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

RESOLUCIÓN DEL MANUFACTURING CELL DESIGN PROBLEM UTILIZANDO FLOWER POLLINATION ALGORITHM

Michele Marco De Conti Rivara
Ronald Andrés Rubio Hurtado

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

INFORME FINAL PROYECTO DE TITULO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO CIVIL EN INFORMÁTICA

Julio 2017

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

RESOLUCION DEL MANUFACTURING CELL DESIGN PROBLEM UTILIZANDO FLOWER POLLINATION ALGORITHM

Michele Marco De Conti Rivara
Ronald Andrés Rubio Hurtado

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

Julio 2017

Dedicatoria

Michele De Conti: Este proyecto está dedicado a mi familia por otorgarme el apoyo emocional y económico para sacar adelante mi carrera universitaria y también a todos los amigos que hice durante esta larga, pero a la vez tan corta travesía.

Ronald Rubio: Dedico este proyecto de título a mi familia por brindarme apoyo incondicional durante todo el proceso de estudio. También a mis amigos, con quienes compartí la mayor parte de esta etapa. Agradezco también a mis profesores de toda etapa estudiantil, en especial a los profesores guía y co-referente quienes nos prestaron ayuda e impulso para lograr este trabajo.

Resumen

El enfoque principal del Manufacturing Cell Design Problem (MCDP) es agrupar la maquinaria de una planta productiva en celdas altamente independientes, de tal manera que piezas similares sean procesadas en la misma celda y así reducir los movimientos de las partes entre estas. Existen múltiples ventajas de realizar esto, como por ejemplo la reducción de los tiempos de producción , costos y desperdicios de material, por nombrar algunos ejemplos. La utilización del Flower Pollination Algorithm (FPA) es propuesta en este documento como la manera de encontrar la configuración óptima de las celdas, debido a que este algoritmo ha demostrado grandes capacidades en la resolución de complejos problemas. Esta aproximación tiene la dificultad añadida de encontrar los parámetros óptimos del algoritmo y por esta razón se prueban los beneficios de Autonomous Search (AS) como una manera automática y más sencilla de encontrar los mejores valores de dos de los parámetros del FPA: población y delta. Finalmente se muestran resultados experimentales, con las 94 instancias del problema de Boctor además de 70 problemas más grandes y complejos donde se observa el buen rendimiento del algoritmo y los resultados del uso de AS.

Palabras Clave: Manufacturing Cell Design, Flower Pollination Algorithm, Metaheuristicas, Optimización, Autonomous Search.

Abstract

The main focus of the Manufacturing Cell Design Problem (MCDP) is to group the machinery of a productive plant into highly independent cells, so that similar parts are processed in the same cell and thus reducing the movements of these pieces among these cells. There are many advantages of doing this, like decreasing production times, costs and product waste to name a few. The usage of the Flower Pollination Algorithm (FPA), one of the many nature-based metaheuristics, is proposed in this project as a way to find the optimum cell setup, because this algorithm has already shown great capabilities in the resolution of complex problems. This approach has the added difficulty of finding the optimal parameters for the metaheuristic and for this reason in this paper the benefits of Autonomous Search (AS) are tested as an automated and easier way to find the best values for two of the FPA parameters, the population and delta. Finally experimental results are shown, with 94 Boctor instances of the problem alongside 70 bigger and mo-

re complex problems where the good performance of the algorithm is shown besides the first results of the usage of AS.

Palabras Clave: Manufacturing Cell Design, Flower Pollination Algorithm, Metaheuristics, Optimization, Autonomous Search.

Índice

Lista de Figuras

Lista de Tablas

1. Introducción

En la industria manufacturera, una de las principales preocupaciones consiste en determinar procesos eficientes para agrupar la maquinaria productiva con la finalidad de reducir los tiempos de manipulación, los desechos industriales y mejorar la gestión del trabajo. Para esto, se ha utilizado la Agrupación Celular [5], que consiste en descomponer un sistema manufacturero en subsistemas (celdas) que son más fáciles de administrar que el sistema completo. A partir de esto surge el Manufacturing Cell Design Problem (MCDP), el cual se ocupa de agrupar las máquinas que trabajan con piezas relacionadas de un tipo de producto de la manera más eficiente posible. De esta manera se minimiza la cantidad de movimientos e intercambios de las piezas entre las celdas.

La dificultad del MCDP comienza a aumentar a medida que se agregan distintas máquinas, piezas, restricciones en la cantidad de celdas y/o la cantidad de máquinas que puede contener una celda. Debido a esta creciente complejidad se llega a cierto punto en que ya no es conveniente explorar todas las soluciones para encontrar el óptimo. Surge entonces la necesidad de utilizar un algoritmo que mediante algún proceso explore parcialmente el espacio de búsqueda de tal manera de encontrar buenas soluciones a una fracción del costo de procesamiento que tendría el realizar una búsqueda exhaustiva, aunque no necesariamente corresponderán a la solución óptima global.

Para resolver este tipo problemas de alta complejidad computacional, se ha observado el comportamiento de muchos fenómenos que la naturaleza ha perfeccionado a lo largo de millones de años y a partir de estos se han desarrollado una enorme cantidad de algoritmos. Como por ejemplo los algoritmos genéticos, basados en la evolución Darwiniana de las especies [25], los basados en enjambres de aves o peces [26, 27], incluso basados en los patrones del parpadeo de la luz que emiten las luciérnagas del trópico [55], entre muchos otros. De toda esta gama de algoritmos inspirados en la naturaleza, en este documento se estudiará el desempeño del Flower Pollination Algorithm (FPA), el cual se encuentra basado en la manera en la que las flores son polinizadas.

Estos algoritmos poseen variados parámetros que cambian la manera en la que este se comporta dentro del espacio de búsqueda, alterando los resultados que entregan, por lo que buscar la configuración óptima de estos

resulta crucial para obtener todo el potencial del algoritmo. Lograr esto es un proceso tedioso que involucra probar distintos valores y compararlos entre sí, básicamente es un problema de optimización por sí mismo. Por esta razón se ha buscado la manera de automatizar este proceso, surgiendo así Autonomous Search (AS) que toma algún tipo de retroalimentación en tiempo de ejecución para hacer ajustes a los parámetros de tal manera de encontrar mejores soluciones. En este documento se explorará la integración del FPA con AS y se entregarán resultados preliminares que serán la base de futuros trabajos.

Esta investigación se encuentra organizada de la siguiente manera: El capítulo 2 indica los objetivos. El capítulo 3 muestra el estado del arte en relación con el problema de MCDP y con FPA. El capítulo 4 explica el MCDP. El capítulo 5 detalla las características de FPA. El capítulo 6 muestra la integración entre FPA y MCDP. El capítulo 7 describe los resultados experimentales realizados al resolver el MCDP utilizando FPA. El capítulo 8 describe los resultados utilizando FPA vs FPA con Autonomous Search. Finalmente, en el capítulo 9 se muestran las conclusiones y el trabajo futuro.

2. Objetivos

2.1. Objetivo general

Resolver el Manufacturing Cell Design Problem utilizando el Flower Pollination Algorithm.

2.2. Objetivos específicos

- Comprender el fundamento teórico del FPA para aplicarlo al MCDP.
- Implementar el FPA y el MCDP en el lenguaje de programación Java.
- Experimentar y evaluar los resultados con diferentes instancias del MCDP.
- Mejorar los resultados obtenidos mediante la utilización de Autonomous Search.

