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

MANUFACTURING CELL DESIGN PROBLEM UTILIZANDO AFSA

JACQUELINE EDITH LAMA GUERRA

EMANUEL ENRIQUE VEGA MENA

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO DE EJECUCIÓN EN INFORMÁTICA

ABRIL 2014

Pontificia Universidad Católica de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Informática

MANUFACTURING CELL DESIGN PROBLEM UTILIZANDO AFSA

JACQUELINE EDITH LAMA GUERRA

EMANUEL ENRIQUE VEGA MENA

Profesor Guía: **Ricardo Soto De Giorgis**

Profesor Co-referente: **Wenceslao Palma Muñoz**

Carrera: **Ingeniería de Ejecución en Informática**

Abril 2014

Dedicado a nuestras familias, quienes nos brindaron su apoyo y cariño incondicional. A nuestros amigos, quienes nos animaron a continuar cuando las noches se tornaban largas. Y a nuestros profesores, quienes nos entregaron las herramientas para formarnos como profesionales.

Índice

Resumen

El diseño de celdas de manufactura es una estrategia de manufacturación que consiste en la creación de un diseño óptimo de plantas de producción, cuyo objetivo principal reside en minimizar los movimientos e intercambio de material entre estas celdas. El propósito del presente proyecto consiste, entonces, en modelar y resolver el problema de diseño de celdas de manufactura, con el fin de reducir costos de producción y aumentar la productividad. Lo anterior se resuelve por medio de un algoritmo de búsqueda, el cual corresponde al algoritmo de cardumen artificial como metaheurística, la cual trata de encontrar o acercarse al óptimo global, minimizando los movimientos al interior de la celda, y entre éstas.

Palabras-claves: diseño de celdas de manufactura, algoritmo de cardumen artificial, manufacturing cell design problem, artificial fish-swarm algorithm, MCDP, AFSA.

Abstract

The design of manufacturing cells is a manufacturing strategy that involves the creation of an optimal design of production plants, whose main objective is to minimize movements and exchange of material between these cells. The purpose of this project is, then, to model and solve the manufacturing cell design problem, in order to reduce production costs and increase productivity. This is resolved by a search algorithm, which corresponds to the artificial fish-swarm algorithm as metaheuristic, which tries to find or approach the global optimum, minimizing movements within the cell, and between them.

Key-words: manufacturing cell design problem, artificial fish-swarm algorithm, MCDP, AFSA.

Lista de Figuras

Lista de Tablas

Lista de Algoritmos

1 Introducción

La tecnología de agrupación (*Group Technology*) hace referencia a la agrupación de partes o productos pertenecientes a una familia, la cual es procesada en una fábrica miniatura llamada celda. A partir de este concepto nace una técnica de fabricación y filosofía para aumentar la eficiencia producción, explotando las identidades subyacentes de los componentes; como formas, dimensiones, rutas de procesos, etc. La realización de que varios problemas pueden ser similares, y que de igual manera sean agrupados, permite la búsqueda de que una solución pueda ser encontrada para satisfacer un grupo de problemas; lográndose optimizar tiempo y esfuerzo en el proceso que este conlleva [1].

El diseño de celdas de manufactura es, entonces, una estrategia de manufacturación que consiste en la creación de un diseño óptimo de plantas de producción, cuyo objetivo principal reside en minimizar los movimientos e intercambio de material entre estas celdas. Esto nace a través de la constante búsqueda de empresas y organizaciones de encontrar soluciones que generen una mayor productividad, reduciendo los costos de producción. Dado esto, el problema de la formulación de celdas ha sido tema de considerables investigaciones, en donde Burbidge, con su análisis del flujo de producción en 1963, convierte su procedimiento en uno de los primeros para resolver este dilema [2]. En particular, el problema de formulación de celdas ha tenido dos líneas de investigación complementarias, las cuales se pueden organizar en dos grupos: Métodos aproximados y optimización global [3].

Este proyecto se centra en la primera línea de investigación, es decir, está enfocado en la búsqueda de una solución aproximada dado un criterio determinado, por lo tanto, no se puede garantizar un óptimo global [3]. Mediante la utilización de una técnica característica en el campo de la optimización, se modela el problema de diseño de celdas de manufactura, por medio de abstracciones, respetando las restricciones propias del mismo. Por otro lado, el presente documento se centra en el estudio de una técnica reciente, que sigue la tendencia sobre la inteligencia de grupos (*Swarm Intelligence*, SI). Esta expresión fue introducida por Gerardo Beni y Jin Wang en 1989 [4]. El objetivo de estudio, algoritmo de cardumen artificial, es un método para la búsqueda del óptimo global, el cual resulta ser la aplicación del conductismo en la inteligencia artificial [5]. Este algoritmo de búsqueda aleatoria, basada en la simulación de comportamientos de bancos de peces, contiene distintas conductas características. En primer lugar, se construyen los comportamientos simples y representativos del pez artificial; y, de manera consecutiva, se hace que el óptimo global aparezca, basándose en los comportamientos individuales de búsquedas locales. Este algoritmo tiene una fuerte capacidad de evitar extremos locales y de lograr extremos globales, cuyas ventajas resultan ser su flexibilidad de uso y velocidad de convergencia. Donde su capacidad de adaptación para el espacio de búsqueda, perteneciente al algoritmo de SI, resulta ser su característica principal.

Para finalizar, el documento plantea, en una primera etapa, los objetivos del proyecto, la metodología utilizada y el estado del arte de la problemática. Se procede con la realización de una introducción al algoritmo de búsqueda del cardumen artificial, para continuar con sus comportamientos básicos. Luego se presenta la problemática a tratar y la implementación de la solución propuesta. Y, para terminar, se presentan las conclusiones del trabajo realizado.

2 Definición de Objetivos

La investigación aplicada que se desarrollará y será explicada mediante el presente documento, busca entregar una solución al problema de diseño de celdas de manufactura (*Manufacturing Cell Design Problem*, MCDP) a través del desarrollo de un *solver*, el cual toma la descripción del problema de manera genérica para comparar y analizar los resultados obtenidos de modo de determinar una solución óptima.

2.1 Objetivo General

El objetivo principal que se busca cumplir es modelar y resolver el MCDP, utilizando el algoritmo de cardumen artificial (*Artificial Fish-Swarm Algorithm*, AFSA).

2.2 Objetivos Específicos

La solución a desarrollar satisface los siguientes objetivos particulares, desprendidos del objetivo general, los cuales consisten en:

- Implementar el algoritmo de AFSA para MCDP.
- Realizar experimentos con la implementación desarrollada.
- Evaluar resultados y, en caso de ser necesario, aplicar mejoras a la implementación.

3 Estado del Arte

Las celdas de manufactura han emergido en estas últimas dos décadas como innovación para la estrategia de manufacturación, la cual recoge las ventajas de la fabricación en serie de productos. Sin embargo, la independencia entre celdas es difícil de generar en la práctica, debido a que algunas partes necesitan ser procesadas en más de una máquina. Por lo tanto, la meta del problema para la formación de celdas, reside en agrupar las máquinas de manera que se minimice el flujo entre las mismas.

Referente a las investigaciones, el problema de la formulación de celdas ha tenido dos líneas de investigación complementarias. Éstas se pueden organizar en dos grupos: Métodos aproximados y optimización global. Los métodos aproximados, como metaheurísticas, se centran en la búsqueda de una solución aproximada en una cantidad fija de tiempo; por lo tanto, no pueden garantizar un óptimo global. Por el contrario, la optimización global tiene como objetivo analizar el espacio de búsqueda, con el fin de garantizar un óptimo global, como consecuencia, el coste computacional en términos de memoria y el tiempo consumido es mayor [3].

El problema de la formulación de celdas ha sido tema de considerables investigaciones. Burbidge, con su análisis del flujo de producción [13], convierte su procedimiento en uno de los primeros para resolver este dilema; aunque no de un enfoque algorítmico. Su procedimiento utiliza la matriz de incidencia máquina×pieza, y se reorganiza en una forma diagonal de bloque (*Block Diagonal Form*, BDF) [2].

Algunos métodos sólo intentan encontrar la familia de piezas, dando como resultado una solución parcial al problema; ya que la identificación de las familias de piezas requiere máquinas que procesen todas las partes dentro de una misma celda. Por lo general, estos métodos están basados en la clasificación y codificación de esquemas, y grupo de partes de acuerdo a su proceso para determinar coeficientes similares. Una formulación p-mediana para formar familia de piezas es dado por Kusiak [14] y [2].

