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

Resolución del problema de diseño de celdas de manufactura utilizando egyptian vulture

Fabián Andrés Aspée Encina

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

Carrera: **Ingeniería de Ejecución en Informática**

Abril 2017

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

Resolución del problema de diseño de celdas de manufactura utilizando egyptian vulture

Fabián Andrés Aspée Encina

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

Carrera: **Ingeniería de Ejecución en Informática**

Abril 2017

Dedicatoria

Proyecto dedicado a amigos y hermano que siempre confiaron en mi; pero especialmente a mi Madre que siempre me dio aliento para seguir avanzando y creciendo.

Fabián Aspée Encina.

Resumen

Este proyecto tiene por objetivo la resolución del Manufacturing Cell Design Problem (MCDP) a través de la metaheurística llamada Egyptian Vulture Optimization Algorithm (EVOA). El MCDP propone beneficios, dentro de los cuales se encuentra el aumento de la productividad y calidad en los procesos. La idea es dividir una planta de producción en un cierto número de celdas, donde cada celda procese piezas de una misma familia. El propósito es minimizar el flujo de piezas entre celdas de tal forma de disminuir los tiempos de producción. Para resolver el modelo de optimización asociado a este problema se utiliza la metaheurística EVOA, la cual se basa en el comportamiento de un ave al momento de alimentarse y que tiene como nombre científico: *Neophron percnopterus*. En esta investigación, se resuelven 90 instancias de este problema alcanzando el óptimo global para todas ellas. **Palabras Claves:** Manufacturing cell design, Egyptian vulture, metaheurística, optimización.

Abstract

This projects is aimed at solving the Manufacturing Cell Design Problem (MCDP) through the metaheuristic called Egyptian Vulture Optimization Algorithm (EVOA). The MCDP proposes benefits, among which we may find the increase in terms of productivity and quality of processes. The idea is to divide a production plant into a number of cells in such a way each cell processes pieces from the same family. The purpose is to minimize the part flow among cells so that production times are reduced. To solve the optimization model associated to this problem we employ the EVOA metaheuristic, which is based on the behavior of a bird at the time of feeding, having as scientific name: *Neophron percnopterus*. In this research work, 90 instances of this problem are solved where the global optimum is reached for all of them. **Key-words:** Manufacturing cell design, Egyptian vulture, metaheuristic, optimization.

Índice

Lista de Figuras

Lista de Tablas

1 Introducción

En la actualidad podemos observar a muchas empresas con un alto grado de automatización en sus procesos de fabricación, que utilizan celdas de manufacturas, el uso de estos dispositivos les permite obtener un mejor grado de eficiencia, mejor nivel de calidad y la capacidad de realizar con rapidez las modificaciones necesarias que requiera el proceso productivo, para adecuarse al mercado competitivo que hoy existe, es por esto que las celdas de fabricación se crearon para facilitar el flujo de trabajo, esto se consigue reuniendo las operaciones (procesos, máquinas, o personas) que participan en una secuencia de proceso de un flujo de productos. La idea de esto es agrupar los procesos unos cerca de otros en distintos grupos, esta agrupación se llama celda, estas celdas se utilizan para el mejoramiento de muchos factores en un entorno de fabricación, por ejemplo: tiempo, costes, producción.

El MCDP es una estrategia de fabricación perteneciente a una sub-sección de just-in-time manufacturing y de lean manufacturing [6] que abarca la tecnología de grupo [18]. El objetivo de MCDP es poder moverse lo más rápido posible realizando el mínimo de desperdicio posible. Como tal, se pretende que una celda de manufactura realice de mejor manera un proceso, pero la re-programación de este proceso es algo delicado y complejo de realizar. La adecuación de una celda requiere tiempo y personal capacitado para su re-programación, la cual al momento de re-programarla interrumpirá el proceso de fabricación que lleve a cabo. El objetivo principal de MCDP es agrupar partes y máquinas similares en celdas, por lo tanto, la celda ideal es independiente, es decir, fabrica completamente su(s) familia(s) de parte(s) [10].

Existen diversos algoritmos que han sido desarrollados para el MCDP buscando dar con buenos resultados. De los cuales podemos mencionar al algoritmo genético [10], tabu search [12], dolphin echolocation algorithm [20]. En este caso se desarrollará MCDP usando EVOA [26]. EVOA es una metaheurística que no posee una explotación en esta área, esta metaheurística está basada en el alimoche, que es un ave carroñera que ha presentado comportamientos llamativos al momento de alimentarse, esto ha llevado a realizar diferentes investigaciones para la resolución de problemas complejos [26, 28, 27].

Este documento se encuentra organizado de la siguiente manera: en el capítulo 2 se definen los objetivos de esta investigación tanto generales como específicos, el capítulo 3 presenta el estado del arte relacionado con MCPD, en el capítulo 4 se explica que son las metaheurísticas y algunas características de estas, en el capítulo 5 se describe la metaheurística EVOA y sus funciones de movimiento, en el capítulo 6 se explica que es el MCDP, en el capítulo 7 se realiza la integración entre MCDP y EVOA, el capítulo 8 muestra los resultados obtenidos por EVOA al aplicarlo en el MCDP y en el capítulo 9 se detallan las conclusiones obtenidas y el trabajo futuro para EVOA.

2 Definición de Objetivos

2.1 Objetivo General

Como objetivo general se busca comprender y resolver el MCDP utilizando la metaheurística EVOA para poder más adelante comparar y analizar dichos resultados obtenidos por EVOA con los resultados establecidos por Boctor [4]

2.2 Objetivos Específicos

- Comprender el MCDP.
- Resolver el MCDP utilizando Egyptian Vulture.
- Comparar y analizar los resultados obtenidos.

3 Estado del Arte

Hoy en día las empresas están en constante mejora de los procesos de manufactura con el fin de ahorrar costes en términos del tiempo y del dinero y así poder aumentar su productividad. Aquí es donde el MCDP busca desarrollar una estrategia que permita a las empresas poder realizar este cometido de una mejor manera. Durante los últimos años, sin embargo, las empresas de fabricación están dirigiendo sus pasos hacia la adopción de una producción del tipo multi-producto y tamaño pequeño de lote, con vistas a adaptarse a un movimiento del mercado caracterizado por una sociedad diversificada y especializada, así como a unos ciclos de vida más corto de los productos. El número de empresas en todo el mundo que utilizan una producción de tipo multi-producto con tamaño pequeño de lote, se espera que se incremente en años venideros.

Inicialmente manufactura significaba una etapa del desarrollo del capitalismo en que la producción era a mano, es decir, producción de los objetos sin intervención de las máquinas; el objeto no es producido por una sola persona; sino por un grupo de ellas, cada una de ellas ejecuta una u otra operación, lo que los lleva a un incremento de la productividad. La manufactura se ha convertido en una porción inmensa de la economía del mundo moderno, según algunos economistas, la fabricación es un sector que produce riqueza en la economía mientras que el sector de servicios es quien tiende a ser el que consume la riqueza, de manera más clara tenemos que: la celda de manufactura es un conjunto de componentes electromecánicos, que trabajan de manera coordinada para el logro de un producto, y que además permiten la fabricación en serie de dicho producto.

El MCDP trae procesos dispersos entre sí para formar caminos cortos, enfocados en el espacio físico concentrado. Además, las celdas llevan una simplificación en costes ya que la producción de artículos está contenida dentro de la celda en vez de estar dispersos a gran distancia.