3. Trabajo relacionado

La formación de celdas es un problema que ha sido foco de los investigadores por muchos años, uno de los primeros trabajos centrados en resolverlo fue el de Burbidge en 1963 [9], en donde se propone el uso de una matriz de incidencia, que es reorganizada en forma de un bloque diagonal (BDF) [54] para resolverlo. Más recientemente se han utilizado variados métodos o algoritmos basados en fenómenos naturales para resolver el MCDP, como por ejemplo los Algoritmos Genéticos (GA) [38] inspirados en la evolución biológica y su base genético-molecular, las Redes Neuronales [17] que toman el comportamientos de las neuronas y conexiones del cerebro humano como inspiración, la Programación con Restricciones [46] donde las relaciones entre las variables son expresadas en términos de restricciones, el Artificial Fish Swarm Algorith (AFSA) [48] que estudia el comportamiento inteligente de un cardumen de peces, el Shuffled Frog Leaping (SFL) [47] inspirado en las características meméticas de las ranas, donde las ranas tratan de saltar en el espacio de búsqueda en busca de un mejor resultado hasta que se cumpla la condición de detención, entre muchos otros ejemplos.

En este documento la forma elegida para resolver el problema de la formación de celdas es a través del Flower Polllination Algorithm, que es relativamente nuevo pero ya ha sido utilizado exitosamente para resolver numerosas problemáticas. Dentro de las más importantes se encuentran en el área de la distribución de energía eléctrica, como por ejemplo en el dimensionado y colocación óptima de generación distribuida en sistemas de distribución eléctrica [19], en donde se pueden reducir las pérdidas de voltaje y mejorar el perfil de tensión, valores que dependen en gran medida de los tipos, tamaños y localización de los generadores. Otras investigaciones lo han utilizado con éxito para realizar análisis de grupo [52], una técnica de análisis y minería de datos, también demostró ser superior a otros algoritmos en la resolución del Economic Load Dispatch (ELD) y del Combined Economic Emission Dispatch (CEED) [1] que son problemas sobre la entrega eficiente de energía de tal manera que se cumpla con la carga del sistema, con el menor costo posible. También fue usado [3] para encontrar los parámetros óptimos en modelos de sistemas fotovoltaicos de energía, que son utilizados para realizar estimaciones precisas sobre este tipo de sistemas, y demostró ser mejor que otras técnicas en términos de precisión y de velocidad de convergencia. FPA también fue propuesto para encontrar las posiciones óptimas y tamaños de condensadores en varios sistemas distribuidos en los cuales los flujos de energía reactiva causan pérdidas de alta potencia, caída de alta tensión y

bajo factor de potencia [2].

Este tipo de algoritmos, las metaheurísticas, poseen parámetros que pueden ser modificados en un proceso generalmente tedioso que implica muchas pruebas y toma prohibitivas cantidades de tiempo, por esta razón que en los últimos años, se ha tratado de utilizar un método automatizado para realizar esta tarea conocido como Autonomous Search (AS). AS ha sido implementado con éxito en metaheurísticas clásicas como los algoritmos evolucionaros [20], donde se usa una modificación on-line de los parámetros basado en un mecanismo de asignación de créditos, que asocia a cada operador una recompensa y una regla de selección, que determina qué operador es usado en cada paso. Otro ejemplo en optimización por enjambre de partículas [18](PSO) donde se propone un nuevo algoritmo adaptativo, llamado TRIBES, que busca evitar la modificación manual de parámetros definiendo reglas de adaptación que buscan cambiar automáticamente la topología del enjambre de acuerdo a su comportamiento y las estrategias de desplazamiento son elegidas según el rendimiento de las partículas. Los algoritmos basados en colonias de hormigas también tienen variados ejemplos de implementaciones de AS como [4, 35, 37]. Con estos trabajos como precedente, la implementación de AS en una metaheurística menos tradicional como FPA es un campo de investigación atractivo, con espacio para nuevas ideas y mejoras.

4. Manufacturing Cell Design Problem

La división conveniente de las máquinas en celdas está basada en los principios del Group Technology (GT) que fue propuesto por Flanders en 1925 [21]. Lo que busca este método de manufactura es la división de un conjunto de partes en familias que comparten similitudes en su geometría, proceso de fabricación y/o funciones, y que son producidas en un mismo lugar. A partir de la GT, se desprende la creación de celdas altamente independientes que agrupan familias de maquinaria, logrando disminuir los tiempos de preparación, trabajo, material de desperdicio, tiempo de entrega, costos de producción, entre otros. Además de incrementar la flexibilidad, mejorando la calidad del producto y el control de la producción. Sobre estos beneficios existe abundante literatura, como por ejemplo [53, 24, 34].

El encontrar la manera óptima de agrupar la maquinaria para que se dedique a procesar las familias de piezas, además de minimizar la cantidad de movimientos e intercambios de partes entre celdas son los objetivos del Manufacturing Cell Design Problem. Para poder lograr esto, el primer paso es esquematizar el problema en una matriz de incidencia, llamada parte-máquina dónde a partir de la reorganización de los puntos dentro de ésta, se pueda minimizar el número de movimientos entre celdas. Esto lo que resulta en la creación de dos nuevas matrices llamadas máquina-celda y parte-celda. A partir de este esquema se utiliza el modelo de optimización descrito a continuación [44, 7]:

- Sea M el número de máquinas.
- Sea P el número de partes.
- Sea C el número de celdas.
- i, el índice de máquinas ($i = 1, 2, ..., M$).
- j, el índice de las partes ($j = 1, 2, ..., P$).
- k, el índice de las celdas ($c = 1, 2, ..., C$).
- M_{max}, el número máximo de máquinas por cada celda.
- $A = [a_{ij}]$, sea la matriz máquina-parte de incidencia binaria, dónde:

$$a_{ij} = \begin{cases} 1 & \text{Si la máquina } i \text{ procesa la parte } j \\ 0 & \text{Otro caso} \end{cases} \quad (1)$$

- $B = [b_{ik}]$, sea la matriz máquina-celda de incidencia binaria, dónde:

$$b_{ik} = \begin{cases} 1 & \text{Si la máquina } i \text{ es asignada a la celda } k \\ 0 & \text{Otro caso} \end{cases} \tag{2}$$

- $C = [c_{jk}]$, sea la matriz parte-celda de incidencia binaria, dónde:

$$c_{jk} = \begin{cases} 1 & \text{Si la parte } j \text{ es asignada a la celda } k \\ 0 & \text{Otro caso} \end{cases} \tag{3}$$

La función objetivo del modelo matemático es la minimización del movimiento de las partes entre las celdas como se muestra en la ecuación 4.

$$min \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} a_{ij} c_{jk} (1 - b_{ik}) \tag{4}$$

Esta función objetivo está sometida a tres restricciones donde la ecuación 5 indica que cada máquina solo puede pertenecer a una única celda, la ecuación 6 asegura que cada parte es asignada a una sola celda y la ecuación 7 determina el número máximo de máquinas que una celda puede tener.

$$\sum_{k=1}^{C} b_{ik} = 1, \forall i \tag{5}$$

$$\sum_{k=1}^{C} c_{jk} = 1, \forall j \tag{6}$$

$$\sum_{i=1}^{M} b_{ik} \leq M_{max}, \forall k \tag{7}$$

5. Algoritmo Flower Pollination

Los sistemas biológicos poseen intrigantes y sorprendentes métodos para maximizar eficientemente sus objetivos y solucionar los problemas. De lo anterior, muchos algoritmos inspirados en la naturaleza han sido desarrollados durante la última década [52, 56]. Es aquí cuando nace el Flower Pollination Algorithm (FPA) [57], desarrollado por Xin-She Yang. Como su nombre lo indica se basa en la polinización de las flores, las cuales se centran en la sobrevivencia del más fuerte y la óptima reproducción de las plantas en términos de número y aptitud. Existen alrededor de un cuarto de millón de especies de plantas que producen flores lo que representa el 80 % de todas la especies de plantas. Las flores han sido muy influyentes en la evolución y sin ellas quizás el mundo no sería como lo es hoy en día.