Otros enfoques tratan de determinar no sólo las familias de piezas, sino también los grupos de máquinas. La mayoría de estos métodos se basan en la matriz de incidencia de la máquina×pieza, y se puede dividir en la agrupación jerárquica y no jerárquica. Como ejemplo a esto se tiene a Shargal [15]; Seifoddini y Hsu [16]; Srinivasan [17]; y el gráfico de programación de métodos teóricos matemáticos de Deutsch [18]; Atmani [19]; Adil [20]; Kusiak y Chow [21]; Purcheck [22]; Olivia López y Purcheck [23]; y Boctor [12], [2] y [3].

La programación por metas (*Goal Programming*, GP), es otro paradigma de optimización global. GP se puede ver como una generalización de la programación lineal, para el manejo de múltiples funciones objetivo, por ejemplo Sankaran [24]; y Shafer y Rogers [25]. Las técnicas híbridas y metaheurísticas de optimización global también se pueden encontrar en este grupo, por ejemplo Boulif y Atif [26] combina la técnica de ramificación y poda (*Branch and Bound*, BB) con un algoritmo genético (*Genetic Algorithms*, GA) [3].

Por otro lado, y para finalizar, se han utilizado diferentes metaheurísticas para la formación de celdas, por ejemplo Aljaber, Baek, y Chen [27] y Lozano, Díaz, Eguía y Onieva

[28], utilizan *Tabu Search*; Wu, Chang y Chung [29] presentan un enfoque de enfriamiento simulado (*Simulated Annealing*, SA) combinándolo con GA; Durán, Rodriguez, y Consalter [30] combina las partículas de optimización de enjambres con una técnica de minería de datos; Venugopal y Narendran [31] proponen el uso de GA; James, Brown, y Keeling [32] presentan una solución híbrida que incluye una búsqueda local y GA; y Nsakanda, Diaby, y Price [33] proponen una solución metodológica basada en la combinación de GA y técnicas de optimización a gran escala.

4 Algoritmo de Cardumen Artificial

El algoritmo de búsqueda AFSA es un conjunto ordenado y finito de operaciones biónicas de optimización, basado en el estudio sobre el comportamiento inteligente de cardúmenes. Esto quiere decir que, en una zona de agua, el pez a menudo puede encontrar lugares que contienen muchos nutrientes por sí mismo; o bien, siguiendo a otros peces. Por lo tanto, el lugar donde existan el mayor número de peces es, por lo general, aquel que tiene la mayoría de los nutrientes [10] y [11]. Según ciertas características, AFSA simula el comportamiento de este banco de peces, lo cual da forma al pez artificial (*Artificial Fish*, AF) que busca una solución óptima en el espacio de solución, correspondiente al medio ambiente en el cual el AF vive.

En este trabajo se genera de manera aleatoria el estado inicial de los AF, en donde las siguientes notaciones son utilizadas para en el algoritmo [8]:

- X: Estado actual del cardumen artificial. En donde $X = (X_1, X_2, \dots, X_n)$, donde X_i $(i = 1, \dots, n)$ representa el estado actual de *i-ésimo* AF, es decir, la solución buscada.
- Y_i: Valor de la función objetivo, correspondiente a X_i, es decir, al valor de *fitness* de la función $Y_i = f(X_i)$. Siendo *fitness* el valor considerado más aceptable como solución óptima.
- δ: Grado de hacinamiento, $0 < \delta < 1$.
- D_{ij}: Distancia el *i-ésimo* y *j-ésimo* AF. Distancia *Hamming* a implementar, la cual corresponde a la cantidad de variables, n, que difieren entre X_i y X_j [9].
- $Visual$: Campo o rango de visual del AF.
- $N(X_i, Visual)$: Conjunto de vecinos dentro de la visual del *i-ésimo* AF. Donde $N(X_i, Visual) = \{X_j | D_{ij} \leq Visual\}$.
- $FishNum$: Tamaño de la población del cardumen artificial.
- $Maxgen$: Máximo de iteraciones de AFSA.
- $Trynumber$: Máximo de iteraciones para el comportamiento de búsqueda.

4.1 Descripción del Algoritmo de Cardumen Artificial

En AFSA, la consistencia de alimentos en el agua se define como la función objetivo, mientras que el estado de un AF, la variable a ser optimizada. El comportamiento de búsqueda es en el cual el AF se mueve de manera aleatoria, de acuerdo a su valor de *fitness*. Por lo tanto, éste es una optimización de los extremos individuales del cardumen y pertenece al auto-estudio del proceso.

El comportamiento de agrupamiento y seguimiento son procesos de interacción, realizados por el AF, en relación con su entorno circundante. Estos dos procedimientos pueden garantizar que no estará demasiado hacinado, de modo que la dirección de movimiento del AF

es consistente con la dirección de movimiento promedio de los otros compañeros. De esta manera, la convergencia del cardumen se puede mantener.

Después de realizar los comportamientos mencionados con anterioridad, el AF llega al lugar donde la consistencia de comida es más grande. Durante el proceso de optimización de AFSA, la auto-información y la información del entorno son utilizadas para ajustar la dirección de búsqueda. Esto con el fin de alcanzar el equilibrio de la diversidad y la convergencia [7].

A continuación se procede a describir el algoritmo AFSA [11]:

1. Inicialización;
2. Calcular el valor de *fitness*;
3. Para cada AF_i, donde $(i = 1,2, \dots, N)$;
   3.1. Seguimiento; evalúa si el estado después del seguimiento es mejor que el estado anterior, de ser así, se avanza al paso 4, de lo contrario se recurre al paso 3.2;
   3.2. Búsqueda; evalúa si el estado de búsqueda es mejor que el anterior un $Trynumber$ veces, de ser así avanza al paso 4, de lo contrario se recurre al paso 3.3;
   3.3. Agrupamiento; evalúa si el estado después de la agrupación es mejor que el estado anterior, de ser así, se avanza al paso 4, de lo contrario se recurre al paso 3.4;
   3.4. Movimiento;
4. Actualizar el mejor valor actual;
5. Si se concluye el $Maxgen$, salir; de lo contrario, volver al paso 3;

4.2 Comportamientos del AF

Cada pez artificial, o AF, ejecuta tres comportamientos básicos, además de aquel realizado en momentos de ocio. Estos cuatro comportamientos definen al AF y forman parte del ciclo de vida del mismo. Lo mencionado de manera previa se describe como [8]:

1. Comportamiento de seguimiento (*follow behavior*): Cuando un pez encuentra comida, otros de manera correlativa llegarán, debido a que éstos siguen a su compañero más cercano.

   De una forma más detallada:

   - Paso 1: Para X_i, seleccionar el mejor estado de X_j con el mínimo Y_j dentro de su visual. Es decir, el *j-ésimo* AF que tenga el valor mínimo de la función objetivo entre los vecinos que estén dentro del rango de visual del *i-ésimo* AF.
   - Paso 2: Calculado del nf, es decir, el número de vecinos que estén dentro del rango de visual del *i-ésimo* AF. Si $Y_j < Y_i$ y $nf / FishNum < \delta$, significa que

había espacio adicional para otro pez, entonces se aplica una función que influencie a X_i con X_j; de lo contrario, ejecutar comportamiento de búsqueda.

La descripción en pseudocódigo del proceso se presenta de la siguiente manera:

Algoritmo 4.1 Pseudocódigo del Comportamiento de Seguimiento.

```
Artificial_fish_follow()
{
    Y_j = Min(f(X_j)), X_j ∈ N(X_i, Visual);
    nf = N(X_i, Visual);
    if (nf/FishNum < δ and Y_j < Y_i) then
        X_i = Inf(X_j);
    else
        AF-prey();
    end
}
```

2. Comportamiento de búsqueda (*prey behavior*): En general, el pez nada al azar y de manera libre en el agua. Cuando éste encuentra comida cerca, se desplaza a la dirección donde la comida aumenta de forma gradual.

   De una forma más detallada:

   - Paso 1: Para X_i , selecciona al azar un estado X_j de los vecinos que estén dentro del rango de visual del *i-ésimo* AF.
   - Paso 2: Si $Y_j < Y_i$, entonces se aplica una función que influencie a X_i con X_j; de lo contrario, ir al Paso 1.

   Si la operación anterior se ejecutó $trynumber$ veces, y no se pudo encontrar un mejor estado, entonces ejecutar comportamiento de agrupamiento.