Luego de la aparición del MCDP se comenzaron a utilizar diferentes algoritmos para la resolución de este, es así como en 1990 se desarrolló el MCDP utilizando una heurística llamada twofold [9]. Luego de su desarrollo, se concluye que twofold tiene una buena adaptación y que además es muy rápido para problemas de larga dimensión, añadiendo que el orden de complejidad es insensible a N, siendo N el largo del problema. Más tarde en 2005 Carlos Andrés y Sebastián Lozano [3] desarrollaron el MCDP con enjambre de partículas (particle swarm) lo que se buscaba era poder optimizar el problema del MCDP ya que resultados computacionales demostraban que el PSO era capaz de encontrar las soluciones óptimas en casi todos los casos, ellos concluyen que el algoritmo propuesto podía generar un óptimo o estar cerca de soluciones que sean óptimas. Después de esto, en 2010 profesores de la Universidad Tecnológica de Pereira desarrollaron el MCDP con un algoritmo de ordenamiento binario (AOB) ellos concluyen que el AOB es una alternativa fácil y rápida para la formación de celdas de manufactura, permitiendo generar una reorganización eficiente de un sistema productivo [13]. Además, se utilizaron técnicas de minería de datos aplicadas en enjambre de partículas para el desarrollo de MCDP [7], en este caso el enfoque del algoritmo se basaba en probabilidades

proporcionales, que se utilizaban en minería de datos. Se concluye en base a un conjunto de experimentos que el algoritmo es estable y presentaba una baja variabilidad. Otras investigaciones realizadas a MCDP son el uso del algoritmo Flower Pollination [22], el algoritmo Bat [19], el algoritmo Invasive Weed [21], el algoritmo Shuffled Frog Leaping [23], el algormitmo Artificial Fish Swarm [24] y Constraint Programming [25].

El problema de la formulación de celdas ha sido tema de considerables investigaciones. Burdidge [5], con sus análisis del flujo de producción, convierte su procedimiento en uno de los primeros para resolver este dilema; aunque no desde un enfoque algorítmico. Su procedimiento utiliza la matriz de incidencia máquina-pieza, y se reorganiza en una forma diagonal de bloque [8].

La manufactura en su sentido más amplio según Schmid [17] es el proceso de convertir la materia prima en producto, es la columna vertebral de cualquier nación industrializada, su importancia queda enfatizada por el hecho que esta es el 20 a 30 por ciento del valor de todos los bienes y servicios producidos. La manufactura también involucra actividades en que el producto se utiliza para crear otros productos, ejemplos de estas son las grandes prensas para conformar la lámina de metal para las carrocerías de automóvil [17].

4 Metaheurística

El término metaheurística fue un término acuñado por F. Glover en el año 1986 [15], etimológicamente deriva de la composición de dos palabras de origen griego que son meta y heurística. Meta (en inglés) se podría traducir como más allá de un nivel superior. Con este término, Glover pretendía definir un procedimiento maestro de alto nivel que guía y modifica otras heurísticas para explorar otras soluciones más allá de un simple óptimo local. Heurística proviene del vocablo griego *heurisken* que podría definirse como encontrar, descubrir o hallar [15]. La definición que aparece en el diccionario de la real academia de la lengua española en su segunda y cuarta acepción es la siguiente:

- Técnica de la indagación y del descubrimiento.
- En algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo o reglas empíricas.

Las metaheurísticas son métodos de más alto nivel e independientes del problema. Por ello no se basan en las características del problema a resolver y por ello, se pueden ver como un método de optimización de caja negra.

Existen algoritmos que permiten obtener soluciones óptimas (algoritmos exactos) [16] para algunos problemas de optimización, los cuales se basan en técnicas de programación dinámica o ramificación y poda, el método de diseño de algoritmos ramificación y poda (también llamado Ramificación y Acotación) es una variante del Backtracking mejorado sustancialmente. El término (del inglés Branch and Bound) se aplica mayoritariamente para resolver problemas de optimización, sin embargo, en la mayoría de los casos, no es posible obtener la solución óptima en tiempo razonable. Las metaheurísticas permiten obtener buenas soluciones en tiempos de búsqueda asumibles. La mayoría de las metaheurísticas se apoyan en una de estas dos características:

- Generación aleatoria de soluciones.
- Utilización de algún tipo de información que permita mejorar o construir mejores soluciones. Este tipo de información se puede obtener de:
    - Características específicas del problema que permiten evaluar o introducir modificaciones en el problema.
    - La estructura de las soluciones previamente evaluadas y sus valores.

4.1 Exploración vs Explotación

El espacio de búsqueda es el conjunto de todas las soluciones posibles. Asumimos que una región del espacio de búsqueda suficientemente pequeña contendrá soluciones similares.

En general, estos algoritmos intentan encontrar un equilibrio entre:

- Exploración o diversificación: La capacidad de explorar nuevas zonas del espacio de búsqueda.
- Explotación o intensificación: La capacidad de mejorar las soluciones prometedoras encontradas en un momento dado.

El algoritmo debe por una parte buscar nuevas soluciones a las encontradas en un momento dado (exploración), pero sin descartar aquellas soluciones que parezcan prometedora (explotación).

4.2 Algunas consideraciones acerca de las metaheurísticas

Ventajas

- Algoritmos de propósito general.
- Funcionan bastante bien en la práctica en cuanto a resultados y tiempo de ejecución a diferencia de las heurísticas.
- Fácilmente paralelizables.

Desventajas

- No garantizan encontrar un resultado óptimo (ni encontrar un resultado de una calidad determinada).
- Aleatoriedad.
- Existen desarrollos teóricos que justifican algunos métodos, pero falta un sustento teórico solido en la mayoría de los casos.

5 Egyptian Vulture

EVOA es una metaheurística compuesta de múltiples funciones de movimiento las cuales se basan en el comportamiento de un buitre. Esta manera de interpretar su forma de obtener la comida ha ayudado a desarrollos de distintos problemas de optimización [27]. EVOA posee diversos tipos de operadores los cuales tratan de resolver un problema de optimización y así obtener mejores resultados. Este se compone de 5 pasos o etapas las cuales se muestran a continuación.

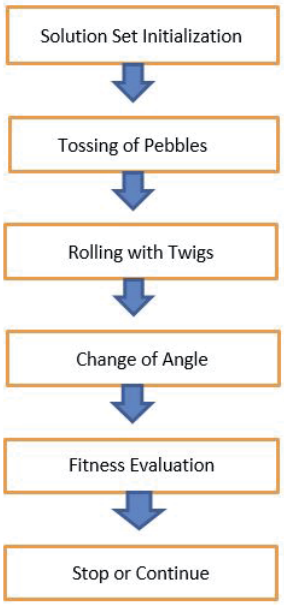

Figura 5.1 Modelo lógico de Egyptian Vulture.

Esta es una pequeña descripción de los pasos lógicos que tiene EVOA, tratando así de dar a conocer cuál es su comportamiento, después de esta explicación se desarrollan de mejor manera 3 conceptos que corresponden a las operaciones principales (funciones de movimiento) que realiza EVOA, ya que el primero es la creación de una solución inicial. La evaluación del fitness es determinar el valor de la solución y parar o continuar es en base a las iteraciones que EVOA posea.

Paso 1: Inicialización del conjunto de solución o vector qué contiene la representación de los parámetros en forma de variables. El vector representa un conjunto de parámetros que globalmente representan un estado solo de solución permisible [27].

Paso 2: Refinamiento de las variables representativas, comprobando las condiciones y limitaciones.

Paso 3: Tossing of Pebbles en puntos seleccionados o aleatorios.

Paso 4: Rolling with Twigs en puntos seleccionados o en toda la solución.