El propósito de una flor es la reproducción vía polinización la que se lleva a cabo en asociación con diferentes tipos de especies como insectos, aves, murciélagos y otros animales, los cuales son llamados polinizadores. La polinización es realizada en dos formas, abiótica y biótica. El 90 % de las plantas que producen flores lo realiza a través de polinización biótica, es decir con la ayuda de un polinizador. El 10 % restante lo hace a través de la forma abiótica, la cual consiste en la difusión de polen a través del agua y el aire principalmente [22].

Algunos polinizadores tienden a visitar exclusivamente ciertas especies de flores, obviando en su recorrido a otras especies, esto se denomina constancia floral [16]. La constancia trae consigo ventajas evolutivas debido a que maximiza la transferencia del polen entre flores de la misma familia, aumentando las posibilidades de reproducción de la especie. A su vez, los polinizadores se benefician del proceso ya que obtienen más néctar utilizando costos mínimos de aprendizaje o exploración.

La polinización puede ser concebida ya sea por polinización cruzada o autopolinización. La polinización cruzada, se refiere a que la polinización ocurre con el polen de una flor de una planta de diferente especie, mientras que la autopolinización es la fertilización de una flor con el polen de la misma o diferentes flores, pero de la misma planta.

La polinización cruzada o biótica es producida a larga distancia, es ahí cuando los polinizadores entran en el proceso y se consideran como polinizadores globales. Además, las abejas y aves tienen a comportarse a través de

un modelado matemático denominado el vuelo de Lévy [41], y sus saltos o distancia de vuelo obedecen una distribución de Lévy. La constancia puede ser usada como un paso de incremento usando la similitud o diferencia entre dos flores.

Las características descritas anteriormente son utilizadas para construir un algoritmo que asemeje dicho comportamiento, siguiendo las siguientes reglas:

I. La polinización cruzada y biótica es considerada como un proceso de polinización global en el cual el polen es llevado por los polinizadores ejecutando un vuelo de Lévy.

II. La auto-polinización y abiótica es considerada como polinización local.

III. La constancia floral puede ser considerada como la probabilidad de reproducción que es proporcional a la similitud entre dos flores involucradas.

IV. La polinización global y local es controlada por un interruptor de probabilidad p entre 0 y 1. $p \in [0, 1]$. Debido a la proximidad física y otros factores como el viento, la polinización local puede tener una fracción p significativa en las actividades generales de polinización.

En la realidad una planta posee múltiples flores, y cada flor libera millones e incluso billones de gametos de polen. Para simplificar lo anterior, se asume que cada planta posee una flor, y que cada flor solo produce un gameto de polen. Es decir, se considera polen, gameto, flor, planta o solución ($\boldsymbol{X_i}$) como la misma cosa.

En FPA existen dos principales pasos, la polinización global y local. En la global, el polen es llevado por polinizadores (insectos, aves, murciélagos u otros), viajando distancias mucho más largas. Esto asegura la reproducción y polinización del más apto, y se denominará como $\boldsymbol{g_*}$. Matemáticamente, la primera regla junto con la constancia floral puede ser representada en la ecuación 8.

$$X_i^{t+1} = X_i^t + L(\boldsymbol{g_*} - X_i^t) \tag{8}$$

Donde X_i^{t+1} es el polen o solución X_i a la iteración t, y $\boldsymbol{g_*}$ es la actual mejor solución encontrada dentro de todas las soluciones en la generación/iteración actual. El parámetro L es la fuerza de la polinización, que es esencialmente el tamaño del salto. Ya que los insectos pueden moverse en una gran distancia con varios saltos de distancia, se puede usar el vuelo de Lévy para imitar esta característica eficientemente. Esto es, podemos trazar un $L > 0$ desde una distribución Levy (ver ecuación 9).

$$L \sim \frac{\lambda\Gamma(\lambda)sin(\pi\lambda/2)}{\pi}\frac{1}{s^1+\lambda}, \quad (S > S_0 > 0). \tag{9}$$

Donde $\Gamma(\lambda)$ es la función gamma estándar, y esta distribución es válida para pasos largos $s > 0$.

La polinización local y la constancia floral puede ser representada a través de la ecuación 10.

$$X_i^{t+1} = X_i^t + \epsilon(X_j^t - X_k^t) \tag{10}$$

,Matemáticamente si X_j^t y X_k^t son de la misma especie, o son seleccionados desde la misma vecindad, esto sería un recorrido local aleatorio si se crea un ϵ desde una distribución en $[0, 1]$.

Muchas tareas de polinización pueden ocurrir global o localmente. En la práctica, las flores adyacentes o dentro de las vecindades cercanas son más probables a ser polinizadas que las que están más lejanas. Para esto, se usa el interruptor de probabilidad que determina una polinización global o intensiva polinización local.

5.1. Pseudocódigo del Flower Pollination Algorithm

Basado en las reglas descritas anteriormente se presenta el siguiente pseudocódigo:

Algorithm 1 Flower Pollination Algorithm

Objetivo min or max $f(x), x = (x_1, x_2, ..., x_d)$
Inicializar la población de n flores/polen gametos con soluciones aleatorias
Encontrar la mejor solución $\boldsymbol{g_*}$ *en la población inicial*
Definir un interruptor de probabilidad $p \in [0, 1]$
while $(t < MaxGeneration)$ **do**
  for $i = 1 : n$ *((todas las n flores en la población)* **do**
    if $rand < p$ **then**
      Crear un vector d-dimensional de salto L que obedece una distribución Lévy.
      Polinización Global vía $X_i^{t+1} = X_i^t + L(\boldsymbol{g_*} - X_i^t)$
    else
      Crear ϵ *de una distribución uniforme en [0,1]*
      Aleatoriamente elegir j y k dentro de todas las soluciones
      Realizar una polinización local vía $X_i^{t+1} = X_i^t + \epsilon(X_j^t - X_k^t)$
    end if
    Evaluar las nuevas soluciones, si las nuevas soluciones son mejores, actualizarlas en la población
  end for
  Encontrar la actual mejor solución $\boldsymbol{g_*}$
  Llamada a la rutina de Autonomous Search
end while

6. Integración entre FPA y MCDP

La representación del MCDP más utilizado es la matricial donde, como se mencionó anteriormente, si la máquina i procesa la parte j tendrá un valor 1, sino 0. Lo anterior se puede ver en el ejemplo de la tabla 1, en donde las filas representan las máquinas y las columnas son las partes a procesar.

Tabla 1: Representación Matricial MCDP con $P = 6$ y $M = 5$

	Parte 1	Parte 2	Parte 3	Parte 4	Parte 5	Parte 6
Máquina 1	1	1	0	0	0	1
Máquina 2	0	0	1	1	0	1
Máquina 3	0	0	0	1	1	1
Máquina 4	1	1	0	0	0	0
Máquina 5	0	0	1	0	1	0

La representación de la tabla 1 se utilizará durante todo el código y en las demás representaciones, excepto cuando se aplique el movimiento de FPA. Esto, debido a que el FPA trabaja sobre el conjunto de números reales, es necesario modelar la solución con que ésta trabaja para así abordar el Manufacturing Cell Design Problem. Para ello se plantea que la solución, flor o gameto sea construido como un vector el cual representará a la matriz máquina-celda (B). Éste será del tamaño o largo de la cantidad de máquinas que el problema presente y cada valor de la posición de la máquina será la asignación a una celda. Para este efecto denotaremos el vector como V y cada valor será un v_i, donde $v_i \in [1, C]$. Así un ejemplo ilustrativo es el siguiente:

- Cantidad de celdas (C): 5.
- Cantidad de máquinas (M): 10.

