Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

ESTADO DEL ARTE Y ANÁLISIS DE MÉTODOS DE OPTIMIZACIÓN DE RECURSOS EN PLANTAS DE PRODUCCIÓN

Autor:

Guerra Diego, Alejandro

Tutora:

Jiménez Gómez, María Isabel
Dpto. CMEIM EGI CGF IM IPF

Valladolid, Junio de 2015

A mi abuelo

Don Martín de Diego Martín (1924 – 2015)

AGRADECIMIENTOS

A mi madre, Pepi, dado que sin su esfuerzo y dedicación esto no sería posible. A mi padre, Don Mariano, de quien he aprendido a ver la vida desde un ángulo diferente. A mi hermana, Carmen, por su apoyo incondicional siempre. A mis abuelos, Encarna y Martín, por todo el cariño y sabiduría recibida.

A mis grandes amigos y amigas, por la fuente de inspiración que suponen para mí, por las grandes conversaciones y los periodos ociosos compartidos, ya que sin ellos esto no tendría sentido.

A numerosos profesores de calidad de la Universidad de Valladolid y a la corta, pero excelente relación con mi tutora, Maribel.

Alguien se sienta hoy en la sombra de un árbol que plantó hace mucho tiempo. Warren Buffet.

RESUMEN

La Programación de la Producción consiste en la asignación de los recursos disponibles de una manera eficiente. Es fundamental obtener soluciones de calidad, que permitan aprovechar al máximo estos recursos y de esta manera aumentar la competitividad de la empresa. Cuando el problema de asignación de recursos tiene tamaño real, abordar este tipo de problemas de manera exacta no es factible. Debido a esto, los métodos de optimización aproximados cobran cada vez mayor importancia e interés. En concreto, los Algoritmos Genéticos son perfectamente adaptables al problema de programación de la producción y obtienen soluciones de alta calidad en tiempos razonables.

ABSTRACT

Manufacturing Scheduling is allocating resources in an efficient manner. It is essential to get quality solutions to obtain the optimal resources usage and thus increase the competitiveness of the company. When the problem of resource allocation has full-address such problems it is not exactly feasible. Because of this, the approximate optimization methods are gaining increasing importance and interest. Genetic algorithms are perfectly adaptable to the problem of manufacturing scheduling and obtain high quality solutions in reasonable time.

PALABRAS CLAVE: Programación de Producción, sistema productivo, gestión de recursos, métodos de optimización, algoritmos genéticos.

ÍNDICE

ÍNDICE DE FIGURAS

INDICE DE TABLAS

CAPÍTULO 1:

INTRODUCCIÓN

1.1. Presentación

La competitividad de las empresas productivas en el siglo XXI es un factor determinante a la hora de operar en un mercado cada vez más globalizado y dinámico, donde las empresas que no aprovechan bien sus recursos tienen grandes dificultades para subsistir.

La Programación de la Producción puede ser definida como la asignación eficiente en el tiempo de los recursos disponibles por una compañía con el fin de satisfacer un conjunto de requisitos.

Uno de los recursos más preciados y escasos de las compañías productivas son las máquinas donde los materiales son procesados con el fin de aumentar su valor añadido. Es por esto, que de alguna forma ha de establecerse la mejor manera de ordenar las tareas que van a ser procesadas con el fin de optimizar algún criterio de producción.

Este Trabajo de Fin de Grado, trata sobre la Programación de la Producción dentro de una empresa productiva con una configuración en línea generalizada o configuración flow-shop híbrida.

Resolver el problema de programación de la producción en este entorno, cuando nos enfrentamos a problemas reales con un gran número de trabajos es costoso, siendo en muchos casos inabordable de manera exacta.

Debido a esta complejidad, han sido desarrollados numerosos métodos y algoritmos que no tienen por qué obtener la solución exacta, sino que obtienen soluciones de muy buena calidad en un tiempo reducido.

Muchos de estos algoritmos desarrollados están inspirados en la naturaleza. Este es el caso de los Algoritmos Genéticos que utilizan mecanismos evolutivos para refinar la solución y alcanzar el óptimo del problema.

En este trabajo, tras comprobar mediante el análisis pertinente, que los métodos exactos no tienen una utilidad práctica cuando nos enfrentamos a problemas complejos, se ha desarrollado un algoritmo genético capaz de resolver el problema de programación de la producción en un tiempo prudencial.

El algoritmo desarrollado, aunque utilizando mecanismos simples, es capaz de obtener la solución óptima del problema de una manera eficiente.

1.2. Objetivos

El objetivo principal de este trabajo es analizar los métodos y algoritmos actuales de optimización del problema de programación de la producción y presentar un método que proporciona soluciones de buena calidad al problema de programación de la producción.

Como objetivos secundarios se tienen los siguientes:

Estudio y análisis de las fuentes científicas referentes a la programación en máquinas, con el fin de conceptualizar correctamente el problema que se pretende resolver.

Estado del arte de los distintos métodos y algoritmos de resolución existentes actualmente, distinguiendo entre métodos que obtienen un resultado exacto o métodos que obtienen un resultado aproximado.

Análisis de métodos aproximados inspirados en la naturaleza, con especial atención sobre los algoritmos genéticos, analizando su arquitectura y características más representativas.

Implementación y validación de un algoritmo genético sobre MatLab, capaz de resolver el problema de programación de la producción.

1.3. Medios hardware y software utilizados

La totalidad de este trabajo ha sido llevada a cabo sobre un ordenador con sistema operativo Windows, procesador Intel i5 a 1,8 GHz y 4 Gb de memoria RAM.

