# “Cooperation Greedy Monkey Algorithm”: Algoritmo paralelo para resolver la clase fuertemente correlacionada del problema de la mochila 0-1

## “Cooperation Greedy Monkey Algorithm”: Parallel algorithm to solve the strongly correlated class of the knapsack problem 0-1

José Crispín Zavala-Díaz[1], Joaquín Pérez-Ortega[2], Nely Nelva Almanza-Ortega[3], Jaqueline López-Calderón[1]

[1] Facultad de contaduría, administración e Informática UAEM, Av. Universidad 1001, Col. Chamilpa, Cuernavaca, Morelos CP: 62210

[2] Departamento de Ciencias Computacionales, TecNM/Centro Nacional de Investigación y Desarrollo Tecnológico, Interior Internado Palmira S/N, Col. Palmira, Cuernavaca, Morelos CP: 62490

[3] División de Estudios de Posgrado e Investigación, TecNM/Instituto Tecnológico de Tlalnepantla, Av. Instituto Tecnológico s/n, Col. La Comunidad, Tlalnepantla de Baz, Estado de México CP: 54070

* Correo-e: crispin_zavala@uaem.mx

PALABRAS CLAVE:

Algoritmo del mono ávido cooperativo, Inteligencia de enjambre, El problema del peso en la mochila 0-1.

RESUMEN

Se presenta la paralelización del Cooperation Greedy Monkey Algorithm y el ajuste de parámetros para resolver el problema KP 0-1 (0-1 Knapsack Problem). Los problemas resueltos son tomados de la literatura especializada hasta las instancias establecidas por Pisinger, las no correlacionadas, las débilmente correlacionadas y las fuertemente correlacionadas. Se amplía la capacidad de solución del algoritmo para resolver instancias con diferentes porcentajes del 25% y 50% de la suma de los pesos de los elementos, y no únicamente el 75% como está diseñado el algoritmo originalmente. Se utilizó un modelo maestro-esclavo para su implementación paralela en un cluster de 5 servidores. Los resultados son alentadores y en algunas ocasiones se calcula la solución óptima.

.

KEYWORDS:

Cooperation Greedy Monkey Algorithm, Swarm Intelligence, Knapsack Problem 0-1.

ABSTRACT

The parallelization of the Cooperation Greedy Monkey Algorithm and the adjustment of parameters to solve the problem KP 0-1 (0-1 Knapsack Problem) is presented. The solved problems are taken from the specialized literature up to the instances established by Pisinger, the uncorrelated, the weakly correlated and the strongly correlated. The solution capacity of the algorithm is extended to solve instances with different percentages of 25% and 50% of the sum of the weights of the elements, and not only 75% as the algorithm was originally designed. A master-slave model was used for its parallel implementation in a cluster of 5 servers. The results are encouraging and the optimal solution is sometimes calculated.

**Recibido:** 12 de diciembre de 2020 • **Aceptado:** 25 de mayo de 2021 • **Publicado en línea:** 4 de junio de 2021

## 1. Introducción.

El problema de la mochila KP 0-1 (0-1 *Knapsack Problem*) ha sido intensamente estudiado desde la segunda mitad del siglo XX [1]. Este problema simula el llenado de una mochila limitada por su capacidad, donde se ingresará un conjunto de elementos, cada uno con un beneficio y un peso específico, el objetivo es obtener el mayor beneficio de la suma de todos los elementos dentro de la mochila sin exceder su capacidad.

El KP 0-1 tiene una gran variedad de aplicaciones en la vida cotidiana como son la planificación de la producción, los modelos financieros, en la selección de proyectos, en la asignación de datos en sistemas distribuidos, en la planificación de la capacidad de instalaciones, entre otros [2].

A través de los años diferentes autores de la comunidad científica han propuesto métodos que ofrecen soluciones exactas o aproximadas para el problema de la mochila. Estos métodos utilizan instancias de prueba para comprobar la eficiencia en la solución del problema como son las siete clases propuestas por Pisinger [3]. De esta manera, si un algoritmo tiene buen desempeño al resolver estas instancias es probable que resuelva un problema de la vida cotidiana.

Tres de esas clases son las no correlacionadas, débilmente correlacionadas y fuertemente correlacionadas. Las clases no correlacionadas y débilmente correlacionadas son resueltas con algoritmos como fuerza bruta (*Brute-force*, BF), ramifica y corta (*Branch and Bound*, BB) y programación dinámica (*Dynamic Programming*, DP), [4]. Por otro lado, la clase fuertemente correlacionada es el tipo de clase más difícil, representa situaciones de la vida diaria [5], en esta clase los pesos son distribuidos a través de un rango, donde el beneficio y el peso están fuertemente correlacionados entre ellos, lo que hace complejo encontrar una solución que cumpla con la restricción de capacidad de la mochila [3].

Se han propuesto variedad de métodos de solución como la hibridación de algoritmos metaheurísticos [6] y estrategias metaheurísticas paralelas, con el fin de ofrecer buenas soluciones en tiempo de cómputo aceptable a problemas difíciles como el KP 0-1.

Alfonso et al [7] mencionan que el paralelismo junto con la hibridación constituyen dos de las ramas más exitosas para la construcción de algoritmos eficientes para resolver los problemas complejos de optimización. Sin embargo, en las últimas décadas el campo de las metaheurísticas paralelas ha estado evolucionando, esto a la necesidad de resolver problemas complejos de la vida cotidiana [8].

En la actualidad las implementaciones paralelas de las metaheurísticas son una alternativa efectiva para acelerar las metaheurísticas secuenciales, se reduce el tiempo de búsqueda y se mejora la calidad de las soluciones [9].

El algoritmo *Cooperation Greedy Monkey Algorithm* (CGMA) ha mostrado ser un algoritmo hibrido que ha resuelto el problema KP0-1 [10], con la limitante de que el algoritmo está configurado para resolver instancias únicamente con una restricción del .75 de la suma de los pesos de los elementos. Por lo que se propone paralelizar este algoritmo y ampliar su aplicación en la solución del problema KP0-1 con otras capacidades de la mochila, así como resolver las instancias más difíciles de este problema, las fuertemente correlacionadas. Esto se logra a través de una sintonización de sus parámetros para la selección de elementos. La paralelización se realiza en un cluster de 120 cores repartidos en 5 servidores.

En la sección dos está el modelo matemático y una breve revisión del estado del arte de las metaheristicas paralelas. En la sección tres está nuestro enfoque para paralelizarlo y realizar el tuning de las variables utilizadas den el proceso CGMA. En la sección cuatro están los datos de la computadora, y aspectos de la implementación paralela. En la sección 5 están los resultados, los que se dividen en cuatro subsecciones. La primera y segunda consiste de las

pruebas sencillas encontradas en la literatura. En las demás secciones se realizan las pruebas de las instancias más complejas de Pisinger [3].