Tabla 2: Representación vectorial de solución con $C = 5$ y $M = 10$

	Máquinas									
Celdas	2	5	1	2	3	4	5	1	3	3

Considerando el vector de la tabla 2, se procede a realizar los movimientos de la metaheurística anteriormente explicados. Los movimientos se aplicarán de la misma forma, pero en nuestro caso se harán en base a los vectores. Cada movimiento aplica la modificación a cada posición del vector utilizando la

ecuación 11 para el movimiento global y la ecuación 12 para el movimiento local.

$$V_j^{t+1}[k] = Aproximar(V_j^t[k] + L * (bestV[k] - V_j^t[k]) \tag{11}$$

$$V_j^{t+1}[k] = Aproximar(V_j^t[k] + E * (Vt_m[k] - V_n^t[k]) \tag{12}$$

En donde:

- t es la iteración actual
- j es la solución a modificar en su k-ésima posición
- bestV es la mejor solución de la población en la iteración actual, que también es un vector
- L el valor del paso Levy
- E el valor del paso local
- m y n soluciones elegidas aleatoriamente en la población

La función *aproximar()* toma el número generado por el movimiento (que representa a qué celda pertenece la máquina) y primero lo redondea al entero más cercano para luego corregir el rango de este, en otras palabras, si supera el máximo de celdas que posee el problema lo lleva al número celdas; o bien, si resulta un número menor a uno lo lleva a uno. Esto es explicado en la siguiente ecuación:

$$Aproximar(x) = \begin{cases} 1 & \text{if } x < 1 \\ x & \text{if } x >= 1 \text{ and } x <= C \\ C & \text{if } x > C \end{cases} \tag{13}$$

Donde x es el número entregado por el movimiento aproximado al entero más cercano, y C la cantidad de celdas del problema. Luego de aplicado el movimiento, el vector es transformado nuevamente en la matriz de asignación binaria.

6.1. Reparación de soluciones

En algunas ocasiones, al aplicar el movimiento local o global, la solución generada no cumple con las restricciones y es necesario repararla, por esta razón, se utiliza una matriz de similitud que representa cuantas partes poseen en común cada máquina con otra, . Si la cantidad de máquinas es n, la matriz similitud será nxn, como se puede ver en la tabla 3.

Tabla 3: Representación Matricial Matriz similitud con $M = 4$

	Máquina 1	Máquina 2	Máquina 3	Máquina 4
Máquina 1	-	1	3	0
Máquina 2	1	-	0	2
Máquina 3	3	0	-	0
Máquina 4	0	2	0	-

De esta forma, la máquina 1 con la máquina 3 posee mayor similitud que con la máquina 4, debido a que con la máquina 3 comparte 3 partes y con la máquina 4 comparte ninguna.

En la tabla 4, se muestra el incumplimiento de algunas restricciones. Las filas representan las máquinas, mientras que las columnas celdas. A la izquierda no se cumple la restricción donde una máquina debe estar asignada a una sola celda, mientras que a la derecha se muestra el incumplimiento de la restricción del numero máximo de máquinas ($Mmax = 2$).

En ambos casos, la nueva asignación de la máquina a la celda es decidido en base a cálculos sobre la matriz de similitud. Para la primera matriz (izquierda), la máquina 3 quedará asignada a la celda 1 ya que tiene mayor similitud que con la máquina 2. En la matriz derecha, la máquina a mover es la que tiene menor similitud dentro del grupo de la celda, es decir la máquina 2. El calculo de este procedimiento se basa en la sumatoria de similitud de cada máquina en el grupo celda, reasignándose el menor a otra celda donde haya mayor similitud.

Finalmente, la matriz parte-celda se deduce en base a la matriz máquina-celda, no es necesario transformarla para el movimiento.

Tabla 4: Tablas de incumplimiento de restricciones

	Celda 1	Celda 2
Máquina 1	1	0
Máquina 2	0	1
Máquina 3	1	1
Máquina 4	0	1

	Celda 1	Celda 2
Máquina 1	0	1
Máquina 2	0	1
Máquina 3	0	1
Máquina 4	1	0

7. Resultados experimentales

La implementación de FPA con MCDP fue realizada en el lenguaje de programación Java versión 1.8 y ejecutado en un notebook con procesador Intel Core i5 3230M (2600 MHz - 3200 MHz), 8GB RAM DDR3 1600 MHz, con Windows 10. Se probaron 10 problemas (90 instancias considerando 5 valores de máximo de máquinas para 2 celdas y 10 instancias considerando 4 valores de máximo de máquinas para 3 celdas). Los parámetros de la metaheurística utilizados en las pruebas y con los que se obtuvieron buenos resultados son: n=160, *iteraciones*=100, p=0.1, *delta*=1.5. Finalmente, cada instancia de prueba fue ejecutada un total de 31 veces.

En las tablas 6 y 7 se contrastan los valores obtenidos por la metaheurística versus el óptimo global para cada problema. Además, se presentan comparaciones con respecto a otras técnicas. La primera columna representa el problema de Boctor [7], la segunda columna el número máximo de máquinas por celda, la tercera columna corresponde al óptimo global. Las siguientes tres columnas son los resultados obtenidos por FPA, el mejor óptimo encontrado, el promedio de estos y el RPD %. Las últimas columnas contienen los resultados de diferentes técnicas como Migrating Birds Optimization (MBO), Simulated Annealing (SA) y Particle Swarm Optimization (PSO) respectivamente. Como se puede apreciar, el algoritmo FPA alcanza los óptimos globales en todos los problemas de Boctor, con un RPD % de 0 en todos los casos y con promedios que se acercan en gran medida al óptimo global.

En las tablas 8 y 9 se describen los resultados con otras instancias de pruebas (ver tabla 5) en donde se puede apreciar que el algoritmo FPA alcanza el óptimo global en todas los problemas que poseen un óptimo global conocido.

Tabla 5: Referencias a las nuevas instancias de prueba

Problem	Source	M	P	Mmax
CFP01	King-Nakornchai [29]	5	7	3
CFP02	Waghodekar-Sahu [51]	5	7	3
CFP03	Seifoddini [42]	5	18	3
CFP04	Kusiak-Cho [32]	6	8	3
CFP05	Kusiak-Chow [33]	7	11	4
CFP06	Boctor [7]	7	11	4
CFP07	Seifoddini-Wolfe [43]	8	12	4
CFP08	Chandrasekharan-Rajagopalan [13]	8	20	4
CFP09	Chandrasekharan-Rajagopalan [12]	8	20	4
CFP10	Mosier-Taube [39]	10	10	5
CFP11	Chan and Milner [11]	10	15	5
CFP12	Askin-Subramanian [6]	14	24	7
CFP13	Stanfel [50]	14	24	7
CFP14	McCormick [36]	16	24	8
CFP15	Srinivasan [49]	16	30	8
CFP16	King [28]	16	43	8
CFP17	Carrie [10]	18	24	9
CFP18	Mosier-Taube [40]	20	20	10
CFP19	Kumar [30]	20	23	10
CFP20	Carrie [10]	20	35	10
CFP21	Boe-Cheng [8]	20	35	10
CFP22	Chandrasekharan-Rajagopalan [15]	24	40	12
CFP23	Chandrasekharan-Rajagopalan [15]	24	40	12
CFP24	Chandrasekharan-Rajagopalan [15]	24	40	12
CFP25	Chandrasekharan-Rajagopalan [15]	24	40	12
CFP26	Chandrasekharan-Rajagopalan [15]	24	40	12
CFP27	Chandrasekharan-Rajagopalan [15]	24	40	12
CFP28	McCormick [36]	27	27	14
CFP29	Carrie [10]	28	46	14
CFP30	Kumar-Vannelli [31]	30	41	15
CFP31	Stanfel [50]	30	50	15
CFP32	Stanfel [50]	30	50	15
CFP33	King-Nakornchai [29]	36	90	18
CFP34	McCormick [36]	37	53	19
CFP35	Chandrasekharan-Rajagopalan [14]	40	100	20