Como software, se ha utilizado Microsoft Word para la edición de textos y MatLab para desarrollar y probar los algoritmos.

1.4. Estructura de la memoria

En el capítulo 2, se hace un breve repaso sobre la historia de la producción industrial y como esta ha ido evolucionando a través del tiempo, hasta llegar a filosofías productivas que maximicen los beneficios y permitan ser flexibles a la firma. Además en este capítulo se enmarca la programación de la producción dentro del sistema productivo, identificando el papel que desempeña dentro de una empresa.

A continuación, con la función de programación de la producción ubicada en su contexto, en el capítulo 3, se introducen todos los conceptos teóricos sobre los que se apoya este proceso y que serán utilizados a lo largo del trabajo.

Una vez introducidos los conceptos teóricos, en el capítulo 4, se revisan los métodos y algoritmos de resolución del problema de programación de la producción, existentes en la literatura consultada, tanto exactos como aproximados.

Con el objetivo de presentar los algoritmos genéticos, en el capítulo 5, se exponen los conceptos teóricos y mecanismos evolutivos en los que se basan estos algoritmos así como las ventajas e inconvenientes de estos.

Con el concepto de algoritmo genético introducido, en el capítulo 6, se detallan los mecanismos utilizados en la implementación del algoritmo desarrollado para este TFG apoyándose en diagramas de flujo y esquemas.

Una vez que el algoritmo desarrollado ha sido detallado, en el capítulo 7, se prueba el algoritmo frente a diversos casos utilizados por varios investigadores para validar sus algoritmos, comprobando que el desarrollado en este trabajo funciona con exíto.

En el capítulo 8, aparecen algunas de las conclusiones obtenidas tras la realización de este trabajo, así como líneas futuras de investigación.

Finalmente, en el capítulo 9, aparece la bibliografía consultada para el desarrollo de este trabajo.

CAPÍTULO 2:

LA PROGRAMACIÓN DE LA PRODUCCIÓN DENTRO DEL SISTEMA PRODUCTIVO

2.1. Introducción

El principal objetivo de este capítulo es ubicar la Programación de la Producción dentro del sistema productivo, además se introducirán algunas definiciones sobre el entorno y las estrategias existentes actualmente.

En primer lugar se realiza un breve repaso histórico sobre la producción industrial y cómo ésta ha ido evolucionando a lo largo del tiempo. Después se describirá el Sistema Productivo para poder analizar la Programación de la Producción en su contexto. Una vez ubicada, y ya centrados en el ámbito de la programación en máquinas, introduciremos algunos conceptos relativos al entorno donde se realiza esta programación, prestando especial atención a la influencia que tiene la naturaleza de este (determinista/estocástico) sobre la toma de decisiones. También dentro de este capítulo se definirán conceptos referentes al momento en el que se toman las decisiones como son políticas estáticas, dinámicas, en línea y en tiempo real. Una vez dadas estas definiciones se repasarán algunas de las estrategias existentes (predictivas-reactivas) cuando el entorno es dinámico y sujeto a incertidumbre. Por último, dedicaremos un apartado a las conclusiones extraídas del análisis anterior.

2.2. Historia de la producción industrial

En este apartado se describe cómo los sistemas productivos han evolucionado a lo largo de los años hasta llegar al estado actual, donde la capacidad de adaptarse al entorno de una manera rápida y eficiente es de gran valor para las compañías que compiten en el mercado internacional.

Hasta mediados del siglo XVIII eran los artesanos los que controlaban el proceso productivo en Europa, generando ellos gran parte de las mercancías consumidas. A mediados de dicho siglo, con la revolución industrial, la economía basada en la artesanía y el trabajo manual fue sustituida por una economía industrial y manufacturera. De este modo, las herramientas de trabajo manual se remplazaron por máquinas y la dirección de operaciones empezó su desarrollo.

Tras esta primera fase de industrialización, se inicia una nueva etapa donde se introduce la producción en serie de bienes de consumo. Debido a la complejidad de los nuevos procesos productivos surge la necesidad de organizar de algún modo la producción. Es aquí cuando Frederic W. Taylor (1856 – 1915), un ingeniero y economista norteamericano, sienta las bases teóricas sobre la cadena de montaje, una manera de organizar la producción donde cada operario o trabajador tiene una función específica en la fabricación de los productos. (Cadena Palagot, 2015)

No obstante, la idea de Taylor no se hace realidad hasta que algunos años después Henry Ford (1863 – 1947), joven empresario de la industria del automóvil, pone en práctica dicha idea con gran éxito, fabricando un automóvil barato y fácil de reparar.

En los años siguientes Ford se dedicó a buscar la manera de optimizar su proceso productivo reduciendo el tiempo de producción y los costes utilizando algunas de las ideas de Frederic Taylor. Es así, como llevando estas ideas a cabo, Henry Ford introdujo en sus plantas la cadena de montaje, consiguiendo un aumento espectacular de la producción. De esta manera, en la producción en serie, la máquina adquiere un papel fundamental en el proceso productivo, pasando el artesano a ser una pieza más del sistema industrial.

El impacto de la implantación de los sistemas desarrollados por Frederic Taylor y Henry Ford fue tal que logro influir de forma significativa en el sistema económico de la época, desarrollando las condiciones para el consumo en masa. Esto fue posible gracias al aumento de los salarios y una clase media con mayor poder adquisitivo encargada de dar salida a la producción.

