PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

RESOLUCIÓN DEL PROBLEMA MANUFACTURING CELL DESIGN APLICANDO LA METAHEURÍSTICA CUCKOO SEARCH

ANA ELIZABETH JAIME BERNAL
MAYKOL ANDRES RAMÍREZ GONZÁLEZ

Profesor Guía : Ricardo Soto de Giorgis
Co-referente : Boris Almonacid Gutiérrez

INFORME FINAL PROYECTO DE TITULO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERIA DE EJECUCIÓN EN INFORMÁTICA

OCTUBRE, 2016

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

RESOLUCIÓN DEL PROBLEMA MANUFACTURING CELL DESIGN APLICANDO LA METAHEURÍSTICA CUCKOO SEARCH

ANA ELIZABETH JAIME BERNAL
MAYKOL ANDRES RAMÍREZ GONZÁLEZ

Profesor Guía : **Ricardo Soto de Giorgis**
Co-referente : **Boris Almonacid Gutiérrez**

Carrera: **Ingeniería de Ejecución en Informática**

OCTUBRE, 2016

Dedicatoria

Dedico este proyecto de título a mis padres quienes siempre me apoyaron en esta etapa. A mis seres queridos y amigos que me dieron su apoyo, alegría y con quienes compartí muchos momentos. A nuestro profesor guía y co-referente quienes siempre creyeron en nuestras capacidades, nos impulsaron a más y nos brindaron ayuda cuando fue necesario.

Ana Jaime Bernal.

Quiero agradecer a mi familia por entregarme su apoyo incondicional en los estudios, por estar siempre en los momentos mas difíciles de mi vida como estudiante. A mis amigos y compañeros con quienes compartí las dificultades y alegrías de esta carrera. Agradezco también a todos mis profesores ya que ellos me enseñaron a valorar los estudios y a superarme cada día.

Maykol Ramírez González.

Resumen

El propósito de este proyecto, es modelar y resolver el problema de diseño de celdas de manufactura (MCDP). Este problema propone la división de una planta de producción en celdas para organizar y agrupar las máquinas y piezas que se procesan dentro de ella. Esta división se hace para crear un diseño óptimo de producción con el objetivo de aumentar la productividad, minimizando el intercambio de material entre las celdas. Se estudió la metaheurística Cuckoo Search para intentar encontrar una solución óptima a las instancias del MCDP.

Palabras Clave: Manufacturing Cell Design Problem, optimización, Cuckoo Search, metaheurística.

Abstract

The purpose of this project is to model and solve the Manufacturing Cell Design Problem (MCDP). This problem proposes the division of an industrial production plant into cells to organize and to group the machines and parts that are being processed inside of it. This division is made to create an optimal production design, in order to increase the productivity, minimizing the exchange of material (parts) between the cells. The metaheuristic Cuckoo Search was studied to try to find an optimal solution to the instances of the MCDP.

Key Words: Manufacturing Cell Design Problem, optimization, Cuckoo Search, metaheuristic.

Índice

Lista de Figuras

Lista de Tablas

1. Introducción

Este informe se basa en la resolución del problema llamado Manufacturing Cell Design (MCDP) utilizando la metaheurística Cuckoo Search (CS). Fundamentalmente se buscará un método con el cual se pueda adaptar el CS a la presentación matricial en que se desarrolla el MCDP. Teniendo en cuenta que actualmente en el ámbito empresarial, específicamente sobre términos de desarrollo y creación, ha existido una constante búsqueda de mayor rendimiento, se plantea el MCDP con el fin de reducir los costos al mínimo y aumentar los beneficios. El MCDP se define como un problema que representa los diferentes elementos y procedimientos que forman parte de una planta manufacturera en matrices. Lo que se plantea es distribuir las máquinas manufactureras en celdas y las partes del producto en familias, con el fin de obtener una mayor productividad y menor tiempo de procesamiento al hacer que cada familia de partes deba viajar la menor cantidad posible de veces entre las celdas. En otras palabras, el objetivo es encontrar un camino óptimo para procesar un producto.

En este proyecto se verá el estudio de la metaheurística Cuckoo Search. La expresión "metaheurística" deriva del griego "meta", que se refiere a ir más allá, y "heurística", que significa encontrar. Por definición, es un método de búsqueda para resolver un problema computacional. Se basan en casos poco convencionales, como el comportamiento de un enjambre y procesos naturales. Dentro de este ámbito se encuentra Cuckoo Search, la cual se inspira en la estrategia de reproducción parasitaria de una especie de ave llamada Cuco. Ésta se caracteriza porque no construye nidos, sino que utiliza los de otras aves para dejar sus huevos y que sean criados por ellas. Cuckoo Search es una metaheuristica eficiente y de rápida convergencia con la cual se da solución al MCDP. En el presente informe, se dan a conocer los resultados de ésta técnica aplicada a 90 instancias del MCDP [2].

Este informe está estructurado de la siguiente manera: El capítulo 2 y 3, respectivamente, presentan los objetivos y una revisión actualizada de avances dentro de la temática. El capítulo 4 presenta el problema y su modelo matemático. En el capítulo 5 se da a conocer la metaheruística Cuckoo Search. Su integración con el problema se describe en el capítulo 6. La descripción de experimentos llevados a cabo y su análisis pertinente se presentan en el capítulo 7. Finalmente las conclusiones y el trabajo futuro se muestran en el capítulo 8.

2. Definición de Objetivos

2.1. Objetivo General

- El objetivo principal de este proyecto es resolver el MCDP mediante la utilización del algoritmo de optimización Cuckoo Search.

2.2. Objetivos Específicos

- Comprender el MCDP.
- Comprender la metaheurística Cuckoo Search.
- Implementar el algoritmo Cuckoo Search para la resolución del MCDP.
- Aplicar el concepto de paralelismo de diferentes formas en el algoritmo implementado.
- Experimentar y evaluar resultados con respecto a la aplicación de este algoritmo en diferentes instancias de MCDP.

3. Estado Del Arte

Esta investigación se enfoca en el estudio de la resolución del Manufacturing Cell Design Problem. Las investigaciones sobre este tema se pueden dividir en dos grupos; métodos de aproximación y métodos de optimización global. En los métodos de aproximación se encuentran las metaheurísticas y la gran mayoría tiende a explotar la diversificación de éstas, es decir, intentar recorrer una mayor cantidad de áreas con el fin de encontrar soluciones, las cuales no siempre logran el óptimo. Por el contrario, las técnicas de optimización global, se dedican a analizar detenidamente todo el espacio con el fin de encontrar el óptimo del problema, sin embargo, gastan una gran cantidad de trabajo, memoria y tiempo.

En el primer grupo de las metaheurísticas usadas para resolver el MCDP, se encuentran: Aljaber, Baek, Chen [1] y Lozano, Díaz, Eguía, Onieva [7] con el uso del Tabu Search. Wu, Chang, Chung [24] utilizando Simulated Annealing (SA), Durán, Rodriguez y Consalter [3] combinan Particle Swarm Optimization con Data Mining, y Venugopal, Narendran [23] proponen usar Genetics Algorithms (GA), más adelante Gupta, Gupta, Kumar, Sundaram [4] deciden utilizar la misma solución representada con los GA pero con una optimización diferente multi-objetivo la cual se dirige a la minimización simultánea del total del número entre células con variación de carga celular. Consolidando este grupo tenemos diversas investigaciones para resolver el MCDP como Artificial Fish Swarm Algorithm (AFSA) [21], Migrating Birds Optimization Algorithm (MBO) [16, 17], Shuffled Frog Leaping Algorithm (SFLA) [20], Invasive Weed Optimization (IWO) [19], Bat Algoritm (BAT) [22] y Dolphin Echolocation Algorithm (DEA) [18].

En el segundo grupo se encuentra el uso de programación lineal por parte de Purcheck [12], Olivia-Lopez y Purcheck [11], modelos cuadráticos propuestos por Boctor [2], Kusiak y Chow [6]. La programación por metas denominadas Goal Programming (GP) es otro paradigma de este grupo cuya función es la toma de decisiones con objetivos múltiples, Sankaran [14], Shafter y Roger [15] propusieron modelos sobre este paradigma.

En esta investigación se utilizó la metaheurística Cuckoo Search (CS), creada por Xin-She Yang [25] en la resolución del MCDP, que para nuestro conocimiento no ha sido investigado aún.

4. Manufacturing Cell Design Problem

La estrategia de producción con celdas de manufactura (Manufacturing Cell Design Problem, MCDP) consiste en organizar una planta de producción en un conjunto de celdas. En donde cada celda, contiene un grupo de máquinas las cuales se encargan de procesar un conjunto de piezas o partes similares. Estos grupos de piezas forman familias de partes, las cuales son determinadas de acuerdo a las similitudes entre ellas. Las similitudes pueden ser de tipo geométrico, por peso, por tipo de materiales de fabricación, de acuerdo a los volúmenes de fabricación, el número de operaciones necesarias, entre otras.

El MCDP tiene por objetivo determinar una disposición y organización óptima de una planta de producción. Para ello, se reorganizan las piezas dentro de una celda, minimizando el flujo entre las celdas, lo que permite reducir tiempos, costos de producción, utilización de máquinas e incrementa la productividad. Este objetivo es logrado si se generan celdas que garanticen la fabricación completa de los productos asignados y en caso de que esto no sea posible, habría piezas que deberán visitar diferentes celdas en su fabricación [26].