## 2. Fundamentación

El problema de la mochila KP 0-1 se define matemáticamente de la siguiente forma:

- Función objetivo:

$$f_{opt} = max \sum_{i=1}^{n} p_i x_i \quad (1)$$

Sujeto a:

$$\sum_{i=1}^{n} w_i x_i \leq c \quad (2)$$

Donde:

$f_{opt}$: Función objetivo

$p_i$: Beneficio del elemento $i$, $p_i \hat{I} Z^+$

$w_i$: Peso del elemento $i$, $w_i \hat{I} Z^+$

$c$: Capacidad de la mochila

$x_i \hat{I} \{0,1\}$, i = 1, 2, …, n

La ecuación (1) hace referencia a maximizar la utilidad de la mochila a partir de la suma de los beneficios de cada uno de los elementos. La restricción (2) indica que es necesario tener en cuenta el peso total de los elementos que serán guardados en la mochila, cuya capacidad está determinada por la variable $c$. La variable $x_i$ toma sus valores del conjunto {0, 1}, donde "0" significa que elemento no se incluye dentro de la mochila y "1" cuando el elemento si se incluye.

El KP 0-1 consiste en determinar qué elementos se deben seleccionar para incluir en la mochila, de tal manera que la utilidad total de los elementos seleccionados sea la máxima sin exceder la capacidad de la mochila.

En el campo de investigación de los algoritmos metaheurísticos inspirados en la naturaleza como la inteligencia del enjambre SI (*Swarm Intelligence*), ha tomado importancia para la solución en problemas de optimización combinatoria a causa de que son eficientes y versátiles para resolver problemas con aplicaciones del mundo real [11].

La SI se basa en el comportamiento inteligente y la adaptación de varios animales, aves e insectos, se considera un nivel próximo al de la inteligencia humana [12].

Las metaheurísticas paralelas y distribuidas abordan los problemas complejos de optimización, su objetivo principal es resolver problemas aún más complejos de grandes dimensiones en tiempos de ejecución razonable, así como lograr una mayor diversidad y exploración del espacio de búsqueda [8].

Las metaheurísticas basadas en la población son naturalmente propensas a paralelizarse, porque la mayoría de sus operadores de variación se pueden realizar fácilmente en paralelo [13]. En la actualidad las implementaciones paralelas de las metaheurísticas son una alternativa efectiva para acelerar las metaheurísticas secuenciales, porque se reduce el tiempo de búsqueda y se mejora la calidad de las soluciones [9].

En la solución de los problemas de optimización Cotta et al [14] mencionaron que para aprovechar los beneficios aportados por los metaheurísticos bioinspirados, híbridos e implementaciones paralelas, de las metaheurísticas, a menudo es necesario combinar las técnicas. En el 2002 Talbi [15] señaló que el paralelismo puede introducirse de diversas maneras en metaheurísticas bioinspirados o híbridas, por lo que se necesita conocer perfectamente el problema que se va a resolver y la metaheurística que se va a utilizar para la paralelización, de esta manera se podrá realizar una correcta implementación de las diferentes técnicas.

En [16] se presenta un algoritmo genético paralelo para resolver el KP 0-1, se utilizan GPU (Unidad de

Procesamiento Grafico) para el cálculo. Proponen un modelo paralelo en el cual optimizan 4 instancias del KP 0-1 y realizan la comparación de calidad de los resultados obtenidos, donde demuestran mantener resultados razonables de la calidad de la solución.

Hajaria et al [17] implementaron un algoritmo paralelo metaheurístico de enjambre inspirado en el comportamiento de luciérnagas parpadeando, utilizaron la arquitectura CUDA (Arquitectura Unificada de Dispositivos de Cómputo) para el procesamiento paralelo. Los resultados muestran que el hardware de la GPU disminuye el tiempo de ejecución, es 320 veces más rápido que la solución del algoritmo secuencial.

Sonuc et al [18] propusieron un algoritmo metaheurístico inspirado en el proceso de recocido del acero y cerámicas, este algoritmo propuesto para el KP 0-1 fue paralelizado utilizando la arquitectura CUDA (Arquitectura Unificada de Dispositivos de Cómputo). Además, se ejecutó con una técnica de arranque múltiple para mejorar la calidad de las soluciones, para probar la eficacia del algoritmo, se utilizaron instancias de dimensión de 1000. Los resultados muestran que el enfoque paralelo supera al secuencial y corre hasta 16 veces más rápido que una CPU (Unidad Central de Procesamiento) de un solo núcleo.

El-Shafei et al [19] desarrollaron un algoritmo metaheurístico hibrido inspirado en la búsqueda de armonía binaria, utilizando la arquitectura paralela basada en FPGA (*Field-Programmable Gate Array*), realizaron la división del algoritmo en etapas independientes para poder realizar el cómputo de manera simultánea mejorando así el rendimiento, además es capaz de reducir el tiempo de cómputo en instancias de grandes dimensiones de KP 0-1.

El algoritmo *Monkey Algorithm*, propuesto por Zhao y Tang [20], es uno de los algoritmos de inteligencia de enjambre que ha tenido relevancia en la literatura, por su capacidad de resolver una gran variedad de problemas difíciles de optimización. Además, de ser capaz de encontrar la solución óptima global en diferentes dominios del espacio de búsqueda. Está inspirado en el comportamiento de los monos para resolver problemas de inteligencia numérica y se ha aplicado con éxito a diferentes problemas de la literatura con dimensiones de 30, 1000 o 10000, entre los cuales se encuentran [21, 22, 23].

Para el KP 0-1 Zhou, et al [10] propusieron una versión binaria del algoritmo mono mejorado CGMA (*Cooperation Greedy Monkey Algorithm*), se deriva de la simulación de los procesos de escalar de los monos, se agregaron dos procesos más; el proceso de estrategia greedy para mejorar las soluciones no factibles y el proceso de cooperación para mantener la diversidad de la población. De acuerdo a los resultados experimentales los autores demostraron que CGMA es un algoritmo que tiene grandes ventajas en la resolución de problemas de mochila KP 0-1 de pequeña y gran escala, comparado con otros algoritmos famosos en la literatura como [24, 25, 26].

Por sus buenos resultados en la solución del KP 0-1 el algoritmo CGMA es un buen candidato a ser paralelizado, con la finalidad de resolver el problema KP 0-1 con capacidades diferentes al 0.75 de la suma de sus elementos. Para lograr esta paralelización y modificación de los elementos se describe a continuación el método.

## 3. Enfoque paralelo del algoritmo de mono mejorado CGMA (*Cooperation Greedy Monkey Algorithm*)

Para el KP 0-1 Zhou, et al [10] El algoritmo mono está diseñado con 6 procesos principales:

1. Inicialización,
2. Ascenso,
3. Cooperación,
4. Estrategia greedy,
5. Observar y saltar, y
6. Salto mortal.

Se podría decir que es una secuencia de pasos, donde en cada uno de ellos se realizan operaciones donde intervienen ciertas variables. Variables que son tomadas en cuenta para el ajuste de los parámetros y resolver instancias más complejas. Los procesos del cálculo se utilizan para proponer la paralelización del método.