Paso 5: Change of Angle a través de parte selectiva reversal del conjunto de solución.

Paso 6: Evaluación del fitness

Paso 7: Condición de Control para parar o continuar buscando una mejor solución.

Las funciones de movimiento explicadas a continuación se basan en un ejemplo el cual no representa de manera real una solución, más bien se trata de dar a conocer cómo se comporta cada función de movimiento que esta metaheurística posee.

5.1 Tossing of Pebbles

Tossing of Pebbles es una habilidad que EVOA posee, en la cual este utiliza guijarros para romper los huevos de otras aves que resultan relativamente más difíciles y del cual solo pueden obtener la comida de su interior. Dos o tres alimoches continuamente golpean el huevo con fuerza con un guijarro hasta que estos se rompan, en este proceso tratan de buscar los puntos débiles o grietas que los llevaran al éxito. Este es el enfoque utilizado en esta metaheurística para la introducción de la nueva solución, aleatoriamente seleccionara ciertas posiciones e hipotéticamente la solución puede irrumpir y traer consigo cuatro posibilidades que dependen de una probabilidad y de dos parámetros generados por la ejecución de las operaciones [27].

Las dos variables para la determinación de la magnitud de la operación son: PS es el tamaño del guijarro (nivel a ocupar) y FT es la fuerza de descarte (nivel de supresión), donde $PS \geq 0$ y $FT \geq 0$. Con esto se generan 4 operaciones como se puede ver en la ecuación 1 y 2 y donde Entrar y Remover denota la ocupación y la extracción, FT y PS son generados aleatoriamente dentro de un límite. PS es el número de nodos que se generará aleatoriamente considerando que la combinación puede producir un nuevo conjunto de soluciones. FT indica el número de nodos que se extraerán. En general hay cuatro combinaciones de operaciones posibles y son:

$$\text{Pebble size} = \begin{cases} \text{Entrar} & \text{si } PS \text{ es mayor que } 0 \\ \text{No entrar} & \text{si } PS \text{ es igual a } 0 \end{cases} \tag{1}$$

$$\text{Discard force} = \begin{cases} \text{Remover} & \text{si } FT \text{ es mayor que } 0 \\ \text{No remover} & \text{si } FT \text{ es igual a } 0 \end{cases} \tag{2}$$

Además, cabe señalar que la aplicación es única si se piensa hasta qué punto la combinación de funcionamiento tendrá lugar, sí es permisible permitir combinaciones donde $PS \neq FT$, lo que significa que la extracción o la ocupación o ambos serán de longitud desigual y, por lo tanto, introduce el concepto de cadena de longitud variable que es necesaria para encontrar la ruta de problemas. Sin embargo, para

problemas como TSP o QAP, es necesario una combinación $FT = PS$ para justificar las restricciones de números constantes.

5.2 Rolling with Twigs

El Rolling with twigs es otra habilidad asombrosa de EVOA la cual le permite rodar objetos con el fin de poder encontrar la posición ideal o puntos débiles o simplemente dar un vistazo sobre la otra parte que estaba mirando al suelo [27].

Esta actividad de EVOA es considerada como el laminado de la solución para el cambio de las posiciones de las variables y por lo tanto es posible crear nuevas soluciones que pueden producir un mejor valor de fitness y también mejor camino cuando se trata de optimización multi-objetivo. También cuando el golpe es menor y el número de opciones son más, puede tomar un tiempo para que ocurra el evento. En tal caso este tipo de operaciones puede ser útil [27].

En este caso tenemos dos variables las cuales son DS y DR donde DS es la cantidad de rolls a realizar y DR es la dirección de laminado dando lugar a la siguiente probabilidad:

$$\text{Type of rolling} = \begin{cases} \text{rolling derecha} & \text{Si } DR \text{ es igual a 0.} \\ \text{rolling izquierda} & \text{Si } DR \text{ es igual a1.} \end{cases} \tag{3}$$

Donde 0 y 1 se generan de manera aleatoria, y de manera determinante la ecuación será:

$$\text{Another Type of rolling} = \begin{cases} \text{Izquierda/Cambio} & \text{Si rolling RightHalf>LeftHalf} \\ \text{Derecha/Cambio} & \text{Si rolling RightHalf<LeftHalf} \end{cases} \tag{4}$$

Donde $RightHalf$ es el fitness secundario para la mitad derecha de la cadena solución y $LeftHalf$ es para la mitad izquierda. La razón detrás de esto es si la mitad derecha es mejor, entonces esta será una disposición de medida de la fuente con la parte del nodo conectado y lo mismo para la mitad izquierda.

5.3 Change of Angle

Esta es otra operación que EVOA posee y que deriva de la analogía con el change of angle del descarte de guijarros, para experimentar con el procedimiento y aumentar la probabilidad de rotura de los huevos. Ahora bien, el cambio de ángulo se representa como una mutación. Paso por donde el nodo vinculado a la secuencia se invierte para así llegar a estar conectados y, por lo tanto, completar la secuencia de nodos.

Este cambio de ángulo puede ser multi-punto y la búsqueda local decide los puntos, el número de nodos a ser considerados depende del número de nodos que el vector posea [27]. Si el vector está llevando a cabo muchos nodos y Pebble Tossing no se puede realizar, entonces este paso es una buena opción para una búsqueda local y tratar de averiguar la ruta completa de ella.

6 Manufacturing Cell Design Problem

Las celdas de manufactura corresponde a una estrategia de manufactura que divide un sistema de producción en pequeños grupos o células cada una de ellas enfocada a la producción de un conjunto de Partes o componentes. Esta estrategia de producción es una clara consecuencia de la aplicación de la filosofía de la tecnología de grupos, la cual plantea que cosas parecidas deberían ser fabricadas de manera parecida.

Una celda de manufactura es dos o más procesos que agregan valor, unidos de una manera óptima, cuyo objetivo es fabricar una o más unidades de un mismo producto en un corto plazo, de modo que fácilmente se puedan adaptar o cambiar para producir otro producto semejante.

6.1 Un poco de historia

La idea de manufactura celular tiene su origen en las metodologías de agrupamiento (tecnología de grupos), de estas metodologías se desprenden los sistemas de producción dentro de los cuales se ubica la producción celular. Sin embargo, es interesante saber que la idea de la agrupación de familias de Partes surge desde mucho tiempo antes.

F. Koenigsberger reporta en su artículo que alrededor del año 2500AC [11], el hombre ya fabricaba herramientas de corte como flechas, lanzas, machetes, etcétera que eran agrupadas y clasificadas de acuerdo a la geometría y se llegó a la conclusión de que el proceso de fabricación de todas ellas era en esencia el mismo.

6.2 Tecnología de grupos

El concepto básico de la tecnología de grupos se aplica desde hace muchos años como parte de la buena práctica de la ingeniería o de la administración científica [14], por ejemplo, en la manufactura de principios de siglo se usó un sistema de clasificación y codificación desarrollado por F.W. Taylor para la formación de familias de Partes. Durante estos años muchas empresas implantaron sus propios sistemas de clasificación y codificación y los han seguido usando en diversas áreas, como el diseño, materiales y herramientas.

La tecnología de grupo se ha practicado en diversas formas y grados en todo el mundo y por muchos años. En las décadas de 1950 y 1960 muchos países se interesaron en ella [14]. En ese tiempo surgieron varios sistemas de clasificación y codificación, se pusieron en práctica algunos conceptos de células de máquinas y se conocieron muchas excelentes prácticas de maquinado en grupo.