Finalmente en las figuras 1 y 2 se observa que FPA tiene una convergencia rápida al óptimo global. Esto, debido a que su comportamiento en base a soluciones vecinas o lejanas crea una nueva solución, asemejándose a movimientos de mutaciones. Para la instancia 56 el algoritmo FPA converge al óptimo a la iteración número 11 y en el caso de la instancia 71 se llega al óptimo global en la iteración número 12.

Tabla 6: Experimentos de Boctor usando $C = 2$.

Boctor	Mmax	Valor	FPA			SA	PSO	MBO	SFLA
Problem		Óptimo	Óptimo	Promedio	RPD %				
1	8	11	11	11,40	0	11	11	11	11
1	9	11	11	11,03	0	11	11	11	11
1	10	11	11	11,07	0	11	11	11	11
1	11	11	11	11,17	0	11	11	11	11
1	12	11	11	11,30	0	11	11	11	11
2	8	7	7	7,07	0	7	7	7	7
2	9	6	6	6,33	0	6	6	6	6
2	10	4	4	4,77	0	10	5	4	4
2	11	3	3	4,07	0	4	4	3	3
2	12	3	3	3,33	0	3	4	3	3
3	8	4	4	4,43	0	5	5	4	4
3	9	4	4	4,37	0	4	4	4	4
3	10	4	4	4,23	0	4	5	4	4
3	11	3	3	3,80	0	4	4	3	3
3	12	1	1	2,17	0	4	3	1	1
4	8	14	14	14,00	0	14	15	14	14
4	9	13	13	13,10	0	13	13	13	13
4	10	13	13	13,23	0	13	13	13	13
4	11	13	13	13,47	0	13	13	13	13
4	12	13	13	13,00	0	13	13	13	13
5	8	9	9	9,73	0	9	10	9	9
5	9	6	6	6,57	0	6	8	6	6
5	10	6	6	6,90	0	6	6	6	6
5	11	5	5	5,90	0	7	5	5	5
5	12	4	4	5,10	0	4	5	4	4
6	8	5	5	5,17	0	5	5	5	5
6	9	3	3	3,70	0	3	3	3	3
6	10	3	3	3,80	0	5	3	3	3
6	11	3	3	3,43	0	3	4	3	3
6	12	2	2	3,10	0	3	4	2	2
7	8	7	7	7,13	0	7	7	7	7
7	9	4	4	4,17	0	4	5	4	4
7	10	4	4	4,30	0	4	5	4	4
7	11	4	4	4,57	0	4	5	4	4
7	12	4	4	4,83	0	4	5	4	4
8	8	13	13	13,10	0	13	14	13	13
8	9	10	10	10,93	0	20	11	10	10
8	10	8	8	8,63	0	15	10	8	8
8	11	5	5	6,60	0	11	6	5	5
8	12	5	5	5,70	0	7	6	5	5
9	8	8	8	9,77	0	13	9	8	8
9	9	8	8	8,40	0	8	8	8	8
9	10	8	8	8,63	0	8	8	8	8
9	11	5	5	6,50	0	8	5	5	5
9	12	5	5	6,87	0	8	8	5	5
10	8	8	8	8,37	0	8	9	8	8
10	9	5	5	6,30	0	5	8	5	5
10	10	5	5	5,63	0	5	7	5	5
10	11	5	5	5,80	0	5	7	5	5
10	12	5	5	5,83	0	5	6	5	5

Tabla 7: Experimentos de Boctor usando $C = 3$

Boctor Problem	Mmax	Valor Óptimo	FPA			SA	PSO	MBO	SFLA
			Óptimo	Promedio	RPD %				
1	6	27	27	27,57	0	28	-	27	-
1	7	18	18	18,07	0	18	-	18	-
1	8	11	11	13,23	0	11	-	11	-
1	9	11	11	12,13	0	11	-	11	-
2	6	7	7	7,27	0	7	-	7	-
2	7	6	6	6,07	0	6	-	6	-
2	8	6	6	7,77	0	7	-	6	-
2	9	6	6	6,93	0	12	-	6	-
3	6	9	9	9,23	0	8	-	9	-
3	7	4	4	4,07	0	8	-	4	-
3	8	4	4	4,57	0	4	-	4	-
3	9	4	4	4,77	0	27	-	4	-
4	6	27	27	27,03	0	18	-	27	-
4	7	18	18	18,00	0	14	-	18	-
4	8	14	14	14,73	0	13	-	14	-
4	9	13	13	14,13	0	11	-	13	-
5	6	11	11	11,70	0	9	-	11	-
5	7	8	8	8,60	0	9	-	8	-
5	8	8	8	10,03	0	8	-	8	-
5	9	6	6	7,47	0	8	-	6	-
6	6	6	6	6,83	0	5	-	6	-
6	7	4	4	4,10	0	5	-	4	-
6	8	4	4	4,50	0	4	-	4	-
6	9	3	3	4,17	0	11	-	3	-
7	6	11	11	11,60	0	5	-	11	-
7	7	5	5	5,27	0	5	-	5	-
7	8	5	5	6,07	0	5	-	5	-
7	9	4	4	4,20	0	14	-	4	-
8	6	14	14	14,47	0	11	-	14	-
8	7	11	11	11,63	0	11	-	11	-
8	8	11	11	12,70	0	10	-	11	-
8	9	10	10	11,67	0	12	-	10	-
9	6	12	12	12,60	0	12	-	12	-
9	7	12	12	12,53	0	13	-	12	-
9	8	8	8	8,40	0	8	-	8	-
9	9	8	8	8,60	0	8	-	8	-
10	6	10	10	10,13	0	10	-	10	-
10	7	8	8	8,57	0	8	-	8	-
10	8	8	8	8,93	0	8	-	8	-
10	9	5	5	7,30	0	5	-	5	-