Para lograr la reorganización del sistema de producción es necesario conocer las rutas de fabricación de cada una de las piezas, lo cual permite determinar las máquinas visitadas por cada una de las partes producidas en la ruta de fabricación. Para representar esta información, se utiliza una matriz binaria llamada matriz parte-máquina. Esta matriz posee una representación binaria de unos o ceros, la cual indica cuales máquinas son necesarias para la fabricación de cada una de las piezas. Esta matriz representa las máquinas en las filas y las piezas en las columnas. En ningún momento se entrega información referente a la secuencia de las operaciones o el número de máquinas existentes de cada tipo en las instalaciones de manufactura [5].

En la tabla 1 se muestra un ejemplo de matriz parte-máquina, en la cual, si la componente de la coordenada (i, j) es 0, significa que la máquina i no procesa la pieza j, de lo contrario, si la componente de la coordenada (i, j) es 1, la máquina i si procesa la pieza j [5].

A partir de la tabla 1 se determina una agrupación de piezas y máquinas en celdas. El propósito de MCDP es organizar la estructura de las familias de manera que el movimiento de piezas entre las celdas ocurra el mínimo de

Máquinas	Partes									
	1	2	3	4	5	6	7	8	9	10
A	1	0	0	0	0	0	0	0	0	1
B	0	0	0	0	1	0	1	0	1	0
C	0	0	0	0	0	1	0	1	0	0
D	0	1	0	0	0	0	0	0	0	1
E	0	0	0	0	1	0	1	0	1	0
F	0	1	0	0	0	0	0	0	0	1
G	0	0	1	1	0	1	0	1	0	0
H	1	1	0	0	0	0	0	0	0	1
I	0	0	0	0	1	0	1	0	1	0
J	0	0	0	0	0	0	0	0	1	0

Tabla 1: Ejemplo de matriz inicial parte-máquina

veces posible (ver tabla 2). Para esto, se hace necesario generar un procedimiento lógico el cual permita transformar la matriz inicial parte-máquina de la tabla 1 en la matriz parte-máquina de la tabla 2 [9]. En la matriz final se observa que la diagonal principal se encuentra formada mayormente por 1 y los otros sectores están formados por 0. Es así como se conforman las celdas o grupos, y en éste caso tenemos 3 celdas que son independientes entre sí.

Máquinas	Partes									
	9	5	7	10	1	2	6	8	4	3
I	**1**	**1**	**1**	0	0	0	0	0	0	0
E	**1**	**1**	**1**	0	0	0	0	0	0	0
B	**1**	**1**	**1**	0	0	0	0	0	0	0
J	**1**	**0**	**0**	0	0	0	0	0	0	0
A	0	0	0	**1**	**1**	**0**	0	0	0	0
F	0	0	0	**1**	**0**	**1**	0	0	0	0
H	0	0	0	**1**	**1**	**1**	0	0	0	0
D	0	0	0	**1**	**0**	**1**	0	0	0	0
G	0	0	0	0	0	0	**1**	**1**	**1**	**1**
C	0	0	0	0	0	0	**1**	**1**	**0**	**0**

Tabla 2: Ejemplo de matriz final parte-máquina

4.1. Representación de Manufacturing Cell Design Problem

El MCDP se concibe como un modelo matemático compuesto por variables, dominios, función objetivo y restricciones que deben satisfacerse para encontrar una organización de celdas que sea óptima en cuanto a costos de producción. Dicha estructura a buscar debe tener como característica principal reducir al mínimo los movimientos de piezas entre grupos [3]. El modelo matemático del MCDP fue propuesto por Boctor y se encuentra representado por los siguiente elementos:

Parámetros

- P, representa el número de piezas (o partes) a ser producidas.
- M, representa el número de máquinas.
- C, representa el número de celdas del modelo.
- M_{max}, corresponde a la cantidad máxima de máquinas que permite una celda.
- $A = [a_{ij}]$, matriz binaria máquina–pieza con dominio $[0, 1]$ y dimensión $M \times P$.

$$a_{ij} \begin{cases} 1 & \text{si la máquina } i \text{ procesa la pieza } j \\ 0 & \text{en otro caso} \end{cases} \tag{1}$$

Variables:

- $B = [y_{ik}]$, matriz binaria máquina–celda con dominio $[0, 1]$ y dimensión $M \times C$.

$$y_{ik} \begin{cases} 1 & \text{si la máquina i pertenece a la celda k} \\ 0 & \text{en otro caso} \end{cases} \tag{2}$$

- $C = [z_{jk}]$, matriz binaria pieza–celda con dominio $[0, 1]$ y dimensión $P \times C$.

$$z_{jk} \begin{cases} 1 & \text{si la pieza j pertenece a la celda k} \\ 0 & \text{en otro caso} \end{cases} \tag{3}$$

Índices:

- i, corresponde al índice de las máquinas $(i = 1, ..., M)$.

- j, corresponde al índice de las piezas ($j = 1, ..., P$).
- k, corresponde al índice de las celdas ($k = 1, ..., C$).

La función objetivo del MCDP, es una función de minimización, en donde lo que se busca es reducir la cantidad de veces que una pieza deba hacer movimientos interceldarios para su procesamiento. Para la representación de ésta se utilizan las variables y constantes declaradas con anterioridad. A continuación tenemos la representación matemática del problema:

$$Min : \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} a_{ij} z_{jk} (1 - y_{ik})$$

Este modelo de MCDP tiene algunas restricciones que son presentadas a continuación:

- Restricción 1: Una máquina pertenece a una celda.

$$\sum_{k=1}^{C} y_{ik} = 1, \quad \forall i (i = 1, ..., M)$$

- Restricción 2: Una pieza pertenece a una celda.

$$\sum_{k=1}^{C} z_{jk} = 1, \quad \forall j (j = 1, ..., P)$$

- Restricción 3: El número de máquinas no debe de exceder el máximo de máquinas que se permite en una celda.

$$\sum_{i=1}^{M} y_{ik} \leq M_{max}, \quad \forall k (k = 1, ..., C)$$

5. Cuckoo Search

El comportamiento de la especie de ave llamada Cuckoo al momento de reproducirse, sirve como inspiración para el algoritmo metaheurístico Cuckoo Search creado por Yang y Deb en 2010 [25]. Esta práctica se caracteriza principalmente por ser una forma agresiva de reproducción, debido a que las aves Cuckoo no construyen nidos, sino que utilizan los nidos de otras aves (normalmente de otras especies) para dejar sus huevos y que éstos sean criados por ellas. Para llevar a cabo este comportamiento, los Cuckoo primero se ponen en búsqueda de un lugar de acogida potencial, el cual sería el nido de otra ave. Cuando es encontrado, el Cuckoo pone un huevo en dicho nido para luego desplazarse nuevamente hacia otros nidos donde incubar más crías. A veces este objetivo se cumple satisfactoriamente y los huevos dejados por el Cuckoo pueden ser criados por la otra ave hasta el momento de eclosionar y luego de eso pueden incluso ser alimentadas por ella. Existen ciertos Cuckoos que son capaces de producir huevos que se mimetizan en color y forma de los huevos del ave de acogida, para que así éstos no sean descubiertos o abandonados. También tienen la característica de que al momento de eclosionar, la cría recién nacida intentará instintivamente botar del nido a los huevos del ave de acogida e incluso imitará el canto de la otra especie de ave para aumentar la probabilidad de ser alimentada. Por otro lado, existen ocasiones en las cuales estos huevos son descubiertos por el ave dueña del nido, lo que hará que ésta intente eliminarlos o abandonarlos, construyendo un nuevo nido en otra parte.

El comportamiento descrito en el párrafo anterior, ha sido aplicado a problemas de optimización y de búsqueda de soluciones. Para lograr esto y describir de forma más simple el algoritmo utilizado, se siguen tres reglas básicas:

1. Cada Cuckoo pone un huevo a la vez, y será en un nido que elige aleatoriamente.
2. Los mejores nidos (soluciones), se mantienen para las siguientes generaciones.
3. El número de nidos de acogida es fijo, y el huevo del Cuckoo es descubierto con una probabilidad de Pa perteneciente a un intervalo de $[0, 1]$. En caso que ocurra, el ave botará el huevo o abandonará su nido (se desechará la solución) construyendo uno nuevo (nueva solución).

También se debe tomar en consideración, que los Cuckoos para encontrar nuevos nidos, utilizan un algoritmo especial llamado Vuelos de Lévy o Lévy Flights (ver sección 5.2). El vuelo de Lévy simula el comportamiento animal del movimiento de varios animales e insectos. La mayor característica del vuelo de Lévy es su trayectoria aleatoria, con cierta restricción de lo que recorre. Sin embargo, gracias a esta característica se logra que el conjunto de soluciones pueda ser recorrido de una manera mucho más amplia para hallar una solución óptima, o cercana a ésta.

5.1. Algoritmo Cuckoo Search

Los pasos básicos del algoritmo Cuckoo Search pueden ser mostrados en el siguiente pseudocódigo:

Algorithm 1 Cuckoo Search

1: Función Objetivo $Min : \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} a_{ij} z_{jk} (1 - y_{ik})$
2: Generar la población inicial de n nidos de acogida $x_i (i = 1, 2, ..., n)$
3: **while** ($t <$ máximo número de generaciones), (óptimo encontrado) o (algún criterio definido de término) **do**
4: Obtener un cuckoo al azar por vuelos Levy, evaluar su calidad Fi
5: Elige un nido entre n (por ejemplo, j) al azar
6: **if** $Fi > Fj$ **then**
7: Reemplazar j por la nueva solución;
8: **end if**
9: Se ordenan los nidos según su calidad de Fi de forma ascendente;
10: Una fracción (Pa) de peores nidos son abandonados;
11: Nuevos nidos se construyen (nuevas soluciones);
12: **end while**
13: Resultados post proceso y visualización;

5.2. Lévy Flights

El vuelo de Lévy es un tipo de combinación de pasos aleatorios (random walk) con un patrón que mezcla trayectorias de saltos cortos que se alternan con saltos lagos siendo movimientos al azar y que utiliza un tipo de distribución de probabilidad (ley de potencia) para trazar el camino.