La paralización consiste en generar diferentes enjambres de monos en cada unidad de procesamiento, de tal forma que cada uno de ellos encuentren diferentes montañas. Para lograr lo anterior se utiliza el modelo maestro-esclavo [8] como se observa en la Figura 1. El modelo maestro-esclavo es uno de los métodos más famosos para la paralelización de algoritmos, el procesador maestro controla toda la actividad de un grupo de procesadores esclavos, asignando las tareas que se realizan en paralelo y ocupándose también de las operaciones de entrada/salida [8].

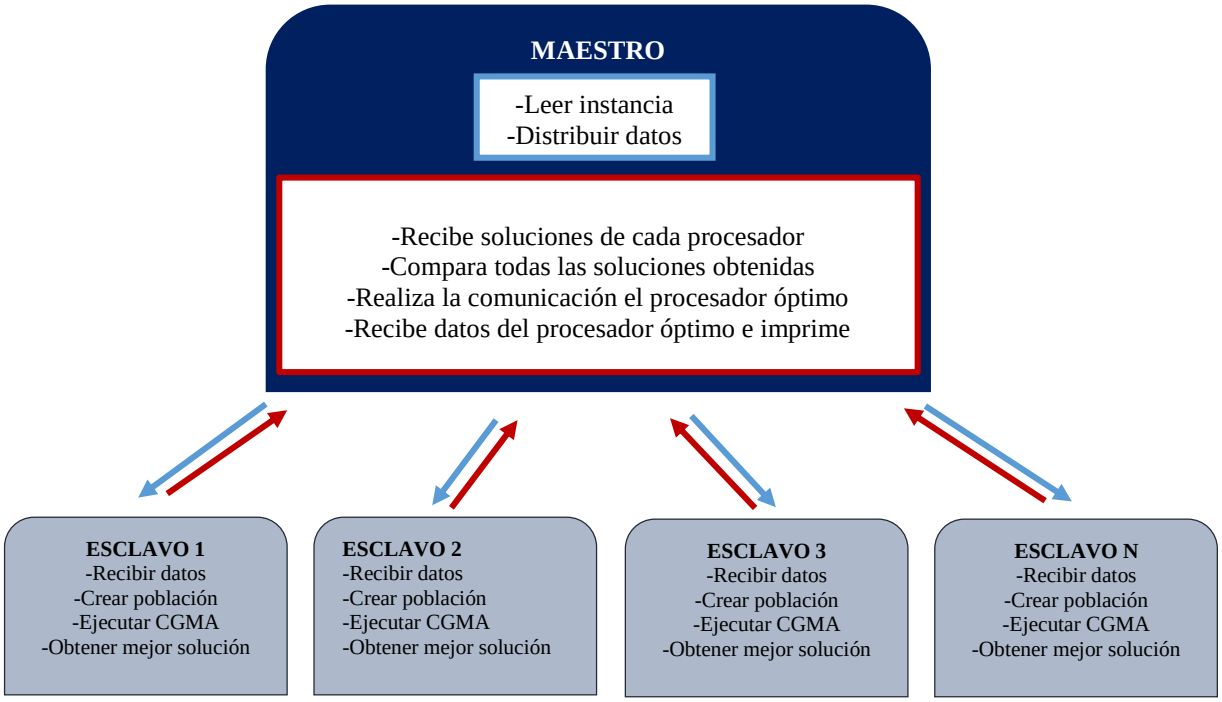


**Figura 1.** Esquema de la paralelización del Cooperation Greedy Monkey Algorithm

El procesador maestro lee la instancia del problema, la envía a los procesadores esclavos. Cada procesador esclavo crea su enjambre y ejecuta el CGMA, al finalizar envía la solución al procesador maestro. Este procesador selecciona la mejor solución y la envía a los esclavos, el esclavo que tenga la mejor solución envía los datos de la misma al procesador maestro. Finalmente, el procesador maestro imprime la solución.

De las variables utilizadas en el proceso se seleccionaron seis. La primera del proceso de inicialización, la inicialización de la población consiste en colocar a cada uno de los monos de la población en una posición de las montañas de forma aleatoria. Para el mono *i* su posición se denota a través de un vector $X_{ij} = ( x_{i1} , x_{i2},\ldots, x_{in})$, en donde j va de 1 hasta n elementos, que es el tamaño del vector que será utilizado para expresar una solución del KP 0-1. Esta variable $x_{ij}$ toma un valor de un número real dentro del intervalo de [0.0, 1.0], cuando el numero aleatorio es menor a 0.50 indicará que el elemento queda afuera, sino, este será incluido dentro en la

mochila. El primer cambio es modificar el 0.50 a otros valores para poner al mono i en otra posición de inicio.

El segundo cambio es el tamaño de la población, el valor recomendado es de 10. Por lo que se modificará este número.

La tercera variable se considera del proceso de escalar, la variable es la que controla el ascenso (Nc), este es el número máximo permitido de iteraciones del proceso de ascenso. El valor original es de 2.

La cuarta variable es el número de repeticiones para saltar, los monos simularán la búsqueda observando una montaña más alta que su actual y saltaran a ella, este proceso se repetirá hasta que se haya alcanzado el número máximo de repeticiones denotado por Nw. El valor de referencia es 2.

La quinta variable es el número de repeticiones del ciclo, el valor de referencia es de 50.

La última variable a considerar es la capacidad de la mochila, de 0.25, 0.50 y 0.75 de la suma de los pesos de los elementos. El valor utilizado en las referencias es de 0.75.

En la tabla siguiente se muestra un resumen de las modificaciones propuestas.

**Tabla 1**. Ajustes propuestos de los parámetros del Monkys

| Sintonización | | | | |
|---|---|---|---|---|
| Parámetros | Literatura | Ajustes | | |
| Variable de decisión en evaluación de 0-1 | 0.50 | 0.45 | 0.55 | 0.60 |
| Tamaño de la población | 10 | 5 | 15 | 20 |
| N° de repeticiones de Escalar | 2 | 1 | 3 | 4 |
| N° de repeticiones de Observar- Saltar | 2 | 1 | 3 | 4 |
| Restricción de capacidad | $0.75 \sum_{i=1}^{n} w_i$ | ✓ $0.75 \sum_{i=1}^{n} w_i$<br>✓ $0.50 \sum_{i=1}^{n} w_i$<br>✓ $0.25 \sum_{i=1}^{n} w_i$ | ✓ $0.75 \sum_{i=1}^{n} w_i$<br>✓ $0.50 \sum_{i=1}^{n} w_i$<br>✓ $0.25 \sum_{i=1}^{n} w_i$ | ✓ $0.75 \sum_{i=1}^{n} w_i$<br>✓ $0.50 \sum_{i=1}^{n} w_i$<br>✓ $0.25 \sum_{i=1}^{n} w_i$ |
| N°de repeticiones del ciclo | 50 | 50 | 50 | 50 |

## 4. Consideraciones para las pruebas finales.

El programa fue escrito en ANSI C y se utiliza la biblioteca MPI para el paso de mensajes. En la implementación del modelo maestro-esclavo es necesario transmitir información entre un grupo de procesadores. A través de las funciones de comunicación colectiva, un proceso envía datos a varios procesos o recoge resultados de otros tantos en una sola operación, las rutinas de comunicación que se utilizan son de tipo MPI_Bcast y MPI_Gather.