Hasta entonces, la demanda había sido creciente y predecible, pero sobre 1960 una serie de acontecimientos sociales y económicos modificaron drásticamente las características de la demanda, haciendo que grandes industrias que implantaron el modelo fordista sufrieran un gran impacto y cayeran en una profunda crisis.

Uno de los puntos débiles del sistema implantado por Henry Ford, fue la sustitución de una demanda de productos similares entre sí por una demanda de un único producto estándar. Esta característica, que hizo triunfar al sistema de Ford, también lo llevó al agotamiento, debido a las nuevas exigencias del mercado que demandaba productos diversos. De esta manera el modelo que había funcionado durante años empezó a agotarse debido a su inflexibilidad ante una demanda plural.

Fue entonces cuando las empresas japonesas, después de la guerra de Japón, se esforzaron al máximo por aprovechar los pocos recursos de los que disponían para maximizar la producción y reconstruir su economía. Una empresa en concreto, Toyota, desarrolló un sistema de gestión que daba respuesta a las nuevas exigencias del mercado. Este sistema se conoce como sistema de producción Toyota y su desarrollo se atribuye principalmente a tres personas, el fundador de Toyota, Sakichi Toyoda, su hijo Kiichiro y el ingeniero Taiichi Ohno. (Lefcovich, 2015)

Mediante este sistema, se pasó de la producción de un único producto estándar a gran escala, a la producción de pequeños lotes de productos diversos capaces de hacer frente a la demanda cada vez más diversificada. Para conseguir que esto sea rentable, el sistema de producción Toyota, trata de eliminar al máximo elementos innecesarios para reducir costes, incidiendo en el empleo excesivo de recursos, exceso de producción y exceso de existencias. Esta optimización continua se conoce hoy en día como filosofía Kaizen.

Debido a la puesta en práctica de estas nuevas ideas, las empresas japonesas alcanzaron un grado de competitividad tal, que los norteamericanos se vieron obligados a tomar algunas de las principales ideas de esta filosofía y adaptarlas a las empresas occidentales. Lo que comenzó siendo un contrataque contra el avance de la industria japonesa, es en la actualidad un pilar fundamental de cara a las adversidades a las que se enfrentan las compañías en el siglo XXI. De esta manera nace el Lean Management, siendo esta forma de producción la que se está imponiendo actualmente en la industria.

El sistema Lean trata de producir en el momento adecuado lo que el mercado requiere, dando especial atención al factor tiempo. Busca la optimización de los recursos y la producción de bienes de alto valor añadido a un precio razonable. Se trata de producir más con menos.

Paralelamente al desarrollo industrial, en las últimas décadas, las herramientas y métodos de Programación de la Producción han sufrido un fuerte desarrollo debido a la cada vez mayor complejidad de los sistemas productivos y las exigencias del mercado.

La tendencia actual en la producción industrial es flexibilizar el sistema productivo para así adaptarse a un entorno cambiante e incierto de la manera más eficiente posible y de este modo, dotar a las compañías de la competitividad necesaria para operar con resultados satisfactorios en el mercado global del siglo XXI.

2.3. El Sistema de Operaciones

Llegados a este punto se ha considerado fundamental realizar una introducción al Sistema de Operaciones para así ubicar correctamente la **Programación de la Producción** dentro del sistema productivo. De esta forma, en este apartado se describirá este sistema y las funciones que realiza dentro de la empresa.

La misión del Sistema de Operaciones es la obtención de productos para satisfacer las necesidades impuestas por el Sistema Comercial de una compañía. Para satisfacer dichas necesidades, este sistema, trata de convertir unos determinados recursos en bienes de valor superior. (Companys & Corominas, 1996)

El proceso de dirección de operaciones comienza definiendo los objetivos a largo plazo, que deben ser coherentes con la política global de la compañía. Estos objetivos definen las metas a conseguir, cómo se conseguirán esas metas y con qué medios. Una vez fijados los objetivos a largo plazo debemos descender al medio y corto plazo para encontrar el proceso de Programación de la Producción.

Dentro del Sistema de Operaciones, se diferencian claramente las funciones del Planificador y del Programador. El Planificador determina las cantidades de productos a fabricar, así como el momento en que se fabricarán. Esta información se recoge en el Plan Maestro de Producción, que contiene las cantidades y las fechas en las que deben

estar disponibles los productos. Por otro lado, el Programador se encarga de asignar los recursos disponibles a los trabajos que han de ser realizados.

La determinación del Plan Maestro de Producción se realiza en función de la previsión de la demanda y la capacidad de la empresa, escogiendo entre los posibles planes el que dé lugar al mínimo coste de operación.

Tras la elaboración del Plan Maestro de Producción, se realiza el cálculo de los Requerimientos de Materiales, del que se obtienen dos resultados, por un lado se determinan el conjunto de trabajos a fabricar y por otro las necesidades materiales para fabricar dichos trabajos.

Uno de los métodos más utilizados para obtener los Requerimientos de Materiales es el MRP (Materials Requirement Planning), donde el cálculo de las órdenes de fabricación y las necesidades materiales se obtiene decalando en el tiempo la demanda impuesta por el Plan Maestro de Producción, de este modo, se puede hablar de sistemas tipo push, que empujan la producción para que cumpla el calendario.

