Trabajo de Fin de Grado
Ingeniería de Organización Industrial

Problema de secuenciación con ensamblado en tres etapas

Autor: Salvador Aragón Parrado
Tutor: José Manuel Framiñán Torres

Dep. Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Trabajo de Fin de Carrera
Ingeniería de Organización Industrial

Problema de secuenciación con ensamblado en tres etapas

Autor:
Salvador Aragón Parrado

Tutor:
José Manuel Framiñán Torres

Dep. De Organización Industrial y Gestión de Empresas
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Trabajo de Fin de Carrera: Problema de secuenciación con ensamblado en tres etapas

Autor: Salvador Aragón Parrado

Tutor: José Manuel Framiñán Torres

El tribunal nombrado para juzgar el Trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

Agradecimientos

Me gustaría que estas líneas sirvieran para expresar mi más profundo y sincero agradecimiento a todas aquellas personas que con su ayuda han colaborado en la realización del presente trabajo, en especial a José Manuel Framiñán Torres, por la orientación, el seguimiento y la supervisión continua, pero sobre todo por la motivación y el apoyo recibido a lo largo del proyecto.

También agradecer a todos los profesores que he tenido a lo largo del grado, en especial a los del departamento de Organización Industrial y Gestión de Empresas, no solo por el contenido de las asignaturas sino también por los consejos. Por otro lado, gracias a mis compañeros por hacer todo más ameno y sencillo.

Llegar hasta este punto ha sido un camino difícil, por este motivo quiero dar las gracias a mi familia y amigos por la comprensión, paciencia y el ánimo recibido durante todos estos años, me permiten seguir siempre adelante con gran confianza.

Salvador Aragón Parrado

Alumno de la Escuela Técnica Superior de Ingeniería

Sevilla, 2018

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO 0: OBJETO DEL PROYECTO

0.1 Objeto del proyecto

El objetivo principal del presente Trabajo Fin de Grado es estudiar y resolver un problema de programación de la producción con una gran similitud al modelo actual que utilizan la gran mayoría de las empresas para elaborar sus productos. Este modelo está compuesto por tres fases: la primera corresponde a la fabricación de las piezas necesarias para la fabricación del producto, la segunda es la fase de transporte de estas hacía la línea de ensamblado, y por último, el acoplamiento de las piezas.

El objetivo del problema será disminuir el tiempo total que transcurre desde el principio hasta el final del proceso, más conocido en la programación de la producción como *makespan*. La dificultad principal viene dada por tratarse de un problema NP-Hard, demostrado por Koulamas and Kyparisis (2001) en el estudio de una situación similar, pero compuesto por dos fases, obviando el transporte de las piezas para su ensamblado. Debido a su complejidad, se opta por la utilización de heurísticas para adquirir una solución, las cuales aportan resultados aceptables y los tiempos de ejecución para encontrarla disminuyen.

Las dos heurísticas utilizadas se basan en el principio de la iteración de soluciones aleatorias que posteriormente se les aplica el método de búsqueda local; dichas heurísticas son GRASP e IDCOA. La primera es considerada un algoritmo muy robusto debido a la gran capacidad de obtener eficientes resultados en diferentes tipos de problemas. La segunda viene por el estudio de G.M. Komaki et al. (2017), la cual consiste en la evolución de la heurística COA, en él se demuestra la mejora de resultados contra su predecesor y otras metaheurísticas como VNS o SA.

La principal diferencia entre las heurísticas estudiadas es la inicialización de las soluciones aleatorias para la realización de la búsqueda local, ya que GRASP no realiza ningún paso previo, mientras que IDCOA filtra dichas soluciones iniciales realizando pequeños cambios en las secuencias para partir de soluciones mejores. Basándose en esta premisa, el objetivo del proyecto será demostrar que partir de soluciones con mejor valor de la función objetivo provoca que los resultados sean mejores y que se obtengan en menor tiempo de computación.

0.2 Sumario

El proyecto que se ha realizado se estructura de la siguiente manera:

- El primer capítulo se divide en tres partes, el primero consta de una visión general de la programación de la producción con el fin de aportar una perspectiva de las principales variables que intervienen en estos problemas. En el segundo, se expone un breve repaso a la historia de las líneas de ensamblado mediante su principal exponente, Henry Ford. Y por último, se finaliza con la descripción detallada del problema a estudiar.

- En el segundo capítulo se explican las principales características y beneficios que poseen las heurísticas, para posteriormente describir de forma detallada las dos que se van a utilizar en el proyecto, IDCOA y GRASP. Por último, un prefacio del programa Code::Blocks, el cual se ha utilizado para codificar ambos algoritmos.

- El tercer capítulo comienza con la descripción de los escenarios que van a ser utilizados para el estudio. A continuación, se realiza un análisis previo de IDCOA con el fin de elegir los valores óptimos de sus respectivas variables. Posteriormente, se procede a la representación y análisis de los resultados de GRASP. Finalizando la sección, con la comparación de los resultados obtenidos, las variables a comparar serán la función objetivo y el tiempo de ejecución, finalizando en qué casos de la batería de pruebas dan mejores resultados cada heurística.

- El último capítulo trata sobre las conclusiones obtenidas tras el estudio y posibles líneas futuras relacionadas con este proyecto

CAPÍTULO I: INTRODUCCIÓN Y DESCRIPCIÓN DEL PROBLEMA

El objetivo del capítulo es aportar al lector una mayor comprensión del problema a estudiar, gracias a una breve introducción de las principales variables de la programación de la producción. Además, se explicará con detalle la parte más crítica del problema, la línea de ensamblado, explicando sus orígenes con su máximo exponente, Henry Ford, y sus principales ventajas e inconvenientes. Por último, se realizará una descripción detallada del problema indicando las restricciones que posee y la función objetivo a resolver.

1.1 Introducción a la programación de la producción

La finalidad de la programación de la producción es establecer un tiempo de inicio y fin de cada trabajo en cada una de las máquinas o estaciones necesarias para obtener el producto final. Para poder conseguir dichos tiempos de inicio y finalización es necesario que previamente cada trabajo haya sido cargado (asignar su realización en una máquina en concreto) y puesto en secuencia, es decir, establecer el orden de ejecución de cada trabajo en cada máquina.

El rango de variación del período de programación puede ser de entre unas pocas horas hasta una semana, tiempo durante el cual únicamente se aceptan órdenes de producción de emergencia (Companys Pascual, 1989). El período ideal de programación sería de cero (Pinedo, 2009), es decir, las órdenes se reciben y se asignan continuamente, de igual modo, el trabajo en proceso y la programación se revisarán de forma continua como resultado de los continuos cambios que ocurren dentro de la planta.

Las decisiones que deben tomarse a la hora de programar la producción deben tener las siguientes características (Framiñán et al., 2014):

- Exhaustivas y con un gran nivel de detalle: el volumen de datos a manejar debe ser muy grande debido a la necesidad de establecer el tiempo de inicio de cada trabajo en cada recurso de la empresa.

- Se deben tomar en intervalos cortos de tiempo, a ser posible en tiempo real:

  - Horizonte rodante (*Rolling horizon*): consiste en la obtención de un programa para un número determinado de intervalos de tiempo, del que solo se ejecuta el primer intervalo. Con esto se consigue incorporar los cambios y variaciones ocurridas en cada período.

  - Reprogramación (*Rescheduling*): se trata de modificar el programa debido a la incertidumbre del entorno de la producción.

- Tiene una gran influencia en los resultados finales.

- Las condiciones de la decisión, las restricciones y los objetivos son muy variables ya que cada empresa o fábrica posee diferentes características u objetivos.

- Las principales variables que intervienen en cualquier problema de la programación de la producción se dividen en:

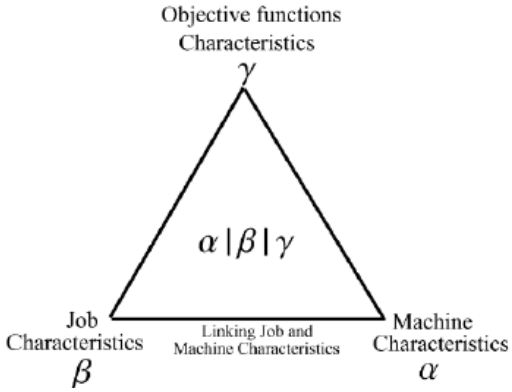

Figura 1.Características de los programas de producción. (Fuente: Paz Pérez, José Framiñán, 2016)

1.2 Características de la programación de la producción

Un problema de programación de la producción se identifica por la letra de los tres campos α , β y γ. Cada una de estas letras corresponde a una característica del modelo, máquinas, trabajos y función objetivo, respectivamente.

1.2.1 Máquinas

Se trata principalmente de la disposición de las máquinas, en el cual se encuentran dos partes diferenciadas, la primera se trata de sistemas con máquinas en serie, cada máquina realiza una operación de todas las que conlleva el trabajo, y por otra parte, los sistemas de máquinas paralelas, se entiende que un determinada operación del trabajo puede ser procesada por cualquier máquina de entre el conjunto.

- Máquinas en serie

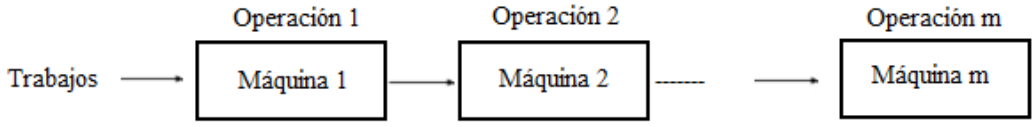

Figura 2.Máquinas en serie. (Fuente: Elaboración propia)

Dentro de las máquinas en serie, existe una clasificación en función del modelo o esquema de paso de los trabajos por las diferentes máquinas:

- α = F: Sistemas de flujo uniforme (Flow shop): El modelo de paso es el mismo para todos los trabajos. Todos los trabajos pasan por cada una de las máquinas del sistema usando el mismo orden de paso por las mismas.

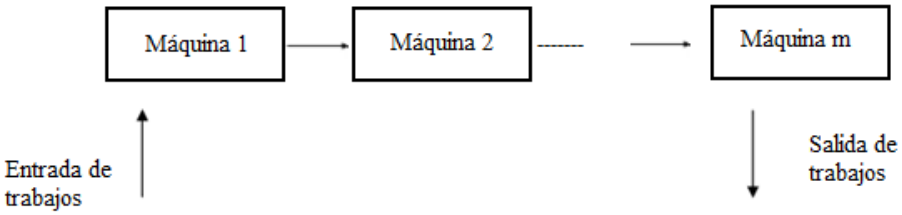

Figura 3.Flow Shop. (Fuente: Elaboración propia)

- α = J: Sistemas de tipo taller (Job shop): Cada trabajo tiene su propio esquema de paso por las máquinas.

