de búsqueda local aplican una transformación o movimiento a la solución de búsqueda y por tanto utilizan explícita o implícitamente, una estructura de entornos. Denotando por N_k, $k=1,...,k_{max}$ a un conjunto finito de estructuras de entornos en el espacio *X*. Los entornos N_k pueden ser inducidos por una o más métricas introducidas en el espacio de soluciones *X*. La mayoría de las heurísticas de búsqueda local usan sólo una estructura de entornos.

Una solución *x'* es un mínimo global del problema planteado en la ecuación (27) si no existe una solución $x \in X$ tal que $f(x) < f(x^*)$. Se dice que $x' \in X$ es un mínimo local con respecto a N_k, si no existe una solución $x \in N_k(x') \subseteq X$ tal que $f(x) < f(x')$.

Una búsqueda local descendente cambia la solución actual por otra solución mejor de su entorno, por tanto tienen el peligro de quedarse atrapada en un mínimo local. Las metaheurísticas basadas en procedimientos de búsqueda local aplican distintas formas de continuar la búsqueda después de encontrar óptimos locales.

La VNS está basada en tres hechos simples:

- Un mínimo local con una estructura de entornos no lo es necesariamente con otra.
- Un mínimo global es mínimo local con todas las posibles estructuras de entornos.
- Para muchos problemas, los mínimos locales con la misma o distinta estructura de entornos están relativamente cerca.

5. ALGORITMO HÍBRIDO PROPUESTO PSO Y VNS

La codificación presentada por Álvarez et al. en [19] es utilizada para realizar la adaptación del algoritmo de optimización propuesto. En [19] el problema de optimización es limitado a encontrar los valores óptimos de los nodos de un árbol binario completo de tres niveles, llamado árbol de distancias de los cortes.

Dado que el algoritmo PSO puede presentar convergencia prematura debido a una inadecuada calibración de los parámetros, se propone una modificación al algoritmo básico del PSO, en el cual se introduce una rutina basada en el algoritmo VNS, que controla el número de nodos del árbol de distancias a los cuales se les actualiza la posición. La inclusión de este algoritmo vuelve el proceso lento pero a su vez incrementa la diversificación de las partículas en el espacio de soluciones. Lo anterior ya que en cada iteración solo calculan las velocidades de los nodos que hacen parte del vecindario de cada partícula. Al igual que el VNS Básico si la nueva partícula no presenta mejora se cambia el vecindario a uno más grande, de lo contrario su vecindario cambia por la mínima vecindad definida.

La figura 1 ilustra el diagrama de flujo de datos y los parámetros utilizados del algoritmo PSO+VNS. Como es expresada en la figura la función Calcular Velocidad SoloVecindario, sólo evaluará la ecuación (25) en los nodos que pertenezcan al vecindario actual de cada partícula.

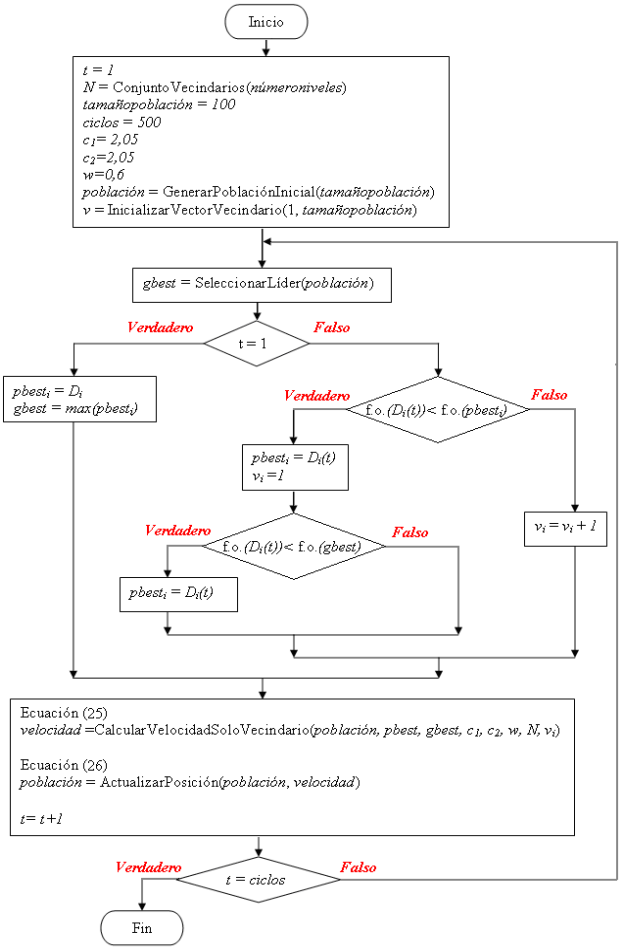

Figura 1. Algoritmo híbrido PSO y VNS.

6. ANÁLISIS DE RESULTADOS

Los casos utilizados para comprobar la metodología propuesta fueron tomados de la referencia [20]. En las tablas 1 y 2 se muestran los resultados obtenidos. Todos los algoritmos fueron implementados en Delphi 7 ® y ejecutados en una máquina de 3 GHz de procesador y 1 Gb de memoria RAM.

Los resultados obtenidos por la metodología propuesta sin rotación de piezas alcanzan 2 de los 3 casos reportados en [20], mientras que la inclusión de la rotación de piezas permite superar 2 de los 3 casos. En las tablas 1 y 2 se presentan los tiempos utilizados por el algoritmo para alcanzar las soluciones.