Tabla 8: Nuevas instancias de prueba usando C = 2

Autor del problema	Mmax	Óptimo Global	Óptimo Encontrado	Óptimo Promedio	RPD %
King-Nakornchai	3	0	0	0,00	0
Waghodekar-Sahu	3	5	5	5,00	0
Seifoddini	3	5	5	5,00	0
Kusiak-Cho	3	2	2	2,00	0
Kusiak-Chow	4	3	3	3,00	0
Boctor	4	2	2	2,00	0
Seifoddini-Wolfe	4	6	6	6,00	0
Chandrasekharan-Rajagopalan	4	7	7	7,00	0
Chandrasekharan-Rajagopalan	4	28	28	28,00	0
Mosier-Taube	5	1	1	1,00	0
Chan-and-Milner	5	4	4	4,00	0
Askin-Subramanian	7	1	1	1,19	0
Stanfel	7	2	2	2,00	0
McCormick	8	16	16	16,55	0
Srinivasan	8	12	12	12,32	0
King	8	15	15	16,00	0
Carrie	9	13	13	13,19	0
Mosier-Taube	10	27	27	28,77	0
Kumar	10	25	25	25,52	0
Carrie	10	1	1	3,26	0
Boe-Cheng	10	-	21	24,06	-
Chandrasekharan-Rajagopalan	12	-	0	2,77	-
Chandrasekharan-Rajagopalan	12	-	3	8,03	-
Chandrasekharan-Rajagopalan	12	-	9	12,48	-
Chandrasekharan-Rajagopalan	12	-	19	21,65	-
Chandrasekharan-Rajagopalan	12	-	22	23,74	-
Chandrasekharan-Rajagopalan	12	-	22	23,23	-
McCormick	14	-	32	32,61	-
Carrie	14	-	36	38,39	-
Kumar-Vannelli	15	-	5	7,87	-
Stanfel	15	-	5	9,19	-
Stanfel	15	-	25	28,65	-
King-Nakornchai	15	-	40	43,29	-
McCormick	19	-	212	228,65	-
Chandrasekharan-Rajagopalan	20	-	13	29,55	-

Tabla 9: Nuevas instancias de prueba usando C = 3

Autor del problema	Mmax	Óptimo Global	Óptimo Encontrado	Óptimo Promedio	RPD %
King-Nakornchai	2	2	2	2,00	0
Waghodekar-Sahu	2	8	8	8,00	0
Seifoddini	2	11	11	11,00	0
Kusiak-Cho	2	7	7	7,00	0
Kusiak-Chow	3	5	5	5,00	0
Boctor	3	2	2	2,00	0
Seifoddini-Wolfe	3	7	7	7,00	0
Chandrasekharan-Rajagopalan	3	14	14	14,00	0
Chandrasekharan-Rajagopalan	3	39	39	39,00	0
Mosier-Taube	4	0	0	0,00	0
Chan-and-Milner	4	0	0	0,00	0
Askin-Subramanian	5	2	2	2,26	0
Stanfel	5	2	2	2,00	0
McCormick	6	22	22	22,42	0
Srinivasan	6	17	17	17,19	0
King	6	-	21	22,32	-
Carrie	6	-	18	19,74	-
Mosier-Taube	7	-	41	42,65	-
Kumar	7	-	34	36,13	-
Carrie	7	-	13	16,61	-
Boe-Cheng	7	-	35	39,29	-
Chandrasekharan-Rajagopalan	8	-	3	8,74	-
Chandrasekharan-Rajagopalan	8	-	4	15,03	-
Chandrasekharan-Rajagopalan	8	-	16	21,84	-
Chandrasekharan-Rajagopalan	8	-	32	35,90	-
Chandrasekharan-Rajagopalan	8	-	34	37,71	-
Chandrasekharan-Rajagopalan	8	-	33	37,35	-
McCormick	9	-	72	76,06	-
Carrie	10	-	53	56,71	-
Kumar-Vannelli	10	-	9	16,94	-
Stanfel	10	-	12	22,32	-
Stanfel	10	-	42	48,10	-
King-Nakornchai	12	-	47	56,39	-
McCormick	13	-	315	352,87	-
Chandrasekharan-Rajagopalan	14	-	43	65,10	-

z

Figura 1: Gráfico de convergencia con C = 3 y M = 6.

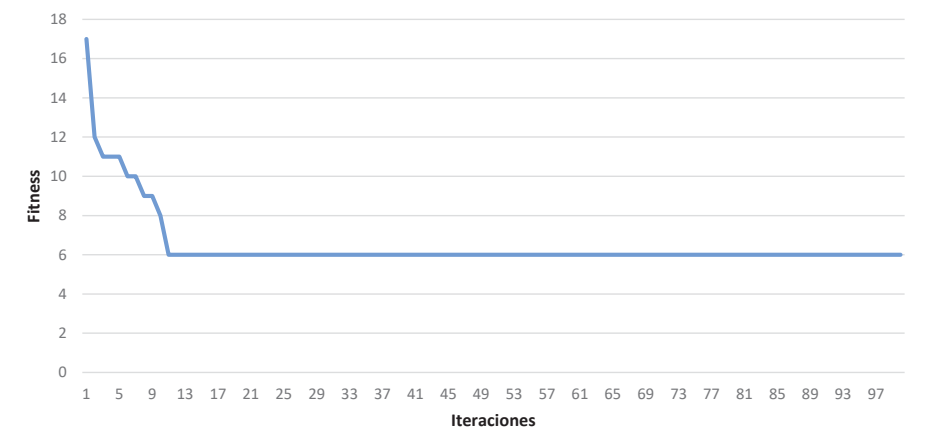

Figura 2: Gráfico de convergencia con C = 3 y M = 7.

8. Autonomous Search

Las metaheurísticas no son una ciencia exacta, su comportamiento está definido por una serie de parámetros que normalmente son ajustados a priori y cuya modificación tiene un impacto incierto en la búsqueda de la solución. Debido a esto se busca una manera automática de ajustar estos parámetros de la manera más eficiente posible, pero esto no es algo sencillo, de hecho se puede decir que es un problema de optimización por sí mismo. Una vez explicado esto, la definición de Autonomous Search (AS) [23] propuesta en este documento será la siguiente: "AS es un sistema que tiene la habilidad de modificar los componentes internos del algoritmo cuando es expuesto a fuerzas externas y oportunidades, con el objetivo de mejorar el rendimiento de este en la resolución de problemas. Para lograr esto adapta la estrategia de búsqueda según las características del problema".

El objetivo de Autonomous Search es mejorar los resultados del algoritmo, esta mejora puede ser cuantificada en distintos parámetros, como por ejemplo en el tiempo de ejecución, la cantidad de iteraciones para converger al óptimo o al disminuir el error relativo entre la respuesta entregada y el óptimo global. Para efectos de este documento los principales objetivos son mejorar la robustez del algoritmo, haciendo que converja la mayor cantidad de veces posible al óptimo global, y también reducir los tiempos de ejecución.

8.1. Implementación de Autonomous Search con FPA

El algoritmo Flower Pollination posee tres parámetros que pueden ser modificados y/o ajustados, el primero es λ que es usado en la función Gamma dentro de la distribución de Lévy, por lo que afecta la forma en la que se genera el vector de salto L (indica qué tanto se aleja la nueva solución con cada iteración). El segundo es la cantidad de flores o soluciones que existen en cada iteración. Finalmente, se tiene el interruptor de probabilidad p que es el que determina cuando se hace un paso local o uno global.