   La descripción en pseudocódigo del proceso se presenta de la siguiente manera:

Algoritmo 4.2 Pseudocódigo del Comportamiento de Búsqueda.

```
Artifical_fish_prey()
{
    m=false;
    for i = 0 to trynumber
        X_j = Random(N(X_i, Visual));
        if (Y_j < Y_i) then
            m=true;
            X_i = Inf(X_j);
```

```
        end
    end
    if (m==false) then
        AF-swarm();
    end
}
```

3. Comportamiento de agrupamiento (*swarm behavior*): Para garantizar la supervivencia y evitar daños, los peces se agrupan de forma natural. Las tres reglas observadas al momento de agruparse son:

   - Separación: Tratar de evitar el hacinamiento.
   - Alineación: Tratar de ser consistente con la dirección media del pez más cercano.
   - Cohesión: Tratar de avanzar hacia el centro del pez más cercano.

   De una forma más detallada:

   - Paso 1: Para cada X_i, calcular nf, el cual corresponde al número de vecinos presentes dentro del rango de visión del *i-ésimo* AF.
   - Paso 2: Si $nf \neq 0$ y $nf / FishNum < \delta$, determinada la posición central X_c de todos los AF, se calcula la función de *fitness* $Y_c = f(X_c)$. Si $Y_c < Y_i$, significa que la concentración de comida de X_c era alta y el entorno no estaba muy lleno, entonces se aplica una función que influencie a X_i con X_c; de lo contrario, ejecutar comportamiento de movimiento.
   - Paso 3: Si $nf = 0$, ejecutar comportamiento de movimiento.

   La descripción en pseudocódigo del proceso se presenta de la siguiente manera:

Algoritmo 4.3 Pseudocódigo del Comportamiento de Agrupamiento.

```
Artificial_fish_swarm()
{
    nf = N(X_i, Visual);
    if (nf ≠ 0 and nf / FishNum < δ) then
        X_c = Center(N(X_i, Visual));
        Y_c = f(X_c);
        if Y_c < Y_i then
            X_i = Inf(X_c);
        else
            AF-move();
        end
    else
```

```
            AF-move();
        end
    }
```

4. Comportamiento de movimiento (*move behavior*): Si la solución óptima no fue mejorada en el progreso de optimización anterior, se genera de forma aleatoria otro estado X_j. Si el valor de la función de la nueva situación es mejor que X_i, a continuación, se adopta como el estado actual del *i-ésimo* AF; de lo contrario, se aplica una función que influencie a X_i con X_j. Este comportamiento podría ayudar a la solución a escapar de un óptimo local, mientras tanto reduce la carga de trabajo de cálculo.

   La descripción en pseudocódigo del proceso se presenta de la siguiente manera:

Algoritmo 4.4 Pseudocódigo del Comportamiento de Movimiento.

```
Artificial_fish_swarm()
{
    new X_j;
    if (Y_j < Y_i) then
        X_i = X_j;
    else
        X_i = Inf(X_j);
    end
}
```

5 Problema de Diseño de Celdas de Manufactura

El MCDP es una estrategia de manufacturación que consiste en la creación de un diseño óptimo de plantas de producción, las cuales se componen de celdas de manufactura y éstas de máquinas que procesan un subconjunto de piezas que forman familias, determinadas de acuerdo a la similitud entre partes [6] y [3].

5.1 Planteamiento del MCDP

El objetivo del MCDP reside en minimizar los movimientos e intercambio de material entre celdas, con el fin de reducir costos de producción y aumentar la productividad. La idea es representar los requisitos de procesamiento de las piezas de máquinas a través de una matriz de incidencia llamada máquina×pieza. Esta matriz contiene dominios binarios y se denota como $A = [a_{ij}]$, donde [3]:

$$a_{ij} = \begin{cases} 1 & \text{Si la pieza } j \text{ visita la máquina } i \\ 0 & \text{En otro caso} \end{cases}$$

Cuando una matriz de incidencia de máquina×pieza se construye, el objetivo principal es la agrupación de máquinas para la formación de conjuntos de máquinas y piezas de trabajo, de modo que el número de transporte intercelular de piezas se reduzca al mínimo. Este reordenamiento tiene por finalidad minimizar del total de movimientos entre celdas y de variación de la carga dentro de las celdas. Una formulación matemática rigurosa del problema de agrupamiento máquina×pieza viene dada por el modelo de optimización que se indica a continuación. Sea:

- M el número de máquinas.
- P el número de piezas.
- C el número de celdas.
- i el índice de máquinas ($i = 1, \dots, M$).
- j el índice de piezas ($j = 1, \dots, P$).
- k el índice de celdas ($k = 1, \dots, C$).
- $A = [a_{ij}]$ la matriz de incidencia binaria $M \times P$.
- M_{max} el número máximo de máquinas por celda.

Se selecciona la función objetivo que debe reducir al mínimo el número de veces que una parte dada debe ser procesada por una máquina que no pertenece a la celda a la cual la parte ha sido asignada. Sea:

$$y_{ik} = \begin{cases} 1 & \text{Si la máquina } i \text{ pertenece a la celda } k \\ 0 & \text{En otro caso} \end{cases}$$

$$z_{jk} = \begin{cases} 1 & \text{Si la parte } j \text{ pertenece a la familia } k \\ 0 & \text{En otro caso} \end{cases}$$

El problema descrito con anterioridad, es representado por el siguiente modelo matemático:

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

Para finalizar, la función objetivo presentada de manera previa, está sujeta a las siguientes restricciones, en donde cada una de ellas asegura si:

- Una máquina pertenece a una celda:

$$\sum_{k=1}^{C} y_{ik} = 1 \quad \forall i$$

- Una pieza pertenece a una familia de componentes:

$$\sum_{k=1}^{C} z_{jk} = 1 \quad \forall j$$

- El número de máquinas no exceda el número máximo de máquinas por celda:

$$\sum_{i=1}^{M} y_{ik} \leq M_{max} \quad \forall k$$

Como ejemplo a lo descrito de manera previa, se presenta en la Figura 5.1 una matriz máquina×pieza inicial, la cual contiene el detalle sobre qué máquinas elaboran ciertas piezas. Por otro lado, en la Figura 5.2 se presenta una posible solución al problema. En ésta, las máquinas están agrupadas, de manera tal, que el intercambio de material entre celdas sea el mínimo.

Pieza

Máquina	1	2	3	4	5	6	7
1		1			1		
2	1						1
3			1	1		1	
4			1	1		1	
5	1						1
6			1	1		1	
7		1			1		

Figura 5.1 Matriz Máquina×Pieza Inicial.

Pieza

Máquina	1	7	3	4	6	2	5
2	1	1					
5	1	1					
3			1	1	1		
4			1	1	1		
6			1	1	1		
1						1	1
7						1	1

Figura 5.2 Matriz Máquina×Pieza Final.

6 Modelado del MCDP

El modelado de sistemas y de procesos, entendido como el establecimiento de relaciones semánticas entre la teoría, los objetos y los fenómenos, es una herramienta en la explicación científica y resolución de problemas. Se representa la realidad por medio de abstracciones, de modo que permite concentrar la atención en aquellas características importantes del sistema, restándoles la atención a otras. El MCDP no queda exento de esta lógica de tratamiento, por lo que se requiere de un análisis hipotético (ver capítulo sobre el Problema de Diseño de Celdas de Manufactura) y una transferencia desde este último a una representación codificable.

6.1 Representación del MCDP en Netbeans

La representación del MCDP, para el proyecto actual, se basa en la sintaxis de Netbeans, entorno de desarrollo integrado libre que utiliza el lenguaje de programación Java. Dada la formulación del MCDP como un CSP, el código presentado está sujeto a restricciones y sintaxis lógica. De manera siguiente, se ilustra y especifican variables, constantes, dominios y restricciones del MCDP bajo los términos de Netbeans.

6.1.1 Formalización del MCDP utilizando AFSA en Netbeans

El *solver* utilizado para resolver el MCDP utiliza AFSA, el cual fue descrito en el capítulo sobre el Algoritmo de Cardumen Artificial. Éste consta de 3 paquetes, denominados:

- Matrices, el cual contiene los problemas a resolver, además de la lectura de estos.
- Proyecto, el cual contiene el modelamiento del MCDP y AFSA.
- Resultados, en donde se van almacenando en un archivo ".csv" los resultados obtenidos cada vez que el programa se ejecuta.

Al interior del paquete "Proyecto" se establecen cuatro clases, llamadas "Principal", "Pez", "Parámetros" y "Restricciones". Cada clase contiene métodos y atributos que serán explicados a continuación.