En manufactura, la productividad y el ahorro en los costos se logra explotando similitudes en las operaciones de manufactura, procedimientos de preparación, herramientas utilizadas y manejo de equipo. Las partes que tengan requerimientos de manufactura similares pueden ser procesadas conjuntamente en células de trabajo dedicadas para eso, conduciendo a la reducción de los tiempos de preparación del equipo, del empleo de herramientas y materiales. La manufactura celular que es una aplicación de la tecnología de grupos en manufactura, proporciona una estrategia para obtener ventajas económicas en un medio de gran

variedad de producto, pero de baja demanda de producción.

En el libro publicado por John L. Burbidge en 1970 sobre tecnología de grupos, se hace una recopilación sobre las ponencias expuestas en el seminario (realizado en 1969 en el centro internacional de Turín, Italia) [5]. Fue en este seminario donde se abordaron tres temas principales de la tecnología de grupos: brindar a los líderes expertos de diferentes países un intercambio de ideas, dar una visión a futuro a los industriales, consultores y representantes del gobierno y analizar los aspectos administrativos, económicos y tecnológicos de la tecnología de grupos.

6.3 Evolución de la tecnología de grupo

El sistema de producción por tecnología de grupos, en particular manufactura celular han evolucionado a través de los años es decir la aplicación que tuvo como sistema de producción, ahora se utiliza como estrategia de producción para competir en el mercado mundial, pero no fue sino hasta 1952, en un taller de ingeniería eléctrica en Francia, donde se implantó por primera vez una línea flexible de manufactura.

Jacques Schaffran replantea la línea de producción lineal por una de producción tipo circular, Schaffran plantea las siguientes ventajas:

- La mayor distancia que deberán recorrer los productos agrupados en círculo será el diámetro del círculo.
- Mayor ocupación del tiempo de uso de las máquinas y el desahogo de otras dependiendo del lote de producción.

A Partir del caso anterior se empieza con el concepto de celdas de manufactura, dentro de la filosofía de tecnología de grupos. En sus inicios este sistema de producción estuvo limitado por los requerimientos tecnológicos de la organización por celdas de manufactura; es decir, para que dicho sistema fuera autosuficiente y flexible se requería:

- Alto costo de inversión
- Que el equipo utilizado dentro de la celda tuviese una mayor automatización y autonomía
- Un sistema de codificación de Partes que tomara en cuenta las variantes que se presentan al formar las familias de piezas y grupos de máquinas.
- Mayor utilización de algunas máquinas y desocupado de otras.

6.4 Ventajas y desventajas de las celdas de manufactura celular en una planta

Las principales ventajas y desventajas de MCDP se muestran en la siguiente tabla.

Ventajas	Desventajas
• Reducción del manejo de material • Reducción del inventario de trabajo en proceso • Reducción del tiempo de producción • Incremento del espacio disponible para producción • Reducción y eliminación de tiempos de preparación • Reducción de la cantidad de herramientas a utilizar • Incremento de la utilización de mano de obra • Crean centros de responsabilidad	• Reduce flexibilidad • Reduce utilización de las máquinas • Demanda operarios con mayores habilidades y entrenamiento • Requiere mayor confiabilidad del sistema

Tabla 6.1 Ventajas y Desventajas de MCDP.

6.5 Estado del problema

Uno de los objetivos principales de la manufactura celular o de celdas del modelo de Boctor [4] es el de minimizar los movimientos e intercambio de material entre los grupos, objetivo que es logrado si se generan celdas que garanticen la fabricación completa de los productos asignados. Si esto no es posible entonces existirán piezas que deberán visitar diferentes grupos en su fabricación.

Para lograr la reorganización del sistema de producción es necesario conocer las rutas de fabricación de cada una de las piezas lo cual permite determinar las máquinas visitadas por cada una de las partes producidas en su ruta de fabricación. Una herramienta que permite de manera ordenada resumir esta información es la matriz pieza-máquina en la cual se puede determinar con unos o ceros las máquinas que son necesarias para la fabricación de cada una de las piezas. En esta matriz de manera general se representan las máquinas en filas y las piezas en las columnas, la información que se da a conocer en ningún momento hace referencia al número de máquinas existentes en una instalación o a la secuencia de operaciones. En las figuras 6.2 y 6.3 se da un ejemplo de una matriz máquina-pieza en su inicio y una matriz máquina-pieza en su fin, cada una

compuesta de 0 y 1 teniendo los siguientes significados: si la celda es igual a 1, entonces la máquina i procesa la pieza j, en caso contrario será 0.

	Parte 1	**Parte 2**	**Parte 3**	**Parte 4**	**Parte 5**	**Parte 6**	**Parte 7**
Máquina 1	1	1	0	1	1	1	0
Máquina 2	1	0	1	0	0	0	1
Máquina 3	1	1	1	0	0	0	1
Máquina 4	0	1	0	1	0	1	0
Máquina 5	1	0	1	0	0	0	1

Figura 6.2 Matriz de incidencia inicial $M \times P$

La figura 6.2 muestra una configuración inicial de una matriz máquina-pieza, en la cual si se observa no existe un orden en el cual trabajen las máquinas que procesan ciertas piezas, a continuación se muestra como deberían quedar agrupadas las máquinas por celdas y se da a entender porque MCDP es importante.

		Celda 1			**Celda 2**			
		Parte 1	**Parte 3**	**Parte 7**	**Parte 2**	**Parte 4**	**Parte 5**	**Parte 6**
Celda 1	**Máquina 2**	1	1	1	0	0	0	0
	Máquina 3	1	1	1	1	0	0	0
	Máquina 5	1	1	1	0	0	0	0
Celda 2	**Máquina 1**	1	0	0	1	1	1	1
	Máquina 4	0	0	0	1	1	0	1

Figura 6.3 Matriz final re-ordenada $M \times P$

La figura 6.3 muestra de manera clara como quedan ordenadas las máquinas con las partes que estas procesan, la idea es agrupar un conjunto de máquinas y partes en celdas de manera que la cantidad de movimientos entre estas sea el mínimo y poder así ahorrar costos.

7 Resolución de el MCDP con Egyptian Vulture

En este trabajo, se resuelve el MCDP con vectores y matrices que serán de tipo $[M]$ y $[P \times C]$ donde cada celda del vector $[M]$ representará en su interior la posición de la máquina en una determinada celda y en el otro caso será una matriz de $Pieza \times Celda$.

7.1 Definición de variables y formulas

El MCDP puede ser representado por un modelo matemático según Boctor (1991) [4] de la siguiente manera:

$$min \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} a_{ij} c_{jk} (1 - b_{ik}) \tag{5}$$

La cual posee las siguientes restricciones:

- Solo una máquina sea asignada a una celda.

$$\sum_{k=1}^{C} b_{ik} = 1 \quad \forall i \tag{6}$$

- Solo una parte sea asignada a una celda.

$$\sum_{k=1}^{C} c_{jk} = 1 \quad \forall j \tag{7}$$

- Que un número máximo de máquinas sean asignadas a una celda.

$$\sum_{i=1}^{M} b_{ik} \leq M_{max} \quad \forall k \tag{8}$$

Debido al uso de vectores la primera restricción se validará por si sola ya que, dentro de cada índice de máquinas se encontrará almacenada la celda a la que esta pertenece.

A continuación, se definen las variables utilizadas en el problema.

Constantes:

M = número de máquinas.

P = número de partes.

C = número de celdas.

M_{max} = número máximo de máquinas por celda.

Variables y dominio:

$A = a_{ij}$ la matriz de incidencia binaria máquina pieza $M \times P$, con dominio $[0, 1]$.