Para entender el patrón matemático utilizado por el vuelo de Lévy hay que imaginar los comportamientos de algunos animales para la búsqueda de alimento en los que se ha demostrado que poseen características similares. Un ejemplo se puede encontrar en el estudio de Reynolds y Frye [13] que muestra que la mosca de la fruta, explora su paisaje con una serie de vuelo recto y caminos interrumpidos por giros repentinos de 90^o, lo que lleva a un estilo de vuelo de Lévy en patrones intermitentes de búsqueda a escala libre. El vuelo de Lévy proporciona esencialmente un proceso de caminata aleatoria cuya longitud del paso al azar se extrae de una distribución de Lévy, que tiene una varianza infinita con una media infinita.

Se estudió el algoritmo de Mantegna [8] para poder formular los saltos aleatorios, en el cual tenemos:

$$step = \frac{u}{(|v|)^{1/\beta}} \tag{4}$$

Donde β es un valor entre 1 y 2, mientras que u y v son obtenidos de una distribución normal , Es decir:

$$u \sim N(0, \sigma_u^2), \quad v \sim N(0, \sigma_v^2) \tag{5}$$

Donde:

$$\sigma_u = [\frac{\Gamma(1+\beta)\ sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]*\beta 2^{(\beta-1)/2}}]^{\frac{1}{\beta}}, \quad \sigma_v = 1 \tag{6}$$

El step es el valor que se obtiene finalmente mediante el vuelo de Lévy.

6. Cuckoo Search aplicado para resolver el Manufacturing Cell Design Problem

Para llevar a cabo este proyecto se utilizó el entorno de desarrollo integrado o IDE Eclipse Mars 4.5, bajo el lenguaje de programación Java 1.8. Las principales razones de esta decisión son la comodidad que presenta el lenguaje, a lo que se le suma el desarrollo bajo el paradigma de la orientación a objetos, el que ayuda a presentar el problema de forma clara y ordenada haciendo bastante legible y versátil el código. Además, se dispone de las bibliotecas que son de ayuda al momento de programar y trabajar con clases y objetos Java.

La implementación de la metaheurística Cuckoo Search aplicada al MCDP, consiste en primer lugar en generar una población inicial de nidos para CS. Básicamente, se crea un arreglo que representa el conjunto inicial de soluciones para la metaheurística. Su tamaño, está dado por la cantidad fija de nidos que existen inicialmente. Cabe aclarar, que para esta parte se definió que cada nido generado contiene en su interior 1 solo huevo, es decir, que la cantidad de soluciones iniciales, es igual a la cantidad de nidos.

Para generar cada solución inicial, se utiliza la matriz A y las constantes de cada instancia del MCDP. Los elementos que componen una solución son, la matriz A, una matriz Y y una matriz Z (ver Fig. 1). Se comienza llenando la matriz binaria Y máquina-celda aleatoriamente, asignando cada una de las máquinas a una celda. Después, utilizando el orden de la matriz Y y realizando un procedimiento lógico, se logra formar la matriz Z. La matriz generada no siempre es factible, por lo que se envía a una función de reparación para obtener una matriz Z viable. Es así, como se obtienen los elementos necesarios para representar una de las soluciones iniciales. Finalmente, se crea un arreglo que contiene en su interior varios objetos que representan cada una de las soluciones, teniendo así, una población inicial generada aleatoriamente.

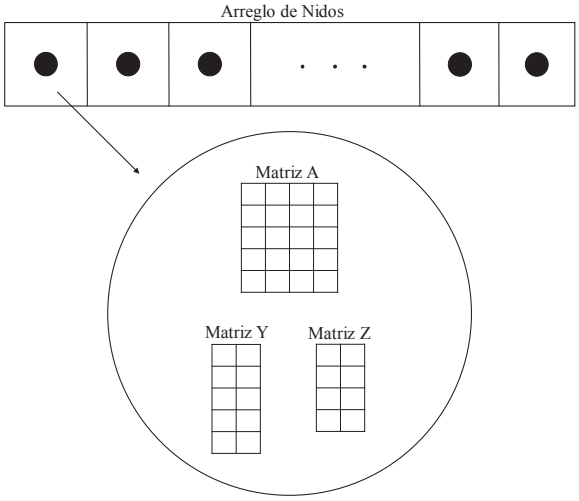

Figura 1: Integración de Cuckoo Search con MCDP.

Para generar una nueva solución via el vuelo de Lévy, primero se toma aleatoriamente una solución de la población inicial. Calculamos el valor del *step* que estará en el rango $[0, M]$ si se modificará una máquina, o en el rango $[0, P]$ si es una parte. Teniendo el valor del *step*, se toma la máquina o parte seleccionada y se genera una nueva solución cambiando la celda a la cual pertenece. Si la solución del vuelo de Lévy muestra una mejora en el fitness, ésta reemplaza a la solución original. Si luego de un cierto número de intentos, el vuelo de Lévy no logra mejorar las soluciones, éste se inicializa nuevamente tomando una solución aleatoriamente de la población inicial. En caso de que la población en general quede atrapada en un óptimo local, es decir el fitness deje de mejorar, el vuelo de Lévy invertirá todos los valores de la matriz Y. Luego de este proceso, se ordena la problación inicial ascendentemente de acuerdo al fitness y una fracción de nidos son abandonados según la probabilidad P_a.

Para utilizar el *step* en CS se realiza una transformación de su valor mediante el uso de una función de trasferencia. La función de transferencia utilizada es la V4 [10], que se define como una función de variable real diferenciable. Sus características más relevantes son que su dominio sea todos los reales y su imagen esté en el intervalo $[0, 1]$.

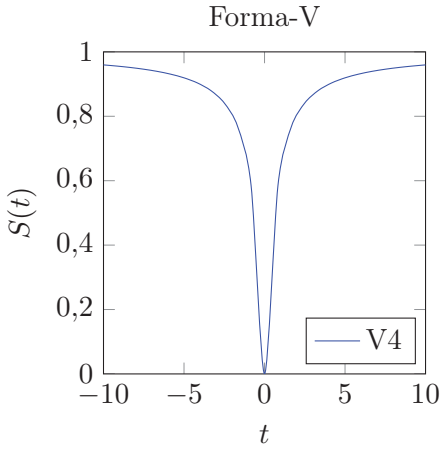

Figura 2: Visualización de la función V4

Con el objetivo de que la función V4 fuera de utilidad en este proyecto, ésta debió ser modificada, debido a que al aplicar el vuelo de Lévy se necesita generar un número dentro de un intervalo deseado. Por lo que el resultado de V4 ($S(t)$), se multiplica por el largo del intervalo que se desea obtener.

La función V4 (ver figura 2) se describe de la siguiente forma:

$$S(t) = |\frac{2}{\pi} * \arctan(\frac{2}{\pi} * t)| \tag{7}$$

Finalmente, un aspecto importante del algoritmo CS implementado es la utilización de paralelismo con el fin de optimizar los tiempos de ejecución. En este caso, hemos aplicado esta forma de trabajo en varias partes del algoritmo. En la sección siguiente de este documento, se detalla más este tema.

7. Experimentos

El proceso de implementación de Cuckoo Search aplicado al MCDP, ha permitido obtener interesantes resultados, los cuales se presentarán en esta sección. Para llevar a cabo las pruebas de este proyecto, se utilizaron los problemas propuestos por Boctor [2]. Éstos, son los 10 problemas propuestos para MCDP y en cada uno varía la cantidad de celdas (C) y el número máximo de máquinas por celda ($Mmax$).

7.1. Características

Para las pruebas se han utilizado los siguientes parámetros en Cuckoo Search:

- $Pa = 0{,}75$
- Generaciones = 120000
- Cantidad de nidos = 1000
- Ejecuciones realizadas para cada problema = 31

7.2. Matrices utilizadas

Las matrices testeadas con la solución implementada son extraídas de un conjunto de problemas estudiados por F. Boctor [2]. Dichas matrices son de dimensión 16×30 y en total se presentan 10 de éstas. Para cada problema se varía la cantidad de celdas (C) entre 2 y 3, teniendo para los problemas con 2 celdas variación en el M_{max} entre 8 y 12 y para los de 3 celdas una variación del M_{max} entre 6 y 9. Los valores óptimos que entrega la investigación de Boctor para cada una de las variaciones de los 10 problemas, están dados a continuación en la Tabla 3 y Tabla 4.

7.3. Resultados

El algoritmo Cuckoo Search aplicado al MCDP, ha sido implementado en Java en un computador con procesador AMD A10 con 8GB de RAM,

Problema	Celdas = 2				
	M_{max} 8	M_{max} 9	M_{max} 10	M_{max} 11	M_{max} 12
1	11	11	11	11	11
2	7	6	4	3	3
3	4	4	4	3	1
4	14	13	13	13	13
5	9	6	6	5	4
6	5	3	3	3	2
7	7	4	4	4	4
8	13	10	8	5	5
9	8	8	8	5	5
10	8	5	5	5	5

Tabla 3: Resultados óptimos obtenidos por Boctor para problemas con 2 celdas.

Problema	Celdas = 3			
	M_{max} 6	M_{max} 7	M_{max} 8	M_{max} 9
1	27	18	11	11
2	7	6	6	6
3	9	4	4	4
4	27	18	14	13
5	11	8	8	6
6	6	4	4	3
7	11	5	5	4
8	14	11	11	10
9	12	12	8	8
10	10	8	8	5

Tabla 4: Resultados óptimos obtenidos por Boctor para problemas con 3 celdas.

ejecutando Microsoft Windows 7 y el IDE Eclipse Mars.