- Clase "Principal", contenedor del "main", el cual especifica dónde debe comenzar la ejecución del programa. Por otro lado, se definen variables globales, debido a su frecuente uso y a que no pueden ser alteradas mientras el programa se encuentra en ejecución, éstas son:
  - arreglo_pez, correspondiente al arreglo de peces o cardumen.
  - arreglo_pez_fo, correspondiente a los valores de *fitness* de cada pez del arreglo de peces.
  - Mejor_Valor_Actual, correspondiente a la posición del AF que contiene al óptimo, el cual no implica que sea el global.
  - Optimo_Local, correspondiente a la posición del AF que contiene el mejor valor de *fitness* encontrado en un punto determinado.

Por otro lado, la clase “Principal” contiene los siguientes métodos:

- ImprimirMatriz, el cual imprime una matriz determinada, en caso de ser necesario.
- CrearArregloPez, el cual crea el cardumen artificial, es decir, inicializa a los AF generando la matriz máquina×celda, en donde cada pez inserto en el “arreglo_pez” cumple con las restricciones propias del MCDP.
- CrearArregloPezFuncionObjetivo, el cual crea el arreglo que contiene los valores de *fitness* de cada AF.

- Clase “Pez”, contiene un único atributo, propio del AF, denominado “maquina_celda”, correspondiente a la matriz máquina×celda. Por otro lado, esta clase contiene los siguientes métodos:
  - Pez, el cual consiste en el constructor de la clase, en donde inicializa de manera aleatoria la matriz máquina×celda.
  - Pez_Seguimiento, el cual ejecuta el comportamiento de seguimiento del AF.
  - Pez_Busqueda, el cual ejecuta el comportamiento de búsqueda del AF.
  - Pez_Agrupamiento, el cual ejecuta el comportamiento de agrupamiento del AF.
  - Pez_Movimiento, el cual ejecuta el comportamiento de movimiento del AF.
  - Influir, el cual altera la matriz máquina×celda del AF X_i con la matriz máquina×celda del AF X_j, en donde se cambia una máquina de celda.
  - CalcularVecindario, el cual calcula los vecino del AF.
  - MinimoFitness, el cual calcula el AF X_j con menor *fitness* dentro de la *Visual* del AF X_i.
  - CalcularCentro, el cual determina el AF con menor distancia, con respecto a su vecindario, de otros AFs.
- Clase “Parametros”, contiene los valores con los cuales trabaja el algoritmo entregando un mejor ordenamiento y manejo de parámetros a ocupar. Dentro de este tenemos:
  - M, correspondiente al número de máquinas.
  - P, correspondiente al número de piezas.
  - C, correspondiente al número de celdas.
  - M_max, correspondiente al número máximo de máquinas permitidas por celda.
  - Maxgen, correspondiente al número de iteraciones realizadas por AFSA.
  - FishNum, correspondiente al número de AFs.
  - grado_hacinamiento, correspondiente al valor de δ.
  - Trynumber, correspondientes al número de iteraciones realizados por el comportamiento de búsqueda.

- Clase “Restricciones”, la cual contiene los métodos que validan las restricciones propias del MCDP, como son:
  - RestriccionMaquinaCelda, el cual evalúa si una máquina pertenece a una celda.
  - RestriccionPiezaCelda, el cual evalúa si una pieza pertenece a una familia de componentes.
  - RestriccionMaquinaCelda_max, el cual evalúa el número de máquinas por celdas.
  - RestriccionFixMaquinaCelda, el cual repara una solución infactible, en lugar de desecharla y crear una nueva. Este método repara las restricciones de que una máquina pertenezca a una celda y de que no se excede el número máximo de éstas permitidas por celda.

  Por otro lado, se tiene métodos adicionales como son:
  - CrearMatrizPiezaCelda, el cual genera la matriz pieza×celda, deducible a través de las matrices máquina×pieza y máquina×celda, con el fin de hacer cumplir las restricciones del MCDP.
  - FuncionObjetivo, el cual calcula los movimientos entre celdas del problema de la matriz máquina×celda generada como posible solución.

6.1.2 Parámetros

Los parámetros utilizados en el problema del MCDP y los del algoritmo AFSA, el cual le da solución, utilizan los siguientes parámetros:

- M=16.
- P=30.
- *Maxgen*=50.000.
- *FishNum*=40.
- δ=[0,9-0,3].
- *Trynumber*=4.
- El valor de C, celdas de manufactura; y *n*, cantidad de variables; se alteran según el problema al cual se esté dando solución.

El grado de hacinamiento (δ) en este algoritmo resulta ser dinámico, esto quiere decir que como valor inicial tendrá el valor de 0,9. A medida que el “Mejor Valor Actual” sea alterado, éste parámetro disminuirá en un 0,1, teniendo como límite el valor de 0,3. Lo anterior es realizado con el fin de agrupar al cardumen, considerando a aquellos AFs que estén muy alejados de lo que se considera como óptimo.

6.1.3 Resultados

El proceso de testeo del algoritmo es utilizado en máquinas con Intel core i7-3630 240GHz con 8GB de Ram, en donde la ejecución del camino completo de AFSA promedia un tiempo de 2 a 10 minutos. Los resultados obtenidos hasta el momento se ven resumidos en la Tabla 6.1. Cada problema fue puesto a prueba 10 veces, en donde en cada ocasión el resultado obtenido fue el mismo. La diferencia entre cada caso está sobre en qué punto se llegó al óptimo. Como referencia se utilizó el *paper* de Fayez F. Boctor, *A linear formulation of the machine-part cell formation problem*, cuya tabla de comparación observada es "*Table 1 Optimal Solution for the 10 test problems*" [12]. A continuación se muestra en la Tabla 6.1 los resultados obtenidos por la implementación de AFSA, en comparación con los óptimos entregados por el *paper* ya especificado.

Tabla 6.1 Resultados al MCDP implementando AFSA.

	C=2																C=3											
Pblm	M_{max}=8				M_{max}=9				M_{max}=10				M_{max}=11				M_{max}=12				M_{max}=6				M_{max}=7				M_{max}=8				M_{max}=9			
	B	P	X	M	B	P	X	M	B	P	X	M	B	P	X	M	B	P	X	M	B	P	X	M	B	P	X	M	B	P	X	M	B	P	X	M
1	11	14	14	14	11	14	14	14	11	14	14	14	11	14	14	14	11	14	14	14	27	34	31	30	18	23	21	21	11	18	14	14	11	17	14	14
2	7	9	9	9	6	8	8	8	4	8	8	8	3	8	8	8	3	7	7	7	7	13	13	13	6	13	12	12	6	12	10	9	6	11	8	8
3	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	3	4	4	4	1	4	4	4	9	12	12	12	4	12	9	9	**4**	**9**	**6**	**4**	**4**	**8**	**4**	**4**
4	**14**	**14**	**14**	**14**	13	14	14	14	13	14	14	14	13	14	14	14	**13**	**13**	**13**	**13**	27	30	28	28	18	21	20	19	**14**	**17**	**14**	**14**	13	17	15	14
5	9	12	12	12	6	9	9	9	6	9	9	9	5	7	7	7	**4**	**4**	**4**	**4**	11	14	14	14	8	11	11	11	8	13	12	11	6	14	9	9
6	5	6	6	6	3	4	4	4	3	4	4	4	3	4	4	4	2	4	4	4	6	15	14	14	4	11	9	9	4	8	7	6	3	8	4	4
7	**7**	**7**	**7**	**7**	4	6	6	6	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	11	17	13	12	5	10	7	7	5	8	7	7	4	10	6	6
8	**13**	**13**	**13**	**13**	10	12	12	12	**8**	**8**	**8**	**8**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	14	17	16	16	11	16	13	13	11	16	13	13	10	13	12	12
9	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	12	20	19	19	12	17	14	14	**8**	**12**	**9**	**8**	**8**	**9**	**8**	**8**
10	**8**	**8**	**8**	**8**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	10	17	13	13	8	13	9	9	**8**	**11**	**8**	**8**	**5**	**9**	**6**	**5**

B=Óptimo alcanzado por Boctor.

P=Peor óptimo alcanzado en AFSA.

X=Promedio de óptimos alcanzados en AFSA.

M=Mejor óptimo alcanzado en AFSA.

A continuación, en la Figura 6.1, se muestra el gráfico del problema 6. Los valores de los parámetros utilizados son C=3 y M_{max}=8. Por otro lado, como comparación, se tiene en la Figura 6.2 el gráfico de solución al problema 7, el cual utiliza como parámetros C=2 y M_{max}=10, el cual da como resultado el mismo óptimo obtenido por Fayez F. Boctor. En ambos gráficos se puede observar cómo se altera el "Mejor Valor Actual" de manera abrupta al inicio de las iteraciones, para luego cambiar su valor de forma más paulatina, como es en el caso de la Figura 6.1.