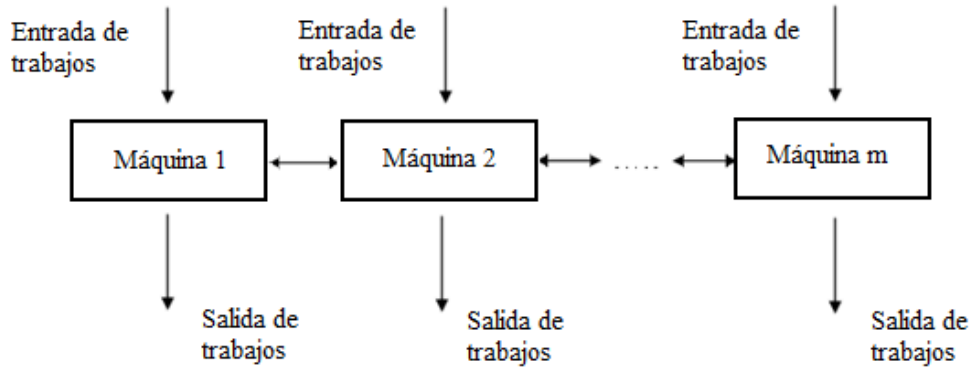

Figura 4.Job Shop. (Fuente: Elaboración propia)

- α = O: taller abierto (Open Shop): Cada una realiza un proceso diferente, siendo necesario que cada trabajo sea procesado en todas las máquinas. La ruta de los trabajos no está determinada, siendo posible cualquier combinación. Siendo así el más general y complejo de los talleres.

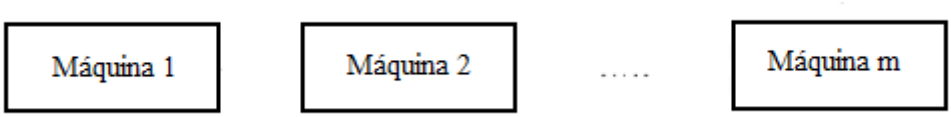

Figura 5.Open Shop. (Fuente: Elaboración propia)

- Máquinas en paralelo:

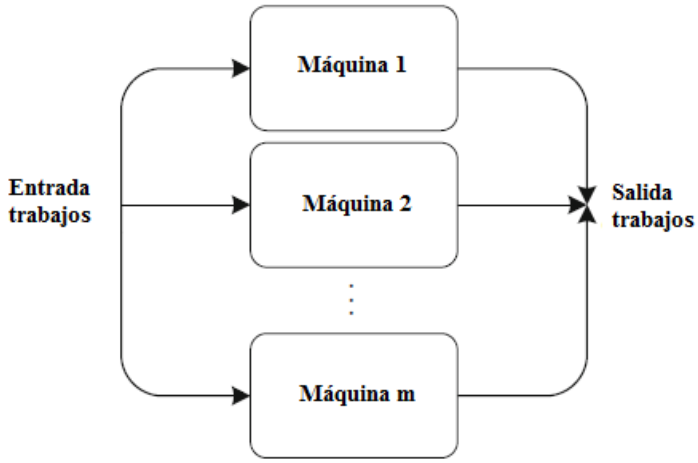

Figura 6.Máquinas en paralelo. (Fuente: Elaboración propia)

Como pasaba anteriormente con las máquinas en serie, las paralelas también se pueden dividir en este en tres grupos:

- α = P: máquinas paralelas idénticas. El tiempo de proceso de cada trabajo en cada una de las máquinas no depende de la máquina donde se procese.
- α = Q: máquinas paralelas uniformes. Cada máquina tiene diferentes velocidades de procesado, siendo la misma para todos los trabajos, es decir, una máquina más rápida procesa los trabajos más rápido que otra más lenta
- α = R: máquinas paralelas no relacionadas. El tiempo en que se procesa un trabajo depende de la máquina en que se procese.

- Entornos híbridos

Se trata de la unión de unión de máquinas paralelas con algún taller. El taller está compuesto por estados en vez de máquinas y cada uno de estos estados (estación) está formado por una o varias máquinas paralelas.

Su representación podría ser α: P2, R4, 1, esto indicaría que el taller dispone de tres estados, el primero se trataría de 2 máquinas paralelas idénticas, el segundo estaría formado por 4 máquinas paralelas no relacionadas y por último, la tercera formada solo por una máquinas.

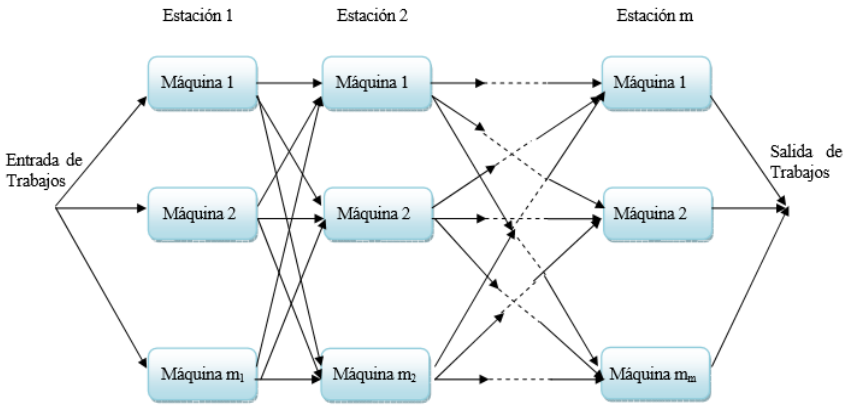

Figura 7.Hybrid Flowshops. (Fuente: Paz Pérez, José Framiñán Torres, 2016)

1.2.2 Trabajos

En este apartado se encuentran características relacionadas con el procesamiento de los diferentes trabajos. Se va a exponer a continuación alguna de ellas:

- β = **Ø:** Englobaría una serie de características como:

  - Trabajos disponibles al principio del horizonte de programación.
  - Los trabajos no se pueden interrumpir.
  - Máquinas siempre disponibles.
  - El buffer entre máquinas se supone infinito.
  - El tiempo de transporte es despreciable
  - Cada máquina puede hacer un trabajo, y un trabajo puede ser realizado sólo en una máquina.

- β = prmu: Taller de flujo en el que la secuencia es la misma para todas las máquinas.
- β = rj: Las fechas de llegada (disponibilidad) de los trabajos son distintas de cero.
- β = dj: Las fechas de entrega de los trabajos son todas iguales (dj=d)
- β = $\overline{d}$j: Las fechas de entrega de los trabajos son deadlines.
- β = prec: precedencia de trabajos.

- β = sij: existe un tiempo de setup en la máquina i antes de procesar el trabajo j.
- β = sijk: existe un tiempo de setup en la máquina i antes de procesar el trabajo k después de procesar el trabajo j.
- β = no-idle: No están permitidos tiempos ociosos de las máquinas entre trabajos.
- β = batch (b): l tiempo de proceso del lote es el mayor de los tiempos de proceso de los trabajos en dicho lote.
- β = pmtn: existen tres tipos:
- β = pmtn-non-resumable: se pierde el trabajo hecho de la tarea interrumpida, y se empieza otra vez cuando se reinicia.
- β = pmtn-semi-resumable: se pierde parte del trabajo realizado, es decir, se reinicia parcialmente.
- β = pmtn-resumable: se reinicia por donde se había dejado tras la interrupción.
- β = nwt: Los trabajos no pueden esperar entre máquinas
- β = buffer=b: Cuando el buffer de una máquina está lleno, la máquina anterior no puede procesar el siguiente trabajo hasta que no quede sitio.

1.2.3 Función objetivo

Las funciones objetivos pueden clasificarse según el coste, tiempo, calidad y flexibilidad. Existen las siguientes medidas a la hora de evaluar la función objetivo de un modelo:

- Cj: tiempo de terminación del trabajo j (completion time). Es el tiempo en que un trabajo termina su procesado en el entorno.
- Fj: tiempo de flujo del trabajo j (flowtime). Es el tiempo en el que un trabajo se encuentra en el entorno.
  Fj = Cj – rj.
- Lj: retraso del trabajo j (lateness). Mide lo que se retrasa o adelanta un trabajo.
  Lj = Cj – dj.

- Tj: tardanza del trabajo j (tardiness). Mide lo que se retrasa un trabajo.
  Tj = máx {0, Lj}.

- Ej: adelanto del trabajo j (earliness). Mide lo que se adelanta un trabajo.
  Ej = máx {0, -Lj}.

- Uj: trabajo tardío (tardy job). Uj vale 1 en el caso en que se retrase y 0 si no.
  Uj =1 si Tj > 0 y Uj = 0 en caso contrario.

Se pueden encontrar funciones objetivos que no están relacionadas con las fechas de entrega:

- Makespan: Cmáx = máx1≤ j ≤n Cj
- Maximum flowtime: máx Fj = máx1≤ j≤ n Fj
- Total completion time: $\sum C_j = \sum_{j=1}^{n} C_j$
- Total flowtime: $\sum F_j = \sum_{j=1}^{n} F_j$

Y por otro lado las relacionadas con fechas de entrega:

- Maximun lateness: máx Lj = máx1≤ j≤ n Lj
- Maximum tardiness: máx Tj = máx1≤ j≤ n Tj
- Maximum earliness: máx Ej = máx1≤ j≤ n Ej
- Total lateness: $\sum L_j = \sum_{j=1}^{n} L_j$
- Total tardiness: $\sum T_j = \sum_{j=1}^{n} T_j$
- Total earliness: $\sum E_j = \sum_{j=1}^{n} E_j$
- Number of tardy jobs: $\sum U_j = \sum_{j=1}^{n} U_j$

1.3 Línea de ensamblado

Una línea de ensamble consiste en la realización de una serie de tareas a un producto mientras este avanza a un ritmo determinado; este proceso puede incluir paradas para que los operarios realicen dichas operaciones.

La primera línea de ensamble según la Asociación cultural Aliusmodi en su proyecto " Venecia y sus lagunas" se atribuye al Arsenal de Venecia en el año 1104, donde la embarcación circulaba cuesta abajo por el canal mientras los diferentes talleres realizaban los trabajos correspondientes. No obstante, hasta la llegada de la Revolución Industrial no se puede definir con precisión las líneas de ensamblado.

La primera línea moderna se le puede asignar a Ransom Olds [Henry Ford, 2014], quien fabricó el primer automóvil en masa, el Oldsmobile Curved Dash. Sin embargo, este logro es eclipsado por Henry Ford, debido a la instalación de cintas transportadoras que permitía llevar el trabajo hasta los trabajadores, y no al revés. Esta idea se basa en una visita por parte de Wiliiam "Pa" Klanm a un matadero de Detroit, donde los cuerpos se movían mediantes cintas y permitía a los trabajadores realizar movimientos simples y automáticos. Esta idea fue transmitida a Peter E. Martin, que pronto se convertiría en jefe de producción de Ford. Esta mejora en el proceso permitió que la fabricación del modelo T pasara a 93 minutos-hombre, en vez de 12.5 horas. En resumen, esta mejora hizo que aumentara la eficiencia un 800%.

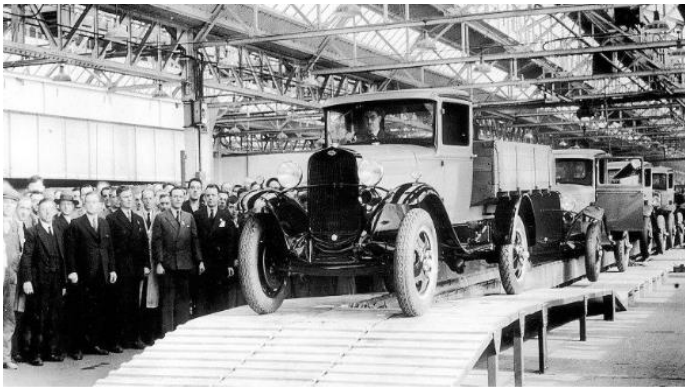

Figura 8.Ford T 1908 (Fuente: Ford, 2018)

Esta innovación provocó un cuello de botella en la nave de pintura debido a que no tenía las prestaciones necesarias para ese cambio de carga. La solución parcial a este problema fue utilizar un tinte japonés de color negro que si se secaba lo suficientemente rápido. Una frase de Henry Ford haciendo burla a este problema fue: "cualquier cliente puede tener el coche del color que quiera siempre y cuando sea negro". No fue hasta 1915 cuando este modelo se empezó a vender en diferentes colores.