Dado que el MRP no tiene en cuenta los recursos disponibles por la empresa, se puede llegar a soluciones no admisibles por falta de recursos. Para solventar este problema, existen sistemas MRP que consideran las restricciones de capacidad y ajustan los Requerimientos de Materiales a los recursos existentes. Estos sistemas se conocen como sistemas MRP II (Manufacturing Resource Planning) o MRP de bucle cerrado. (EGUIA SALINAS, 1996)

Existen también otros métodos como la Tecnología de la Producción Optimizada (OPT, Optimized Production Technology) que tienen en cuenta las limitaciones de capacidad para obtener la planificación de la producción. Estos sistemas se basan en la teoría de las restricciones (TOC, Theory Of Constraints) identificando los cuellos de botella donde la capacidad es escasa e intentando maximizar la eficiencia en dichos puntos.

Otro enfoque basado en la descentralización de la producción lo forman los sistemas justo a tiempo o JIT (Just In Time). La idea fundamental en la que se basan estos sistemas es producir la cantidad necesaria en el instante preciso para satisfacer la demanda, de esta manera todos los materiales se encuentra activos en el proceso, logrando reducir drásticamente los inventarios y los costes asociados que estos conllevan.

Como se puede ver en la figura 2-1, es en el ámbito del corto plazo donde se encuentra la **Programación de la Producción**, esta consiste en la asignación en el tiempo de los recursos disponibles para satisfacer de una forma óptima las órdenes derivadas de realizar la planificación de la producción. Dicha programación está relacionada y debe interactuar con otras funciones de la organización.

La Programación de la Producción es el paso previo a la ejecución de las operaciones de fabricación y a pesar de su importancia dentro de la empresa, en numerosos casos, no se presta la atención necesaria a esta tarea, siendo llevada a cabo por personal del nivel operativo como técnicos y operarios.

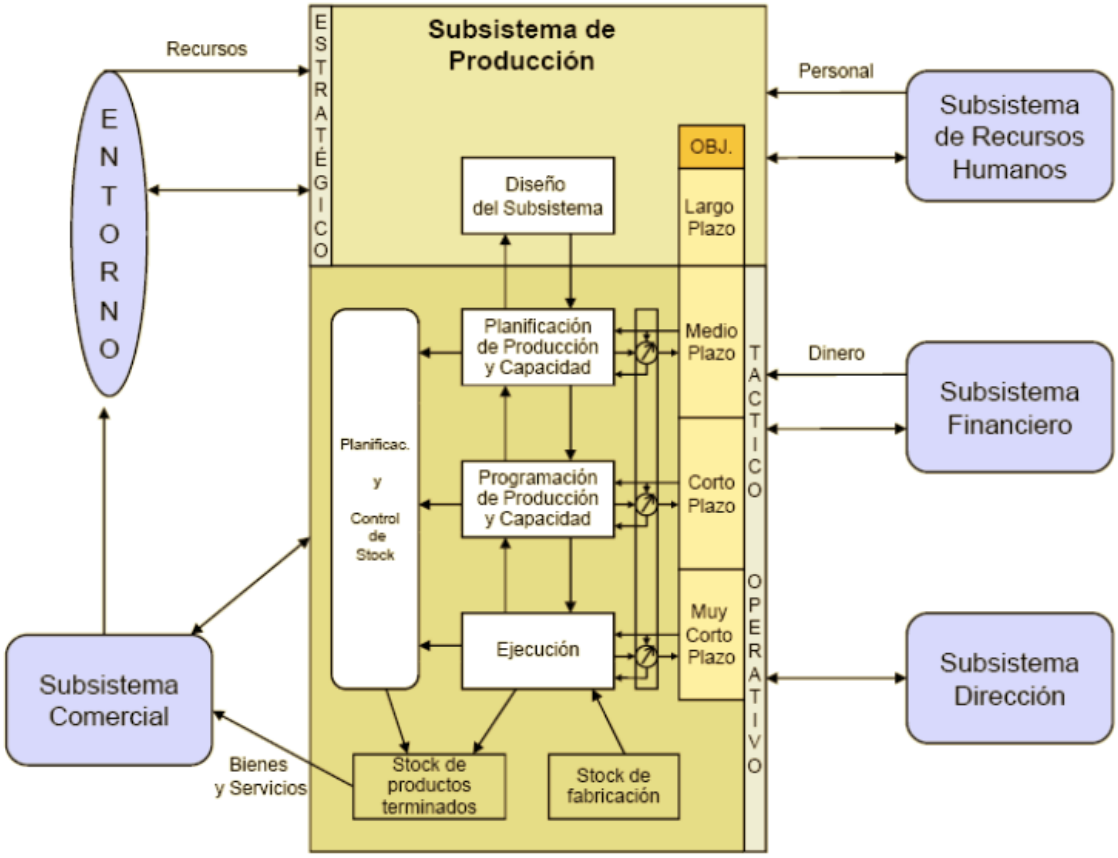

Figura 2 - 1. Estructura del Sistema de Operaciones [Vicens, 1999].

Descendiendo hasta el ámbito del muy corto plazo y tiempo real, encontramos las actividades relacionadas con el Control de la Producción cuyo objetivo es monitorizar las operaciones en curso con el fin de detectar anomalías y así poder actuar sobre ellas.

Como se puede observar, la función de programación esta interrelacionada con otras funciones de la empresa como son el control de la producción y la planificación. Si examinamos estas relaciones, el Sistema de Control de la producción debe comunicar el estado del sistema productivo al Sistema de Programación de la Producción. Este, a su vez, informa al Sistema de Planificación de la Producción, pudiendo solicitar cambios en el Plan Maestro de Producción debido a restricciones dentro del sistema productivo.