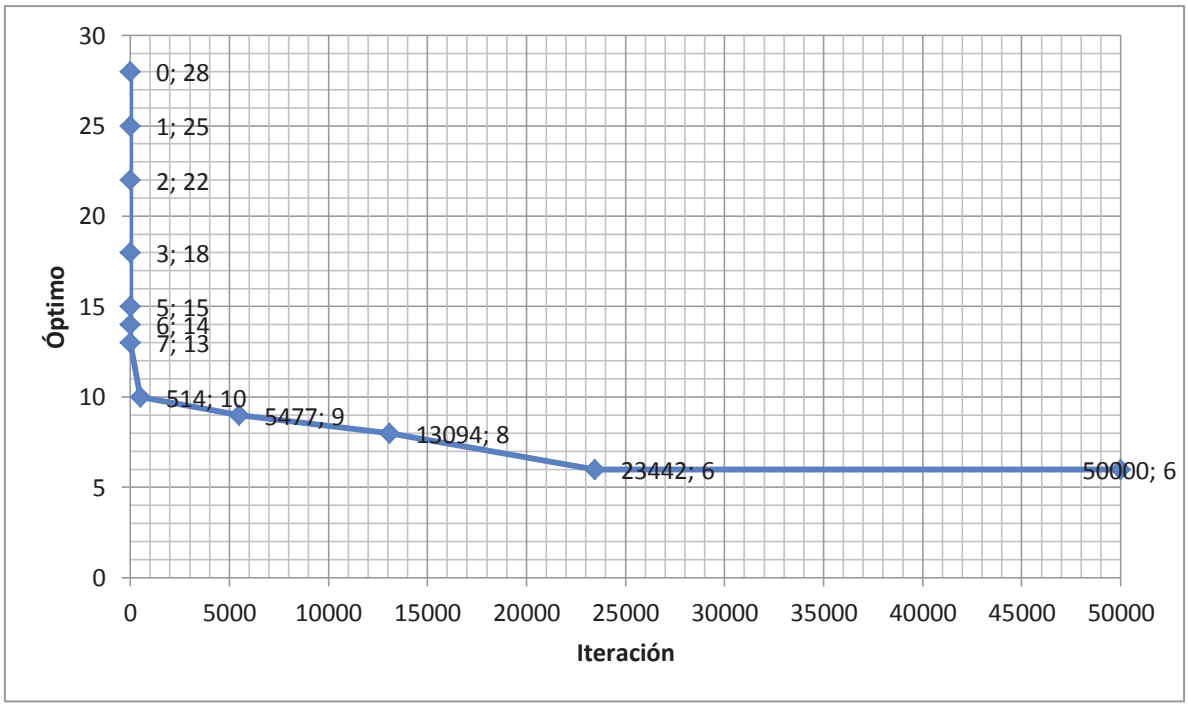

Figura 6.1 Resultado al problema 6.

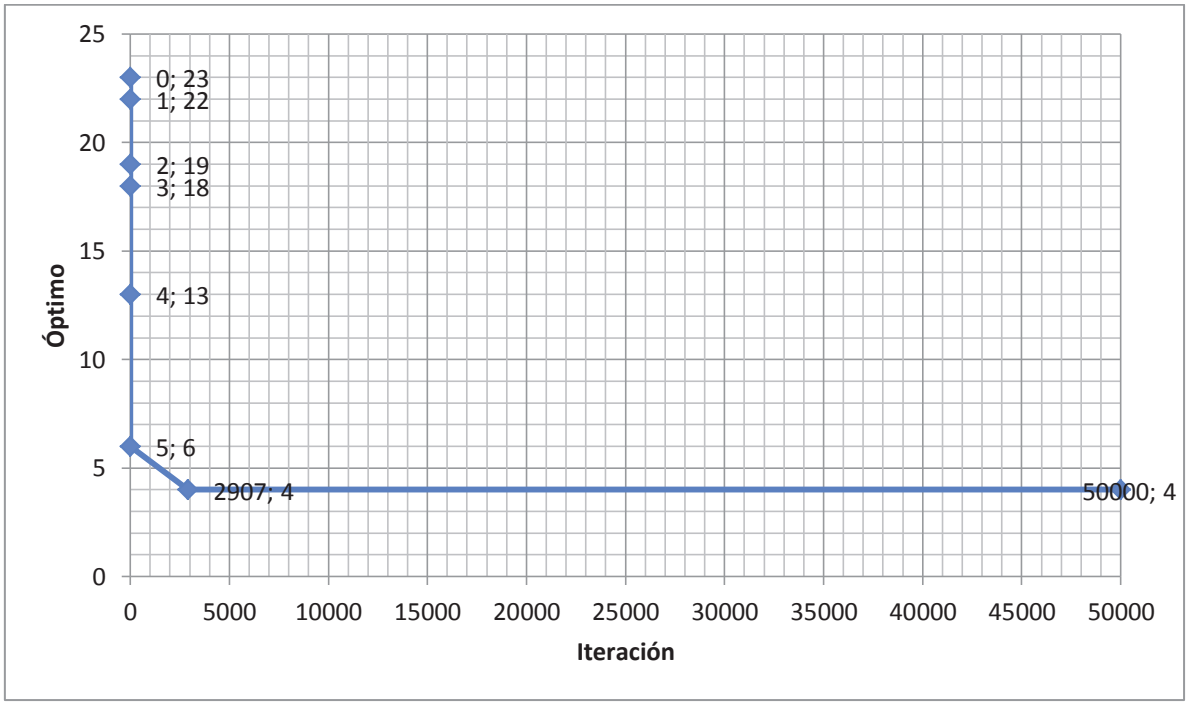

Figura 6.2 Resultado al problema 7.

Durante el problema 1, la diferencia entre los óptimos alcanzados por el algoritmo implementado, AFSA, con respecto a los resultados obtenidos por Fayez F. Boctor, se ven diferenciados por un valor de 3. Antes de continuar con la experimentación, se intentó disminuir este número, sin embargo, pese a la modificación del método "Influir", el cual

modifica la solución del AF, no se logró dicho cambio. De manera inicial se alteraban 5 máquinas de celdas, cuyos cambios resultaban ser grandes, escalando de forma abrupta a un óptimo. Siguiendo esta lógica, se procede a modificar sólo una máquina de celda, alcanzando un método de escalado más pausado, donde se obtiene un mejor comportamiento del algoritmo. De igual manera, al realizarse algún cambio en la matriz, ésta nueva, al no cumplir con las restricciones propias del MCDP, se desechaba. Por este motivo, se procede a reparar aquel valor que viola la restricción, en lugar de eliminarla.

En la Figura 6.3, presentada a continuación, muestra el gráfico del problema 1. Los valores de los parámetros utilizados son C=3 y M_{max}=8. Éste permite observar que se requieren sólo 8 iteraciones para alcanzar un óptimo determinado por AFSA, por lo que otros intentos por mejorar este valor resultan no tener éxito en este caso particular.

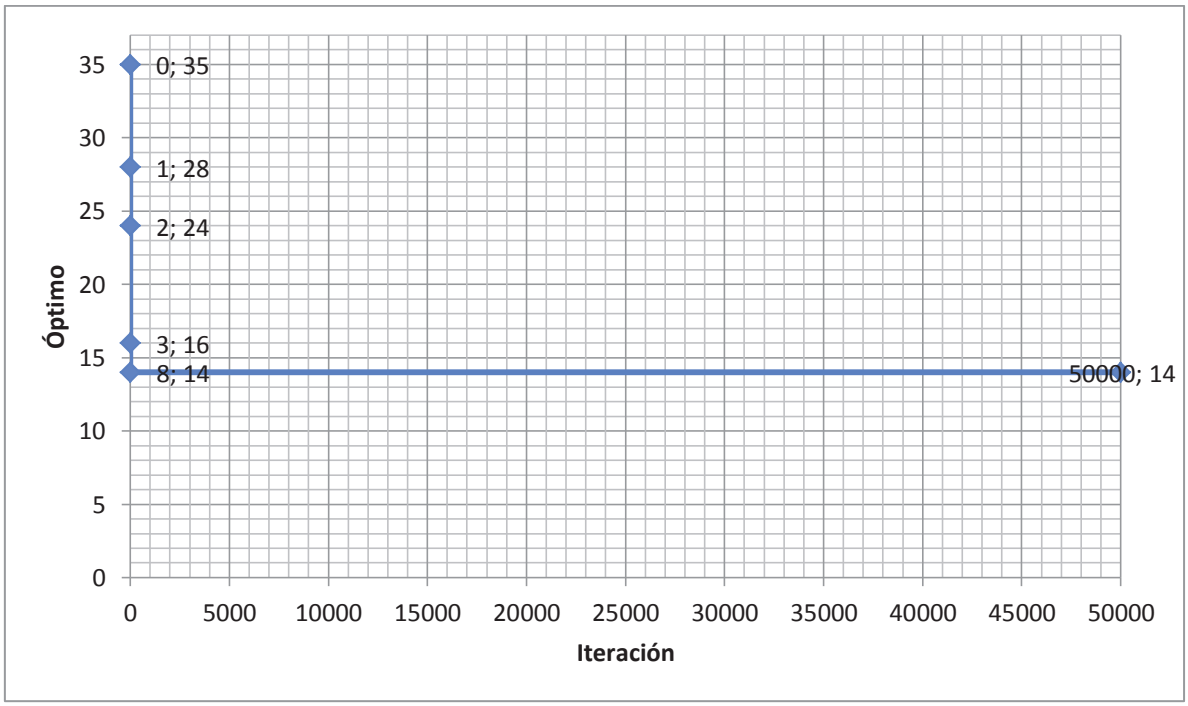

Figura 6.3 Resultado al Problema 1.