Durante la fase de implementación se intentó ver el comportamiento del problema bajo ciertos parámetros como P_a, iteraciones de la metaheurística y n. Los cuales, luego de ser variados en numerosas ocasiones, fueron finalmente observados con la finalidad de ver la mejor forma de trabajo con el que la metaheurística pueda lograr llegar a los óptimos globales. Los parámetros utilizados son : $P_a = 0,75$, Iteraciones= 120000 y $n = 1000$. Además, al momento de realizar las pruebas, los resultados que se obtenían eran comparados con los óptimos de las tablas 3 y 4 para así observar el nivel eficiencia que alcanzaba la metaheurística e intentar mejorarla.

Se realizaron 31 ejecuciones del algoritmo implementado, teniendo los resultados presentados en la tabla 5 y 6. La metaheurística aplicada al MCDP logra alcanzar el óptimo en todos los problemas al menos una vez, presentando para todos los problemas con dos celdas, una desviación estándar de 0, a diferencia de variadas desviaciones para los problemas de 3 celdas. Así también, observando el contador, se concluye que para los problemas de 2 celdas, el óptimo se alcanza en las generaciones tempranas, a diferencia de los problemas con 3 celdas, para los cuales se debe llegar casi al final de las generaciones para alcanzarlo.

Tabla 5: Resultados de $C = 2$ usando : Población = 1000, $Pa = 0{,}75$ y $Generaciones = 120000$

Problema	Mmax	Óptimo	Mejor	Media Fitness	Desviación Estándar	Iteraciones	RPD %
1	8	11	11	11,00	0,00	3382	0,00 %
1	9	11	11	11,00	0,00	2921	0,00 %
1	10	11	11	11,00	0,00	2408	0,00 %
1	11	11	11	11,00	0,00	2353	0,00 %
1	12	11	11	11,00	0,00	2532	0,00 %
2	8	7	7	7,00	0,00	2053	0,00 %
2	9	6	6	6,00	0,00	3369	0,00 %
2	10	4	4	4,00	0,00	2861	0,00 %
2	11	3	3	3,00	0,00	2833	0,00 %
2	12	3	3	3,00	0,00	1342	0,00 %
3	8	4	4	4,00	0,00	1884	0,00 %
3	9	4	4	4,00	0,00	2028	0,00 %
3	10	4	4	4,00	0,00	889	0,00 %
3	11	3	3	3,00	0,00	2389	0,00 %
3	12	1	1	1,00	0,00	2728	0,00 %
4	8	14	14	14,00	0,00	3148	0,00 %
4	9	13	13	13,00	0,00	1908	0,00 %
4	10	13	13	13,00	0,00	2669	0,00 %
4	11	13	13	13,00	0,00	2050	0,00 %
4	12	13	13	13,00	0,00	356	0,00 %
5	8	9	9	9,00	0,00	3347	0,00 %
5	9	6	6	6,00	0,00	2291	0,00 %
5	10	6	6	6,00	0,00	2356	0,00 %
5	11	5	5	5,00	0,00	2482	0,00 %
5	12	4	4	4,00	0,00	2375	0,00 %
6	8	5	5	5,00	0,00	1491	0,00 %
6	9	3	3	3,00	0,00	2065	0,00 %
6	10	3	3	3,00	0,00	2331	0,00 %
6	11	3	3	3,00	0,00	1489	0,00 %
6	12	2	2	2,00	0,00	840	0,00 %
7	8	7	7	7,00	0,00	2863	0,00 %
7	9	4	4	4,00	0,00	2719	0,00 %
7	10	4	4	4,00	0,00	1936	0,00 %
7	11	4	4	4,00	0,00	967	0,00 %
7	12	4	4	4,00	0,00	1443	0,00 %
8	8	13	13	13,00	0,00	2397	0,00 %
8	9	10	10	10,00	0,00	2612	0,00 %
8	10	8	8	8,00	0,00	352	0,00 %
8	11	5	5	5,00	0,00	3239	0,00 %
8	12	5	5	5,00	0,00	3222	0,00 %
9	8	8	8	8,00	0,00	3128	0,00 %
9	9	8	8	8,00	0,00	1152	0,00 %
9	10	8	8	8,00	0,00	1557	0,00 %
9	11	5	5	5,00	0,00	2196	0,00 %
9	12	5	5	5,00	0,00	1777	0,00 %
10	8	8	8	8,00	0,00	3015	0,00 %
10	9	5	5	5,00	0,00	2428	0,00 %
10	10	5	5	5,00	0,00	1981	0,00 %
10	11	5	5	5,00	0,00	2885	0,00 %
10	12	5	5	5,00	0,00	1896	0,00 %

Tabla 6: Resultados de $C = 3$ usando : Población = 1000 , $Pa = 0{,}75$ y $Generaciones = 120000$

Problema	Mmax	Óptimo	Mejor	Media Fitness	Desviación Estándar	Iteraciones	RPD %
1	6	27	27	28,00	1,38	47964	0,00 %
1	7	18	18	20,00	1,69	45824	0,00 %
1	8	11	11	14,00	2,19	46002	0,00 %
1	9	11	11	13,00	2,02	44707	0,00 %
2	6	7	7	11,00	2,58	43643	0,00 %
2	7	6	6	8,00	1,83	47823	0,00 %
2	8	6	6	7,00	1,37	48602	0,00 %
2	9	6	6	7,00	0,72	48009	0,00 %
3	6	9	9	10,00	1,28	44886	0,00 %
3	7	4	4	7,00	1,59	48466	0,00 %
3	8	4	4	5,00	1,68	42749	0,00 %
3	9	4	4	5,00	1,61	44322	0,00 %
4	6	27	27	27,00	0,54	28599	0,00 %
4	7	18	18	19,00	1,31	46748	0,00 %
4	8	14	14	15,00	1,82	42877	0,00 %
4	9	13	13	14,00	1,40	49582	0,00 %
5	6	11	11	12,00	1,72	40480	0,00 %
5	7	8	8	11,00	1,68	47447	0,00 %
5	8	8	8	10,00	1,31	46701	0,00 %
5	9	6	6	8,00	1,85	42746	0,00 %
6	6	7	7	9,00	1,41	50000	0,00 %
6	7	4	4	6,00	1,56	49740	0,00 %
6	8	4	4	5,00	1,11	46947	0,00 %
6	9	3	3	5,00	1,28	46768	0,00 %
7	6	11	11	14,00	1,69	46846	0,00 %
7	7	5	5	8,00	1,98	48301	0,00 %
7	8	5	5	7,00	1,92	44675	0,00 %
7	9	4	4	7,00	1,67	49197	0,00 %
8	6	14	14	16,00	2,19	44788	0,00 %
8	7	11	11	15,00	1,97	46661	0,00 %
8	8	11	11	14,00	1,19	48429	0,00 %
8	9	10	10	12,00	0,92	46341	0,00 %
9	6	12	12	17,00	1,90	48599	0,00 %
9	7	12	12	14,00	1,48	48761	0,00 %
9	8	8	8	11,00	2,21	47204	0,00 %
9	9	8	8	11,00	1,90	43079	0,00 %
10	6	10	10	13,00	1,93	47544	0,00 %
10	7	8	8	11,00	1,68	47914	0,00 %
10	8	8	8	9,00	1,24	43106	0,00 %
10	9	5	5	8,00	1,98	45730	0,00 %

7.4. Paralelismo

Una inclusión importante de esta etapa del proyecto, fue la utilización del paralelismo en el algoritmo, con el objetivo de reducir los tiempos de ejecución. Luego de analizar el algoritmo, se llegó a implementar varias formas de paralelismo aplicados estratégicamente en diferentes partes del código. Finalmente, se llegó a obtener resultados con paralelismo con los métodos explicados a continuación.

7.4.1. Merge Sort

Se utilizó un algoritmo para realizar ordenamientos paralelos a través de hebras. Éste se aplica cada vez que es necesario reordenar la población de nidos de manera ascendente. Luego de varias pruebas, se llegó a la conclusión de que con 4 hebras se obtenía el mejor comportamiento aplicando este método. Los resultados se muestran en las tablas 7 y 8.

7.4.2. Creación paralela de la población inicial

Se utilizó paralelismo para crear el arreglo de soluciones iniciales (nidos). Se crearon dos hebras y cada una de éstas es responsable de crear una mitad de la población inicial simultáneamente. Los resultados se muestran en las tablas 9 y 10.

7.4.3. Lévy Flights en problemas de 3 celdas

Al aplicar Lévy Flights en una de las matrices, se utiliza paralelismo para crear dos soluciones vecinas modificadas a la vez y elegir la mejor de ellas inmediatamente. Los resultados se muestran en la tabla 11.

7.4.4. Combinación de Paralelismos con Merge Sort

Se realizaron experimentos utilizando las tres formas de paralelismo anteriores (Merge sort, en la creación de la población inicial y en la creación de

una solución vecina con $C = 3$). Los resultados se muestran en las tablas 12 y 13.

7.4.5. Combinación de Paralelismos con Sort

Esta vez, se utilizaron sólo dos de los paralelismos anteriores (creación de la población inicial y en la creación de una solución vecina en problemas con $C = 3$) y realizando los ordenamientos con el método sort que viene incluído en Java SE 1.8. Los resultados se muestran en las tablas 14 y 15.

Tabla 7: Resultados de $C = 2$ usando : Población = 1000, Pa = 0,75, $Generaciones = 120000$ y $Ejecuciones = 31$ (Merge Sort).