Esta innovación permitió la expansión del automóvil en América debido a la reducción de costes durante la producción, disminuyendo por tanto el de venta. Gracias a este avance, el precio del Modelo T bajó de $825 a $575. Esta reducción en el precio es comparable con una reducción de $15,000 a $10,000 en términos de dólares desde el año 2000. Se estima que 250 empresas quebraron debido a que no adoptaron este proceso productivo en 1930.

Las principales ventajas que aportan este modelo de producción son el control de tiempo productivo y la reducción de costos en la producción. Sin embargo, la industrialización de procesos es un procedimiento costoso, por lo que es aconsejable adoptar este modelo siempre que el proceso sea estable en el horizonte temporal.

1.4 Programación de la producción con ensamblado

Una vez descrito las variables que intervienen en la programación de la producción y en qué consiste una línea de ensamblado, se procede a la descripción de la situación a estudiar.

La situación tiene una gran similitud con el modelo seleccionado por las empresas actuales, la cual se basa en dividir la producción en tres etapas diferenciadas. La primera fase se compondría de un conjunto de máquinas paralelas para la producción de las piezas que componen el producto final, estas piezas serán transportadas en serie hasta la última etapa, donde se ejecuta el ensamblado de las piezas producidas.

La situación descrita queda resumida en la posterior imagen:

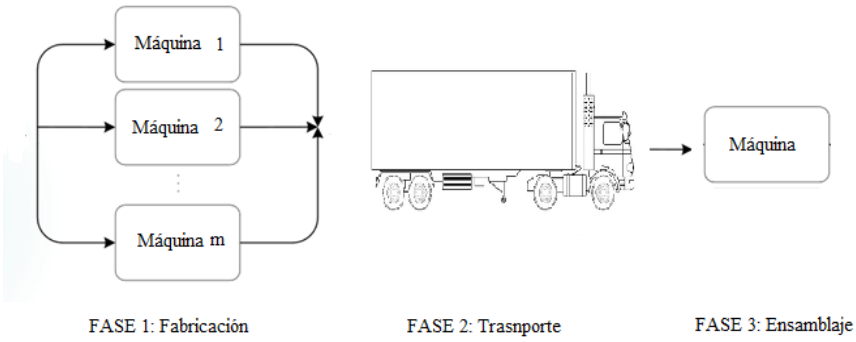

Figura 9.Representación del problema. (Fuente: Elaboración propia)

Este problema se encuentra sometido a las siguientes restricciones:

- Todos los productos están disponibles desde el principio del problema y son conocidos sus tiempos en los diferentes procesos.
- Una vez el trabajo comienza no se puede interrumpir
- Todas las máquinas están operativas a lo largo del horizonte temporal
- El proceso del trabajo comenzará cuando una máquina este libre.
- Todas las máquinas pueden estar trabajando en el mismo momento y cada trabajo tiene que ser procesado en las diferentes fases

El objetivo del problema es disminuir el tiempo que trascurre entre la realización de la primera operación y la finalización de la última operación, es decir, el makespan. La dificultad de hallar la solución viene dada por la dependencia del cálculo de la función objetivo entre las diferentes fases.

Una vez explicado, el problema quedaría expresado según la programación de la producción como:

$$AF3\ (M,\ 1{,}1)\ ||\ C_{MAX}$$

La principal problemática es el cuello de botella que se origina entre la primera etapa y la segunda, esto se debe al conjunto de máquinas paralelas que permite realizar diferentes trabajos al mismo tiempo provocando con el paso del tiempo una congestión en la etapa de transporte. Por consiguiente, el cálculo de la función objetivo viene dada por la siguiente expresión:

$$C_{max} = max_{1\leq k_1\leq n}\left\{max_{1\leq j\leq m}\left\{\sum_{k=1}^{k_1} t_{i_k j}\right\} + \sum_{k=k_1}^{k_2} t_{i_k T} + \sum_{k=k_2}^{n} t_{i_k A}\right\}$$

A continuación, se va a proceder a cómo se resolvería el problema, pero en un escenario pequeño para poder explicar cómo se calcula la función objetivo y las dificultades que se van a presentar en los escenarios que se van a estudiar. Para ello, el escenario está compuesto por dos máquinas paralelas en la primera etapa, en las dos etapas siguientes se mantiene la existencia de una única máquina. Por último, la cantidad de trabajos serán de cinco y su carga de tiempo en cada una de las etapas es la siguiente:

Nº de trabajo	Fase 1: Máquinas paralelas	Fase 2: Transporte	Fase 3: Ensamblado
1	7	3	3
2	3	2	2
3	6	1	6
4	8	4	6
5	3	2	2

Tabla 1. Tiempos del escenario

Como se ha mencionado anteriormente, se trata de un sistema de flujo uniforme (*Flowshop*), por lo que el orden de entrada en la primera etapa definirá la secuencia en el resto de etapas. Es decir, si la secuencia que se utilizara fuera [1, 2, 3, 4,5], el resultado sería el siguiente:

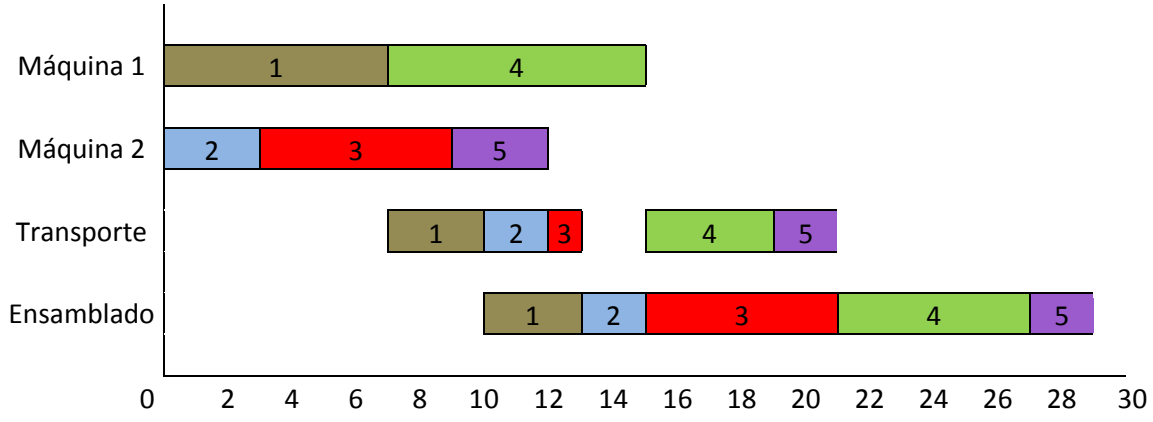

Figura 10. Diagrama de Gantt

Con este orden se obtiene un *makespan* de 29 u.t, para poder calcular el óptimo en este pequeño escenario sería necesario calcular todas las posibles secuencias que correspondería a 5! (120 combinaciones). Para pocos trabajos, calcular todas las posibles soluciones no costaría una gran cantidad de tiempo, pero si solo aumentamos la cantidad de trabajos a 8, se obtendría 40320 posibles combinaciones.

En resumen, en este capítulo se han explicado las principales características de la programación de la producción y un breve repaso a la historia de la línea de ensamblado, finalizando con la explicación en detalle del problema a resolver, estancado la idea de la complejidad que conlleva un problema que parece muy simple, para ello en el siguiente capítulo se definirán qué son las metaheurísticas y por qué se utilizan para resolver la mayoría de estos problemas, finalizando con las descripción de las dos que se van a utilizar para resolver el problema, IDCOA y GRASP.

CAPÍTULO II: DESCRIPCIÓN DE LA METODOLOGÍA

La resolución de la mayoría de problemas de la programación de la producción se realiza mediante el uso de metaheurísticas debido a que la mayoría de ellos están categorizados como NP-Hard. A continuación, se realiza una breve introducción de estos métodos de resolución para continuar con la descripción detallada de las dos heurísticas que van a ser utilizadas para resolver el problema. Finalmente, se describe el software utilizado para programar las heurísticas que facilitará los resultados para su posterior estudio.

2.1 Metaheurísticas

Las metaheurísticas son metodologías aproximadas que aportan una buena solución (no necesariamente la óptima) a problemas con gran complejidad de forma fácil y sencilla. La mayoría de ellas son procedimientos muy simples, basados principalmente en el sentido común. Se utilizan principalmente para problemas que tienen un método exacto pero consume mucho tiempo para hallar el óptimo y es necesario una respuesta rápida, cuando no existe ningún método exacto de resolución o como paso intermedio para poder obtener una solución inicial con el fin de aplicar otro algoritmo para alcanzar una solución final.

Algunos autores (Silver et al., 1980) proponen la siguiente clasificación de métodos de resolución mediante heurísticos:

- Métodos constructivos, que se caracterizan por construir una solución definiendo diferentes partes de ella en sucesivos pasos.
- Métodos de descomposición, dividen el problema en varios más pequeños y la solución se obtiene a partir de la solución de cada uno de estos.
- Métodos de reducción, tratan de identificar alguna característica de la solución que permita simplificar el tratamiento del problema.
- Métodos de manipulación del modelo, obtienen una solución del problema original a partir de otra de otro problema simplificado.
- Métodos de búsqueda por entornos, en las que se parte de una solución inicial a la que se realizan modificaciones en sucesivas iteraciones para obtener una solución final. En cada iteración existe un conjunto de soluciones vecinas candidatas a ser nueva solución en el proceso. En este grupo se encuadran las técnicas metaheurísticas.

Una vez comentada la clasificación de las diferentes heurísticas, las utilizadas en este documento se encuentran en el último grupo, el motivo es que tanto IDCOA como GRASP parten de soluciones iniciales que van a partir del azar, cuya única condición es que dicha solución sea factible. Y además, estas posibles soluciones que aporten las heurísticas se moverán entre las vecinas con la ayuda de otra heurística llamada LS (Local Search o Búsqueda Local).

2.2 IDCOA

Esta metaheurística desarrollada en el artículo (G.M. Komaki et al. ,2017) está basada en otra llamada COA (Cuckoo Optimization Algorithm), gracias al estudio exhaustivo que se realiza en el artículo realizaron varias modificaciones tanto en los procesos a seguir como en valores que se escogen, provocando mejores resultados. Están basados en el ciclo de vida de los cucos, que gracias a una de las principales características de ellos, surge la idea para explorar los diferentes conjuntos de soluciones vecinas candidatas.

La característica es que estos pájaros no hacen nidos, por consiguiente ponen sus huevos en los nidos de las otras aves, la gran probabilidad que tiene de sobrevivir es gracias a su gran habilidad para adaptar sus huevos al color y tamaño de los del nido, y por otra parte los huevos eclosionan de los cucos eclosionan antes provocando que sus polluelos se coman o tiren los huevos del nido. Luego cuando son adultas migran hacía mejores hábitats, que llevándolo al problema sería el conjunto de mejores soluciones. A continuación, se explicará con detalle todos los pasos de la metaheurística.

El proceso comienza con la creación del hábitat del cuco, cada hábitat representa una posible solución, se asemeja a la idea de los algoritmos genéticos, en los que se desarrollan varias soluciones posibles y a partir de la mejor se va desarrollando otra mejor y así sucesivamente. En la primera iteración, se empiezan con una cantidad x de cucos, y en las siguientes iteraciones, esta cantidad dependerá de la siguiente fórmula:

$$N_{egg} = w_i \left(\frac{f_i}{f_{best}}\right)$$

En dicha fórmula:

- w_i: corresponde a número al azar entre un límite inferior y superior que ira variando para cada cuco.
- f_i: valor de la función objetivo de cada cuco.
- f_{best}: mejor valor de función objetivo de todos los cucos en cada iteración.