Por otro lado, el problema 2 presenta más cambios, cuyos valores difieren entre los rangos de 2 a 6 movimientos entre celdas. En el problema 3, las diferencias entre valores del óptimo obtenidos tras la utilización de AFSA y el utilizado por Fayez F. Boctor varían entre los rangos de valores de 1 a 3 movimientos entre celdas, siendo los parámetros C=3 y M_{max}=7 la excepción, ya que el problema generara un valor de 5 movimientos entre celdas. Por el contrario, los parámetros C=2 y M_{max}=8; C=2 y M_{max}=9; C=2 y M_{max}=10; C=3 y M_{max}=8; y C=3 y M_{max}=9 alcanzan los óptimos propuestos por Boctor. En la Figura 6.4 se puede observar la curva generada por AFSA con parámetros C=3 y M_{max}=9, donde el óptimo alcanzado se logra en la iteración 42.367.

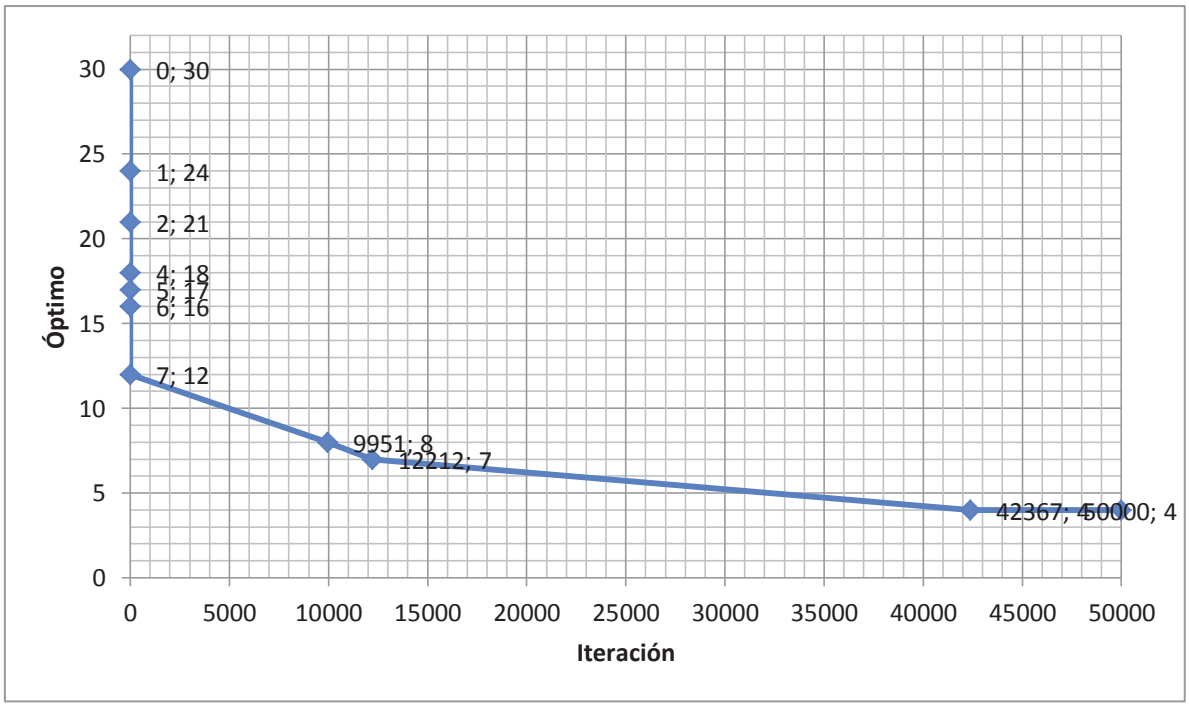

Figura 6.4 Resultado al Problema 3.

Continuando, en el problema 4 se alcanzan tres óptimos, cuyos parámetros son C=2 y M_{max}=8; C=2 y M_{max}=12; y C=3 y M_{max}=8. Sin embargo, los otros resultados arrojan valores que difieren en 1 movimiento entre celdas. Al contrario del problema 5, en este se alcanza un óptimo, con parámetros C=2 y M_{max}=12, en donde los otros valores entregados tras la ejecución del problema varían en un movimiento de 3; siendo la excepción C=2 y M_{max}=11 con un movimiento de 2, y C=3 y M_{max}=8 con un movimiento de 4.

El problema 6 no alcanza ningún óptimo, y además es el primer problema que presenta una de las diferencias más grande entre los resultados alcanzados. Este valor es de 8 movimientos entre celdas, con parámetros C=3 y M_{max}=6. En contraste con el problema 7, que alcanza 4 óptimos, los demás problemas difieren entre 1 a 2 movimientos entre celdas. Los parámetros de los óptimos logrados por AFSA tienen los parámetros C=2 y M_{max}=8; C=2 y M_{max}=10; C=2 y M_{max}=11; y C=2 y M_{max}=12. Se debe destacar que este problema no cumplía con la restricción del MCDP que dice que cada pieza debe pertenecer a una familia, por lo que la solución a este dilema se resuelve asignando la pieza 11 a la máquina 16.

Como resultado al problema 8, se logran 4 óptimos, cuyos parámetros son C=2 y M_{max}=8; C=2 y M_{max}=10; C=2 y M_{max}=11; y C=2 y M_{max}=12. Los otros valores difieren con 2 movimientos entre celdas, lo que se asimila al problema 1, al presentar una misma diferencia entre los valores alcanzados por AFSA y los entregados por el *paper* de Boctor. Al contrario del problema 9, este logra llegar a 7 óptimos, de parámetros C=2 y M_{max}=8; C=2 y M_{max}=9; C=2 y M_{max}=10; C=2 y M_{max}=11; C=2 y M_{max}=12; C=3 y M_{max}=8; y C=3 y M_{max}=9. Sin embargo, este es el segundo problema que presenta más diferencias en relación a los resultados que se tienen como referencia y los alcanzados por la metaheurística implementada. Esta desigualdad tiene un valor de 7, cuyos parámetros son C=3 y M_{max}=6. La gráfica de este problema se presenta en la Figura 6.5.

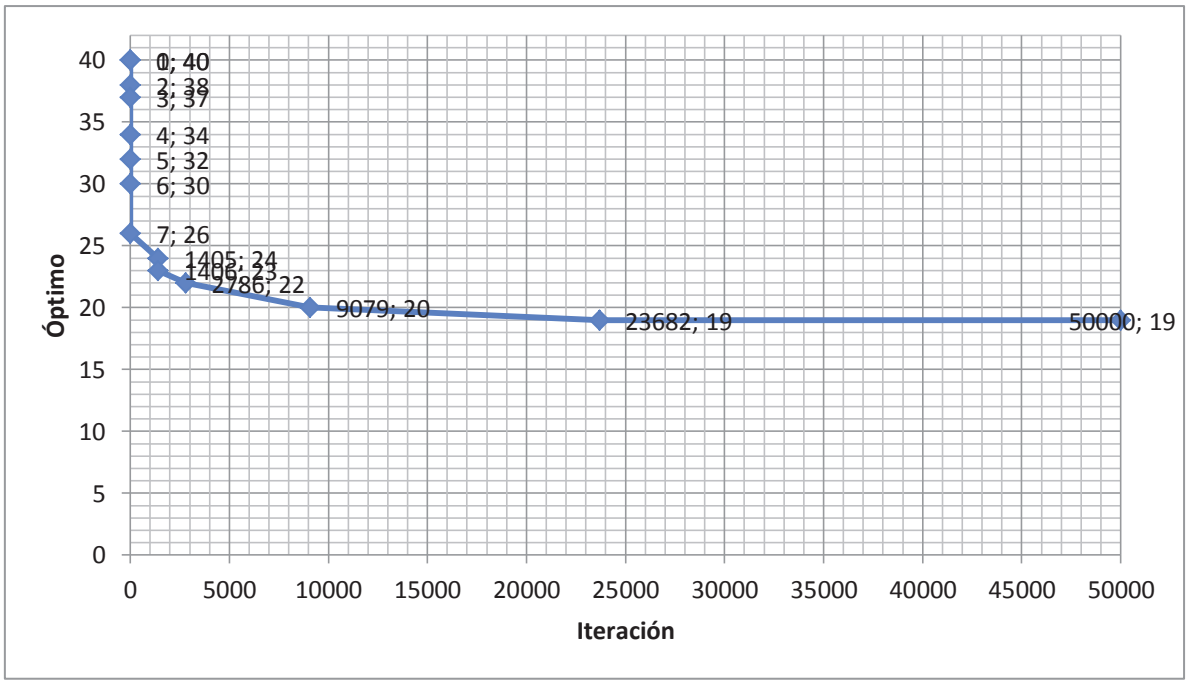

Figura 6.5 Resultado al Problema 9.