Las características del clúster Ioevolution donde se llevó a cabo la implementación tiene las características siguientes:

**Tabla 2.** Características del cluster ioevolution

| Clúster Ioevolution | |
|---|---|
| Máquinas | Cores disponibles |
| Ioevolution | 32 |
| compute-0-0 | 72 |
| compute-0-1 | 16 |
| compute-0-2 | 16 |
| compute-0-3 | 16 |

A causa de que cada core calcula el algoritmo CGMA es necesario tener un buen balance de carga entre los procesadores, lo que se busca es que en forma equitativa cada servidor alcance su 100% de carga. El balance de carga que se propone se muestra en la tabla siguiente

**Tabla 3.** Distribución de la carga en el cluster ioevolution

| | Cores | | | | | | |
|---|---|---|---|---|---|---|---|
| Maquinas | 20 | 40 | 60 | 80 | 100 | 120 | 140 |
| Ioevolution | 4 | 11 | 15 | 22 | 26 | 32 | 32 |
| compute-0-0 | 4 | 11 | 15 | 22 | 26 | 40 | 60 |
| compute-0-1 | 4 | 6 | 10 | 12 | 16 | 16 | 16 |
| compute-0-2 | 4 | 6 | 10 | 12 | 16 | 16 | 16 |
| compute-0-3 | 4 | 6 | 10 | 12 | 16 | 16 | 16 |

## 5. Resultados de la experimentación computacional

El ajuste se realizó con un problema no correlacionado de un tamaño de 100 elementos. Los mejores resultados se obtuvieron con los valores que se muestran en la tabla 4

**Tabla 4.** Valores obtenidos de la mejor sintonización

| Mejor Sintonización | |
|---|---|
| Criterios | Ajuste |
| Variable de decisión en evaluación de 0-1 | 0.60 |
| Tamaño de la población | 20 |
| N° de repeticiones de Escalar | 4 |
| N° de repeticiones de Observar- Saltar | 3 |
| Restricción de capacidad | ✓ 0.75 $\sum_{i=1}^{n} w_i$<br>✓ 0.50<br>✓ 0.25 |
| N°de repeticiones del ciclo | 50 |

### 5.1 Comparación con Instancias publicadas

Se toman 8 instancias de [10] para comprobar la eficiencia del CGMA paralelo, se mide el desempeño del algoritmo paralelo respecto al secuencial, los resultados se muestran en la Tabla 5.

**Tabla 5** Comparación de las 8 instancias de Zhou et al [10]

| Instancias de la literatura | | | | | | |
|---|---|---|---|---|---|---|
| Instancia | Dimensión | Capacidad | Valor Óptimo Referencia | Algoritmo secuencial vs paralelo | Valor de la Solución Aproximada | Cores |
| Kp_01 | 10 | 269 | 295 | CGMA_Sec | 295 | 1 |
| | | | | CGMA_Par | | 20 |
| Kp_02 | 20 | 878 | 1024 | CGMA_Sec | 1024 | 1 |
| | | | | CGMA_Par | | 20 |
| Kp_03 | 4 | 20 | 35 | CGMA_Sec | 35 | 1 |
| | | | | CGMA_Par | | 20 |
| Kp_04 | 10 | 60 | 52 | CGMA_Sec | 52 | 1 |
| | | | | CGMA_Par | | 20 |
| Kp_05 | 7 | 50 | 107 | CGMA_Sec | 107 | 1 |
| | | | | CGMA_Par | | 20 |
| Kp_06 | 23 | 10,000 | 9767 | CGMA_Sec | 9767 | 1 |
| | | | | CGMA_Par | | 20 |
| Kp_07 | 5 | 80 | 130 | CGMA_Sec | 130 | 1 |
| | | | | CGMA_Par | | 20 |
| Kp_08 | 20 | 879 | 1025 | CGMA_Sec | 1025 | 1 |
| | | | | CGMA_Par | | 20 |

Los algoritmos secuencial y paralelo resuelven correctamente las 8 instancias. En el algoritmo paralelo se utilizó el valor mínimo de 20 cores para su ejecución.

Otras de las pruebas del CGMA paralelo fueron las 4 instancias de prueba del KP 0-1 publicadas en [16], se anexan los resultados que ellos obtuvieron. Ellos utilizaron un algoritmo basada en inteligencia de enjambre, enfocado en genéticos (GA). La paralelización de ellos se ejecutó en un GPU. A continuación, en la Tabla 6 se observan los resultados.

**Tabla 6** Comparación contra Pospishal et al [16]

| Instancia | Dimensión | Capacidad | Óptimo | Algoritmo | Solución | % Calidad |
|---|---|---|---|---|---|---|
| KP 0-1 | 4 | 100 | 473 | CGMA- Par | 473 | 100 |
| | | | | CGMA- Sec | 473 | 100 |
| | | | | GA | - | 94.83 |
| KP 0-2 | 10 | 100 | 798 | CGMA- Par | 798 | 100 |
| | | | | CGMA- Sec | 798 | 100 |
| | | | | GA | - | 86.67 |
| KP 0-3 | 25 | 300 | 3307 | CGMA- Par | 3307 | 100 |
| | | | | CGMA- Sec | 3307 | 100 |
| | | | | GA | - | 96.64 |
| KP 0-4 | 40 | 600 | 4994 | CGMA- Par | 4940 | 98.92 |
| | | | | CGMA- Sec | 4597 | 92.05 |
| | | | | GA | - | 98.39 |

La calidad de los resultados obtenidos se mide de la siguiente manera [16].

$$Puntaje = 100\frac{\sum_1^n F_a}{nF_b}[\%] \quad (3)$$

Donde:

n = número de ejecuciones

$F_a$ = Mejor solución obtenida

$F_b$ = Solución óptima

En la fórmula 3 se observa el puntaje de calidad del algoritmo a través de las soluciones obtenidas y el número de ejecuciones realizadas. Como se observa en la Tabla 6, el CGMA paralelo en las instancias KP 0-1, KP 0-2, KP 0-3 obtiene el resultado óptimo, lo cual indica una calidad del 100%, en la KP 0-4 no se obtuvo una calidad del 100% pero se logró mejorar el porcentaje de calidad con el 98.92% respecto al porcentaje obtenido por GA del 98.39%. El porcentaje de calidad del CGMA paralelo es mejor comparado con el CGMA secuencial y GA, ya que en las 4 instancias se observan mejores resultados.