La forma de modificar estos parámetros está basada en la cantidad de iteraciones en las que no se mejora la solución, por ejemplo después de cinco iteraciones, se empieza por aumentar el valor del parámetro y si después de cinco iteraciones más se sigue sin mejorar, se cambia la tendencia y se comienza a reducir el valor hasta que se logre mejorar la solución actual. La idea es que si la solución se está mejorando es mantener la tendencia (aumento o

reducción), pero si después de varias modificaciones no se logra este objetivo, se invierte en la búsqueda de mejores resultados. Este comportamiento es explicado en el siguiente pseudo código:

8.2. Pseudo código de Autonomous Search

Algorithm 2 Autonomous Search algorithm

Definir la cantidad de iteraciones ***n*** *a esperar antes de modificar*
Definir la cantidad de iteraciones ***m*** *donde no se mejora la solución*
Definir el tamaño de la modificación ***L***
Definir un interruptor de inicio ***p*** *0 or 1*
Sea ***i*** *el número de iteraciones sin modificación en la solución*
Inicio del algoritmo de AS
if i>n Iteraciones sin mejoras **then**
  if **p**=0, aumentar el parámetro **then**
    Aumentar en ***L*** *el parámetro definido*
    if i> modificaciones del parámetro sin mejora de la solución **then**
      Cambiar el interruptor a 1 (para comenzar a disminuir el parámetro)
      Reiniciar el contar de iteraciones sin modificación
    end if
  end if
  if **p**=1, disminuir el parámetro **then**
    Disminuir el parámetro en ***L***
    if i>=m **then**
      Cambiar el interruptor a 1 (para comenzar a disminuir el parámetro)
      Reiniciar el contar de iteraciones sin modificación
    end if
  end if
end if

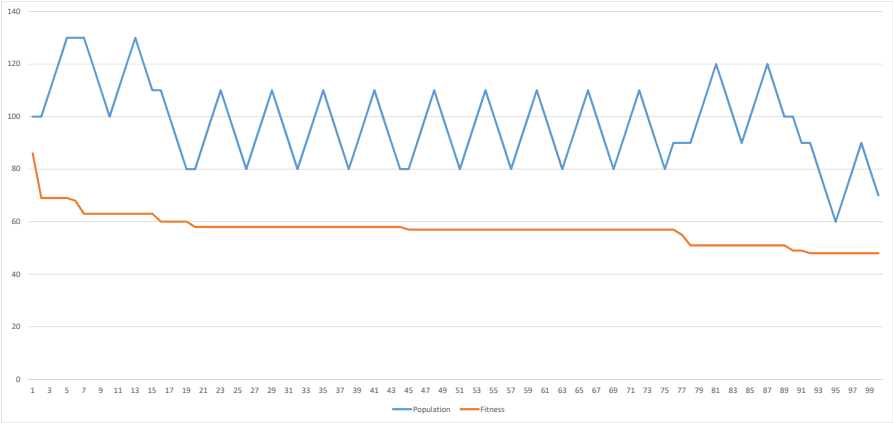

Figura 3: Fitness comparado con la población

Finalmente para mostrar gráficamente como el anteriormente mencionado algoritmo modifica los parámetros, en la figura 3 se compara el fitness y la población en el algoritmo en el segundo problema de King Nakornchai (tres celdas, 12 el máximo de máquinas por celda) con una configuración de AS donde n=1, m=2 y aumenta en 10 la población en cada modificación. Como puede observarse la población solo mantiene su cantidad durante una iteración cuando el algoritmo hace progreso reduciendo el fitness, mientras que cuando se estanca trata de encontrar una nueva cantidad de población de tal manera de encontrar mejores soluciones.

Para comparar las distintas configuraciones se eligió el promedio de todos los óptimos encontrados en las 31 iteraciones de cada problema de Boctor y también la cantidad de iteraciones que en promedio se necesitaron para llegar a ese resultado. A continuación se presenta en la tabla 10 un resumen con los resultados obtenidos. Se muestran dos resultados distintos de cada parámetro donde el segundo tiene incrementos más extremos que el primero.

Tabla 10: Resumen resultados Autonomous Search

	Base	λ 1	λ 2	Pob. 1	Pob. 2	Switch 1	Switch 2
Ópt. prom.	8,199	7,959	7,848	8,203	8,155	8,679	9,019
It. prom.	11,577	12,735	15,03	11,982	12,577	9,34	8,097

En esta tabla en la primera columna se observa los resultados (el pro-

medio de los óptimos y de las iteraciones hasta encontrarlo) con la mejor configuración del algoritmo, encontrada manualmente. Las siguientes columnas muestran, respectivamente, dos configuraciones distintas de AS de λ, de la población y del switch de probabilidad.

Se puede apreciar que la modificación de la población básicamente no tiene efectos en los resultados obtenidos. En el caso del switch de probabilidad los promedios de los óptimos obtenidos tienden a aumentar un 5-10 %, pero la cantidad de iteraciones necesaria disminuye un 25-30 %, finalmente con la modificación del λ se obtiene una mejora en la resultados obtenidos a expensas de un ligero aumento en la cantidad de iteraciones. Basados en estos resultados, se cambiará en un futuro trabajo el enfoque de AS con respecto de la población, ya que si bien los fitness no mejoran, puede dar lugar para alcanzar resultados de igual calidad, pero con una menor población inicial. Esto se traduciría en menores tiempos de ejecución debido a que este parámetro es crítico en este sentido, ya que mientras más aumenta la población, mas largos son los cálculos a realizar.

9. Conclusiones

En el desarrollo de esta investigación se resolvió el Manufacturing Cell Design Problem, que se encarga de la colocación eficiente de la maquinaria en una planta manufacturera. Se abordó esta problemática mediante la utilización del Flower Pollination Algorithm, una metaheurística que ha demostrado grandes capacidades para resolver complejos problemas en diversas áreas, como en Data Mining o en la generación eficiente de energía eléctrica en un sistema distribuido. A pesar de que el problema ha sido ampliamente investigado a lo largo de los años, sigue siendo una excelente manera de comprobar las bondades del algoritmo.

Una vez analizado tanto el problema como el algoritmo, se realizó con una exitosa integración entre ambos. Esto se puede apreciar en los resultados experimentales, el FPA logró obtener óptimos globales en todas las instancias en las que este es conocido, logrando esto con una rápida convergencia y unos reducidos tiempos de ejecución. Cabe destacar que estos resultados fueron obtenidos luego de un largo proceso de pruebas, donde se logró una buena configuración de los distintos parámetros del algoritmo en base a experimentación.

Como se mencionó en el párrafo anterior, en una primera instancia los parámetros de la metaheurística fueron solamente calibrados con experimentación. Es por ello que fue necesario implementar Autonomous Search, un método de optimización a las variables influyentes en tiempo real y con ello mejorar el rendimiento en óptimos y tiempos de ejecución. Estos prometedores resultados preliminares se muestran en las tablas y gráficos anteriormente expuestos, los cuales dan pie a un futuro trabajo donde estos puedan ser mejorados aún más.

Finalmente, se ha realizado una publicación científica basada en este trabajo en el *International Workshop on Multi-disciplinary Trends in Artificial Intelligence* [45].

Referencias

[1] A. Abdelaziz, E. Ali, and S. Elazim. Combined economic and emission dispatch solution using flower pollination algorithm. *International Journal of Electrical Power and Energy Systems*, 80:264–274, 2016.

[2] A. Abdelaziz, E. Ali, and S. Elazim. Flower pollination algorithm and loss sensitivity factors for optimal sizing and placement of capacitors in radial distribution systems. *International Journal of Electrical Power & Energy Systems*, 78:207–214, 2016.

[3] D. Alam, D. Yousri, and M. Eteiba. Flower pollination algorithm based solar pv parameter estimation. *Energy Conversion and Managemente*, 111:410–422, 2015.

[4] L. Albuquerque, F. Baptista, and E. Costa. C-ant: A multi-colony ant algorithm. *EA*, volume 5975 of LNCS:25–36, 2009.

[5] K. Andrew and S. Wing. Decomposition of manufacturing systems. *IEEE Journal of Robotics and Automation*, pages 457–471, 1988.

[6] R. Asktn and P. Subramantan. A cost-based heuristic for group technology configuration. *International Journal of Production Research*, 25(1):101–113, 1987.

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

[8] W. Boe and C. Cheng. A close neighbour algorithm for designing cellular manufacturing systems. *The International Journal of Production Research*, 29(10):2097–2116, 1991.

[9] J. Burbidge. Production flow analysis for planning group technology. *Production Engineer*, pages 742–752, 1991.

[10] A. Carrie. Numerical taxonomy applied to group technology and plant layout. *The International Journal of Production Research*, 11(4):399–416, 1973.