Problema	Mmax	Óptimo	Media	Mejor Fitness	Desviación Estándar	Varianza	Iteraciones	Tiempo (ms)
1	8	11	11.0	11	0.0	0.0	40	28954
1	9	11	11.0	11	0.0	0.0	35	22904
1	10	11	11.0	11	0.0	0.0	39	25191
1	11	11	11.0	11	0.0	0.0	47	29955
1	12	11	11.0	11	0.0	0.0	50	31905
2	8	7	7.0	7	0.0	0.0	59	37395
2	9	6	6.0	6	0.0	0.0	36	23457
2	10	4	4.0	4	0.0	0.0	38	24077
2	11	3	3.0	3	0.0	0.0	38	24277
2	12	3	3.0	3	0.0	0.0	17	11565
3	8	4	4.0	4	0.0	0.0	38	24192
3	9	4	4.0	4	0.0	0.0	24	15653
3	10	4	4.0	4	0.0	0.0	15	10373
3	11	3	3.0	3	0.0	0.0	55	34284
3	12	1	1.0	1	0.0	0.0	35	22378
4	8	14	14.0	14	0.0	0.0	40	26429
4	9	13	13.0	13	0.0	0.0	39	25760
4	10	13	13.0	13	0.0	0.0	39	25502
4	11	13	13.0	13	0.0	0.0	59	37548
4	12	13	13.0	13	0.0	0.0	10	7289
5	8	9	9.0	9	0.0	0.0	32	20581
5	9	6	6.0	6	0.0	0.0	39	24770
5	10	6	6.0	6	0.0	0.0	49	30989
5	11	5	5.0	5	0.0	0.0	32	20798
5	12	4	4.0	4	0.0	0.0	44	27712
6	8	5	5.0	5	0.0	0.0	19	12978
6	9	3	3.0	3	0.0	0.0	51	31929
6	10	3	3.0	3	0.0	0.0	47	29809
6	11	3	3.0	3	0.0	0.0	21	13987
6	12	2	2.0	2	0.0	0.0	13	8756
7	8	7	7.0	7	0.0	0.0	42	27860
7	9	4	4.0	4	0.0	0.0	35	23086
7	10	4	4.0	4	0.0	0.0	21	14425
7	11	4	4.0	4	0.0	0.0	21	14604
7	12	4	4.0	4	0.0	0.0	18	12251
8	8	13	13.0	13	0.0	0.0	41	26475
8	9	10	10.0	10	0.0	0.0	54	34528
8	10	8	8.0	8	0.0	0.0	4	3720
8	11	5	5.0	5	0.0	0.0	49	31212
8	12	5	5.0	5	0.0	0.0	36	23443
9	8	8	8.0	8	0.0	0.0	48	30831
9	9	8	8.0	8	0.0	0.0	16	10670
9	10	8	8.0	8	0.0	0.0	19	12628
9	11	5	5.0	5	0.0	0.0	42	26477
9	12	5	5.0	5	0.0	0.0	37	26305
10	8	8	8.0	8	0.0	0.0	46	32521
10	9	5	5.0	5	0.0	0.0	35	22977
10	10	5	5.0	5	0.0	0.0	43	27776
10	11	5	5.0	5	0.0	0.0	44	28352
10	12	5	5.0	5	0.0	0.0	32	21357

Tabla 8: Resultados de $C = 3$ usando : Población = 1000, Pa = 0,75, $Generaciones = 120000$ y $Ejecuciones = 31$ (Merge Sort).

Problema	Mmax	Óptimo	Media	Mejor Fitness	Desviación Estándar	Varianza	Iteraciones	Tiempo (ms)
1	9	27	27,0	27	0,00	0,00	9329	7043394
1	8	18	18,0	18	0,00	0,00	8779	7681691
1	7	11	11,0	11	0,00	0,00	9246	6987652
1	6	11	11,0	11	0,00	0,00	10522	8018302
2	9	7	7,0	7	0,00	0,00	4688	3536038
2	8	6	6,0	6	0,00	0,00	10027	8144870
2	7	6	6,0	6	0,00	0,00	9001	7742090
2	6	6	6,0	6	0,00	0,00	4078	3039743
3	9	9	9,0	9	0,00	0,00	2900	1912633
3	8	4	4,0	4	0,00	0,00	10690	6993524
3	7	4	4,0	4	0,00	0,00	5715	4206812
3	6	4	4,0	4	0,00	0,00	3062	2241230
4	9	27	27,0	27	0,00	0,00	706	547680
4	8	18	18,0	18	0,00	0,00	2939	2284085
4	7	14	14,0	14	0,00	0,00	9980	6712932
4	6	13	13,0	13	0,00	0,00	11370	65115575
5	9	11	11,0	11	0,00	0,00	3404	2542461
5	8	8	8,0	8	0,00	0,00	9418	6975998
5	7	8	8,0	8	0,00	0,00	11428	9302955
5	6	6	6,0	6	0,00	0,00	9137	6750563
6	9	6	6,0	6	0,00	0,00	10484	8296970
6	8	4	4,0	4	0,00	0,00	9999	8354951
6	7	4	4,0	4	0,00	0,00	13121	10975225
6	6	3	3,0	3	0,00	0,00	9090	6607143
7	9	11	11,0	11	0,00	0,00	7300	5912000
7	8	5	5,0	5	0,00	0,00	8985	7550956
7	7	5	5,0	5	0,00	0,00	11377	9570363
7	6	4	4,0	4	0,00	0,00	10864	7940783
8	9	14	14,0	14	0,00	0,00	5531	4417750
8	8	11	11,0	11	0,00	0,00	8084	6743702
8	7	11	11,0	11	0,00	0,00	9374	7726954
8	6	10	10,0	10	0,00	0,00	10365	7553285
9	9	12	12,0	12	0,00	0,00	10732	8631651
9	8	12	12,0	12	0,00	0,00	4704	3959258
9	7	8	8,0	8	0,00	0,00	9447	7741213
9	6	8	8,0	8	0,00	0,00	3370	2458598
10	9	10	10,0	10	0,00	0,00	10232	11073436
10	8	8	8,0	8	0,00	0,00	8553	7250888
10	7	8	8,0	8	0,00	0,00	2894	2393082
10	6	5	5,0	5	0,00	0,00	8344	6181278

Tabla 9: Resultados de $C = 2$ usando : Población= 1000, $Pa = 0{,}75$, $Generaciones = 120000$ y $Ejecuciones = 31$ (Creación paralela de la población inicial).

Problema	Mmax	Óptimo	Media	Mejor Fitness	Desviación Estándar	Varianza	Iteraciones	Tiempo (ms)
1	8	11	11.0	11	0.0	0.0	41	15594
1	9	11	11.0	11	0.0	0.0	43	18797
1	10	11	11.0	11	0.0	0.0	37	13802
1	11	11	11.0	11	0.0	0.0	21	7728
1	12	11	11.0	11	0.0	0.0	52	18421
2	8	7	7.0	7	0.0	0.0	39	13962
2	9	6	6.0	6	0.0	0.0	37	12792
2	10	4	4.0	4	0.0	0.0	44	17050
2	11	3	3.0	3	0.0	0.0	46	17360
2	12	3	3.0	3	0.0	0.0	19	7115
3	8	4	4.0	4	0.0	0.0	35	12335
3	9	4	4.0	4	0.0	0.0	21	7616
3	10	4	4.0	4	0.0	0.0	26	9601
3	11	3	3.0	3	0.0	0.0	30	10311
3	12	1	1.0	1	0.0	0.0	40	13537
4	8	14	14.0	14	0.0	0.0	36	13599
4	9	13	13.0	13	0.0	0.0	46	16886
4	10	13	13.0	13	0.0	0.0	42	15370
4	11	13	13.0	13	0.0	0.0	40	14560
4	12	13	13.0	13	0.0	0.0	10	3906
5	8	9	9.0	9	0.0	0.0	33	11819
5	9	6	6.0	6	0.0	0.0	58	20049
5	10	6	6.0	6	0.0	0.0	34	12092
5	11	5	5.0	5	0.0	0.0	46	16561
5	12	4	4.0	4	0.0	0.0	41	14021
6	8	5	5.0	5	0.0	0.0	23	8269
6	9	3	3.0	3	0.0	0.0	48	16987
6	10	3	3.0	3	0.0	0.0	38	13232
6	11	3	3.0	3	0.0	0.0	14	5691
6	12	2	2.0	2	0.0	0.0	14	5282
7	8	7	7.0	7	0.0	0.0	33	14217
7	9	4	4.0	4	0.0	0.0	48	17157
7	10	4	4.0	4	0.0	0.0	22	8476
7	11	4	4.0	4	0.0	0.0	19	6931
7	12	4	4.0	4	0.0	0.0	22	8652
8	8	13	13.0	13	0.0	0.0	47	17258
8	9	10	10.0	10	0.0	0.0	63	22251
8	10	8	8.0	8	0.0	0.0	6	2677
8	11	5	5.0	5	0.0	0.0	37	12929
8	12	5	5.0	5	0.0	0.0	41	13913
9	8	8	8.0	8	0.0	0.0	43	15062
9	9	8	8.0	8	0.0	0.0	22	7821
9	10	8	8.0	8	0.0	0.0	22	8184
9	11	5	5.0	5	0.0	0.0	52	18502
9	12	5	5.0	5	0.0	0.0	40	16335
10	8	8	8.0	8	0.0	0.0	40	15414
10	9	5	5.0	5	0.0	0.0	38	14574
10	10	5	5.0	5	0.0	0.0	50	18221
10	11	5	5.0	5	0.0	0.0	48	16907
10	12	5	5.0	5	0.0	0.0	56	19906

Tabla 10: Resultados de $C = 3$ usando : Población = 1000, $Pa = 0{,}75$, $Generaciones = 120000$ y $Ejecuciones = 31$ (Creación paralela de población inicial).