Continuando con las pruebas del CGMA paralelo, se tomaron 5 instancias de prueba de la literatura KP 0-1 utilizadas en [27] un algoritmo de murciélago binario Inyectivo- basado en un esquema general (IBBA-RSS), las soluciones son comparadas contra las soluciones del CGMA paralelo, CGMA secuencial y el algoritmo de inteligencia de cohorte (CI) [28]. Los resultados se muestran en la Tabla 7

**Tabla 7.** Comparación contra Rizk-Allah & Hassanien [27]

| Instancia | Dimensión | Capacidad | Óptimo | Algoritmo | Solución | % Calidad |
|---|---|---|---|---|---|---|
| KP-001 | 30 | 577 | 1437 | CGMA- Par | 1393 | 96.94 |
| | | | | CGMA- Sec | 1355 | 94.29 |
| | | | | IBBA-RSS | 1437 | 100 |
| | | | | CI | 1437 | 100 |
| KP-002 | 40 | 819 | 1821 | CGMA- Par | 1741 | 95.61 |
| | | | | CGMA- Sec | 1702 | 93.47 |
| | | | | IBBA-RSS | 1821 | 100 |
| | | | | CI | 1816 | 99.73 |
| KP-003 | 50 | 882 | 2448 | CGMA- Par | 2243 | 91.59 |
| | | | | CGMA- Sec | 2170 | 88.61 |
| | | | | IBBA-RSS | 2448 | 100 |
| | | | | CI | 2440 | 99.67 |
| KP-004 | 60 | 1006 | 2917 | CGMA- Par | 2673 | 91.64 |
| | | | | CGMA- Sec | 2599 | 89.10 |
| | | | | IBBA-RSS | 2917 | 100 |
| | | | | CI | 2917 | 100 |
| KP-005 | 65 | 1319 | 2818 | CGMA- Par | 2564 | 91.12 |
| | | | | CGMA- Sec | 2459 | 87.26 |
| | | | | IBBA-RSS | 2818 | 100 |
| | | | | CI | 2818 | 100 |

En la Tabla 7, se observa que el algoritmo CGMA paralelo obtiene un buen porcentaje de calidad de la solución por encima del 90% en las 5 instancias de prueba utilizadas. El CGMA paralelo resulta ser más eficiente que la versión secuencial ya que logra mejorar las instancias, sin embargo, en comparación con IBBA-RSS y el CI la calidad de la

solución del CGMA paralelo es buena ya que se obtienen porcentajes muy cercanos al 100%.

Los ejemplos resueltos no considran la complejidad de la instancia del PK0-1 propuesta por Pisinger [3], por tal motivo el algoritmo CGMA paralelo no tiene ningún problema para resolverlos. En la sección siguiente se resuelven los problemas considerados complejos.

### 5.2 Solución de las clases No correlacionado, débilmente correlacionado y fuertemente correlacionada

Para comparar los resultados del algoritmo paralelo sus valores se comparan con los obtenidos de la solución exacta *branch and Bound* [29]. En esta sección se presentan los problemas del tamaño que fue posible obtener su solución óptima.

En las tablas que se presentan a continuación se incluye la solución del algoritmo secuencial, con la finalidad de comparar los resultados con ambos algoritmos.

**Tabla 8** Clase de datos no correlacionados CGMA paralelo

| Clase de datos "No correlacionados" | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Instancia | Dimensión | Restricción | Capacidad | Valor óptimo | Algoritmo secuencial vs paralelo | Valor de la solución | % Error | Cores |
| Ncor025_100 | 100 | .25 | 135935 | 242169 | CGMA_Sec | 182842 | 24.50 | 1 |
| | | | | | CGMA_Par | 197881 | 18.29 | 120 |
| Ncor025_200 | 200 | .25 | 259056 | 542054 | CGMA_Sec | 354146 | 34.67 | 1 |
| | | | | | CGMA_Par | 364138 | 32.82 | 80 |
| Ncor050_100 | 100 | .50 | 271871 | 349541 | CGMA_Sec | 289211 | 17.26 | 1 |
| | | | | | CGMA_Par | 298209 | 14.69 | 100 |
| Ncor050_200 | 200 | .50 | 518112 | 774308 | CGMA_Sec | 590978 | 23.68 | 1 |
| | | | | | CGMA_Par | 602370 | 22.21 | 20 |
| Ncor075_100 | 100 | .75 | 407807 | 414081 | CGMA_Sec | 362563 | 12.44 | 1 |
| | | | | | CGMA_Par | 372923 | 9.94 | 20 |
| Ncor075_200 | 200 | .75 | 777168 | 910690 | CGMA_Sec | 767002 | 15.78 | 1 |
| | | | | | CGMA_Par | 774090 | 15.00 | 40 |

El CGMA paralelo mejora significativamente cada uno de los resultados obtenidos por el CGMA secuencial, como se observa en la tabla 8. El número de cores del CGMA paralelo es con el que se obtuvieron los mejores resultados.

En Tabla 9 se presentan los resultados obtenidos de las instancias de la clase de datos débilmente correlacionados, se compara la calidad de la solución obtenida del secuencial y del paralelo, se calcula el porcentaje de error respecto a la solución óptima obtenida por el algoritmo de solución exacta BBBA [29].

**Tabla 9** Clase de datos débilmente correlacionados CGMA paralelo

| Clase de datos "Débilmente correlacionados" | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Instancia | Dimensión | Restricción | Capacidad | Valor Óptimo | Algoritmo | Valor de la Solución Aproximada | % Error | Cores |
| Dcor025_100 | 100 | .25 | 127589 | 227827 | CGMA_Sec | 200297 | 12.08 | 1 |
| | | | | | CGMA_Par | 202074 | 11.30 | 40 |
| Dcor025_200 | 200 | .25 | 245773 | 456928 | CGMA_Sec | 373336 | 18.29 | 1 |
| | | | | | CGMA_Par | 382816 | 16.22 | 20 |
| Dcor025_300 | 300 | .25 | 360358 | 677749 | CGMA_Sec | 534357 | 21.16 | 1 |
| | | | | | CGMA_Par | 539338 | 20.42 | 60 |
| Dcor050_100 | 100 | .50 | 255179 | 405770 | CGMA_Sec | 375673 | 7.42 | 1 |
| | | | | | CGMA_Par | 379150 | 6.56 | 40 |
| Dcor050_200 | 200 | .50 | 491546 | 817624 | CGMA_Sec | 727804 | 10.99 | 1 |
| | | | | | CGMA_Par | 731513 | 10.53 | 80 |
| Dcor050_300 | 300 | .50 | 720717 | 1220404 | CGMA_Sec | 1083384 | 11.23 | 1 |
| | | | | | CGMA_Par | 1087270 | 10.91 | 20 |
| Dcor075_100 | 100 | .75 | 382768 | 557217 | CGMA_Sec | 526051 | 5.59 | 1 |
| | | | | | CGMA_Par | 527820 | 5.28 | 20 |
| Dcor075_200 | 200 | .75 | 737319 | 1120547 | CGMA_Sec | 1030349 | 8.05 | 1 |
| | | | | | CGMA_Par | 1035036 | 7.63 | 80 |
| Dcor075_300 | 300 | .75 | 1081075 | 1528107 | CGMA_Sec | 1528107 | 8.62 | 1 |
| | | | | | CGMA_Par | 1541224 | 7.84 | 60 |