Caso	Solución óptima	Solución Alcanzada	Tiempo Utilizado(s)
CHW1	2892	2768	115
CHW2	1860	1860	155
CHW3	244	244	77

Tabla 1. Resultados sin rotación.

Caso	Solución óptima	Solución Alcanzada	Tiempo Utilizado(s)
CHW1	2892	2859	225
CHW2	1860	1900	310
CHW3	244	258	140

Tabla 2. Resultados con rotación.

7. CONCLUSIONES Y RECOMENDACIONES

La combinación del algoritmo PSO con el VNS evita una convergencia prematura en la solución del problema de la mochila. Como desventaja se observa una convergencia lenta del proceso, a pesar de esto, se asegura la obtención de soluciones de muy alta calidad al observarse un mejoramiento en el proceso de diversificación.

Se alcanzaron 2 de 3 de los casos reportados en [20] sin rotación de las piezas y se superaron 2 de 3 permitiendo rotación de las piezas, lo cual demuestra que la metodología propuesta presenta un buen desempeño.

La alternativa de rotar las piezas 90° en el problema de la mochila bidimensional, aumenta el grado de complejidad del problema pero mejora ostensiblemente la calidad de las respuestas al igual que sucede en otros problemas de corte y empaquetamiento.

8. BIBLIOGRAFÍA

[1] R. Gallego, E. Toro y A. Escobar, *Técnicas Metaheurísticas de Optimización*, Textos Universitarios, Universidad Tecnológica de Pereira, 2008.

[2] A. Caprara y M. Monaci, “On the two dimensional knapsack problem”, *Operations Research Letters*, Vol. 32, pp. 5-14, 2004.

[3] P. C. Gilmore, R. E. Gomory, “Multistage cutting problems of two and more dimensions”, *Oper. Res.*, Vol. 13, pp. 94–119, 1965.

[4] H. Dyckhoff, G. Scheithauer y J. Terno, “Cutting and Packing (C&P)”, in: M. Dell’Amico, F. MaUoli, S. Martello (Eds.), Annotated Bibliographies in Combinatorial Optimization, Wiley, Chichester, 1997, pp. 393–413.

[5] J. E. Beasley, “An exact two-dimensional non-guillotine cutting tree search procedure”, *Oper. Res.*, Vol. 33, pp. 49–64, 1985.

[6] E. Hadjiconstantinou, N. Christo1des, “An exact algorithm for the orthogonal, 2D cutting problems using guillotine cuts”, *European J. Oper. Res.*, Vol. 83, pp. 21–38, 1995.

[7] M. A. Boschetti, E. Hadjiconstantinou y A. Mingozzi, “New upper bounds for the two-dimensional orthogonal non guillotine cutting stock problem”, *IMA Journal of Management Mathematics*, Vol. 13, pp. 95–119, 2002.

[8] A. Lodi, S. Martello, D. Vigo, “Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems”, *INFORMS J. Comput.* Vol. 11, pp. 345–357, 1999.

[9] J. J. Liang, A. K. Qin, P. N. Suganthan y S. Baskar, “Comprehensive learning particle swarm optimizer for global optimization of multimodal functions”, *IEEE Trans. Evol. Comput.*, Vol. 10, pp. 281–295, 2006.

[10] R. C. Eberhart y Y. Shi, “Guest editorial”, *IEEE Trans. Evol. Comput.—Special Issue Particle Swarm Optimization*, Vol. 8, pp. 201–203, 2004.

[11] P. J. Angeline, "Using selection to improve particle swarm optimization", *in Proc. IEEE Congr. Evol. Comput.*, Anchorage, AK, pp. 84–89 , 1998.
[12] Y. P. Chen, W. C. Peng y M. C. Jian, "Particle swarm optimization with recombination and dynamic linkage discovery", *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 6, pp. 1460–1470, 2007.
[13] P. S. Andrews, "An investigation into mutation operators for particle swarm optimization", *in Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Canada, pp. 1044–1051, 2006.
[14] J. J. Liang y P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with local search," *in Proc. IEEE Congr. Evol. Comput.*, pp. 522–528, 2005.
[15] P. Hansen, N. Mladenović y J. Moreno, "Búsqueda de entorno variable", Revista Iberoamericana de Inteligencia Artificial, No. 19, pp. 77-92, 2003.
[16] P. Hansen y N. Mladenović. "Variable neighborhood search for the p-median". *Location Science*, Vol. 5, pp. 207–226, 1997.
[17] N. Mladenović, "A variable neighborhood algorithm a new metaheuristic for combinatorial optimization", *In Abstracts of Papers Presented at Optimization Days*, page 112, 1995.
[18] N. Mladenović y P. Hansen, "Developments of variable neighborhood search". *Les Cahiers du GERAD*, Vol. 24, 2001.
[19] D. Álvarez, E. Toro y R. Gallego, "Algoritmo de optimización cúmulo de partículas aplicado en la solución del problema de empaquetamiento óptimo bidimensional con y sin rotación", *Scientia et Technica*, Vol. 42, pp. 204-209, 2009.
[20] N. Christofides y C. Whitlock, "An algorithm for two-dimensional cutting problems", Operations Research, Vol. 25, pp. 30-44, 1977.