Problema	Mmax	Óptimo	Media	Mejor Fitness	Desviación Estándar	Varianza	Iteraciones	Tiempo (ms)
1	9	27	27,0	27	0,00	0,00	10288	5407413
1	8	18	18,0	18	0,00	0,00	9517	4595464
1	7	11	11,0	11	0,00	0,00	8737	3866400
1	6	11	11,0	11	0,00	0,00	11985	5962378
2	9	7	7,0	7	0,00	0,00	4492	2350262
2	8	6	6,0	6	0,00	0,00	13475	6620515
2	7	6	6,0	6	0,00	0,00	9992	4924683
2	6	6	6,0	6	0,00	0,00	4690	2158036
3	9	9	9,0	9	0,00	0,00	2875	1023415
3	8	4	4,0	4	0,00	0,00	8488	2988833
3	7	4	4,0	4	0,00	0,00	4899	1908668
3	6	4	4,0	4	0,00	0,00	4026	1570658
4	9	27	27,0	27	0,00	0,00	1215	497646
4	8	18	18,0	18	0,00	0,00	3235	1310385
4	7	14	14,0	14	0,00	0,00	6742	2443920
4	6	13	13,0	13	0,00	0,00	10791	4340470
5	9	11	11,0	11	0,00	0,00	2313	923283
5	8	8	8,0	8	0,00	0,00	12066	4795877
5	7	8	8,0	8	0,00	0,00	10713	4229340
5	6	6	6,0	6	0,00	0,00	11462	4533008
6	9	6	6,0	6	0,00	0,00	9361	3873083
6	8	4	4,0	4	0,00	0,00	8740	3507634
6	7	4	4,0	4	0,00	0,00	8378	4303148
6	6	3	3,0	3	0,00	0,00	9278	4508493
7	9	11	11,0	11	0,00	0,00	8117	3431747
7	8	5	5,0	5	0,00	0,00	7690	3126003
7	7	5	5,0	5	0,00	0,00	11450	5408118
7	6	4	4,0	4	0,00	0,00	8654	3738369
8	9	14	14,0	14	0,00	0,00	4638	1946078
8	8	11	11,0	11	0,00	0,00	10884	4375271
8	7	11	11,0	11	0,00	0,00	11646	6106223
8	6	10	10,0	10	0,00	0,00	11162	4544514
9	9	12	12,0	12	0,00	0,00	9008	3799483
9	8	12	12,0	12	0,00	0,00	4015	1629933
9	7	8	8,0	8	0,00	0,00	7128	2437002
9	6	8	8,0	8	0,00	0,00	3839	1475071
10	9	10	10,0	10	0,00	0,00	9585	4084368
10	8	8	8,0	8	0,00	0,00	10657	4554571
10	7	8	8,0	8	0,00	0,00	2764	1647713
10	6	5	5,0	5	0,00	0,00	11084	4490857

Tabla 11: Resultados de $C = 3$ usando : Población = 1000, $Pa = 0{,}75$, $Generaciones = 120000$ y $Ejecuciones = 31$ (Vuelo de Lévy en problemas de 3 celdas).

Problema	Mmax	Óptimo	Media	Mejor Fitness	Desviación Estándar	Varianza	Iteraciones	Tiempo (ms)
1	9	11	11,0	11	0,17	0,13	64521	5239121
1	8	27	27,0	27	0,39	0,62	82812	6786762
1	7	18	18,0	18	0,23	0,48	64646	5260456
1	6	11	11,0	11	0,77	0,88	64237	5207421
2	9	6	7,0	7	0,29	0,54	41169	3344450
2	8	6	6,0	6	0,39	0,62	60498	5041659
2	7	5	6,0	6	0,16	0,40	43669	3516259
2	6	6	6,0	6	0,16	0,40	55901	4499566
3	9	9	9,0	9	0,00	0,00	21475	1630676
3	8	4	4,0	4	0,15	0,12	74185	5593812
3	7	4	4,0	4	0,06	0,25	45159	3874221
3	6	4	4,0	4	0,00	0,00	22805	1962715
4	9	26	26,0	27	0,97	0,98	7571	6866109
4	8	18	18,0	18	0,00	0,00	34159	3089846
4	7	14	14,0	14	0,32	0,57	60670	4683843
4	6	10	13,0	13	0,74	0,86	54594	4796343
5	9	11	11,0	11	0,06	0,25	37793	3382203
5	8	8	8,0	8	0,84	0,92	70355	6226930
5	7	8	8,0	8	0,19	0,44	62974	5301447
5	6	6	6,0	6	0,23	0,48	60825	5492771
6	9	6	6,0	6	0,32	0,57	66887	5958056
6	8	4	4,0	4	0,26	0,51	64179	5751218
6	7	4	4,0	4	0,23	0,48	53122	4762254
6	6	3	3,0	3	0,39	0,62	64055	5317414
7	9	11	11,0	11	0,45	0,67	78562	7159101
7	8	5	5,0	5	0,13	0,12	68766	6345812
7	7	5	5,0	5	0,52	0,72	51007	4617337
7	6	3	4,0	4	0,10	0,10	50696	4393178
8	9	14	14,0	14	0,00	0,00	32177	2885167
8	8	10	11,0	11	0,25	0,16	69049	6325794
8	7	11	11,0	11	0,13	0,11	62336	5571854
8	6	10	10,0	10	0,42	0,65	65474	18972314
9	9	12	13,0	12	0,25	0,16	75181	6785151
9	8	11	11,0	12	0,97	0,98	34519	3184068
9	7	8	8,0	8	0,90	0,95	65517	5893676
9	6	8	8,0	8	0,00	0,00	19848	1784941
10	9	10	10,0	10	0,12	0,11	77882	7101051
10	8	6	8,0	8	0,42	0,65	47828	4453001
10	7	8	8,0	8	0,00	0,00	23959	2177665
10	6	5	5,0	5	0,22	0,15	76079	6736140

Tabla 12: Resultados de $C = 2$ usando : Población = 1000, Pa = 0,75, $Generaciones = 120000$ y $Ejecuciones = 31$ (Combinación Merge Sort).

Problema	Mmax	Óptimo	Media	Mejor Fitness	Desviación Estándar	Varianza	Iteraciones	Tiempo (ms)
1	8	11	11.0	11	0.0	0.0	29	14195
1	9	11	11.0	11	0.0	0.0	36	16310
1	10	11	11.0	11	0.0	0.0	41	18692
1	11	11	11.0	11	0.0	0.0	39	18539
1	12	11	11.0	11	0.0	0.0	46	21601
2	8	7	7.0	7	0.0	0.0	35	16649
2	9	6	6.0	6	0.0	0.0	37	17142
2	10	4	4.0	4	0.0	0.0	43	19082
2	11	3	3.0	3	0.0	0.0	38	17815
2	12	3	3.0	3	0.0	0.0	18	8674
3	8	4	4.0	4	0.0	0.0	32	14685
3	9	4	4.0	4	0.0	0.0	23	10638
3	10	4	4.0	4	0.0	0.0	17	8125
3	11	3	3.0	3	0.0	0.0	35	16370
3	12	1	1.0	1	0.0	0.0	59	25235
4	8	14	14.0	14	0.0	0.0	24	11350
4	9	13	13.0	13	0.0	0.0	47	21688
4	10	13	13.0	13	0.0	0.0	58	26217
4	11	13	13.0	13	0.0	0.0	37	17597
4	12	13	13.0	13	0.0	0.0	10	5278
5	8	9	9.0	9	0.0	0.0	58	26399
5	9	6	6.0	6	0.0	0.0	38	17755
5	10	6	6.0	6	0.0	0.0	45	19729
5	11	5	5.0	5	0.0	0.0	43	19213
5	12	4	4.0	4	0.0	0.0	30	13037
6	8	5	5.0	5	0.0	0.0	25	11534
6	9	3	3.0	3	0.0	0.0	45	19619
6	10	3	3.0	3	0.0	0.0	46	20123
6	11	3	3.0	3	0.0	0.0	20	9092
6	12	2	2.0	2	0.0	0.0	14	6719
7	8	7	7.0	7	0.0	0.0	40	17692
7	9	4	4.0	4	0.0	0.0	45	20045
7	10	4	4.0	4	0.0	0.0	15	6934
7	11	4	4.0	4	0.0	0.0	16	7636
7	12	4	4.0	4	0.0	0.0	22	10011
8	8	13	13.0	13	0.0	0.0	49	21479
8	9	10	10.0	10	0.0	0.0	41	18181
8	10	8	8.0	8	0.0	0.0	6	3338
8	11	5	5.0	5	0.0	0.0	32	14014
8	12	5	5.0	5	0.0	0.0	39	17220
9	8	8	8.0	8	0.0	0.0	47	20571
9	9	8	8.0	8	0.0	0.0	23	10615
9	10	8	8.0	8	0.0	0.0	21	9576
9	11	5	5.0	5	0.0	0.0	46	20106
9	12	5	5.0	5	0.0	0.0	37	16361
10	8	8	8.0	8	0.0	0.0	48	21452
10	9	5	5.0	5	0.0	0.0	39	17166
10	10	5	5.0	5	0.0	0.0	44	19459
10	11	5	5.0	5	0.0	0.0	35	15426
10	12	5	5.0	5	0.0	0.0	36	15899

Tabla 13: Resultados de $C = 3$ usando : Población = 1000, $Pa = 0{,}75$, $Generaciones = 120000$ y $Ejecuciones = 31$ (Combinación con Merge Sort).