En la Tabla 9 las instancias de la clase de datos débilmente correlacionada son resueltas por el algoritmo en paralelo CGMA. El CGMA paralelo demostró tener una buena eficiencia para la solución de las instancias de la clase de datos débilmente correlacionada, ya que como se observa en los resultados para la restricción de capacidad del .25 se obtiene un porcentaje de error hasta del 11.30%, en restricción del .50 el porcentaje de error es hasta del 6.56%, y con restricción del .75 el porcentaje de error es hasta del 5.28%.

En la Tabla 10 se muestran los resultados obtenidos de las instancias de la clase de datos fuertemente correlacionados, se compara la calidad de la solución obtenida por el algoritmo secuencial y paralelo con restricciones de capacidad del .25, .50 y .75 de la suma de los pesos de los elementos.

**Tabla 10** Comparación con los fuertemente correlacionados

| Clase de datos "Fuertemente correlacionados" | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Instancia | Dimensión | Restricción | Capacidad | Valor Óptimo | Algoritmo | Valor de la Solución Aproximada | % Error | Cores |
| Fcor025_50 | 50 | .25 | 64571 | 89525 | CGMA_Sec | 85267 | 4.76 | 1 |
| | | | | | CGMA_Par | 86166 | 3.75 | 60 |
| Fcor025_100 | 100 | .25 | 119922 | 171920 | CGMA_Sec | 156641 | 8.89 | 1 |
| | | | | | CGMA_Par | 157147 | 8.59 | 40 |
| Fcor050_50 | 50 | .50 | 129143 | 164140 | CGMA_Sec | 160021 | 2.51 | 1 |
| | | | | | CGMA_Par | 160457 | 2.24 | 20 |
| Fcor050_100 | 100 | .50 | 239845 | 311845 | CGMA_Sec | 295858 | 5.13 | 1 |
| | | | | | CGMA_Par | 300014 | 3.79 | 20 |
| Fcor075_50 | 50 | .75 | 193714 | 236713 | CGMA_Sec | 233312 | 1.44 | 1 |
| | | | | | CGMA_Par | 235310 | 0.59 | 140 |
| Fcor075_100 | 100 | .75 | 359767 | 446767 | CGMA_Sec | 439479 | 1.63 | 1 |
| | | | | | CGMA_Par | 440993 | 1.29 | 60 |

Como se indica en la Tabla 10 las instancias de la clase de datos fuertemente correlacionada son resueltas por el CGMA paralelo, como se observa en los resultados con la restricción de capacidad del .25 se obtiene un porcentaje de error hasta del 3.75%, con la capacidades de .50 el porcentaje de error es hasta del 2.24% y restricción del .75 el porcentaje de error es hasta del 0.59%, el número de cores de tabla es con el que se obtuvo el mejor resultado.

Los resultados muestran que conforme se incremente la complejidad del KP 0-1, el CGMA paralelo obtiene soluciones más aproximadas a las soluciones exactas.

### 5.3. Comparación con grandes instancias fuertemente correlacionadas

El problema de resolver estas instancias es que no se puede obtener la solución óptima por un algoritmo exacto, por lo que en este trabajo se recurrió a modificar el algoritmo *Branch and Bound* [29] para obtener una solución aproximada, que es la que se utiliza como referencia. Los resultados de los experimentos llevados a cabo de dimensión hasta 4,000 están en la tabla 11.

**Tabla 11.** Clase de datos fuertemente correlacionados dimensión 4,000 CGMA paralelo

| Clase de datos "Fuertemente correlacionados" | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Instancia | Dimensión | Restricción | Capacidad | Valor Mejor Conocido | Algoritmo Secuencial vs Paralelo | Valor de la Solución Aproximada | % Error | Cores |
| Fcor025_1000 | 1000 | .25 | 123013 | 172808 | CGMA_Sec | 147038 | 14.91 | 1 |
| | | | | | CGMA_Par | 149647 | 13.40 | 60 |
| Fcor050_1000 | 1000 | .50 | 246026 | 316724 | CGMA_Sec | 292890 | 7.53 | 1 |
| | | | | | CGMA_Par | 295086 | 6.83 | 40 |
| Fcor075_1000 | 1000 | .75 | 369039 | 455738 | CGMA_Sec | 444743 | 2.41 | 1 |
| | | | | | CGMA_Par | 446730 | 1.98 | 100 |
| Fcor025_2000 | 2000 | .25 | 242612 | 343812 | CGMA_Sec | 292832 | 14.83 | 1 |
| | | | | | CGMA_Par | 295525 | 14.04 | 20 |
| Fcor050_2000 | 2000 | .50 | 485225 | 627725 | CGMA_Sec | 559135 | 10.93 | 1 |
| | | | | | CGMA_Par | 568630 | 9.41 | 40 |
| Fcor075_2000 | 2000 | .75 | 727838 | 901638 | CGMA_Sec | 880518 | 2.34 | 1 |
| | | | | | CGMA_Par | 881835 | 2.20 | 20 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Fcor025_4000 | 4000 | .25 | 488690 | 690390 | CGMA_Sec | 542495 | 21.42 | 1 |
| | | | | | CGMA_Par | 572579 | 17.06 | 40 |
| Fcor050_4000 | 4000 | .50 | 977380 | 1261780 | CGMA_Sec | 1066730 | 15.46 | 1 |
| | | | | | CGMA_Par | 1084273 | 14.07 | 40 |
| Fcor075_4000 | 4000 | .75 | 1466070 | 1813470 | CGMA_Sec | 1770857 | 2.35 | 1 |
| | | | | | CGMA_Par | 1772303 | 2.27 | 20 |

Los resultados alcanzados por el CGMA paralelo con restricción .25 se obtiene un porcentaje de error de hasta del 13.40%, en restricciones de .50 el porcentaje de error de hasta del 6.83% y para restricciones del .75 el porcentaje de error es de hasta el 1.98%, en donde el número de cores utilizados permitieron encontrar la solución y mejorar el resultado anteriormente obtenido por el algoritmo secuencial.

A continuación, en la Tabla 12 se muestran los experimentos llevados a cabo con hasta 10,000 elementos de la clase fuertemente correlacionada.