Una vez calculado el número de polluelos que pondrá cada cuco se procede al cálculo de una característica específica de cada cuco que será utilizado en los pasos posteriores, el DELR (Dynamic Egg Laying Radius), este valor representará la distancia que cada cuco viajará para poder establecerse en un hábitat mejor, es decir, el número de cambios máximos en su secuencia para poder mejorar la función objetivo.

La fórmula es la siguiente:

$$DELR_i = \propto_{itr} * n * \frac{N_{egg}}{TN_{egg}}$$

Donde:

- n: número total de trabajos.
- Negg: número de huevos que pone cada cuco.
- TNegg: Total de huevos puesto en dicha iteración.

α_{itr}, viene dado por otra fórmula:

$$\propto_{itr} = \propto_{max} - \ N_{itr} \ \frac{(\propto_{max} - \propto_{min})}{TN_{itr}}$$

En la cual:

- TN_{itr}: es el número total de iteraciones del algoritmo, dicho valor será 400.
- N_{ite}: iteración i en el que se encuentre el algoritmo
- α_{max} y α_{min}: son dos variables que empiezan con un valor determinado pero que con el transcurso de las iteraciones puede cambiar si el valor de α_{itr} es mayor o menor que α_{max} y α_{min} respectivamente.

Como se comentó, los cucos ponen los huevos en los nidos de otros pájaros existiendo la posibilidad de que las madres de estos nidos detecten los huevos o los polluelos de los cucos siendo eliminados o que no se desarrollen de la forma adecuada. Para llevar la realidad al algoritmo existe una probabilidad del 10% de que los huevos no sobrevivan.

Esta metaheurística utiliza un método de agrupación usando una medida de similitud entre los posibles candidatos basado en la idea de aproximación de Jaccard. En esta parte se busca hacer una matriz con los coeficientes de similitud entre los candidatos de cada iteración con el fin de tener una mayor diversidad en la población, para ello se utiliza el CSC (Cuckoo Similarity Coefficient):

$$CSC\ (a,b) = \frac{S_b^a + 2 * z_b^a}{n}$$

Cuyas variables son:

- n: número de trabajos que existen.
- S_b^a: equivale al número de pares que existen entre dos candidatos, es decir, se coge la secuencia por pares y se compara si ese par existe en la secuencia del otro cuco al que se le está realizando la comparación.
- Z_b^a: en este caso se corresponde a que si los trabajos de cada secuencia coinciden en el orden.

Para facilitar la compresión de este método de agrupación, se explicará cada uno de ellos con un ejemplo numérico:

SECUENCIA CUCKOO 1

1	2	8	7	4	6	5	3

SECUENCIA CUCKOO 2

4	8	5	7	1	2	6	3

Si observamos la secuencia CUCKOO 1, se tendría que coger el primer par que corresponde a la pareja (1,2) y comparar con cada una de las parejas de la secuencia CUCKOO 2. Las parejas de CUCKOO 2 son: (4,8), (8,5), (5,7), (7,1)… Si observamos ambas secuencias se repiten las parejas (1,2). Por lo que el valor de S_2^1 correspondería a 1 para este par de candidatos.

Por otro lado, ahora vamos a hallar Z_2^1, como se ha dicho anteriormente este solo trata de ver, por ejemplo, si en la posición 6 de cada secuencia se corresponde a la realización del mismo trabajo. En el ejemplo se observa que en la posición 8 de cada secuencia corresponde al trabajo 3, igual pasa con el trabajo 7 en la posición 4. Entonces, nos queda el valor de 2 para Z_2^1 .

Si volvemos a la fórmula:

$$CSC(1,2) = \frac{S_2^1 + 2 * Z_2^1}{n}$$

Con los valores hallados, el resultado del CSC (1,2) sería de 0´75. Como es de imaginar el resultado para CSC (2,1) será también de 0,75. Hay que tener en consideración que cuando los valores son decimales, se ha optado a un redondeo de que si el decimal es mayor o igual a 0,5, se toma el número entero superior. Con el ejemplo anterior de 0,75, en la matriz aparecería con el valor de 1.

En cada iteración, el número de filas y columnas serán igual al número de cucos de dicha iteración, a cada matriz cuadrada hay que extraer el valor mínimo. Es decir, si tenemos la siguiente matriz:

3	1	1	2	2
1	3	1	2	1
1	1	3	1	1
2	2	1	3	1
2	1	1	1	3

Tabla 1.Ejemplo numérico de CSC (a, b).

El valor mínimo sería 1, dicho valor correspondería al valor de una nueva variable, k, una vez hallado el mínimo de la matriz hay que retomar el valor del DELR.

Como se explicó anteriormente, DELR es un valor característico de cada cuco, que se va a dar uso junto a k, para realizar una serie de cambios a la secuencia. El número de cambios en las secuencias

dependerá de:

$$k \leq DELR_i$$

Con esto se quiere aplicar una seria intercambios en la secuencia para intentar comenzar la búsqueda local con una ``mejor´´ solución. Los cambios serán igual a k siempre que el número sea menor que el propio DELR de dicho cuco. Si k fuese mayor que DELR, se realizaría tantos cambios como valga DELR.

Es decir, nos encontramos con una situación donde k tiene un valor de 2 y el DELR del cuco vale 1, entonces se realizará solo un intercambio en la secuencia de dicho cuco. Dichos intercambios en la secuencia se realizarán al azar. A continuación se explicará con un ejemplo. Se tiene la siguiente secuencia:

1	2	8	7	4	6	5	3

Suponemos que este cuco tiene un DELR=2 y el valor de k corresponde a 1, por lo que se realizaría solo un intercambio. Se toman dos valores al azar entre la cantidad de trabajos que haya, en este caso 8. Estos valores pueden ser 3 y 5, entonces la secuencia ahora quedaría:

1	2	4	7	8	6	5	3

Estos intercambios se realizan con el fin de tener una mayor variedad de posibles soluciones y no así no quedarse estancado con las mismas.

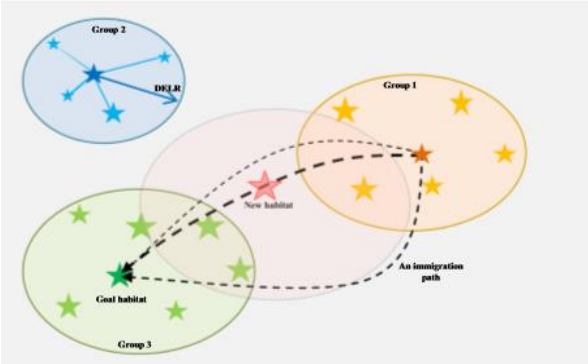

Figura 11.Migración de los cucos.(Fuente: G.M. Komaki et al. ,2017)

Una vez realizado los intercambios, se escogen un conjunto de soluciones, que se van a conocer con el nombre de soluciones elites, estas se corresponde a las mejores secuencias de cada iteración. El número de soluciones élites dependerá del número de cucos de la iteración. Durante este estudio el número de soluciones de élite variarán con estos porcentajes 3,5 y 7%, entonces si en una iteración se obtiene 60 cucos y se está realizando el estudio con un 5% para escoger las soluciones élites, el números de estos será igual a 3. A estas soluciones se le aplicará la heurística Local search o búsqueda local, la cual será explicada en el siguiente apartado.

2.2.1 Local Search

El término local tiene un uso muy común en las metaheurísticas de búsqueda tanto en la teoría como en la práctica. Se asocia al uso de estructuras de entorno, reflejando el concepto de proximidad o vecindad entre las soluciones alternativas del problema.

Todas las soluciones incluidas en el entorno de la solución actual, que viene delimitado por un operador de generación de soluciones, se denominan soluciones vecinas.

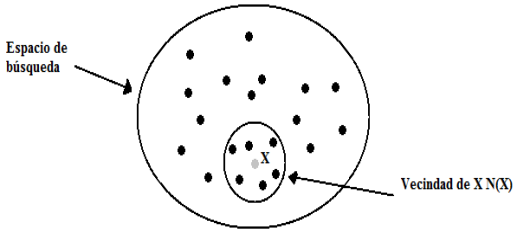

Figura 12.Búsqueda local. (Fuente: Elaboración propia)

Una búsqueda local es un proceso que, dada la solución actual en la que se encuentra el recorrido, selecciona iterativamente una solución de su entorno para continuar la búsqueda. Su *modus operandi* se resume en variar cada posición de la secuencia y escoger aquel cambio que mejore más la función objetivo.

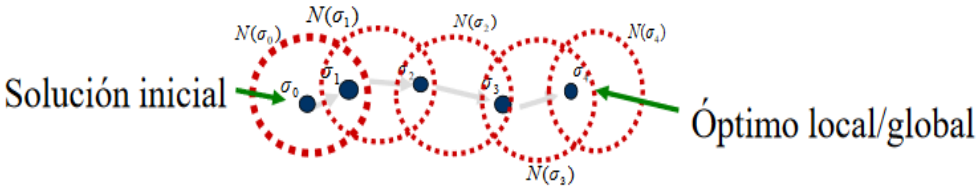

Figura 13.Búsqueda local.(Fuente: J.M.Benitez, 2012)

Por ejemplo, se obtiene la secuencia (5, 2, 1, 4, 3, 6) con una función objetivo de 120 y se le aplica dicha heurística, el primer paso corresponde a intercambiar la posición 1 correspondiente al trabajo 5 con el resto de posiciones. A continuación, el cambio de la posición 1 con las 3 mejora la función objetivo a 118.

1	2	5	4	3	6

Pero por otro lado, el cambio de la posición 1 por la 5 la mejora, obteniendo un valor de 108.

3	2	1	4	5	6

Una vez realizado todos los cambios de la posición 1 con el resto, se ha obtenido una mejor secuencia (3, 2, 1, 4, 5, 6). Para analizar la posición 2, se parte con la secuencia obtenida al haber estudiado la posición 1 y así hasta analizar todas las posiciones.

2.2.2 Resumen IDCOA

A continuación, se va a explicar con un resumen paso a paso de la heurística IDCOA:

1. Inicialización de los parámetros iniciales referentes a la Tabla 2. (Número inicial de cucos, Número máximo de cucos (NC_{max}), Porcentaje de soluciones élites (Ne), Límite de α mínimo (α_{min}), Límite de α máximo (α_{max})).

2. Generación de una población inicial que desembocará en los nuevos huevos. (N_{eggi})

$$N_{eggi} = w_i \left(\frac{f_i}{f_{best}}\right)$$

3. Cálculo de α y DELR de cada cuco. Es necesario actualizar α_{max} o α_{min}, si el valor de α_{itr} durante el transcurso de las iteraciones su valor es menor o mayor al rango establecido entre ambas α.

$$\propto_{itr} = \propto_{max} - \; N_{itr} \; \frac{(\propto_{max} - \propto_{min})}{TN_{itr}} \quad DELR_i = \propto_{itr} * n * \frac{N_{eggi}}{TN_{egg}}$$

4. Eliminación de algunos de los huevos por parte de la madre del nido o por falta de desarrollo (esta eliminación se realiza a con un p%).