Problema	Mmax	Óptimo	Media	Mejor Fitness	Desviación Estándar	Varianza	Iteraciones	Tiempo (ms)
1	9	27	27,0	27	0,00	0,00	11209	8213304
1	8	18	18,0	18	0,00	0,00	10900	57343001
1	7	11	11,0	11	0,00	0,00	7606	4473934
1	6	9	10,0	11	1,00	1,00	6814	4247347
2	9	6	6,0	7	0,97	0,98	5748	3255765
2	8	6	6,0	6	0,00	0,00	6603	3675501
2	7	5	5,0	6	0,94	0,97	6351	5005871
2	6	5	5,0	6	0,90	0,95	2792	1594962
3	9	9	9,0	9	0,00	0,00	1889	851004
3	8	4	4,0	4	0,00	0,00	12617	5653610
3	7	3	3,0	4	0,97	0,98	3905	1937642
3	6	4	4,0	4	0,00	0,00	3045	2904751
4	9	24	26,0	27	0,11	0,10	906	470822
4	8	18	18,0	18	0,00	0,00	3538	1820595
4	7	14	14,0	14	0,00	0,00	6960	3181364
4	6	12	12,0	13	0,90	0,95	8012	4160308
5	9	11	11,0	11	0,00	0,00	3056	1556041
5	8	7	7,0	8	0,97	0,98	7817	3950051
5	7	7	7,0	8	0,97	0,98	10726	5394633
5	6	4	5,0	6	1,00	1,00	7029	3532544
6	9	6	6,0	6	0,00	0,00	6437	3220865
6	8	4	4,0	4	0,00	0,00	10290	4883449
6	7	4	4,0	4	0,00	0,00	7897	6406516
6	6	3	3,0	3	0,00	0,00	7734	3735330
7	9	11	11,0	11	0,00	0,00	9419	5048465
7	8	5	5,0	5	0,00	0,00	12168	5859906
7	7	5	5,0	5	0,00	0,00	8234	7410774
7	6	4	4,0	4	0,00	0,00	8969	4402688
8	9	13	13,0	14	0,87	0,93	2944	1581248
8	8	10	10,0	11	0,90	0,95	4411	2107355
8	7	11	11,0	11	0,00	0,00	8868	2626083
8	6	10	10,0	10	0,00	0,00	7143	3474468
9	9	11	11,0	12	0,97	0,98	9298	5044301
9	8	11	11,0	12	0,97	0,98	4932	2371051
9	7	8	8,0	8	0,00	0,00	8336	2287657
9	6	5	7,0	8	0,11	0,10	4007	1998601
10	9	10	10,0	10	0,00	0,00	5731	3093485
10	8	7	7,0	8	0,97	0,98	6995	3451079
10	7	7	7,0	8	0,94	0,97	2392	1239939
10	6	5	5,0	5	0,00	0,00	10365	5393934

Tabla 14: Resultados de $C = 2$ usando : Población = 1000, Pa = 0,75, $Generaciones = 120000$ y $Ejecuciones = 31$ (Combinación Sort).

Problema	Mmax	Óptimo	Media	Mejor Fitness	Desviación Estándar	Varianza	Iteraciones	Tiempo (ms)
1	8	11	11.0	11	0.0	0.0	38	13492
1	9	11	11.0	11	0.0	0.0	32	11118
1	10	11	11.0	11	0.0	0.0	42	14722
1	11	11	11.0	11	0.0	0.0	40	13948
1	12	11	11.0	11	0.0	0.0	39	13213
2	8	7	7.0	7	0.0	0.0	40	13771
2	9	6	6.0	6	0.0	0.0	37	12546
2	10	4	4.0	4	0.0	0.0	50	16581
2	11	3	3.0	3	0.0	0.0	38	12759
2	12	3	3.0	3	0.0	0.0	22	7794
3	8	4	4.0	4	0.0	0.0	56	18783
3	9	4	4.0	4	0.0	0.0	17	6180
3	10	4	4.0	4	0.0	0.0	20	7051
3	11	3	3.0	3	0.0	0.0	36	11909
3	12	1	1.0	1	0.0	0.0	53	17457
4	8	14	14.0	14	0.0	0.0	47	16646
4	9	13	13.0	13	0.0	0.0	51	17447
4	10	13	13.0	13	0.0	0.0	28	9940
4	11	13	13.0	13	0.0	0.0	35	12262
4	12	13	13.0	13	0.0	0.0	5	2371
5	8	9	9.0	9	0.0	0.0	32	11299
5	9	6	6.0	6	0.0	0.0	33	11268
5	10	6	6.0	6	0.0	0.0	48	15941
5	11	5	5.0	5	0.0	0.0	51	17083
5	12	4	4.0	4	0.0	0.0	25	8482
6	8	5	5.0	5	0.0	0.0	20	7200
6	9	3	3.0	3	0.0	0.0	40	13489
6	10	3	3.0	3	0.0	0.0	54	17818
6	11	3	3.0	3	0.0	0.0	20	7124
6	12	2	2.0	2	0.0	0.0	11	4197
7	8	7	7.0	7	0.0	0.0	49	17005
7	9	4	4.0	4	0.0	0.0	50	17306
7	10	4	4.0	4	0.0	0.0	15	5604
7	11	4	4.0	4	0.0	0.0	21	7526
7	12	4	4.0	4	0.0	0.0	23	8204
8	8	13	13.0	13	0.0	0.0	49	16972
8	9	10	10.0	10	0.0	0.0	38	13114
8	10	8	8.0	8	0.0	0.0	5	2344
8	11	5	5.0	5	0.0	0.0	34	11640
8	12	5	5.0	5	0.0	0.0	43	14417
9	8	8	8.0	8	0.0	0.0	42	14636
9	9	8	8.0	8	0.0	0.0	15	5466
9	10	8	8.0	8	0.0	0.0	19	6846
9	11	5	5.0	5	0.0	0.0	40	13570
9	12	5	5.0	5	0.0	0.0	49	16286
10	8	8	8.0	8	0.0	0.0	50	17322
10	9	5	5.0	5	0.0	0.0	41	14071
10	10	5	5.0	5	0.0	0.0	51	18027
10	11	5	5.0	5	0.0	0.0	49	17167
10	12	5	5.0	5	0.0	0.0	44	15214

Tabla 15: Resultados de $C = 3$ usando : Población = 1000 , $Pa = 0{,}75$, $Generaciones = 120000$ y $Ejecuciones = 31$ (Combinación Sort).

Problema	Mmax	Óptimo	Media	Mejor Fitness	Desviación Estándar	Varianza	Iteraciones	Tiempo (ms)
1	9	27	27,0	27	0,00	0,00	5555	3641785
1	8	18	18,0	18	0,00	0,00	11763	7146054
1	7	11	11,0	11	0,00	0,00	8566	3955666
1	6	11	11,0	11	0,00	0,00	7257	3479890
2	9	6	6,0	7	0,97	0,98	3892	9771397
2	8	6	6,0	6	0,00	0,00	9301	3580729
2	7	5	5,0	6	0,97	0,98	8858	4003762
2	6	6	6,0	6	0,00	0,00	3641	1641609
3	9	9	9,0	9	0,00	0,00	2075	1767822
3	8	4	4,0	4	0,00	0,00	9320	3428949
3	7	4	4,0	4	0,00	0,00	5174	2257346
3	6	4	4,0	4	0,00	0,00	2685	1160757
4	9	27	27,0	27	0,00	0,00	890	3779203
4	8	18	18,0	18	0,00	0,00	3581	1510585
4	7	14	14,0	14	0,00	0,00	7047	2661477
4	6	12	12,0	13	0,84	0,92	5580	2329799
5	9	11	11,0	11	0,00	0,00	2220	915625
5	8	8	8,0	8	0,00	0,00	12349	5066417
5	7	8	8,0	8	0,00	0,00	11853	4842718
5	6	6	6,0	6	0,00	0,00	11018	4482839
6	9	6	6,0	6	0,00	0,00	8157	3385407
6	8	3	3,0	4	0,94	0,97	7707	3262081
6	7	3	3,0	4	0,97	0,98	7682	4322999
6	6	3	3,0	3	0,00	0,00	8186	3851692
7	9	11	11,0	11	0,00	0,00	7254	3090992
7	8	4	4,0	5	0,97	0,98	12056	5175298
7	7	5	5,0	5	0,00	0,00	9646	4341827
7	6	4	4,0	4	0,00	0,00	8308	3775552
8	9	14	14,0	14	0,00	0,00	3941	1675250
8	8	10	10,0	11	0,97	0,98	6353	2698637
8	7	11	11,0	11	0,00	0,00	9242	3931244
8	6	10	10,0	10	0,00	0,00	10808	4637372
9	9	12	12,0	12	0,00	0,00	10679	4560926
9	8	12	12,0	12	0,00	0,00	3809	5647630
9	7	8	8,0	8	0,00	0,00	10603	2902622
9	6	8	8,0	8	0,00	0,00	4223	1819371
10	9	10	10,0	10	0,00	0,00	10818	4662864
10	8	7	7,0	8	0,94	0,97	6051	2696081
10	7	7	7,0	8	0,97	0,98	2147	2323746
10	6	5	5,0	5	0,00	0,00	7104	3098751

7.5. Análisis de Resultados

Para la presente investigación, obtuvimos resultados para todos los problemas de Boctor aplicando los paralelismos explicados anteriormente. A continuación, en las figuras 3 y 4 se presentan los gráficos con el tiempo total de ejecución (en milisegundos) de los 90 problemas de Boctor con los diferentes tipos de paralelismos probados. Lo primero que se observa es que el método de ordenamiento paralelo Merge Sort resulta ser ineficiente en cuanto al tiempo, siendo más lento que el algoritmo normal implementado sin ningún paralelismo (25 % más lento en problemas de 2 celdas y 100 % más lento en problemas de 3 celdas). Por otro lado, el método que entrega el menor tiempo de ejecución es el de crear la población inicial de forma paralela, siendo el más eficiente de todos, disminuyendo el tiempo con respecto al algoritmo normal en un 37 % en problemas de 2 celdas y 21 % en problemas de 3 celdas.