**Tabla 12** Clase de datos fuertemente correlacionados dimensión 10,000 CGMA paralelo

| Clase de datos "Fuertemente correlacionados" | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instancia | Dimensión | Restricción | Capacidad | Valor mejor conocido | Algoritmo Secuencial vs Paralelo | Valor de la Solución Aproximada | % Error | Tiempo de ejecución (s) | Cores |
| Fcor025_6000 | 6000 | .25 | 742016 | 1042416 | CGMA_Sec | 842585 | 28.82 | 16.59 | 1 |
| | | | | | CGMA_Par | 865025 | 17.02 | 75.33 | 40 |
| Fcor050_6000 | 6000 | .50 | 1484032 | 1909032 | CGMA_Sec | 1574551 | 17.52 | 10.66 | 1 |
| | | | | | CGMA_Par | 1581905 | 17.14 | 50.77 | 60 |
| Fcor075_6000 | 6000 | .75 | 2226048 | 2746148 | CGMA_Sec | 1511807 | 44.95 | 14.69 | 1 |
| | | | | | CGMA_Par | 1579530 | 42.48 | 29.93 | 20 |
| Fcor025_9000 | 9000 | .25 | 1122611 | 1571711 | CGMA_Sec | 1253019 | 20.28 | 58.47 | 1 |
| | | | | | CGMA_Par | 1283241 | 18.35 | 110.70 | 40 |
| Fcor050_9000 | 9000 | .50 | 2245223 | 2881523 | CGMA_Sec | 2322557 | 19.40 | 59.01 | 1 |
| | | | | | CGMA_Par | 2341428 | 18.74 | 71.05 | 20 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Fcor075_9000 | 9000 | .75 | 3367834 | 4147134 | CGMA_Sec | 2274185 | 45.16 | 29.80 | 1 |
| | | | | | CGMA_Par | 2333641 | 43.73 | 114.09 | 20 |
| Fcor025_10000 | 10,000 | .25 | 1245730 | 1744730 | CGMA_Sec | 1415691 | 18.86 | 44.96 | 1 |
| | | | | | CGMA_Par | 1453716 | 16.68 | 182.99 | 40 |
| Fcor050_10000 | 10,000 | .50 | 2491460 | 3198460 | CGMA_Sec | 2556124 | 20.08 | 48.70 | 1 |
| | | | | | CGMA_Par | 2588948 | 19.06 | 80.82 | 60 |
| Fcor075_10000 | 10,000 | .75 | 3737190 | 4603190 | CGMA_Sec | 2568934 | 44.19 | 35.72 | 1 |
| | | | | | CGMA_Par | 2585000 | 43.84 | 120.82 | 40 |

En la Tabla 12 el CGMA paralelo presenta mejor calidad respecto al CGMA secuencial en la solución de las instancias fuertemente correlacionadas del KP 0-1. Por otra parte, se observa que el tiempo de ejecución del algoritmo paralelo resulta ser mayor que el secuencial, lo anterior es a causa de que los dos resuelven el mismo algoritmo, sólo que en el algoritmo paralelo se resuelven el número de veces igual al número de cores utilizados y en el secuencial una sola vez. Por ejemplo, para la instancia Fcor075_10000 en algoritmo secuencial requirió de 35.72 segundos, si este algoritmo se repite 40 veces para cubrir el mismo espacio de búsqueda que el paralelo, se requerirá de 35.72*40 = 1428.8 segundos. Entonces se observa que el algoritmo paralelo es más rápido para cubrir el mismo espacio de búsqueda, por lo que *Speedup* paralelo de Sp = (T1/Tp) = (1428.8/120.82) = 11.82. Si se hacen las mismas consideraciones con los demás resultados se obtiene la tabla siguiente:

**Tabla 13** Sp de los valores de la tabla 12

| Número de elementos | 0.25 | 0.50 | 0.75 |
|---|---|---|---|
| 6,000 | 8.81 | 12.6 | 9.81 |
| 9,000 | 21.12 | 16.61 | 5.22 |
| 10,000 | 9.83 | 36.15 | 11.82 |

Como se muestra en la tabla 13 no existe una tendencia en el Sp que nos indique cual es el valor alcanzado para el algoritmo CGMC secuencial, lo que si se puede decir que es hasta 36.15 veces más rápido que el secuencial.

Con respecto a la precisión de la solución es claro que los resultados nos indican que se tienen que probar otras combinaciones de variables del proceso para mejorar la solución. Aunque es importante notar que la comparación es con la solución aproximada y no la exacta. Ya que como se mostró en los ejemplos previos la solución del CGMC paralelo fue muy aproximada a la solución óptima.

## 6. Conclusiones

Con base a los resultados obtenidos se concluye:

Para la solución del problema de la mochila KP 0-1 (0-1 *Knapsack Problem*), es posible resolver las instancias y mejorar los resultados de una metaheurística secuencial con un enfoque paralelo.

La mejora de la propuesta de ajustar las variables es muy versátil y eficiente, dado que se obtienen mejores resultados para resolver el problema de la mochila 0-1, ya que se puede dar solución a diferentes capacidades de la mochila. Lo cual amplía el campo de resolución de la metaheurística de inteligencia de enjambre inspirada en el comportamiento de montañismo de los monos.

Las pruebas experimentales muestran qué al llevar a cabo el procesamiento paralelo, utilizando hasta 140 unidades procesadoras, se mejora la calidad de la solución obtenida con el algoritmo mono mejorado secuencial. Esto se debe a que se produce mayor diversidad de la población y una mejor exploración del espacio de búsqueda para hallar la solución óptima. Al llevar a cabo una mayor exploración del espacio de búsqueda y al tener una mayor población debido a la ejecución simultánea las unidades procesadoras, la velocidad con la que se obtiene la solución es más rápida respecto a la manera en que se realiza el procedimiento secuencial.

El algoritmo mono mejorado con paralelo puede ser una alternativa eficiente y eficaz para resolver el problema de la mochila 0-1.

**Referencias**


[1] Dantzig G. B. Discrete-Variable Extremum Problems. *Operations Research*. 1957, 5(2), 266-277.

[2] Bretthauer K. M., & Shetty B., The nonlinear knapsack problem-algorithms and applications. *European Journal of Operational Research*, 2002. 138(3) 459-472.

[3] Pisinger D., Where are the hard knapsack problems? *Computers & Operations Research*, 2005, 32(9), 2271-2284.

[4] Hristakeva M., & Shrestha D., Different approaches to solve the 0/1 knapsack problem. In *The Midwest Instruction and Computing Symposium*. 2005

[5] Kellerer H., Pferschy U., & Pisinger D., *Knapsack Problems. Springer, Berlin. 2004*, ISBN 978-3-540-24777-7.

[6] Zhang, J., Comparative study of several intelligent algorithms for knapsack problem. *Procedia Environmental Sciences*, 2011, 11, 163-168

[7] Alfonso H., Salto C., Minetti G. F., Stark, N., Bermúdez, C., Orellana, A., & Sanz Troiani, F. (2010). *Metaheurísticas aplicadas a problemas de optimización. In XII Workshop de Investigadores en Ciencias de la Computación*. 2010, 72-76

[8] Alba E., Luque G., & Nesmachnow S., Metaheurísticas paralelas: avances recientes y nuevas tendencias. *Transacciones internacionales en investigación operativa*, 2013, 20(1), 1-48.