La manera en la que se comunican los diferentes subsistemas dentro del sistema de operaciones es de vital importancia para la programación de la producción pudiendo solicitarse cambios en esta o informando a niveles superiores de la no viabilidad del plan de producción determinado. En la figura 2-2 aparece un esquema del flujo de

información a lo largo del Sistema de Operaciones, además se detalla que tipo de información es la que se intercambia entre los diferentes subsistemas.

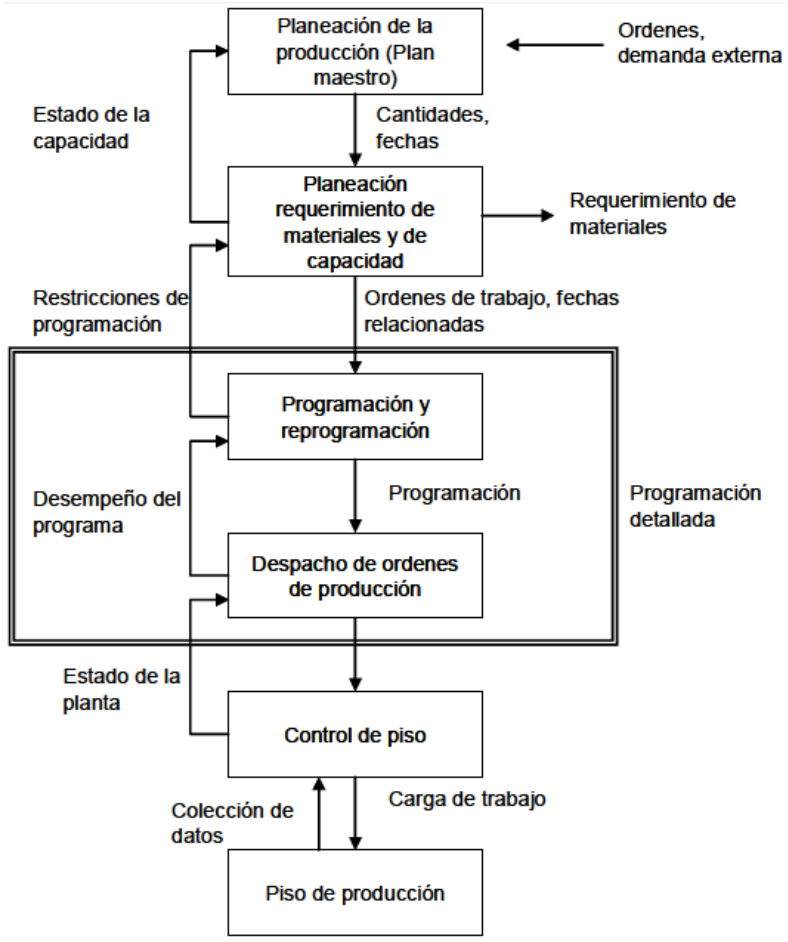

Figura 2 - 2 Flujo de información en un sistema de producción. [Pinedo, 2005]

Como vemos la Programación de la Producción se encuentra en el ámbito del corto plazo y puede sufrir diferentes modificaciones dependiendo del estado de la planta y las restricciones de capacidad.

2.4. La Programación de la Producción

Según (Companys & Corominas, 1996) la Programación de la Producción se define como "el proceso de asignación de órdenes de fabricación y/o las operaciones en las que se descomponen, a recursos específicos dentro de intervalos de tiempo concretos".

En este punto, es importante definir algunos términos comúnmente utilizadas en este trabajo ya que estas no están unificadas en la literatura, dependiendo su significado del autor o autores consultados. Un **Lote** es la unidad básica en la que puede descomponerse el Plan Maestro de Producción y queda definido por el artículo

comercial, la cantidad a fabricar y la fecha de disponibilidad prevista. La unidad básica del Programa de Producción es el **Trabajo** y este queda definido por la referencia, comercializable o no (número de pieza, montaje, sub-montaje, etc.), la cantidad a fabricar, la operación que hay que realizar y con qué recurso se realizará esta además de la fecha de inicio y fin de la operación. Además, se puede definir **Orden de Fabricación** al conjunto de trabajos agrupados con un fin común. (GOMEZ GASQUET, 2010)

Los autores consultados distinguen tres funciones dentro del proceso de programación, que pueden ejecutarse simultáneamente o no:

- **Función carga (loading):** Consiste en la asignación de las operaciones a los centros de trabajo disponibles.

- **Función secuenciación (sequencing):** consiste en la secuenciación de las operaciones, previamente asignadas a un centro de trabajo, con el fin de establecer su orden de ejecución.

- **Función temporización (scheduling):** consiste en la determinación de los instantes de incio y fin de cada operación.

Algunos de los principales objetivos perseguidos por la Programación de la Producción son los siguientes:

- Cumplir los plazos, terminando los trabajos dentro del plazo un alto porcentaje de órdenes.

- Obtener una alta utilización de los recursos, sean máquinas o personal.

- Reducir al mínimo recursos extraordinarios, como pueden ser las horas extra.

- Reducir al mínimo la cantidad de trabajos en curso.

Además de los objetivos citados anteriormente existen muchos otros dependiendo de las políticas globales de la empresa. Como se ha podido observar estos objetivos son antagónicos entre sí, lo que lleva a establecer una jerarquía entre ellos con el fin de eliminar conflictos a la hora de realizar la programación.