5. Cálculo de la matriz CSC (a, b) con la finalidad de obtener el valor de la variable k de cada iteración, correspondiente al valor mínimo obtenido en la matriz.

$$CSC\ (a, b) = \frac{S_b^a + 2 * z_b^a}{n}$$

6. Realizar los cambios a las secuencias siempre que dicho cambio mejore la función objetivo, el número de cambios realizado a cada cuco viene determinado por la restricción $k \leq DELR_i$

7. Formación de la lista de candidatos entre todos los cucos originados durante la iteración j. Dicha lista se limita a $NC_{máx}$, dicho valor se establece en el punto 1. Todos aquellos cucos que queden fuera de la lista, serán eliminados de la iteración.

8. Escoger el porcentaje Ne de la lista de candidatos formada en el punto 7 para formar el grupo de soluciones élites.

9. Aplicar Local search al conjunto de soluciones élites.

10. Fase de actualización, renovar la secuencia final si se obtiene en el punto anterior una función objetivo mejor.

11. El proceso finaliza cuando se llega al límite de iteraciones (TN_{itr}) o si la función objetivo no varía durante un número de iteraciones (NI_{itr}). Si no es así, volver a paso 2.

2.3 GRASP

La metodología GRASP fue desarrollada por Feo y Resende al final de la década de los ochenta con el objetivo inicial de resolver problemas de cubrimientos de conjuntos. La palabra GRASP proviene de las siglas de "Greedy Randomized Adaptative Search Procedure", que en castellano se traduciría como Procedimientos de Búsqueda basados en funciones "Greedy" Aleatorizadas Adaptativas. Dicha metaheurística [Álvaro García Sánchez, 2006] se basa en un procedimiento de búsqueda codiciosa, aleatoria y adaptativo que garantiza buenas soluciones a una gran variedad de problemas de optimización, aunque no necesariamente se obtiene la óptima.

GRASP se caracteriza principalmente por tener ser un proceso robusto, pero a su vez muy sencillo debido a que es difícil encontrar ejemplos patológicos en donde el método funcione arbitrariamente mal. Dicha heurística esta es un procedimiento formado por una fase de construcción, una de mejora y otra de actualización.

La fase de construcción consiste en crear una lista de candidatos, esta elección se realiza mediante la creación de soluciones factibles de forma aleatoria. Posteriormente, de dicho listado se escoge uno o varios candidatos que serán utilizados en la siguiente fase.

Esta solución escogida al azar pasa a la fase de mejora, que consiste únicamente en aplicarle la búsqueda local (2.2.1) a la secuencia, si la secuencia obtenida mejora la solución inicial se actualiza el valor de la secuencia, fase de actualización. La heurística finaliza cuando se completan el número de iteraciones máximas establecidas al principio de la experimentación.

Por lo que GRASP se podría resumir en los siguientes pasos:

1. Inicialización de la fase constructiva con la búsqueda aleatoria de soluciones factibles.
2. Se escoge un elemento de la fase constructiva para realizar la fase de mejora basada en el proceso de búsqueda local.
3. Fase de actualización, si la solución obtenida en el punto 2 mejora la solución final, esta se renueva.
4. El proceso finaliza cuando se llega al límite de iteraciones X o la función objetivo no varía durante un número de iteraciones (NI_{itr}). Si no es así, volver a paso 1.

2.4 Code::Blocks

Como se ha mencionado en la introducción de este capítulo, el programa informático que se va a utilizar para modelar las heurísticas y los escenarios que se van a comparar. El lenguaje utilizado por este programa es C o C++.

Code::Blocks es una plataforma muy dinámica y potente, no solo por la facilidad con que puede incluirse nueva funcionalidad, sino por la capacidad de poder usarla para construir otras herramientas de desarrollo tan solo añadiendo complementos.

Gracias a la característica mencionada se ha aprovechado introduciendo la librería Schedule facilitada por mi tutor, José Manuel Framiñán Torres, y el profesor Víctor Fernández-Viagas Escudero. Esta librería contiene diversas funciones que permite realizar diferentes operaciones con una mayor rapidez. Permite un gran ahorro de tiempo ya que permite de creación de vectores o matrices y realizarles diferentes operaciones como copiar una fila entera de una matriz en un vector, esto permite un código más reducido y muchas facilidades para poder realizarlo. Las librerías son

schedule_lib.a y schedule_lib.h.

Su gran potencia de procesado de operaciones y la facilidad aportada por la librería nos ha permitido codificar las heurísticas a lenguaje C y obtendremos los diferentes resultados de los escenarios que nos permitirá saber en qué situaciones puede funcionar mejor cada heurística. Los códigos de ambas heurísticas se encuentran disponibles en la versión electrónica del documento.

La principal conclusión de este capítulo tras la explicación detallada de ambas heurísticas es que ambas utilizan el método de búsqueda local para encontrar una solución, por otro lado la principal diferencia radica en el los cambios que hace IDCOA para intentar mejorar la solución aleatoria inicial, sin embargo GRASP parte de una totalmente aleatoria. En el siguiente capítulo, se mostrarán los resultados obtenidos con los diferentes escenarios de IDCOA para demostrar los resultados obtenidos en el artículo de G.M. Komaki (2017) y por otro lado, los de GRASP, para finalmente comparar ambas metaheurísticas.

CAPÍTULO III: APLICACIÓN DE LA METODOLOGÍA

Tras la explicación teórica y detallada de ambas metaheurísticas y su correspondiente programación en Code:Blocks, el contenido de este capítulo engloba los resultados en las diferentes casuísticas en cada una. No obstante, primero se comenzará con la elección de las variables más prometedoras para IDCOA, finalizando con la comparación de resultados.

3.1 IDCOA

En el artículo de G.M. Komaki (2017) se muestra un rango de valores para cada una de las variables que intervienen en el desarrollo de la metaheurística. La finalidad de este apartado es encontrar y verificar qué valores aportan mejores resultados. A continuación, se muestran las cifras para cada variable:

Número inicial de cucos	2, 4, 5 y 6
Número máximo de cucos (NC_{max})	20, 40 y 60
Porcentaje de soluciones élites (N_e)	3%, 5% y 7%
Límite de α mínimo (α_{min})	0.3, 0.4 y 0.5
Límite de α máximo (α_{max})	1.7, 2 y 2.2

Tabla 2.Rangos IDCOA.

Con el fin de no hacer un estudio total de todas las posibilidades que engloban, el estudio se va a resumir en tres su métodos, definidos cada uno de ellos por el valor mínimo ($IDCOA_{min}$), medio (IDCOA*) y máximo ($IDCOA_{max}$) de cada variable. Como resultado, obtenemos tipos de IDCOA cuyas variables se resumen en:

	$IDCOA_{min}$	IDCOA*	$IDCOA_{max}$
Número inicial de cucos	2	5	6
Número máximo de cucos (NC_{max})	20	40	60
Porcentaje de soluciones élites (N_e)	3%	5%	7%
Límite de α mínimo (α_{min})	0.3	0.4	0.5
Límite de α máximo (α_{max})	1.7	2	2.2

Tabla 3.Rangos seleccionados IDCOA.

El proceso puede finalizar de dos formas posibles, la primera es debido a que el número máximo posible de iteraciones está fijado en 400, y la segunda corresponde a la limitación en la que el resultado obtenido en cada iteración no se puede repetir x veces de forma consecutiva durante el transcurso de la heurística. Para cada uno de los modelos de IDCOA definidos este límite se establece en 3, 5 y 7 iteraciones.

El tiempo que conlleva cada trabajo de forma teórica en cada una de las fases se establecen según los siguientes rangos:

	Fase 1: Máquinas paralelas	Fase 2: Transporte	Fase 3: Ensamblado
Tiempos	U[0,100]	U[0,10]	U[0,100]

Tabla 4.Rango de tiempos para cada fase.

También se modificará el número de trabajos y máquinas, permitiendo una visualización en los resultados cuando existen cuellos de botella, frecuentes en la producción. Dicho reparto será el siguiente:

Nº de trabajos	20, 40, 60, 80
Nº de máquinas	2, 4, 6, 8

Tabla 4.Rango de valores de máquinas y trabajos.

3.1.1 Resultados IDCOA

Para la comparación de las variaciones de IDCOA, se llevará a cabo calculando el ARPD de la función objetivo y, el AGT, correspondiente al tiempo de computación. En primer lugar se comparará por número de iteraciones sin mejorar para luego terminar con la comparación de las seleccionadas. Para cada combinación del número de trabajos n, número de máquinas m, se han originado 10 instancias.

$$\text{ARPD} = \frac{(Resultado\ iteración\ i - mejor\ solución\ del\ escenario)\ x\ 100}{mejor\ solución\ del\ escenario}$$

$$\text{AGT} = \frac{(Tiempo\ de\ computación\ i - mejor\ tiempo\ de\ computación\ )\ x\ 100}{mejor\ tiempo\ de\ computación}$$

Los resultados de cada escenario se muestran por medio de la media (AVG) y su desviación (STD).

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \qquad \sigma = \sqrt{\frac{\sum_{i=1}^{n} (x - \hat{x})^2}{N}}$$

A continuación, se procede a mostrar los resultados para cada uno de los tipos de IDCOA definidos anteriormente en cada uno de los escenarios propuestos. En las Tablas 8, 10 y 12 se presentan la media y la desviación obtenida del ARPD con el criterio de parada de 3, 5 y 7 iteraciones. En las Tablas 9, 10 y 11 se muestra la misma información, pero en este caso del ATG:

		$IDCOA_{min}$		IDCOA*		$IDCOA_{máx}$	
n	m	AVG	STD	AVG	STD	AVG	STD
2	20	19.545	8.850	9.701	7.036	17.589	11.338
2	40	6.435	5.125	7.752	4.382	10.853	7.159
2	60	3.695	2.744	10.755	6.841	11.221	7.167
2	80	5.726	3.661	12.808	6.670	9.312	5.629
4	20	24.899	15.214	13.915	9.824	14.629	12.591
4	40	13.593	7.463	10.907	7.738	9.339	6.309
4	60	11.307	5.951	7.817	5.121	7.903	5.868
4	80	6.671	4.807	9.386	5.754	6.455	3.767
6	20	22.916	17.285	17.731	7.109	8.665	10.016
6	40	20.675	8.530	13.959	9.522	7.814	8.302
6	60	9.068	4.879	9.508	6.427	9.068	4.879
6	80	8.574	6.419	5.070	3.118	8.560	4.806
8	20	19.778	11.028	13.998	8.312	19.528	12.504
8	40	12.329	8.065	11.801	7.696	12.983	7.582
8	60	7.605	3.868	6.654	5.566	10.238	5.917
8	80	14.347	8.008	7.842	3.870	8.926	5.165
		12.948	7.619	10.600	6.562	10.818	7.438

Tabla 5.Resultados IDCOA 3 iteraciones sin mejora (ARPD).

		IDCOAmin		IDCOA*		IDCOAmáx	
n	m	AVG	STD	AVG	STD	AVG	STD
2	20	95.429	52.011	135.833	87.256	188.889	186.542
2	40	67.380	50.971	72.099	59.661	484.962	463.008
2	60	35.446	30.339	89.790	46.164	145.505	108.081
2	80	58.541	41.099	233.590	259.955	317.801	340.046
4	20	81.020	50.368	117.857	101.960	216.022	166.595
4	40	90.784	48.836	144.296	102.830	117.070	182.312
4	60	81.168	63.464	7.903	5.868	444.950	367.854
4	80	44.024	35.540	122.297	69.595	92.572	115.007
6	20	94.583	57.039	153.833	102.195	224.844	250.020
6	40	48.382	38.286	228.436	106.712	267.348	338.283
6	60	75.683	54.984	177.963	98.711	279.136	258.933
6	80	20.566	14.023	12.613	8.978	448.800	415.526
8	20	39.730	43.338	43.839	43.750	435.789	300.701
8	40	179.182	130.474	80.579	64.503	390.047	345.914
8	60	41.145	42.421	60.084	43.184	218.491	258.266
8	80	20.884	20.197	119.262	106.093	349.439	259.938
		67.122	48.337	112.517	81.713	288.854	272.314

Tabla 6.Resultados IDCOA 3 iteraciones sin mejora (ATG).