[11] H. Chan and D. Milner. Direct clustering algorithm for group formation in cellular manufacture. *Journal of Manufacturing systems*, 1(1):65–75, 1982.

[12] M. Chandrasekharan and R. Rajagopalan. An ideal seed non-hierarchical clustering algorithm for cellular manufacturing. *International Journal of Production Research*, 24(2):451–463, 1986.

[13] M. Chandrasekharan and R. Rajagopalan. Modroc: an extension of rank order clustering for group technology. *International Journal of Production Research*, 24(5):1221–1233, 1986.

[14] M. Chandrasekharan and R. Rajagopalan. Zodiac-an algorithm for concurrent formation of part-families and machine-cells. *International Journal of Production Research*, 25(6):835–850, 1987.

[15] M. Chandrasekharan and R. Rajagopalan. Groupabil1ty: an analysis of the properties of binary data matrices for group technology. *The International Journal of Production Research*, 27(6):1035–1052, 1989.

[16] L. Chittka, J. Thomson, and N. Waser. Flower constancy, insect psychology, and plant evolution. *Naturwissenschaften*, 86:367–377, 1999.

[17] M. Christodoulou and V. Gaganis. Neural networks in manufacturing cell design. *Computers in Industry*, pages 133–138, 1998.

[18] Y. Cooren, M. Clerc, and P. Siarry. Performance evaluation of tribes, an adaptive particle swarm optimization algorithm. *Swarm Intelligence*, page 149–178, 2009.

[19] P. Dinakara, V. Veera, and T. gowri. Application of flower pollination algorithm for optimal placement and sizing of distributed generation in distribution systems. *Journal of Electrical Systems and Information Technology*, pages 14–22, 2016.

[20] A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Extreme value based adaptive operator selection. *PPSN*, volume 5199 of LNCS:175–184, 2008.

[21] R. Flanders. Design, manufacture, and production control of a standard machine. *Transactions of ASME*, 46, 1925.

[22] B. Glover. Understanding flowers and flowering: An integrated approach. *Oxford University Press*, 2009.

[23] Y. Hamadi, E. Monfroy, and F. Saubion. Autonomous search. *Academic Press*, 2012.

[24] S. Heragu. Group technology and cellular manufacturing. *IEEE Transactions of Systems, Man, and Cybernetics*, pages 203–215, 1994.

[25] J. Holland. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. *University of Michigan Press*, page 183, 1975.

[26] J. Kennedy and R. Eberhart. Particle swarm optimization. *Proc. of IEEE International Conference on Neural Networks, Piscataway*, pages 1942–1948, 1995.

[27] J. Kennedy, R. Eberhart, and Y. Shi. Swarm intelligence. *Academic Press*, 2001.

[28] J. King. Machine-component grouping in production flow analysis: an approach using a rank order clustering algorithm. *International Journal of Production Research*, 18(2):213–232, 1980.

[29] J. King and V. Nakornchai. Machine-component group formation in group technology: review and extension. *The International Journal of Production Research*, 20(2):117–133, 1982.

[30] K. Kumar, A. Kusiak, and A. Vannelli. Grouping of parts and components in flexible manufacturing systems. *European Journal of Operational Research*, 24(3):387–397, 1986.

[31] K. Kumar and A. Vannelli. Strategic subcontracting for efficient disaggregated manufacturing. *BEBR faculty working paper; no. 1252*, 1986.

[32] A. Kusiak and M. Cho. Similarity coefficient algorithms for solving the group technology problem. *The International Journal Of Production Research*, 30(11):2633–2646, 1992.

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

[34] S. Mansouri, S. Moattar, and S. Newman. A review of the modern approaches to multi-criteria cell design. *International Journal of Production Research*, pages 1201–1218, 2010.

[35] D. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens. Classification with ant colony optimization. *Evolutionary Computation*, page 651–665, 2009.

[36] W. McCormick Jr, P. Schweitzer, and T. White. Problem decomposition and data reorganization by a clustering technique. *Operations Research*, 20(5):993–1009, 1972.

[37] D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. *Evolutionary Computation*, volume 5199 of LNCS:333–346, 2002.

[38] C. Moon and M. Gen. Genetic algorithm-based approach for design of independent manufacturing cells. *Production Economics*, pages 421–426, 1999.

[39] C. Mosier and L. Taube. The facets of group technology and their impacts on implementation-a state-of-the-art survey. *Omega*, 13(5):381–391, 1985.

[40] C. Mosier and L. Taube. Weighted similarity measure heuristics for the group technology machine clustering problem. *Omega*, 13(6):577–579, 1985.

[41] I. Pavlyukevich. Lévy flights, non-local search and simulated annealing. *Computational Physics*, 226:1830–1844, 2007.

[42] H. Seifoddini. A note on the similarity coefficient method and the problem of improper machine assignment in group technology applications. *The International Journal of Production Research*, 27(7):1161–1165, 1989.

[43] H. Seifoddini and P. Wolfe. Application of the similarity coefficient method in group technology. *IIE transactions*, 18(3):271–277, 1986.

[44] R. Soto, B. Crawford, J. Lama, and F. Paredes. A firefly algorithm to solve the manufacturing cell design problem. *Advances in Intelligent Systems and Computing*, pages 103–114, 2016.

[45] R. Soto, B. Crawford, R. Olivares, M. De Conti, R. Rubio, B. Almonacid, and S. Niklander. Resolving the manufacturing cell design problem using the flower pollination algorithm. In *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, pages 184–195. Springer, 2016.

[46] R. Soto, B. Crawford, E. Vega, and F. Paredes. Solving manufacturing cell design problems using constraint programming. *Volume 7345 of the series Lecture Notes in Computer Science*, pages 400–406, 2012.

[47] R. Soto, B. Crawford, E. Vega, and F. Paredes. Solving manufacturing cell design problems using a shuffled frog leaping algorithm. *Beni Suef, Egypt Volume 407 of the series Advances in Intelligent Systems and Computing*, pages 253–261, 2015.

[48] R. Soto, B. Crawford, E. Vega, and F. Paredes. Solving manufacturing cell design problems using an artificial fish swarm algorithm. *Volume 9413 of the series Lecture Notes in Computer Science*, pages 282–290, 2015.

[49] G. Srinlvasan, T. Narendran, and B. Mahadevan. An assignment model for the part-families problem in group technology. *The International Journal of Production Research*, 28(1):145–152, 1990.

[50] L. Stanfel. Machine clustering for economic production. *Engineering costs and production economics*, 9(1):73–81, 1985.

[51] P. Waghodekar and S. Sahu. Machine-component cell formation in group technology: Mace. *The International Journal of Production Research*, 22(6):937–948, 1984.

[52] R. Wang and Y. Zhou. Flower pollination algorithm with bee pollinator for cluster analysis. *Information Processing Letters*, 116:1–14, 2016.

[53] U. Wemmerlov and H. Nancy. Cellular manufacturing in the u.s. industry: A survey of users. *International Journal of Production Research*, pages 1511–1530, 1988.

[54] A. Xambre and P. Vilarinho. A simulated annealing approach for manufacturing cell formation with multiple identical machines. *European Journal of Operational Research*, pages 434–446, 2011.

[55] X. Yang. Firefly algorithms for multimodal optimization. *In Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications*, 285:169–178, 2009.

[56] X. Yang. Nature-inspired metaheuristic algorithms. *Luniver Press*, 2010.

[57] X. Yang. Flower pollination algorithm for global optimization. *Unconventional Computation and Natural Computation*, 7445:240–249, 2012.