$i =$ el índice de las máquinas $(i = 1. . . ., M)$.

$j =$ el índice de las partes $(j = 1. . . .., P)$.

$k =$ el índice de las celdas $(k = 1. . . ..C)$.

$$a_{ij} = \begin{cases} 1 & \text{Sí la parte } J_{th} \text{ visita la máquina } i_{th}. \\ 0 & \text{En otro caso.} \end{cases} \tag{9}$$

$$b_{ik} = \begin{cases} 1 & \text{Sí la máquina } i \text{ está en la celda } k. \\ 0 & \text{En otro caso.} \end{cases} \tag{10}$$

$$c_{jk} = \begin{cases} 1 & \text{Sí la pieza } j \text{ está en la celda } k. \\ 0 & \text{En otro caso.} \end{cases} \tag{11}$$

Al inicializar nuestra solución inicial con vectores la restricción 6 se validará por si sola como se mencionó anteriormente. A continuación, se desarrollara el MCDP utilizando las funciones de movimiento que Egyptian Vulture posee:

7.2 Tossing of Pebbles

Esta función realiza la inserción y extracción de elementos, las dos variables para la determinación de las operaciones son PS como el tamaño de la piedra, en este caso es el número de máquinas a insertar, donde $PS \geq 0$ y FT es igual a la fuerza de extracción que indica la cantidad de máquinas a remover donde FT $\geq$ 0, entonces tenemos que si $PS > 0$ entonces "Entrar" sino "No Entrar", si $FT > 0$ entonces "Remover" sino "No Remover" con esto tenemos 4 operaciones posibles las cuales son: Caso 1: Entrar/ No Remover. Caso 2: No Entrar/Remover. Caso 3: Entrar/Remover. Caso 4 No Entrar/No Remover [26]. La selección de cada caso se realizará de manera aleatoria con una variable de probabilidad que en la práctica definiremos, además en estos casos trabajaremos con vectores como se mencionó anteriormente.

Para algunos problemas como TSP o QAP. Como se mencionó en el capítulo 5 es requerida una combinación de tipo $PS = FT$ para justificar la restricción constante de ciudades. En el MCDP se aplicará la misma combinación para justificar la restricción constante de máquinas. Con esta restricción el caso 1, 2 y 4 no se implementarán, más adelante se explicará cómo se comportarán estos casos con FT y PS iguales.

- Caso 1: Entrar/No Remover

  Se ingresa una máquina y no se extrae ninguna, esto generaría una desigualdad ya que la máquina ingresada puede ser que esté duplicada o que simplemente sea una máquina ficticia y en el único caso que no generaría una desigualdad sería en que $FT = PS = 0$. Por lo tanto, en el caso que sea 0 no realizaría ningún cambio en el vector y la probabilidad de que se realice este caso será nula.

- Caso 2: No Entrar/Remover

  Este caso en particular lo único que realizará será extraer máquinas produciendo una solución no factible, al igual que el caso anterior su probabilidad de realizarse será nula.

- Caso 3: Entrar/Remover

  Este caso será forzado a realizarse la mayor cantidad de veces posibles a través de una variable de probabilidad que es la misma que se mencionó anteriormente, la idea es que Tossing of Pebbles genere cambios en el vector (Explotación) los cuales nos acerquen a una solución más óptima, por lo tanto, si tenemos $FT = PS = 3$ se escogerán 3 máquinas y se insertarán y removerán otras 3 máquinas, dentro de este caso aplicaremos dos formas de extraer y remplazar las cuales son:

  - **Caso 1:** escoger 3 máquinas de manera aleatoria, es decir que se realizarán 3 hit point y en las posiciones que estos caigan se extraerán las máquinas y se reemplazarán por máquinas aleatorias que deberán cumplir con la siguiente restricción.

    Estar dentro del rango de las celdas existentes:

    $0 \leq C \leq C_{max}$

    Que la cantidad máxima de máquinas dentro de esa celda no sea superior al establecido.

    $0 \leq M \leq M_{max}$

    La restricción que una máquina no esté en más de una celda se elimina automáticamente al utilizar un vector ya que se asume que una máquina no está duplicada dentro del vector.

  - **Caso 2:** hacer un hit point aleatorio y que desde ese hit point se cuenten tantas máquinas hacia la derecha como extracciones deban realizarse, al igual que el caso anterior este caso deberá satisfacer las dos restricciones anteriormente mencionadas.

- Caso 4: No Entrar/No Remover

  El caso 4 como su nombre lo indica no ingresa y no remueve por lo tanto este caso no estará en uso ya que su impacto en la metaheurística es nulo.

  A continuación, se expondrá un ejemplo de cómo se comporta Egyptian Vulture y su función de movimiento Tossing of Pebbles con el MCDP.

  Ejemplo:

Array Inicial

M1	M2	M3	M4	M5	M6	M7	M8	M9
C1	C2	C1	C3	C1	C2	C2	C3	C1

Caso 3: Hit Point

Get in/Removal ↓

M1	M2	M3	**M4**	**M5**	M6	M7	M8	M9
C1	C2	C1	**C3**	**C1**	C2	C2	C3	C1

Extrae C3 & C1 e ingresa C3 & C3

M1	M2	M3	**M4**	**M5**	M6	M7	M8	M9
C1	C2	C1	**C3**	**C3**	C2	C2	C3	C1

Figura 7.4 Pebbles of Tossing con MCDP.

Al realizar este ejemplo suponemos de manera directa que poseemos una matriz inicial $[MxP]$ y que aleatoriamente hemos generado nuestras soluciones.

$FT = 2PS = 2$

Las celdas de las máquinas son generadas aleatoriamente en este caso la primera máquina pertenecerá a la celda 3 y la segunda máquina a la celda 3, cabe recordar que son celdas generadas aleatoriamente es decir que no seleccionamos la primera y la segunda más bien son dos celdas que serán ingresadas en las posiciones que el hit point diga.

En este caso realizamos un hit point aleatorio y se extraerán las máquinas que se encuentren a la derecha.

De esta manera Tossing of Pebbles nos genera una nueva solución, además esta solución debe ser factible es por esto que al momento de generar las celdas a donde pertenecerán dichas máquinas se debe comprobar la restricción 8 que se menciona al inicio del capítulo 7.

7.3 Rolling with Twigs

Esta función de Egyptian Vulture nos permite realizar una exploración. Posee dos variables de decisión las cuales son DS y DR.

DS denota la cantidad de rolls a realizar esto es un número natural que va desde $[0 - N]$ y DR indica la dirección del Rolling donde la probabilidad será $DR = 0$ para un Rolling a la derecha y $DR = 1$ para un Rolling a la izquierda donde 0 y 1 se generarán de manera aleatoria [28]. Para aplicarlo a MCDP se realizará un hit point el cual será de manera aleatoria.

Se seleccionará una máquina y se realizarán tantos rolls como DS se denoten, en este caso consideraremos después del hit point las siguientes máquinas como una lista circular y las que estén antes del hit point como una lista.

Ejemplo:

- DR = 3
- DS = 1

Array Inicial

M1	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**
C1	C2	C1	C3	C1	C2	C2	C3

Realizamos una división

division

↓

M1	M2	M3	M4	**M5**	**M6**	**M7**	**M8**
C1	C2	C1	C3	**C1**	**C2**	**C2**	**C3**

Realizar movimientos de celdas como DS existan

$DS = 1 \quad DR = 0$. Primer Roll.