		$IDCOA_{min}$		IDCOA*		$IDCOA_{máx}$	
n	m	AVG	STD	AVG	STD	AVG	STD
2	20	9.551	5.652	9.984	7.620	13.232	6.292
2	40	9.253	4.987	12.362	7.885	15.506	7.946
2	60	7.339	4.456	11.367	6.545	12.580	8.042
2	80	6.608	5.977	4.552	3.571	7.886	5.761
4	20	14.141	10.039	4.490	11.806	20.531	12.003
4	40	10.070	5.778	11.173	8.705	7.280	5.945
4	60	5.955	4.252	8.624	6.866	5.112	4.339
4	80	6.664	3.634	7.769	4.276	7.874	4.387
6	20	9.842	5.296	7.945	6.343	19.009	10.221
6	40	8.049	7.033	8.077	5.269	17.170	10.030
6	60	15.991	10.446	4.445	3.175	8.955	4.810
6	80	7.043	4.820	2.914	2.408	14.165	9.104
8	20	19.583	9.977	12.049	7.257	10.645	9.403
8	40	13.968	9.148	17.106	7.721	13.158	7.260
8	60	7.648	7.939	5.401	5.234	13.683	7.820
8	80	9.432	7.103	2.274	1.511	10.892	7.705
		10.071	6.659	8.158	6.012	12.355	7.567

Tabla 7.Resultados IDCOA 5 iteraciones sin mejora (ARPD).

		$IDCOA_{min}$		IDCOA*		$IDCOA_{máx}$	
n	m	AVG	STD	AVG	STD	AVG	STD
2	20	83.407	190.826	38.521	53.220	143.281	144.861
2	40	32.809	30.539	66.775	59.342	151.022	134.481
2	60	684.926	291.442	22.137	31.163	266.935	167.218
2	80	50.344	29.343	25.212	14.869	271.197	182.142
4	20	50.833	31.761	33.629	18.093	280.348	197.437
4	40	26.536	26.240	108.679	68.061	40.758	39.310
4	60	54.773	32.269	40.265	45.089	247.573	165.722
4	80	48.622	35.953	42.583	27.331	100.602	107.793
6	20	54.156	34.098	46.065	35.912	117.531	64.526
6	40	35.952	25.968	55.770	41.238	167.372	118.922
6	60	130.693	22.592	41.500	22.974	197.127	130.301
6	80	65.373	39.487	18.344	13.273	181.324	185.978
8	20	856.154	430.801	58.731	51.540	304.671	145.250
8	40	74.646	44.392	55.855	26.581	271.104	138.005
8	60	37.498	27.518	32.917	28.674	107.913	61.011
8	80	53.139	27.241	10.892	7.705	178.922	181.046
		146.241	82.529	43.617	34.067	189.230	135.250

Tabla 8.Resultados IDCOA 5 iteraciones sin mejora (ATG).

		$IDCOA_{min}$		IDCOA*		$IDCOA_{máx}$	
n	m	AVG	STD	AVG	STD	AVG	STD
2	20	22.984	11.624	12.731	8.730	15.979	11.059
2	40	18.438	8.435	13.783	10.444	10.884	9.031
2	60	11.416	7.077	10.555	6.140	10.545	5.893
2	80	9.229	5.525	3.073	2.229	4.865	3.610
4	20	14.867	9.577	22.576	9.105	9.189	4.972
4	40	4.279	4.091	10.188	5.523	12.236	8.022
4	60	9.913	7.005	5.354	4.201	11.189	8.515
4	80	6.485	4.188	4.544	3.640	10.029	7.343
6	20	44.950	22.587	12.298	7.546	18.250	16.156
6	40	3.994	4.505	9.696	7.099	8.578	7.379
6	60	13.816	7.662	3.658	2.568	16.775	7.715
6	80	10.590	5.089	9.059	4.413	7.067	5.049
8	20	18.262	8.923	17.995	8.329	33.415	21.465
8	40	10.657	6.413	8.639	6.953	15.926	9.536
8	60	8.185	6.224	10.442	5.431	18.766	10.379
8	80	4.770	3.235	8.757	5.819	16.037	7.293
		13.302	7.635	10.209	6.136	13.733	8.964

Tabla 9.Resultados IDCOA 7 iteraciones sin mejora (ARPD).

		$IDCOA_{min}$		IDCOA*		$IDCOA_{máx}$	
n	m	AVG	STD	AVG	STD	AVG	STD
2	20	38.953	21.386	65.177	37.131	170.843	83.779
2	40	34.755	25.596	47.912	35.320	75.066	59.432
2	60	23.103	29.134	88.078	54.673	149.100	85.476
2	80	24.599	22.210	73.460	41.479	180.716	165.316
4	20	30.926	23.820	43.303	29.205	56.450	46.487
4	40	59.718	32.865	75.849	39.606	66.968	58.067
4	60	59.273	40.502	66.176	42.556	280.474	162.844
4	80	54.824	29.189	62.405	39.049	125.538	50.726
6	20	77.471	42.275	63.188	34.654	92.557	52.798
6	40	292.456	138.851	35.807	23.098	186.904	93.992
6	60	28.644	25.222	12.566	8.087	48.407	27.406
6	80	5.552	20.978	20.901	17.509	237.081	166.588
8	20	33.154	20.801	99.728	46.395	43.770	46.748
8	40	80.879	41.259	35.329	22.045	143.737	79.129
8	60	33.269	20.671	76.272	44.735	64.008	44.566
8	80	42.133	22.828	71.633	54.336	150.843	84.453
		57.482	34.849	58.611	35.617	129.529	81.738

Tabla 10.Resultados IDCOA 7 iteraciones sin mejora (ATG).

Con los valores obtenidos en las Tablas 8, 10 y 12 referentes al estudio del ARPD, se puede observar un patrón decreciente que se repite en gran parte de los escenarios a medida que crecen el número de trabajos para los modelos $IDCOA_{min}$ e $IDCOA_{max}$, sin embargo IDCOA no presenta ningún patrón y obtiene los mejores valores en las tres tablas, si comparamos la media.

Para las Tablas 9,11 y 13 relacionadas con el ATG, ningún modelo presenta algún patrón, esto se puede deber principalmente a la aleatoriedad de las soluciones iniciales que provoca que los tiempos de computación varíen más.

3.1.2 Comparación resultados IDCOA

A continuación, se muestra un resumen de los valores obtenidos en el estudio de IDCOA en los diferentes escenarios, mostrando solo la media obtenida según el número máximo de iteraciones en la que la el óptimo se repite (Tablas 8, 10 y 12):

nº de iteraciones	IDCOA	AVG	STD
3	$IDCOA_{min}$	10.600	6.561
5	IDCOA*	8.158	6.011
7	$IDCOA_{min}$	10.209	6.135

Tabla 11. Resultados según función objetivo (ARPD).

Y en relación con el tiempo de ejecución referentes a las Tablas 9,11 y 13:

nº de iteraciones	IDCOA	AVG	STD
3	$IDCOA_{min}$	67.121	48.336
5	IDCOA*	43.617	34.066
7	$IDCOA_{min}$	57.481	34.849

Tabla 12 .Resultados según tiempos de computación (ATG).

.

Apreciando los valores aportados por las Tablas 14 y 15, queda demostrado que los valores establecidos para IDCOA* aportan mejores resultados mucho más estables, corroborando la afirmación establecida por Komachi en su artículo. En la Figura 13, se representan la media obtenida según la función objetivo agrupado por el número de máquinas 2, 4, 6 y 8, y a su vez, por el límite de iteraciones sin mejora 3,5 y 7.

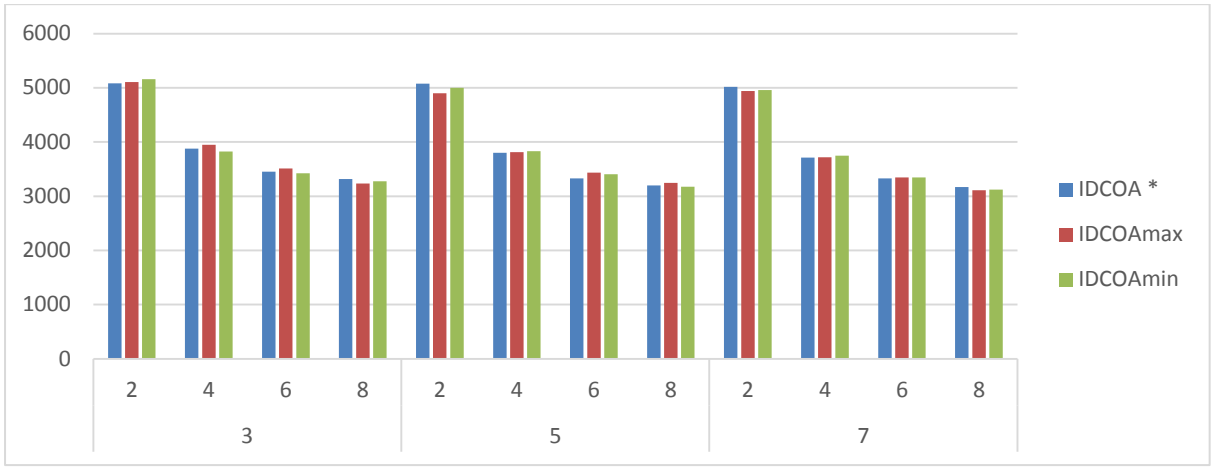

Tabla 13.Media tiempos FO por número de máquinas IDCOA.

Si no se tiene en cuenta la desviación de los resultados, la media de la función objetivo de cada una de las versiones de IDCOA que son muy similares. Por eso, la importancia de no medir solo el resultado en cuanto al valor propio de la función objetivo, ya que una mayor estabilidad de los resultados supone que se adapta mejor a la resolución del problema y permite una mayor confianza de las soluciones obtenidas.