Aunque existe suficiente teoría sobre la Programación de la Producción, en el entorno industrial, no se suelen aplicar los resultados obtenidos a partir de las investigaciones y publicaciones científicas que podrían ayudar a mejorar l competitividad de las empresas. Este hecho, puede ser debido a diferentes factores como el desconocimiento o la dificultad de los métodos utilizados para programar la producción. La solución a este problema es desarrollar interfaces y procedimientos más amigables que permitan la

resolución del problema de programación a personal no excesivamente familiarizado con la teoría de programación de la producción.

Otro de las barreras que impiden utilizar la teoría desarrollada en el ámbito de la investigación deriva de que las situaciones analizadas en la teoría se alejan enormemente de la práctica, debido al análisis de problemas simplificados difíciles de extrapolar al entorno industrial.

Existen multitud de procedimientos y técnicas para abordar un problema de Programación de la Producción. Estos se verán y desarrollarán en capítulos posteriores donde se hará una revisión de los métodos a los que se ha tenido acceso. Estos métodos utilizan técnicas matemáticas y algoritmos para la toma de decisiones, con el fin de asignar los recursos de la forma más eficiente posible.

2.5. Entornos Deterministas o Inciertos

Se entiende por entorno **Determinista** aquél donde el modelo se conoce con precisión y certidumbre (Maccarthy, B. L. & Liu, J. Y. 1993;Mula, J. 2004). La precisión hace referencia a que los parámetros del modelo representan fielmente el sistema modelado. Por otro lado, la certidumbre asume que el modelo y sus parámetros se conocen definitivamente y no existen dudas sobre éstos.

Se entiende por **Incertidumbre** a la diferencia entre la cantidad de información necesaria para ejecutar una tarea y la cantidad de información que realmente se dispone para ejecutar dicha tarea (Galbraith, J. 1973). Normalmente, este tipo de incertidumbre, se ha abordado utilizando las teorías de la probabilidad y la estadística, complementadas con teorías sobre la toma de decisiones.

La mayor parte de las herramientas desarrolladas hasta la fecha y utilizadas para modelar y resolver los problemas relativos a la Programación de la Producción son Deterministas. Sin embargo, en las situaciones reales a las que se enfrentan las compañías de producción son normalmente afectadas por la Incertidumbre. Muchos de los procesos de programación de la producción en el mundo real ocurren en entornos donde las restricciones, las metas y las consecuencias de aplicar unas acciones concretas no se conocen con certidumbre.

Esta imprecisión puede ser de dos tipos, **Incertidumbre Estocástica** o aleatoria, debida a las imprecisiones del mundo real, o **Borrosidad** debida a la imprecisión en la descripción del problema, los modelos o los eventos que se producen en éste.

Una vez presentados los entornos donde han de operar las organizaciones, es conveniente explorar las causas que la producen y como afectan al sistema productivo.

Algunos autores clasifican la Incertidumbre en dos grandes grupos, la correspondiente al entorno y la correspondiente al sistema productivo. Atendiendo al alcance o como afectan las imprecisiones a la toma de decisiones se distinguen tres tipos:

- Completamente desconocidas: se refiere a eventos impredecibles ante los que solo cabe reaccionar.

- Sospechosas: eventos intuidos en base a la experiencia del programador que son difíciles de incorporar en el programa productivo.

- Conocidas: aquellas incertidumbres que pueden ser previstas e incorporarse en los algoritmos y programas.

A causa del desarrollo de la teoría sobre Programación de la Producción, los problemas que se solían abordar bajo un punto de vista Determinista, se han ido modificando de tal manera que pretenden modelar la Incertidumbre presente en cualquier proceso productivo del mundo real. Bajo estas condiciones, los métodos asumen que los datos de los trabajos (tiempos de procesado y fechas) pueden no conocerse con precisión, sino que de lo que se dispone es de una distribución probabilística de los mismos, o funciones de pertenencia.

Algunos de los eventos que suceden normalmente en el mundo real y aportan incertidumbre al sistema son los siguientes: fallos en máquinas, trabajos urgentes, cancelación de trabajos, cambio en las fechas de entrega, retrasos o falta de materias prima, cambio en las prioridades de los trabajos, desviación en los tiempos de proceso, problemas de calidad, etc. Actuar con eficiencia frente a algunas de estas situaciones aporta gran competitividad a las empresas, así como ahorros considerables.

2.6. Conclusiones

La tendencia actual en la producción industrial es optimizar el proceso productivo para reducir costes y aumentar la competitividad. Además, las compañías fabriles deben ser extremadamente flexibles debido al corto ciclo de vida de los productos y la rapidez con la que suceden los cambios en el siglo XXI.

Dichos cambios, generan un entorno incierto donde la demanda y la evolución tecnológica de los competidores son difíciles de prever a largo plazo. Debido a esto, para que un programa productivo sea eficaz, debe incluir en él la incertidumbre, tanto la derivada del entorno como la derivada del proceso productivo.

CAPÍTULO 3:

LA PROGRAMACIÓN DE LA PRODUCCIÓN: MARCO TEÓRICO Y DESARROLLO CONCEPTUAL

3.1. Introducción

El principal objetivo dentro de este capítulo es introducir el marco teórico sobre el que se desarrolla el problema de programación de la producción. Además se definirá el problema concreto que se analiza dentro de este trabajo.

En primer lugar se definirá formalmente el problema general que se pretende resolver a la hora de programar la producción, después se introducirán una serie de hipótesis sobre las que se desarrolla la teoría existente en la bibliografía consultada. Una vez definido el problema general, se estudiarán las diferentes configuraciones productivas existentes en la industria fabril y cómo, dependiendo de una u otra, el proceso de programación varía. Conocidas las diferentes configuraciones productivas, se expondrá el problema sobre el que se centra este trabajo, la programación de la producción de un taller tipo "flowshop".