M1	M2	M3	M4	**M5**	**M6**	**M7**	**M8**
C1	C2	C1	C3	**C3**	**C1**	**C2**	**C2**

$DS = 2 \quad DR = 0$. Segundo Roll.

M1	M2	M3	M4	**M5**	**M6**	**M7**	**M8**
C1	C2	C1	C3	**C2**	**C3**	**C1**	**C2**

$DS = 3 \quad DR = 0$. Tercer Roll.

M1	M2	M3	M4	**M5**	**M6**	**M7**	**M8**
C1	C2	C1	C3	**C2**	**C2**	**C3**	**C1**

Como resultado obtenemos el array.

M1	M2	M3	M4	**M5**	**M6**	**M7**	**M8**
C1	C2	C1	C2	**C2**	**C2**	**C3**	**C1**

Figura 7.5 Rolling with Twigs con MCDP.

Esto nos ayuda a buscar una mejor solución de manera local. En caso que fuera $DS = 0$ se realiza lo mismo que en el paso anterior solo que en vez de ir hacia la derecha ira hacia la izquierda.

7.4 Change of Angle

La función change of angle lo que intenta conseguir es una mejor solución haciendo una búsqueda local seleccionando aquellas celdas que hacen que la solución no sea factible y cambiándolas por celdas que consigan que la solución sea factible. Las celdas que se generan deberán respetar las restricciones descritas en la sección 7, en si la función cambio de ángulo será una función que repare aquellas soluciones que no sean factibles.

Ejemplo:

- C = 3
- $M_{max} = 3$

Vector Inicial

M1	M2	M3	M4	M5	M6	M7	M8	M9
C1	C2	C1	C3	C1	C2	C2	C3	C2

Hit Point

↓

M1	M2	M3	M4	M5	M6	M7	M8	M9
C1	**C2**	C1	C3	C1	C2	C2	C3	C2

Ingresamos la máquina en una celda que aun tenga espacio para tener una máquina y obtenemos el siguiente resultado.

M1	M2	M3	M4	M5	M6	M7	M8	M9
C1	**C3**	C2	C3	C1	C2	C2	C3	C1

Figura 7.6 Change of Angle con MCDP

7.5 Pseudo-código Egyptian Vulture

Algoritmo 7.1 Egyptian Vulture

1: Inicializar Parámetros:
2: - V: Número de Egyptian Vultures.
3: - G: Máximo número de generaciones.
4: - t: Índice que indica las generaciones.
5: - i: Índice que indica la posición del Egyptian Vulture (EV) en el array de población.
6: Genera población inicial V de Egyptian Vultures.
7: BA = Extrae a Egyptian Vulture (mejor solución) en la población.
8: **while** $t < G$ **do**
9: **for** $i = 1$ a V **do**
10: Realizar pebbles tossing en el EV_i.
11: Realizar rolling with twigs en el EV_i.
12: Realizar change of angle en el EV_i.
13: Realizar la evaluación del fitness en el EV_i.
14: **if** fitness(BA) < fitness(EV_i) **then**
15: $BA = EV_i$
16: **end if**
17: **end for**
18: **end while**
19: **return** BA

7.6 Creación de la matriz $[P \times C]$

Para realizar la matriz $P \times C$, se crea una matriz de dimensiones igual a la cantidad de celda por piezas, lo que se hace aquí es:

- Cada pieza-celda poseerá un 1 si tiene asignada la parte o un 0 en otro caso.
- Con la matriz $P \times C$ ya creada y rellenada con todas las piezas que cada celda posea, existirá una segunda matriz la cual se rellenará cumpliendo la restricción 7 planteada en el capítulo 7 esta segunda matriz también $P \times C$ se generará una cantidad de veces determinada la idea de esto es poder evaluar las distintas combinaciones que puedan existir y evaluar cuál de todas es más óptima almacenando cada una y luego conservando la de mejor fitness, a continuación se ejemplifica la idea de creación de la matriz $P \times C$.

M = 7 P = 7

Matriz Máquina x Pieza

	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
M1	1	0	1	1	0	0	1
M2	0	1	0	1	1	0	0
M3	0	1	1	1	0	1	1
M4	0	0	0	1	1	0	0
M5	1	1	0	1	0	0	1
M6	0	1	1	0	0	0	1
M7	0	0	1	0	1	1	0

Vector Máquina x Celda

M1	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**
1	2	1	3	2	1	3

C1

M1	**M3**	**M5**
P1	P2	P2
P3	P3	P3
P4	P4	P7
P7	P6	
	P7	

C2

M2	**M5**
P2	P1
P4	P2
P5	P4
	P5

C3

M4	**M7**
P4	P3
P5	P5
	P6

Figura 7.7 Ejemplo de Agrupación de Piezas

Matriz Pieza x Celda Inicial

	C1	**C2**	**C3**
M1	1	1	0
M2	1	1	0
M3	1	0	1
M4	1	1	1
M5	0	1	1
M6	1	0	1
M7	1	0	0

Figura 7.8 Ejemplo Matriz $P \times C$

En la imagen anterior aparece la matriz inicial (Matriz de ejemplo) que es máquinas por piezas el vector máquina celda y luego se ejemplifica como irían agrupadas las piezas según la máquina y celda a la que corresponden, notar que las piezas se repiten a través de las celdas pero no las máquinas, luego de esto se rellena una matriz $P \times C$ con las piezas pertenecientes a cada celda, la cual se ejemplifica a continuación. Al tener esta matriz creamos otra matriz la cual cumplirá las restricciones que se piden, esta matriz mostrada anteriormente es una matriz binaria la cual utilizaremos para cubrir cobertura, saber que piezas están en cada celda y poder así después en una segunda matriz probar combinaciones. Siguiendo el ejemplo anterior tendríamos (Caso aleatorio).

Matriz

Máquina x Pieza Final

	C1	**C2**	**C3**
M1	1	0	0
M2	0	1	0
M3	1	0	0
M4	0	1	0
M5	0	0	1
M6	0	0	1
M7	1	0	0

Figura 7.9 Ejemplo Matriz $P \times C$ Final

Luego de esto aplicamos la siguiente fórmula para calcular su fitness.

$$min \sum_{k=1}^{C} \sum_{i=1}^{M} \sum_{j=1}^{P} a_{ij} c_{jk} (1 - b_{ik}) \tag{12}$$

7.7 Otra manera de crear la matriz $[P \times C]$

Otra manera de generar la matriz máquina x pieza se dará a conocer en esta sección la idea de todo esto es poder encontrar la manera más correcta de generar esta matriz de modo tal que nos lleve a resultados óptimos en la ejecución de nuestra metaheurística sobre el problema planteado, como vimos en la sección 7.6 una manera de generar esta matriz sería tener una matriz auxiliar y probar todas las posibles combinaciones para poder así dar la oportunidad de encontrar un resultado óptimo, bueno en esta sección la idea es verificar cuantas veces una pieza pertenece a una máquina y así asignarla a la celda de dicha máquina, en otras palabras tenemos:

Matriz Inicial

	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**
M1	1	0	1	0	1	0	1	1	1
M2	1	1	0	0	0	1	1	0	0
M3	0	0	0	1	1	1	0	1	0
M4	1	1	0	0	1	1	0	0	1
M5	0	0	1	1	0	0	1	1	0
M6	1	0	1	0	1	0	1	0	1

Figura 7.10 Matriz $P \times C$ Inicial

- Celdas = 2
- Mmax = 4

Vector que representa las máquinas asignadas a sus celdas.