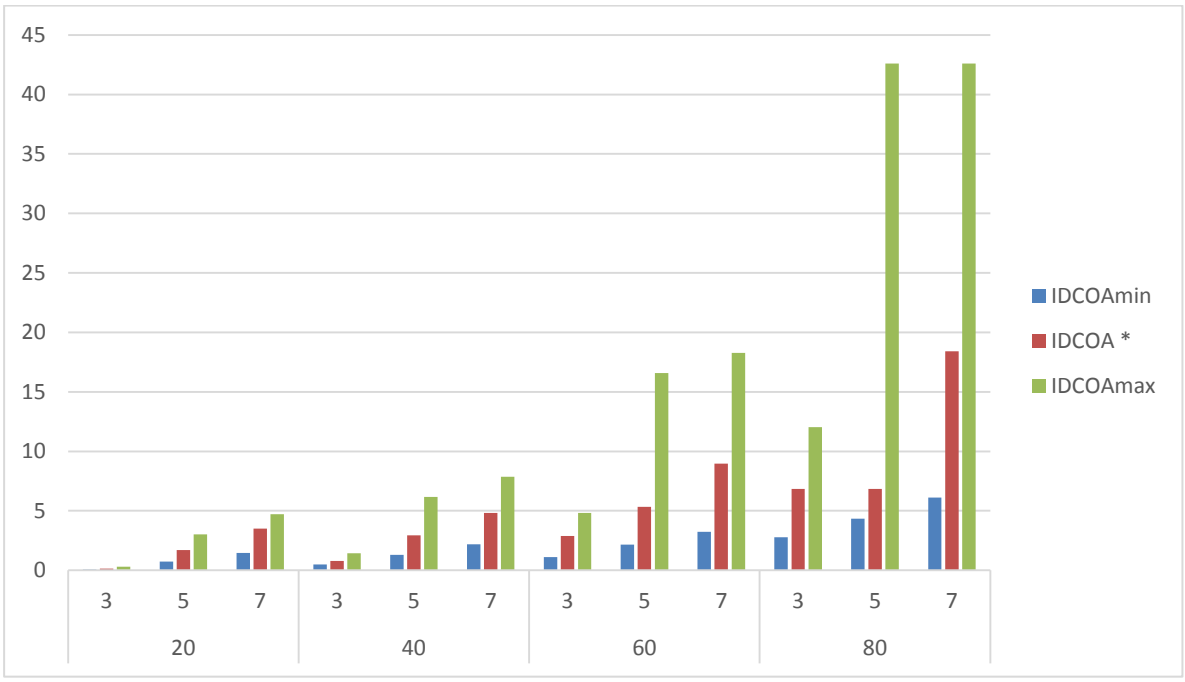

Tabla 14.Media tiempos de computación por número de trabajos IDCOA.

Analizando los tiempos de computación con la Figura 14, se observa que la mayor diferencia de tiempos se produce a partir de los 60 trabajos donde $IDCOA_{max}$ triplica el tiempo de ejecución de los otros dos. $IDCOA_{min}$ aporta los tiempos de ejecución más cortos, sin embargo como se ha mencionado anteriormente sus valores son los más inestables. Por lo tanto, el modelo que aporta mejores resultados es IDCOA* debido a su gran estabilidad en los resultados y que su tiempo de computación solo se acrecienta cuando se sitúa en 80 trabajos donde la diferencia es el doble.

3.2 GRASP

Al contrario que IDCOA, esta metaheurística como se ha explicado anteriormente carece de variables, las únicas son el número máximo de iteraciones sin ofrecer mejora y el número total de movimientos que se van a ejecutar. Los valores seleccionados de dichas variables se presentan en la Tabla 15:

Variables	Valor
Nº de iteraciones sin mejora	5
Nº de iteraciones total	200

Tabla 15. Valores correspondientes a GRASP.

En este caso, el proceso no finalizaría con 5 soluciones sin mejorar, si no que empezaría otra vez la metaheurística con otra solución al azar hasta llegar a las 200 iteraciones en total. A continuación, en la Tabla 16 se representan los valores obtenidos de ARPD y ATG para cada combinación del número de trabajos n y número de máquinas m obtenidos con la heurística GRASP:

		GRASP			
		ARPD		ATG	
n	m	AVG	STD	AVG	STD
2	20	17.612	11.455	52.143	86.782
2	40	17.948	6.713	15.854	30.605
2	60	9.482	6.237	5.929	11.654
2	80	5.827	3.716	2.520	1.562
4	20	10.234	8.599	49.375	131.590
4	40	9.367	5.076	9.821	9.149
4	60	12.628	7.515	2.073	1.989
4	80	6.567	3.866	2.371	2.025
6	20	15.367	9.240	23.611	43.767
6	40	9.956	7.864	3.750	3.516
6	60	7.562	5.563	0.856	1.265
6	80	8.308	7.122	10.523	0.907
8	20	24.372	12.851	71.000	198.050
8	40	16.477	9.009	5.730	10.193
8	60	10.097	5.963	3.055	2.737
8	80	10.523	5.798	2.157	1.428
		12.021	7.287	16.298	33.576

Tabla 16.Resultados GRASP.

3.3 Comparación de metaheurísticas

En esta sección se van a comparar los resultados obtenidos de las dos metaheurísticas, estas comprobaciones se van a realizar mediante diferentes registros, los cuales son la media y el ARPD correspondiente a la función objetivo y los tiempos de ejecución en los diferentes escenarios.

3.3.1 IDCOA vs GRASP

Cabe recordar que en la sección 3.1.2 se estableció que debido a la estabilidad que refleja el modelo IDCOA* con un límite de iteraciones sin mejora de 5. A continuación, en la Tabla 17 se reflejan la media de los valores de ARPD y AGT obtenidos durante la experimentación:

	ARPD		AGT	
	AVG	STD	AVG	STD
IDCOA*	8.158	6.012	43.617	34.067
GRASP	12.021	7.286	16.300	33.576

Tabla 17.Resultados IDCOA vs. GRASP.

Analizando los resultados que se muestran en la tabla anterior se aprecia que son relativamente parecidos, aunque la principal diferencia se contempla en la media del ATG; esto se debe a que el tiempo de computación de IDCOA aumenta con el número de trabajos que intervienen, situación totalmente contraria a GRASP, debido a la escasa variación en los diferentes escenarios. Esta afirmación se refleja gráficamente en la Figura 15.

En la siguiente tabla se muestran los datos correspondientes a la media de la función objetivo mostrados por ambas metaheurísticas dividido por número de máquinas y trabajos con el fin de demostrar la mejora que ofrece IDCOA:

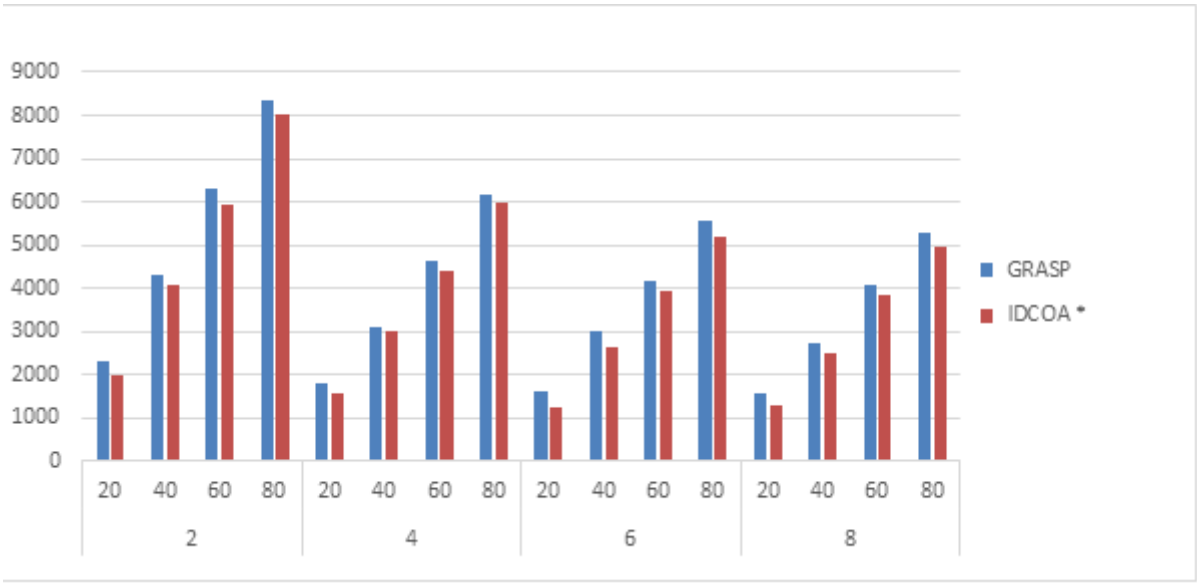

Figura 14.Media función objetivo IDCOA vs. GRASP.

Los resultados ofrecidos por ambos poseen una cercana similitud pero se aprecia que IDCOA consigue mejores resultados en todos los casos. Recordamos que IDCOA en su proceso selectivo antes de utilizar la fase de búsqueda local, mejora las soluciones iniciales y eliminas las de peor función objetivo, al contrario que GRASP, la cual utiliza soluciones totalmente aleatorias en cada una de sus iteraciones. Se concluye que el proceso previo que utiliza IDCOA permite obtener mejores óptimos locales, consiguiendo una ventaja en la media de la función objetivo en cada uno de los escenarios.

Por otro lado, los pasos previos de IDCOA provocan que la duración del tiempo de computación se mucho mayor que GRASP. A continuación, se muestra de forma gráfica dicha diferencia:

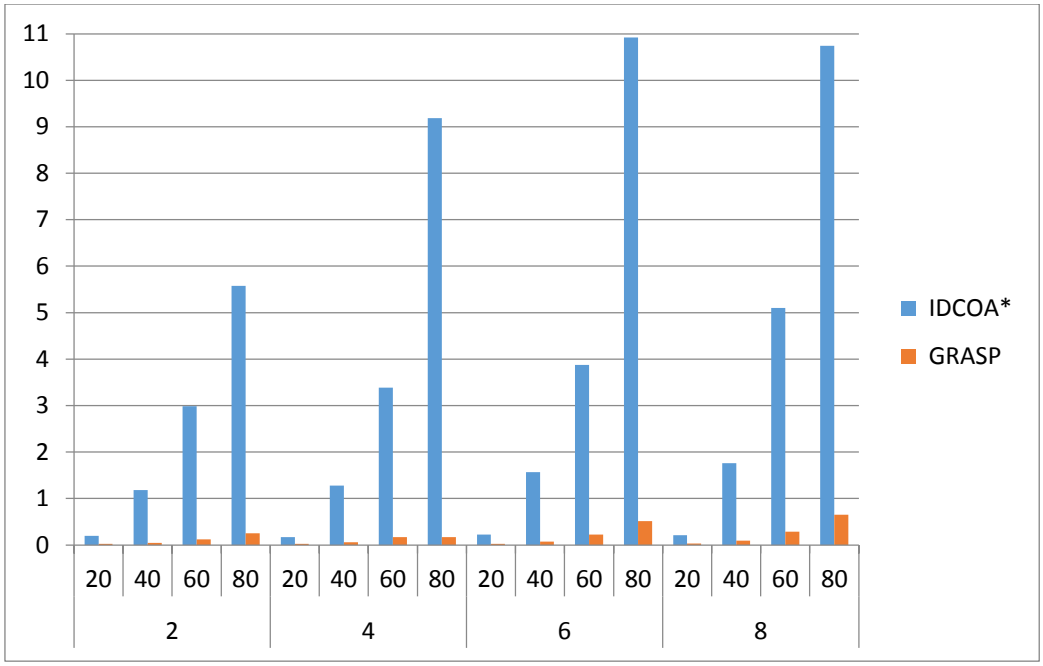

Figura 15.Tiempos de computación IDCOA vs. GRASP.

El análisis de los datos muestra que ambas heurísticas incrementan sus tiempos de computación cuanto mayor es el número de trabajos. Si nos fijamos en IDCOA, la duración del proceso aumenta el doble por cada incremento de 20 unidades en los trabajos. Sin embargo, en GRASP esta diferencia es menospreciable, se debe a las escasas comprobaciones que hace GRASP durante su proceso. No obstante, IDCOA es todo al contrario, si recordamos su desarrollo antes de llegar a realizar la búsqueda local, la cual hace comprobaciones de cada uno de las posibles soluciones y pequeñas variaciones de la secuencia con el fin de mejorarlas con la finalidad de comenzar la búsqueda local con u buenas soluciones para conseguir mejores óptimos local o conseguir el óptimo global.