Por otro lado, el problema 10 alcanza 7 óptimos, al igual que el problema 9; cuyos parámetros son C=2 y M_{max}=8; C=2 y M_{max}=9; C=2 y M_{max}=10; C=2 y M_{max}=11; C=2 y M_{max}=12; C=3 y M_{max}=8; y C=3 y M_{max}=9. Sin embargo, la pieza 21 no pertenece a una familia de componentes, por lo que es asignada a la máquina 16. La diferencia entre celdas en los problemas restantes varía entre los valores 1 y 3. El gráfico a este problema con parámetros C=3 y M_{max}=8 se presenta en la Figura 6.6.

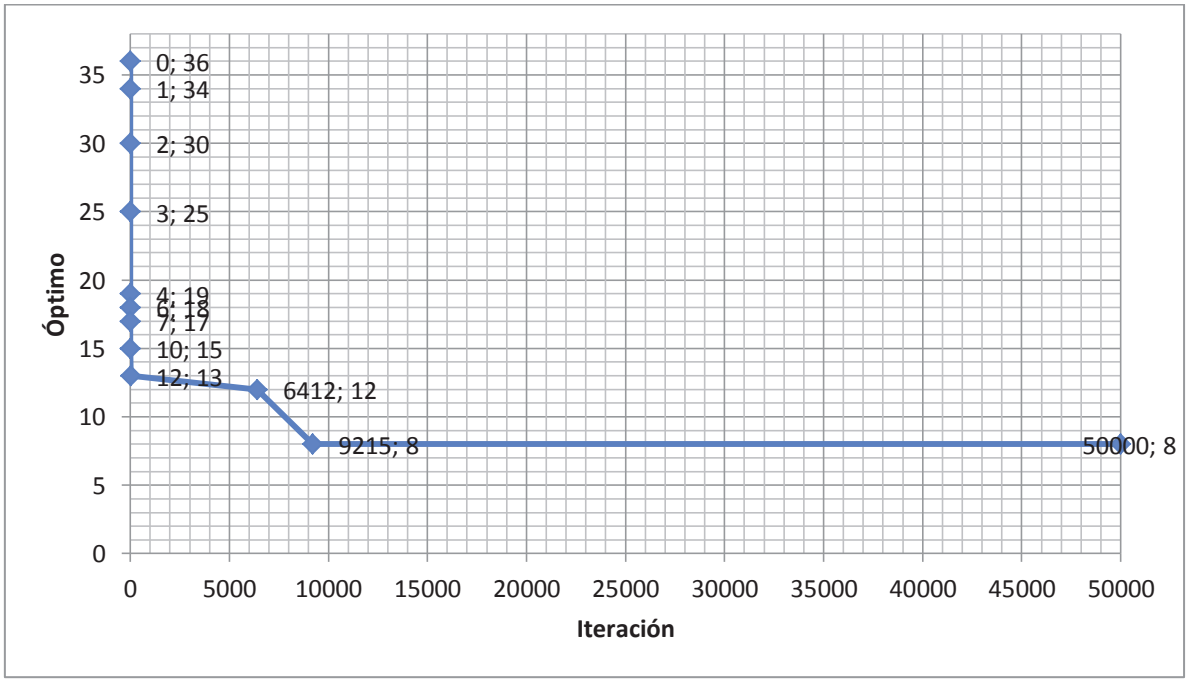

Figura 6.6 Resultado al Problema 10.

Hay que destacar que el cambio entre el primer valor elegido como óptimo, y los siguientes, continúan alterándose de manera abrupta tras la ejecución del programa. En donde trabajar con C=2 entrega resultados óptimos en menor cantidad de iteraciones que cuando se aplica C=3 al problema. De igual manera, los resultados óptimos alcanzados corresponden en su mayoría cuando C=2. Se concluye que la mayor parte de valores considerados como óptimos son alcanzados antes de realizar 50.000 iteraciones, por lo que en sus inicios se trabajó con 100.000 iteraciones, donde en ocasiones la mitad del tiempo los valores no se alteraban. Por esta razón se decide bajar el valor del atributo "*Maxgen*", sin embargo, el mejor valor alcanzado no siempre resulta ser el óptimo entregado por Fayez F. Boctor. Como ejemplo a esta situación, se puede observar en la Figura 6.7 la gráfica de valores del problema 5, con parámetros C=3 y M_{max}=8, el cual presenta una convergencia paulatina al mejor valor encontrado por AFSA en la iteración 43.995.

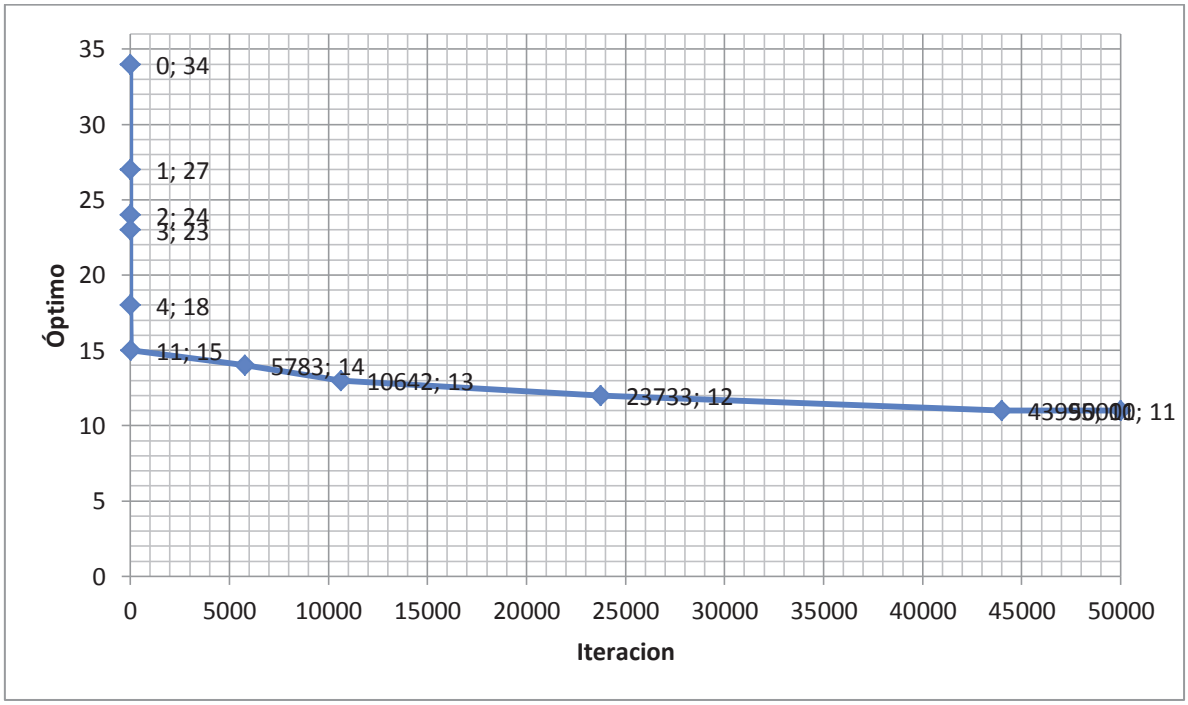

Figura 6.7 Resultados al problema 5.