M1	M2	M3	M4	M5	M6
C1	C2	C1	C1	C1	C2

Figura 7.11 Vector Máquina [Celda]

Teniendo estos datos principales y asumiendo que estamos dentro del proceso de formación de la matriz máquina x pieza, se dará a conocer a continuación como quedaría si la creamos de otra manera, por lo tanto, tenemos en resumen:

- C1 = M1, M3, M4, M5.
- C2 = M2, M6.

A continuación, se agruparán las máquinas por celdas con sus respectivas piezas asociadas.

C1			
M1	**M3**	**M4**	**M5**
P1	P4	P1	P3
P3	P5	P2	P4
P5	P6	P5	P7
P7	P8	P6	P8
P8		P9	
P9			

C2	
M2	**M6**
P1	P1
P2	P3
P6	P5
P7	P7
	P9

Figura 7.12 Máquinas x Piezas Asignadas por celda

A continuación, vamos a resumir los datos mostrados anteriormente para comenzar a aclarar la idea de lo que queremos hacer. Por lo tanto tenemos que P1 pertenece a C1 como a C2, entonces para determinar a qué celda debería pertenecer dicha pieza contaremos cuantas veces está esa pieza en esa celda, de una manera más claras cuantas máquinas que pertenecen a esa celda tienen esa pieza, en este caso tenemos que P1 pertenece a M1, M4 de C1 y pertenece a M2, M6 de C2, podemos observar que la cantidad es igual P1 está dos veces en C1 y dos veces en C2 por lo tanto el lugar a donde se asigna no alterará el resultado, ahora supongamos que tenemos que asignar P3 a una celda hacemos el mismo desarrollo descrito anteriormente y por lo tanto tenemos que P3 está 2 veces en C1 mientras que en C2 solo se encuentra una vez, tras sucederé esto asignamos P3 a C1 ya que ayudaría a encontrar una solución más óptima. Entonces como resultado final de la matriz $[P \times C]$ obtendríamos:

	C1	**C2**
P1	0	1
P2	0	1
P3	1	0
P4	1	0
P5	1	0
P6	1	0
P7	0	1
P8	1	0
P9	1	0

Figura 7.13 Matriz $P \times C$ final

8 Experimentación

El proceso de experimentación del algoritmo fue llevado a cabo en un notebook con procesador AMD A10-4600M APU con Radeon(tm) HD Graphics 2.30 GHz Turbo Speed 3.20 GHz con 8 GB RAM y un sistema operativo Windows 10. El lenguaje utilizado es Java y se trabajó en el entorno de desarrollo Eclipse. Los problemas que se resolvieron difieren en las matrices de origen (a_{ij}), el número de celdas (C) y el número máximo de máquinas por celdas (M_{max}). La primera configuración que fue puesta a prueba tiene los valores $C = 2$ y con 50 casos de prueba, mientras que la segunda configuración que fue puesta a prueba tiene los valores $C = 3$ y 40 casos de prueba. Cada configuración se iteró 31 veces. Los resultados obtenidos se pueden apreciar en la tabla 8.2 para $C = 2$ y 8.3 para $C = 3$.

	C=2																			
P	$M_{max} = 8$				$M_{max} = 9$				$M_{max} = 10$				$M_{max} = 11$				$M_{max} = 12$			
	O	EV	A	RPD	O	EV	A	RPD	O	EV	A	RPD	O	EV	A	RPD	O	EV	A	RPD
1	11	11	11	0.00	11	11	11	0.00	11	11	11	0.00	11	11	11	0.00	11	11	11	0.00
2	7	7	7	0.00	6	6	6	0.00	4	4	4,1	0.00	3	3	3	0.00	3	3	3	0.00
3	4	4	4	0.00	4	4	4	0.00	4	4	4	0.00	3	3	3	0.00	1	1	1	0.00
4	14	14	14	0.00	13	13	13	0.00	13	13	13	0.00	13	13	13	0.00	13	13	13	0.00
5	9	9	9,1	0.00	6	6	6	0.00	6	6	6	0.00	5	5	5	0.00	4	4	4	0.00
6	5	5	5	0.00	3	3	3	0.00	3	3	3	0.00	3	3	3	0.00	2	2	2,1	0.00
7	7	7	7	0.00	4	4	4	0.00	4	4	4	0.00	4	4	4	0.00	4	4	4,1	0.00
8	13	13	13	0.00	10	10	10	0.00	8	8	8	0.00	5	5	5	0.00	5	5	5	0.00
9	8	8	8	0.00	8	8	8	0.00	8	8	8	0.00	5	5	5	0.00	5	5	5	0.00
10	8	8	8	0.00	5	5	5	0.00	5	5	5	0.00	5	5	5	0.00	5	5	5	0.00

Tabla 8.2 Resultados Experimental con $C = 2$

	C=3															
P	$M_{max} = 6$				$M_{max} = 7$				$M_{max} = 8$				$M_{max} = 9$			
	O	EV	A	RPD	O	EV	A	RPD	O	EV	A	RPD	O	EV	A	RPD
1	27	27	28	0.00	18	18	18,6	0.00	11	11	11	0.00	11	11	11	0.00
2	7	7	7	0.00	6	6	6	0.00	6	6	6	0.00	6	6	6	0.00
3	9	9	9,1	0.00	4	4	4,7	0.00	4	4	4	0.00	4	4	4	0.00
4	27	27	27	0.00	18	18	18,1	0.00	14	14	14	0.00	13	13	13	0.00
5	11	11	11,1	0.00	8	8	8	0.00	8	8	8	0.00	6	6	6,4	0.00
6	6	6	6	0.00	4	4	4,2	0.00	4	4	4,2	0.00	3	3	3,2	0.00
7	11	11	11,3	0.00	5	5	5	0.00	5	5	5,1	0.00	4	4	4,1	0.00
8	14	14	14	0.00	11	11	11	0.00	11	11	11	0.00	10	10	10,4	0.00
9	12	12	12	0.00	12	12	12	0.00	8	8	8,5	0.00	8	8	8,1	0.00
10	10	10	10,4	0.00	8	8	8	0.00	8	8	8	0.00	5	5	6,2	0.00

Tabla 8.3 Resultado Experimental con $C = 3$

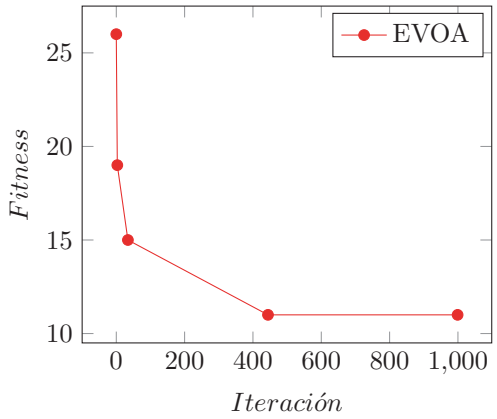

Figura 8.14 Gráfico de convergencia del problema 1 de Boctor con $C = 2$ y $M_{max} = 8$

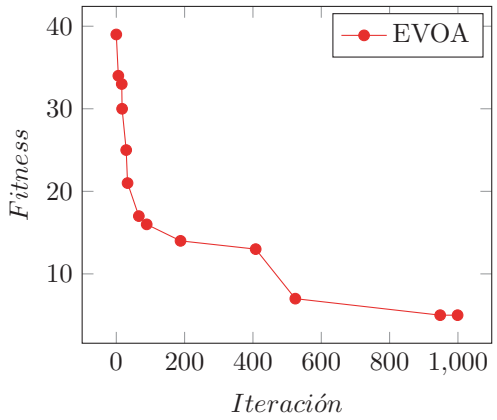

Figura 8.15 Gráfico de convergencia del problema 10 de Boctor con $C = 3$ y $M_{max} = 9$