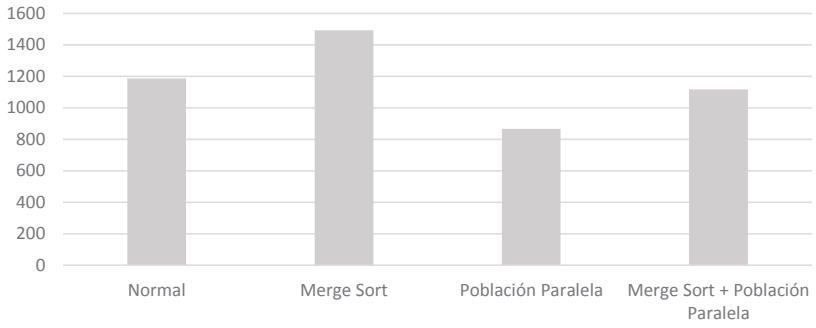

Figura 3: Tiempos de ejecución para problemas de 2 celdas aplicando paralelismo

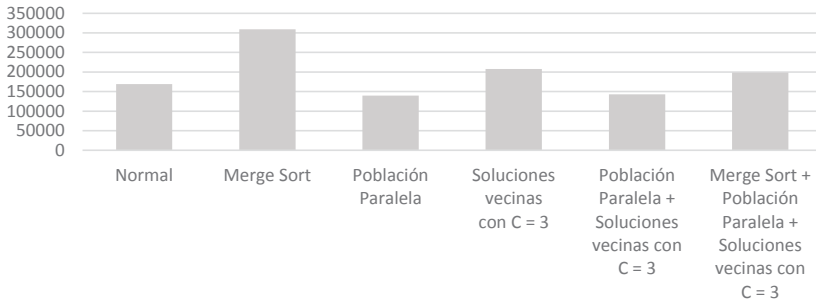

Figura 4: Tiempos de ejecución para problemas de 3 celdas aplicando paralelismo

8. Conclusión

En la actualidad, es necesario y recurrente el desarrollo y la utilización de herramientas informáticas que ayuden en la solución de problemas y en la optimización de procesos en variadas áreas. La aplicación de algoritmos metaheurísticos resulta ser de utilidad en ésto, debido a que puede ser aplicada a una gran cantidad de problemas planteados en la vida real y da la posibilidad de resolverlos de forma óptima. Entre uno de los problemas que pueden ser abordados, tenemos el Manufacturing Cell Design Problem.

Este proyecto reúne la investigación realizada y está basado en gran cantidad de estudios y análisis que fueron realizados para presentar una solución informática que tiene como objetivo la optimización del MCDP. Se logró registrar la información necesaria con respecto al problema que significa el diseño de celdas de manufactura y por otra parte, la presentación de la metaheurísitica Cuckoo Search.

Con respecto al tema planteado, podemos decir que éste fue aplicado satisfactoriamente, obteniendo los resultados deseados en un principio. Ésto debido a que el algoritmo implementado para el problema propuesto demostró ser completamente eficiente y eficaz, llegando al óptimo global de todas las instancias planteadas en un tiempo razonable. A ésto se le suma la aplicación de paralelismo, lo que resultó en todo un éxito considerando la importante disminución en los tiempos de ejecución.

Finalmente, podemos decir que los objetivos definidos en un principio fueron cumplidos completamente, siendo capaces de tomar el tema planteado, estudiarlo e implementar una solución adecuada para obtener los resultados correctos. Con respecto a la optimización de nuestro algoritmo, podemos decir que estudiamos bastantes posibilidades de las cuales implementamos algunas, alcanzando resultados mejores de los esperados con respecto a los tiempos de ejecución. El trabajo realizado en la primera parte del presente proyecto nos permitió obtener una publicación científica con los resultados obtenidos en esa instancia y por otra parte, nos encontramos trabajando en una segunda publicación científica la cual incluirá el trabajo presentado en este proyecto final de tesis.

Referencias

[1] Aljaber, N., Baek, W., Chen, C.L.: A tabu search approach to the cell formation problem. Computers & industrial engineering 32(1), 169–185 (1997)

[2] Boctor, F.F.: A jinear formulation of the machine-part cell formation problem. The International Journal of Production Research 29(2), 343–356 (1991)

[3] Durán, O., Rodriguez, N., Consalter, L.A.: Collaborative particle swarm optimization with a data mining technique for manufacturing cell design. Expert Systems with Applications 37(2), 1563–1567 (2010)

[4] Gupta, Y., Gupta, M., Kumar, A., Sundaram, C.: A genetic algorithm-based approach to cell composition and layout design problems. International Journal of Production Research 34(2), 447–482 (1996)

[5] J. Gutierrez, A.L.: Modelado y resolución del manufacturing cell design problem (mcdp) utilizando programación con restricciones (2011)

[6] Kusiak, A., Chow, W.S.: Efficient solving of the group technology problem. Journal of manufacturing systems 6(2), 117–124 (1987)

[7] Lozano, S., Adenso-Diaz, B., Eguia, I., Onieva, L.: A one-step tabu search algorithm for manufacturing cell design. Journal of the Operational Research Society 50(5), 509–516 (1999)

[8] Mantegna, R.N.: Fast, accurate algorithm for numerical simulation of levy stable stochastic processes. Physical Review E 49(5), 4677 (1994)

[9] Medina, P.D., Cruz, E.A., Pinzón, M.: Generación de celdas de manufactura usando el algoritmo de ordenamiento binario (aob). Scientia Et Technica 1(44), 106–110 (2010)

[10] Mirjalili, S., Lewis, A.: S-shaped versus v-shaped transfer functions for binary particle swarm optimization. Swarm and Evolutionary Computation 9, 1–14 (2013)

[11] Oliva-Lopez, E., Purcheck, G.: Load balancing for group technology planning and control. International Journal of Machine Tool Design and Research 19(4), 259–274 (1979)

[12] Purcheck, G.F.: A linear–programming method for the combinatorial grouping of an incomplete power set (1975)

[13] Reynolds, A.M., Frye, M.A.: Free-flight odor tracking in drosophila is consistent with an optimal intermittent scale-free search. PLoS ONE 2(4) (Apr 2007)

[14] Sankaran, S., Rodin, E.Y.: Multiple objective decision making approach to cell formation: a goal programming model. Mathematical and Computer Modelling 13(9), 71–81 (1990)

[15] Shafer, S.M., Rogers, D.F.: A goal programming approach to the cell formation problem. Journal of Operations Management 10(1), 28–43 (1991)

[16] Soto, R., Crawford, B., Almonacid, B., Paredes, F.: A migrating birds optimization algorithm for machine-part cell formation problems. In: Mexican International Conference on Artificial Intelligence. pp. 270–281. Springer (2015)

[17] Soto, R., Crawford, B., Almonacid, B., Paredes, F.: Efficient parallel sorting for migrating birds optimization when solving machine-part cell formation problems. Scientific Programming 2016 (2016)

[18] Soto, R., Crawford, B., Carrasco, C., Almonacid, B., Reyes, V., Araya, I., Misra, S., Olguín, E.: Solving manufacturing cell design problems by using a dolphin echolocation algorithm. In: International Conference on Computational Science and Its Applications. pp. 77–86. Springer (2016)

[19] Soto, R., Crawford, B., Castillo, C., Paredes, F.: Solving the manufacturing cell design problem via invasive weed optimization. In: Artificial Intelligence Perspectives in Intelligent Systems, pp. 115–126. Springer (2016)

[20] Soto, R., Crawford, B., Vega, E., Johnson, F., Paredes, F.: Solving manufacturing cell design problems using a shuffled frog leaping algorithm. In: The 1st International Conference on Advanced Intelligent System and Informatics (AISI2015), November 28-30, 2015, Beni Suef, Egypt. pp. 253–261. Springer (2016)

[21] Soto, R., Crawford, B., Vega, E., Paredes, F.: Solving manufacturing cell design problems using an artificial fish swarm algorithm. In: Mexican International Conference on Artificial Intelligence. pp. 282–290. Springer (2015)

[22] Soto, R., Crawford, B., Zec, C., Alarcón, A., Almonacid, B.: A bat algorithm to solve the manufacturing cell design problem. In: 2016 11th Iberian Conference on Information Systems and Technologies (CISTI). pp. 1–6. IEEE (2016)

[23] Venugopal, V., Narendran, T.: A genetic algorithm approach to the machine-component grouping problem with multiple objectives. Computers & Industrial Engineering 22(4), 469–480 (1992)

[24] Wu, T.H., Chang, C.C., Chung, S.H.: A simulated annealing algorithm for manufacturing cell formation problems. Expert Systems with Applications 34(3), 1609–1617 (2008)

[25] Yang, X.S., Deb, S.: Cuckoo search via lévy flights. In: Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on. pp. 210–214. IEEE (2009)

[26] Yin, Y., Yasuda, K.: Manufacturing cells'design in consideration of various production factors. International journal of production research 40(4), 885–906 (2002)