Debido a la gran cantidad de problemas existentes, se introducirá una nomenclatura común, ampliamente aceptada en la literatura sobre el tema, para definir el tipo de problema al que se enfrenta el programador. A continuación se repasarán los criterios de optimización y restricciones comunes en la industria que condicionan la programación de la producción. Después se expondrá el cálculo de uno de estos criterios denominado tiempo de ejecución o makespan. Por último, se estudiará la complejidad computacional de este tipo de problemas.

3.2. Definición formal del problema

El problema de la programación de trabajos en máquinas puede definirse como la asignación en el tiempo de los recursos disponibles de forma que satisfaga un conjunto de criterios y/o restricciones. Se trata de secuenciar un conjunto de trabajos que han de ser procesados en un conjunto de máquinas. Cada trabajo lleva asociado una secuencia de operaciones y cada operación se realiza en una máquina concreta durante un periodo de tiempo conocido. (PINEDO, 2012)

Para definir formalmente el problema, en la literatura consultada, se propone la siguiente notación que será la que utilizaremos a lo largo del trabajo:

- Sea $\mathrm{J} = \{\mathrm{J}_1, \mathrm{J}_2, \dots, \mathrm{J}_j\}$ el conjunto de trabajos a realizar.

- Sea $\mathrm{M} = \{\mathrm{M}_1, \mathrm{M}_2, \dots, \mathrm{M}_k\}$ el conjunto de máquinas o procesadores donde se van a realizar las operaciones correspondientes a cada trabajo.

- Sea $O = \{ O_{hkj} : h = 1,2, \dots, H_k ; j = 1,2, \dots, n ; k = 1,2, \dots, m \}$ el conjunto de operaciones que hay que realizar. El trabajo J_j llevará asociado el conjunto de operaciones O_{hks}.

Existen además una serie de datos asociados a los trabajos que podemos resumir en:

- **Tiempo de procesado** (p_{ij}): hace referencia a la duración de la operación O_{ij}.

- **Fecha de disponibilidad del trabajo** (r_i): momento en el que se pueden iniciar las operaciones del trabajo J_i.

- **Fecha de vencimiento del trabajo** (d_i): momento en el que debe estar disponible el trabajo J_i.

- **Fecha de inicio** (S_i): momento en el que se comienzan las operaciones sobre el trabajo J_i.

- **Fecha de finalización** (C_i): momento en el que se terminan las operaciones sobre el trabajo J_i.

- **Holgura** (L_i): tiempo comprendido entre la fecha de vencimiento y la fecha de finalización del trabajo J_i.

- **Peso** (W_i): tiempo comprendido entre la fecha de disponibilidad y la fecha de inicio del trabajo J_i. Entendido como la prioridad del trabajo.

En la figura 3-1 se pueden ver de forma esquemática los parámetros definidos anteriormente:

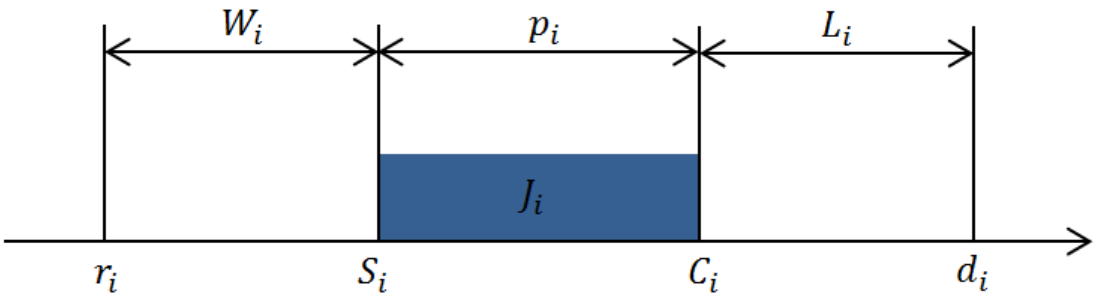

Figura 3 - 1. Datos y variables asociadas a un trabajo [elaboración propia]

Además de variables temporales asociadas a los trabajos, también existen variables económicas como las penalizaciones que representan el coste unitario en el que se incurre por cada unidad de tiempo de retraso del trabajo J_i, respecto a la fecha de entrega o por la espera del trabajo J_i antes de ser procesado.

De esta manera, el problema de programación de la producción consiste en encontrar la secuencia de un conjunto de trabajos $J = \{J_1, J_2, \dots, J_j\}$, que se procesarán en un conjunto de máquinas $M = \{M_1, M_2, \dots, M_k\}$, estando cada trabajo formado por una serie de operaciones $O = \{O_{hkj}\}$. Los programas quedan definidos dando los tiempos de inicio (S_i) y finalización (C_i) de cada trabajo (J_i) en cada máquina (M_i).

Un ejemplo de programa muy simple con dos máquinas y dos trabajos es el que se muestra en la figura 3-2, donde existen dos trabajos que son procesados secuencialmente por dos máquinas M_1 y M_2.