Respecto al estudio del ARPD de ambas heurísticas se puede observar en que situaciones han funcionado de forma más deficiente:

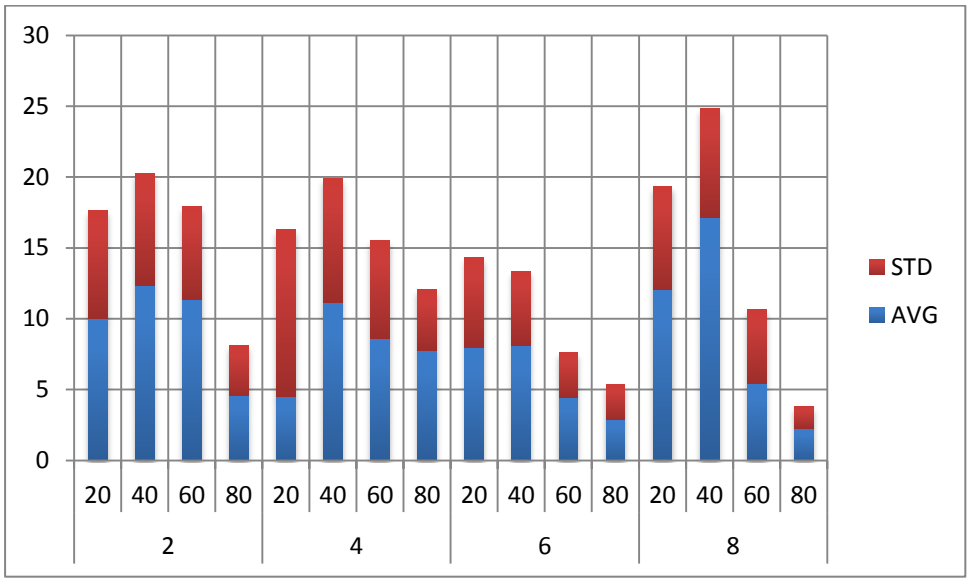

Figura 16.Estudio ARPD IDCOA.

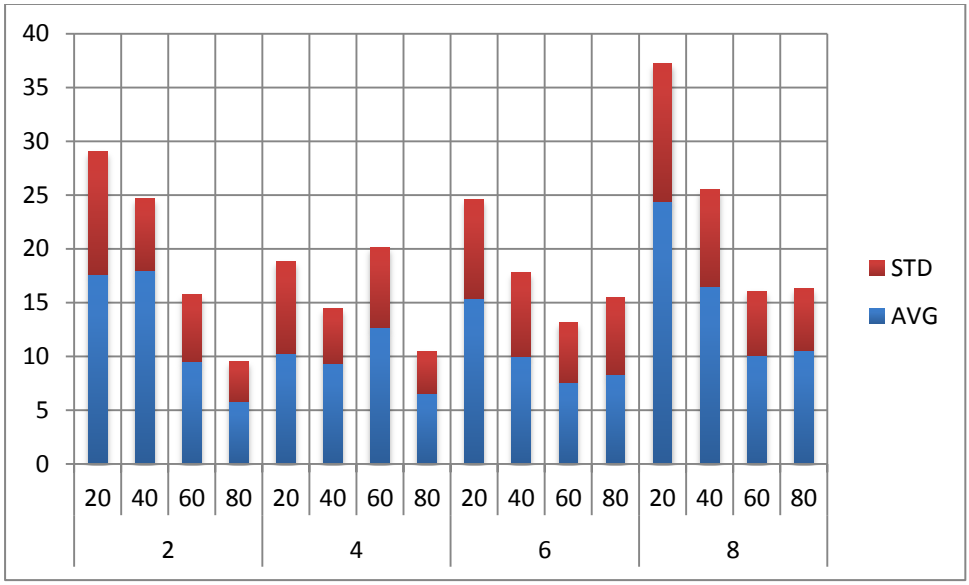

Figura 17.Estudio ARPD GRASP.

Observando ambas figuras, los peores resultados ofrecidos por IDCOA se corresponden a un escenario con 40 trabajos, y por otro lado, GRASP los ofrece en la situación con 20 trabajos. Sin embargo, ambas figuras muestran una característica similar, la cual es que cuanto mayor es el número de trabajos, los resultados son mejores. Basándose en la lógica, esta afirmación sería totalmente contraria, ya que un mayor número de variables sugiere una mayor dificultad para encontrar una solución, y por lo tanto, los resultados pueden tener una mayor variabilidad.

El lado beneficioso de considerar un mayor número de trabajos, es la aportación de un mayor número de óptimos locales, justamente la situación contraria al poseer menos. Evocando a la esencia principal de las heurísticas, su finalidad es la búsqueda de óptimos locales, y que gracias al método de búsqueda local, alcanza posiblemente óptimos inicialmente deficientes pero que evolucionan de forma progresiva en el transcurso de las iteraciones de ambas heurísticas. Se concluye que los escenarios con menor número de máquinas, puede provocar grandes diferencias entre los óptimos locales obteniendo así una mayor variabilidad en los resultados, situación muy contraria al poseer más trabajos ya que posibilita encontrar más posibles soluciones y parecidas.

4. CONCLUSIONES

En el Trabajo Fin de Grado se ha estudiado un problema de programación de la producción dividido en tres etapas destacando la parte final de éste, el ensamblado, que compone la parte más ardua de este proyecto. Dicho escenario está formado en primer lugar por una estructura de m máquinas paralelas para la fabricación de piezas, que correspondientemente serán trasladadas a la parte final donde la principal complejidad es el orden de precedencia de los elementos para la concluir el producto final.

Para llevar a cabo este estudio, el Trabajo Fin de Grado comienza con una introducción de la programación de la producción, explicando sus características, particularidades y tipos de problema, características de la línea de ensamble y Henry Ford, padre de las cadenas de producción, para finalizar con la descripción del problema a estudiar, cuya función objetivo consiste en minimiza el *makespan*. A continuación, se plantea la idea de las heurísticas y el estudio consiguiente para determinar y explicar la utilización de IDCOA y GRASP para solucionar este problema.

Tras estos capítulos teóricos, se procede en primer lugar a la programación en C con el fin de adquirir resultados de la batería de problemas para cada una de las situaciones abordadas por IDCOA con un estudio íntegro de dicho problema para cada una de las variables que intervienen en dicho proceso. Una vez obtenidos los resultados pertinentes se realiza una comparación de los resultados en cuanto a la función objetivo y tiempo de ejecución, demostrando así la veracidad de los resultados obtenidos en el artículo G.M. Komaki et al. (2017). A raíz de ello, se procede a la programación de GRASP con el objetivo de comprobar si la realización de los pasos previos realizados por IDCOA para obtener mejores soluciones iniciales antes de ser realizar la búsqueda local son trascendentales en los resultados finales.

Una vez finalizado el estudio exhaustivo de ambas, se procede a la comparación de los resultados, por la parte del IDCOA, que se encuentran divididos en tres heurísticas debido a la configuración previa, una vez comparados, los resultados verifican la afirmación realizada por Komachi. Tras este hito, se realiza ya la comparación de ambas heurísticas. Al comparar los datos relacionados con la función objetivo (*makespan*), los resultados muestran una diferencia media en torno a las 200 u.t a favor de IDCOA. Sin embargo, en cuanto al tiempo de ejecución, GRASP obtiene valores mínimos, superando rara vez el segundo en el tiempo de computación, sin embargo, IDCOA muestra un aumento progresivo a mayor número de trabajos y máquinas.

Para finalizar, GRASP ha demostrado otra vez que se trata de una heurística muy robusta capaz de adaptarse a cualquier entorno, su característica principal, pero IDCOA, gracias a intentar partir de una mejor solución inicial, presenta valores mucho más estables y para el entorno que se ha verificado los datos, la mejora que hace IDCOA en la función objetivo es suficiente para obviar su tardanza en el tiempo de computación.

REFERENCIAS

[Christos Koulamas, George J. Kyparisis, 1999] The three-stage assembly Flowshop scheduling problem. Computers & Operations Research 28 (2001), 689-704.

[G.M. Komaki a, Ehsan Teymourian , Vahid Kayvanfar , Zahra Booyavi, 2017]. Improved discrete cuckoo optimization algorithm for the three-stage assembly flowshop scheduling problem. Computers & Industrial Engineering 105 (2017), 158–173.

[Paz Pérez González, José Manuel Framiñán Torres, 2016] Material docente: Programación y Control de la Producción. Universidad de Sevilla.

[Ramón Companys Pascual, 1989] Planificación y programación de la producción.S.A. Marcombo.

[Michael L. Pinedo, 2009] Planning and Scheduling in Manufacturing and Services. Springer.

[Framiñán, Leisten, & Ruiz García, 2014] Manufacturing Scheduling System: An Integrated View on Models, Methods and Tools.Springer.

[Fernando Lozano García, Paz Pérez González**,** 2016] Trabajo de Fin de Grado: Problema de Secuenciación en Máquinas Paralelas no Relacionadas y Restricción de Elegibilidad. Universidad de Sevilla.

[Julián David Moreno Dradá, Luisa María Montealegre López, 2013] Trabajo de Fin de Grado: Problema de balance de línea con múltiples líneas en paralelo y enfoque multiobjetivo. Universidad de Sevilla.

[Henry Ford, 2014]. Henry Ford: Mi vida y obra. CreateSpace.

[Arturo Faraone, 2018] Arturo Faraone, Venecia y sus lagunas, Venecia: ciudad industrial. http://www.venicethefuture.com/schede/es/331?aliusid=331 [Último acceso: Junio 2018]

[Ford, 2018] El legado de Henry Ford, https://www.ford.es/acerca-de-ford/historia [Último acceso: Junio 2018].

[Álvaro García Sánchez, 2006] Material docente: Técnicas metaheurísticas.Universidad Politécnica de Madrid.

[Silver, E. A. et al ,1980]. A tutorial on Heuristic Methods. European Journal of Operational Reseach. Vol. 5.

[J.M.Benitez, 2012]. Algoritmos de búsqueda local básicos.Algorítmica, http://sci2s.ugr.es/graduateCourses/Algoritmica [Último acceso: Junio 2018]. Ingeniería Informática. Universidad de Granada.

[Sara Hatami & Sadalah Ebrahimnejad & Reza Tavakkoli-Moghaddam & Yasaman Maboudian ,2009]. Two meta-heuristics for three-stage assembly flowshop scheduling with sequence-dependent setup times. The International Journal of Advanced Manufacturing Technology (2010) 50:1153–1164.

[Saulo Cunha, José Elias Claudio, 2014] NSGA-II with Iterated Greedy for a Bi-objective Three-stage Assembly Flowshop Scheduling Problem. Genetic and Evolutionary Computing Conference, 429-436.

[Aref Maleki-Daronnkolaei, Iman Seyedi, 2014] Taguchi Method for Three-stage Assembly Flow Shop Scheduling Problem with Blocking and Sequence-dependent Set up Times. Journal of Engineering Science and Technology Vol. 8, No. 5 (2013), 603 – 622.

[Fuli Xiong , Keyi Xing , Feng Wang, 2014] Scheduling a hybrid assembly-differentiation flowshop to minimize total flow time. European Journal of Operational Research 240 (2015), 338–354.

[Niloofar Shoaardebili & Parviz Fattahi, 2014] Multi-objective meta-heuristics to solve threestage assembly flow shop scheduling problem with machine availability constraints. International Journal of Production Research, 2015 Vol. 53, No. 3, 944–968.