Podemos observar que los resultados obtenidos con Egyptian Vulture son en su mayoría óptimos, teniendo solo 4 sobre 50 casos en que $C = 2$ tuvo un promedio sobre el fitness óptimo, mientras que para $C = 3$ tenemos que 18 sobre 40 casos tuvieron un promedio sobre el fitness pero, aun así se logró llegar al óptimo en cada caso, el promedio refleja las 31 iteraciones que se realizaron sobre cada caso de prueba, algunas no llegaron a su óptimo, pero aun así su promedio no supera en 1 a su fitness ideal, cabe destacar además, que los gráficos de convergencia 8.14 8.15 muestran de que manera se comportó la metaheurística con el paso de las iteraciones, podemos concluir que claramente para problemas de $C = 2$ este converge de una manera más rápida, mientras que para problemas de $C = 3$ la metaheurística tiene mayor dificultad a la hora de alcanzar un óptimo, notando así que para $C = 2$ el óptimo se alcanza en la iteración 444 y para el problema con $C = 3$ el óptimo se alcanza en la iteración 948, más del doble que un problema con $C = 2$.

9 Conclusión

Tras realizar esta investigación, se ha podido observar el comportamiento que tiene EVOA al momento de resolver problemas de optimización, en este caso en particular, MCDP, pudiendo así dar claro conocimiento que esta metaheurística tiene un buen comportamiento para este problema, obteniendo el óptimo en todos los casos y solo obteniendo en pocos casos un promedio de su fitness en no más de 1 en comparación con su fitness ideal, además cabe destacar que esta es una metaheurística que no tiene mucho campo de explotación y que a pesar de ser nueva presenta muy buenos resultados, además de ser una metaheurística fácil de comprender y además de codificar.

Comprender y adaptar EVOA al problema no fue algo sumamente complicado pero, luego de su codificación, implementación y buenos resultados, se observó que si esta posee mucha población, alcanza los óptimos sin problemas pero, el tiempo que consume para hacerlo es mucho mayor al que posee con la configuración utilizada en esta investigación, además para problemas mucho más grandes como los que se pueden encontrar en la referencia [1] y que los resultados de estos se pueden encontrar en la referencia [2], EVOA a pesar de obtener buenos resultados, consume mucho tiempo en poder encontrarlos.

Para generalizar y no entrar en un desarrollo tan amplio EVOA mostró ser mejor de lo que se esperaba y el MCDP ser un problema que aún puede seguir siendo explotado.

Como trabajo futuro se espera que se realicen pruebas en EVOA utilizando Autonomous Search, además de analizar su comportamiento realizando multi-thread para luego así determinar de que manera es posible hacer EVOA más eficiente al momento de resolver problemas de optimización, además se espera que EVOA tenga el mismo comportamiento en cuanto a rendimiento en el desarrollo de otros problemas de optimización y que sus resultados sean igual de bueno como los obtenidos en el MCDP.

Para finalizar se menciona que se ha realizado una publicación WoS en donde se compara el rendimiento de EVOA y Firefly Algorithm para resolver MCDP, dicha publicación se realizó en IET Software [2]

Referencias

[1] B. Almonacid and F. Aspee. Dataset - modified binary firefly algorithm and egyptian vulture optimization algorithm for solving manufacturing cell design problem. `http://www.inf.ucv.cl/~balmonacid/MPCFP/IET_AKISM2016`, 2016. [Online; accessed 20-March-2017].

[2] B. Almonacid and F. Aspee. Solving manufacturing cell design problem using modified binary firefly algorithm and egyptian vulture optimization algorithm. `http://digital-library.theiet.org/content/journals/10.1049/iet-sen.2016.0196`, 2016. [Online; accessed 20-March-2017].

[3] C. Andres and S. Lozano. A particle swarm optimization algorithm for part–machine grouping. *Robotics and Computer-Integrated Manufacturing*, 22(5):468–474, 2006.

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

[5] J. L. Burbidge. *Group technology: proceedings of an international seminar held at the Turin International Centre from 8th to 13th September 1969*. International Centre for Advanced Technical and Vocational Training, 1973.

[6] A. Ceballos Gutiérrez. Celda-de-manufactura. IPN ESIME Azcapotzalco pp 1 Disponible vía web https://es.scribd.com/doc/92569395/Celda-de-manufactura. Revisada por última vez el 6 de mayo del 2012.

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

[8] H. W. Eves. *Elementary matrix theory*. Courier Corporation, 1980.

[9] G. Harhalakis, R. Nagi, and J. Proth. An efficient heuristic in manufacturing cell formation for group technology applications. *The International Journal of Production Research*, 28(1):185–198, 1990.

[10] J. A. Joines, C. T. Culbreth, and R. E. King. Manufacturing cell design: an integer programming model employing genetic algorithms. *IIE transactions*, 28(1):69–85, 1996.

[11] F. Koenigsberger. The use of group technology in the industries of various countries. *CIRP Annalen*, 21(2):209–14, 1972.

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

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

[14] N. Miranda Pasos. Tecnología de grupo y manufactura celular. Master's thesis, Universidad de Sonora .División de Ingeniería, 1997.

[15] A. D. Muñoz. *Metaheurísticas*, volume 22. Librería-Editorial Dykinson, 2007.

[16] F. L. Ruiz. Nuevos métodos para la obtención de soluciones iniciales en el problema de transporte. *Revista de Dirección y Administración de Empresas*, 1(10), 2014.

[17] S. R. Schmid. *Manufactura, Ingenieria y Tecnologia*. Prentice Hall, 2002.

[18] N. Singh. Design of cellular manufacturing systems: an invited review. *European journal of operational research*, 69(3):284–291, 1993.

[19] R. Soto, B. Crawford, A. Alarcón, C. Zec, E. Vega, V. Reyes, I. Araya, and E. Olguín. Solving manufacturing cell design problems by using a bat algorithm approach. In *International Conference in Swarm Intelligence*, pages 184–191. Springer, 2016.

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

[21] R. Soto, B. Crawford, C. Castillo, and F. Paredes. Solving the manufacturing cell design problem via invasive weed optimization. In *Artificial Intelligence Perspectives in Intelligent Systems*, pages 115–126. Springer, 2016.

[22] 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.

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

[24] R. Soto, B. Crawford, E. Vega, and F. Paredes. Solving manufacturing cell design problems using an artificial fish swarm algorithm. In *Mexican International Conference on Artificial Intelligence*, pages 282–290. Springer, 2015.

[25] R. Sotoyz, B. Crawford, B. Almonacid, F. Paredes, and E. Loyola. Machine-part cell formation problems with constraint programming. In *Chilean Computer Science Society (SCCC), 2015 34th International Conference of the*, pages 1–4. IEEE, 2015.

[26] C. Sur, S. Sharma, and A. Shukla. Egyptian vulture optimization algorithm–a new nature inspired meta-heuristics for knapsack problem. In *The 9th International Conference on Computing and InformationTechnology (IC2IT2013)*, pages 227–237. Springer, 2013.

[27] C. Sur, S. Sharma, and A. Shukla. Solving travelling salesman problem using egyptian vulture optimization algorithm–a new approach. In *Language Processing and Intelligent Information Systems*, pages 254–267. Springer, 2013.

[28] C. Sur and A. Shukla. Road traffic management using egyptian vulture optimization algorithm: A new graph agent-based optimization meta-heuristic algorithm. In *Networks and Communications (NetCom2013)*, pages 107–122. Springer, 2014.