[9] Kaur MD., Revisión de diferentes técnicas metaheurísticas para computación paralela. *Revista de investigación avanzada en computación en la nube, virtualización y aplicaciones web*, 2018, 1(2), 28-32.

[10] Zhou Y., Chen X., & Zhou G., An improved monkey algorithm for a 0-1 knapsack problem. *Applied Soft Computing*, 2016, 38, 817-830.

[11] Yang X. S., & Karamanoglu M., Swarm intelligence and bio-inspired computation: an overview. *In Swarm intelligence and bio-inspired computation, Elsevier*, 2013, 3-23.

[12] Chakraborty A., & Kar A. K., *Swarm intelligence: A review of algorithms. In Nature-Inspired Computing and Optimization. Springer,* 2017, 475-494.

[13] Adi S., & Aldasht M., Parallel Evolutionary Algorithms for Feature Selection in High Dimensional Datasets. *International Journal of Computer Science and Information Security* (IJCSIS), 2018, 16(3).

[14] Cotta C., Talbi EG., y Alba E., *Metaheurísticas híbridas paralelas. Metaheurísticas paralelas: una nueva clase de algoritmos*, 2005 Vol.47, No.347.

[15] Talbi E. G., A taxonomy of hybrid metaheuristics. *Journal of heuristics*, 2002, 8(5), 541-564.

[16] Pospichal P., Schwarz J., y Jaros J., Algoritmo genético paralelo que resuelve 0/1 problema de mochila que se ejecuta en el gpu. *En la 16ª Conferencia Internacional sobre Soft Computing MENDEL*, 2010, 64-70.

[17] Hajarian M., Shahbahrami A., & Hoseini F., A parallel solution for the 0–1 knapsack problem using firefly algorithm. *Conference on Swarm Intelligence and Evolutionary Computation (CSIEC*), 2016 pp. 25-30.

[18] Sonuc E., Sen B., & Bayir S., A parallel approach for solving 0/1 knapsack problem using simulated annealing algorithm on CUDA platform. *International Journal of Computer Science and Information Security*, 2016, 14(12).

[19] El-Shafei M., Ahmad I., & Alfailakawi M. G., Hardware accelerator for solving 0–1 knapsack problems using binary harmony search. *International Journal of Parallel, Emergent and Distributed Systems*, 2018, 33(1), 87-102.

[20] Zhao R. Q., & Tang W. S., Monkey algorithm for global numerical optimization. *Journal of Uncertain Systems*, 2008, 2(3), 165-176.

[21] Yi T. H., Li H. N., & Zhang X. D., A modified monkey algorithm for optimal sensor placement in structural health monitoring. *Smart Materials and Structures*, 2012, 21(10),

[22] Zheng L., An improved monkey algorithm with dynamic adaptation. *Applied Mathematics and Computation*, 2013, 222, 645-657.

[23] Yi T. H., Li H. N., Song G., & Zhang X. D., Optimal sensor placement for health monitoring of high-rise structure using adaptive monkey algorithm. *Structural Control and Health Monitoring*, 2015, 22(4), 667-681

[24] Zou D., Gao L., Li S., & Wu J., Solving 0–1 knapsack problem by a novel global harmony search algorithm. *Applied Soft Computing*, 2011, 11(2), 1556-1564.

[25] Feng Y., Wang G. G., Feng Q., & Zhao X. J., An effective hybrid cuckoo search algorithm with improved shuffled frog leaping algorithm for 0-1 knapsack problems. *Computational intelligence and neuroscience*, 2014, pp. 36

[26] Feng Y., Wang G. G., & Gao X. Z., A novel hybrid cuckoo search algorithm with global harmony search for 0-1 Knapsack problems. *International Journal of Computational Intelligence Systems*, 2016, 9(6), 1174-1190.

[27] Rizk-Allah R. M., & Hassanien A. E., New binary bat algorithm for solving 0-1 knapsack problem. *Complex & Intelligent Systems*, 2018, 4(1), 31-53.

[28] Kulkarni AJ., y Shabir H., Resolver un problema de mochila 0-1 usando el algoritmo de inteligencia de cohorte. *Revista internacional de aprendizaje automático y cibernética*, 2016, 7(3), 427-441.

[29] Zavala Díaz J. C., *Optimización con cómputo paralelo*, México DF. México Ed. AM EDITORES., 2013. ISBN 9786074372267

Dr. José Crispín Zavala Díaz. Obtuvo el grado de Doctor en Ciencias Computacionales por el Instituto Tecnológico y de Estudios Superiores de Monterrey. Actualmente tiene la distinción de Investigador Nacional Nivel I del SNI. Está adscrito a la Facultad de Contaduría, Administración e Informática de la Universidad Autónoma del Estado de Morelos, desde 1990 a la fecha. Sus áreas de interés en investigación son optimización combinatoria, cómputo paralelo y algorítmica.

Dr. Joaquín Pérez Ortega. Obtuvo los grados de Doctorado y Maestría en Ciencias Computacionales por el Instituto Tecnológico y de Estudios Superiores de Monterrey, y el de Ingeniero Industrial en Producción por el Instituto Tecnológico de León. Actualmente tiene la distinción de Investigador Nacional Nivel II del SNI. Está adscrito al Departamento de Ciencias Computacionales del Centro Nacional de investigación y Desarrollo Tecnológico (CENIDET), desde 1989 a la fecha. Sus áreas de interés en investigación son Ciencia de Datos, Minería de Datos, Algoritmia, Optimización y Modelación Matemática. A llevado a buen fin la dirección de más de doce tesis de doctorado y más de cuarenta de maestría. Algunas de las tesis que ha dirigido han obtenido premios nacionales e internacionales. Cuenta con un artículo reconocido como Best Paper en un congreso internacional especializado y de gran prestigio.

Dra. Nelva Nely Almanza Ortega. Profesora adscrita al Departamento de División de Estudios de Posgrado e Investigación del IT de Tlalnepantla (ITTLA) desde abril de 2019. Doctorado y Maestría en Ciencias de la Computación otorgados por el Centro Nacional de investigación y Desarrollo Tecnológico (CENIDET). Miembro del Sistema Nacional de Investigadores (SNI), Nivel C. Las áreas de interés en investigación son Ciencia de Datos, Minería de Datos, Análisis y Desarrollo de Algoritmos Heurísticos.

MOCA Jaqueline López Calderón. Obtuvo el grado de Maestra en Optimización y Cómputo Aplicado por la Facultad de Contaduría, Administración e Informática de la Universidad Autónoma del Estado de Morelos en el 2019. Sus áreas de interés algorítmica y cómputo paralelo.