7 Conclusión

La velocidad, fiabilidad y robustez que han alcanzado los modernos métodos de optimización, hace posible que el académico pueda contribuir de forma real y significativa a las investigaciones sobre un determinado problema. Un factor clave para el éxito de un método de optimización, es el incorporar a su estrategia otras técnicas o algoritmos. Estos principios generales han permitido utilizar procedimientos de planificación y optimización, que de mejor manera se adecuen al perfil computacional del modelo. Dentro de estos se encuentran la programación lineal, programación con restricciones y métodos híbridos.

Como técnica utilizada se tiene la metaheurística que, por lo general, se aplica a problemas que no tienen un algoritmo o heurística específica que entregue una solución satisfactoria. De esta manera, con el uso del algoritmo de búsqueda o técnica AFSA, se trabaja sobre el diseño de celdas de manufactura. El cual, a lo largo de su ejecución, gracias a sus comportamientos, minimiza los movimientos e intercambio de material entre celdas; buscando una optimización en tiempo y costos para la empresa.

La velocidad con la cual AFSA converge, está relacionada con los valores de sus parámetros. En un inicio se utilizan altos valores, como por ejemplo, un grado de hacinamiento de 0,8 o un $trynumber$ de 20. Por otro lado, al ejecutar los comportamientos del AF, la posible solución del mismo se alteraba moviendo 5 máquinas de sus celdas. Tras el análisis de los resultados se llega a la conclusión de que la utilización de parámetros con valores más pequeños, o bien, dinámicos, como es el caso del grado de hacinamiento, los resultados convergen de manera más paulatina.

Cuando la matriz máquina×pieza, cuya cantidad de celdas tiene un valor de 3, genera un mayor gasto de recursos en cuanto al tiempo empleado para resolverla. Además, el valor obtenido no en todos los efectos resulta ser el óptimo. En estos casos es posible observar que el gráfico de resultados escala de manera más pausada, por el contrario de cuando el valor de la cantidad de celdas es 2, el cual presenta una mayor descendencia al valor conocido como "Mejor Valor Actual".

Para finalizar, se puede especificar que los objetivos definidos al inicio del documento son cumplidos. Cada investigación realizada de forma independiente sobre el MCDP y AFSA, permitieron la comprensión de éstos, para luego concretar la teoría en la elaboración de un *solver* que resuelva el problema del diseño de celdas de manufactura con el algoritmo de búsqueda de cardumen artificial. Sin embargo, aunque los valores obtenidos no siempre resultaron ser los óptimos conocidos por el *paper* de Boctor, el promedio de diferencias entre valores entregados por el mismo y AFSA tiene un valor de 2,644 (sin considerar los óptimos alcanzados).

8 Referencias

[1] Richard Lee Storch, Industrial Engineering, *Group Technology*, University of Washington Courses Web Server, IND E 337/537: Introduction to Manufacturing Systems, Autumn 2010.

Disponible en: http://courses.washington.edu/ie337/gt.pdf

[2] Ana R. Xambre and Pedro M. Vilarinho, *A simulated annealing approach for manufacturing cell formation with multiple identical machines*, European Journal of Operational Research 151, pp. 434–446, 2003.

[3] Ricardo Soto, Hakan Kjellerstrand, Orlando Durán, Broderick Crawford, Eric Monfroy and Fernando Paredes, *Cell formation in group technology using constraint programming and Boolean satisfiability*, Expert Systems with Applications 39, pp. 11423-11427, 2012.

[4] Beni, G. and Wang, J., *Swarm Intelligence in Cellular Robotic Systems*, Proceedings of NATO Advanced Workshop on Robots and Biological Systems, Tuscany (Italy), Vol. 102, Junio de 1989.

[5] LI Xiao lei, SHAO Zhi-jiang and QIAN Ji-xin, *An optimizing method based on autonomous animals: Fishswarm algorithm*, Systems Engineering Theory & Practice, Vol. 22, pp. 32-38, 2002.

[6] Pedro Daniel Medina V., Eduardo Arturo Cruz T. y Manuel Pinzon C., *Generación de Celdas de Manufactura Usando el Algoritmo de Ordenamiento Binario (AOB)*, Universidad Tecnológica de Pereira, Scientia et Technica Año XVI, N° 44, Abril de 2010.

[7] Yong PENG, *An improved artificial fish swarm algorithm for optimal operation of cascade reservoirs*, Journal of Computers, Vol. 6, N° 4, Abril 2011.

[8] Yun Cai, *Artificial Fish School Algorithm Applied in a Combinatorial Optimization Problem*, Wuhan University of Science and Technology/Country College of Machinery and Automation, Wuhan, China, pp. 37-43, 2010.

[9] Md. Abul Kalam Azad, Ana Maria A.C. Rocha and Edite M.G.P. Fernandes, *Solving Multidimensional 0-1 Knapsack Problem with an Artificial Fish Swarm Algorithm*, University of Minho, 4710-057 Braga, Portugal.

[10] Danial Yazdani, Sara Golyari and Mohammad Reza Meybodi, *A New Hybrid Approach for Data Clustering*, 5th International Symposium on Telecommunication (IST), Tehran, pp. 932-937, 2010.

[11] Lei WANG and Leijuan MA, *A Hybrid Artificial Fish Swarm Algorithm for Bin-packing Problem*, International Conference on Electronic & Mechanical Engineering and Information Technology, pp.27-29, 2011.

[12] Fayez F. Boctor, *A linear formulation of the machine-part cell formation problem*, International Journal of Production Research, Volume 29, N° 2, pp. 343-356, 1991.

[13] Burbidge, J.L., *Production flow analysis*, Production Engineer 42 (12), 742–752, 1963.

[14] Kusiak, A., *The part families problem in Flexible Manufacturing Systems*, Annals of Operational Research 3, 279–300, 1985.

[15] Shargal, M., Shekhar, S., Irani, S.A., *Evaluation of search algorithms and clustering efficiency measures for machine-part matrix clustering*, IIE Transactions 27 (1), 43–59, 1995.

[16] Seifoddini, H., Hsu, C.-P., *Comparative study of similarity coefficients and clustering algorithms in cellular manufacturing*, Journal of Manufacturing Systems 13 (2), 119–127, 1994.

[17] Srinivasan, G., *A clustering algorithm for machine cell formation in group technology using minimum spanning tree*, International Journal of Production Research 32 (9), 2149–2158, 1994.

[18] Deutsch, S.J., Freeman, S.F., Helander, M., *Manufacturing cell formation using an improved p-median model*, Computers and Industrial Engineering 34 (1), 135–146, 1998.

[19] Atmani, A., Lashkari, R.S., Caron, R.J., *A mathematical programming approach to joint cell formation and operation allocation in cellular manufacturing*, International Journal of Production Research 33 (1), 1–15, 1995.

[20] Adil, G.K., Rajamani, D., Strong, D., *A mathematical model for cell formation considering investment and operational costs*, European Journal of Operational Research 69 (3), 330–341, 1993.

[21] Kusiak, A., and Chow, W., *Efficient solving of the group technology problem*, Journal of Manufacturing Systems, 6, 117–124, 1987.

[22] Purcheck, G., *A linear-programming method for the combinatorial grouping of an incomplete set*, Journal of Cybernetics, 5, 51–58, 1975.

[23] Olivia-Lopez, E., and Purcheck, G., *Load balancing for group technology planning and control*, International Journal of MTDR, 19, 259–268, 1979.

[24] Sankaran, S., *Multiple objective decision making approach to cell formation: A goal programming model*, Mathematical Computer Modeling, 13, 71–77, 1990.

[25] Shafer, S., and Rogers, D., *A goal programming approach to cell formation problems*, Journal of Operations Management, 10, 28–34, 1991.

[26] Boulif, M., and Atif, K., *A new branch-&-bound-enhanced genetic algorithm for the manufacturing cell formation problem*, Computers and Operations Research, 33, 2219–2245, 2006.

[27] Aljaber, N., Baek, W., and Chen, C., *A tabu search approach to the cell formation problem*, Computers and Industrial Engineering, 32(1), 169–185, 1997.

[28] Lozano, S., Díaz, A., Eguía, I., and Onieva, L., *A one-step tabu search algorithm for manufacturing cell design*, Journal of the Operational Research Society, 50(5), 1999.

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

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

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

[32] James, T., Brown, E., and Keeling, K., *A hybrid grouping genetic algorithm for the cell formation problem*, Computers and Operations Research, 34(7), 2059–2079, 2007.

[33] Nsakanda, A., Diaby, M., and Price, W., *Hybrid genetic approach for solving large-scale capacitated cell formation problems with multiple routings*, European Journal of Operational Research, 171(3), 1051–1070, 2006.