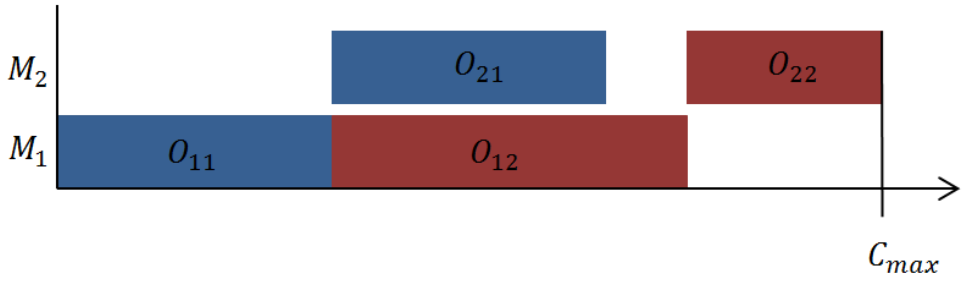

Figura 3 - 2. Programa con dos máquinas y dos trabajos [elaboración propia].

En el apartado 3.9. se resolverá un programa ejemplo para un criterio de optimización concreto.

3.3. Relaciones de precedencia

La precedencia se refiere a la secuencia de operaciones a seguir en la elaboración de un determinado trabajo. De esta manera, un diagrama de precedencias se realiza para visualizar con mayor claridad las relaciones entre las operaciones a seguir para llevar a cabo un trabajo. Estos diagramas tienen como objetivo determinar únicamente las precedencias de las operaciones pero no las tareas críticas, es decir aquellas tareas que al adelantarse o retrasarse afecta al tiempo total del programa.

Si se supone conocido el orden de operaciones para cada trabajo, éstas se ordenan linealmente según su orden de precedencia:

$$\text{si } O_r, O_s \in J_j \text{ y } r < s \rightarrow O_r \text{ precede a } O_s$$

Figura 3 - 3. Diagrama de precedencia con dos operaciones [elaboración propia].

3.4. Hipótesis comunes en programación de máquinas

En este apartado expondremos una serie de hipótesis que aparecen frecuentemente en la literatura sobre secuenciación de trabajos en máquinas. La mayoría de estas fueron introducidas por (Conway, R. W. et al. 1967):

- Las máquinas nunca dejan de funcionar y están siempre disponibles.
- Cada máquina solo puede procesar un trabajo a la vez.
- Cada trabajo solo puede procesarse únicamente en una máquina a la vez.
- Los tiempos de preparación de los trabajos son nulos, es decir, al comienzo del proceso todos los trabajos están disponibles.
- Cuando una operación ha comenzado, debe terminarse antes de empezar otra en la misma máquina. No se permiten interrupciones.
- Los tiempos de cambio son independientes del programa y están incluidos dentro del tiempo de procesado.
- Los tiempos de cambio y las restricciones tecnológicas son deterministas y se conocen de antemano.

Como podemos ver, estas hipótesis hacen que la teoría se aleje de la realidad, dado que los entornos fabriles donde se realiza la producción son por naturaleza dinámicos y sujetos a una serie de perturbaciones de carácter estocástico. Estas perturbaciones pueden modificar el estado del sistema productivo y afectar al rendimiento del mismo.

Existen eventos relacionados con los recursos como los fallos en máquinas, bajas de los operarios, no disponibilidad de materiales o herramientas, etc. Por otro lado, existen eventos relacionados con los trabajos como son la llegada de órdenes, la cancelación de trabajos, cambios en las fechas de entrega, etc.

3.5. Tipos de configuraciones productivas

Una de las decisiones estratégicas que deberá tomar el Sistema de Operaciones será el tipo de configuración productiva que se instalará en la planta. Esta decisión se tomará en función de criterios tales como proporcionar suficiente capacidad de producción, reducir el coste asociado al manejo y transporte de materiales, flexibilidad, etc.

Se pueden considerar tres tipos de sistemas en función de cómo se realizan las operaciones en las máquinas, teniendo esta secuencia de operaciones especial impacto a la hora de programar la producción:

- **Sistemas de flujo uniforme** ("flow shop"): todos los trabajos tienen el mismo número de operaciones que coincide con el número de máquinas, además todos los trabajos tienen la misma ruta y las operaciones se ordenan en la misma secuencia.

- **Sistemas de taller abierto** ("open shop"): todos los trabajos tienen el mismo número de operaciones pero no existe definido un flujo para cada trabajo. El orden de las operaciones para cada trabajo tiene que coincidir con el de otro.

- **Sistemas tipo taller** ("job shop"): el número de operaciones para cada trabajo puede variar y además cada trabajo tiene su ruta específica, lo que complica enormemente el problema de programación de la producción.

En el caso en el que se considera la existencia de más de una máquina capaz de ejecutar la misma operación aparece el concepto de Etapa. Una Etapa es un conjunto de máquinas capaces de realizar la misma operación. De esta manera, se pueden considerar las siguientes configuraciones productivas derivadas de introducir este concepto en las definidas anteriormente:

- **Máquinas paralelas:** M máquinas idénticas sólo en una etapa de trabajo. Cada trabajo necesita solamente una de las máquinas.

- **Taller de flujo híbrido:** cada trabajo sufre una serie de operaciones, donde cada operación puede ser ejecutada por cualquiera de las máquinas presentes en cada etapa. Esta configuración productiva puede ser vista como un caso general del taller de flujo donde para cada operación existe más de una máquina.

- **Taller con máquinas duplicadas:** es un taller donde existen M máquinas capaces de ejecutar cada operación y cada trabajo que requiere esa operación solo se procesa en una de esas máquinas.

3.6. El taller de flujo y el taller de flujo híbrido

Una vez definidas las diferentes configuraciones productivas existentes, nos centraremos en definir el problema que ha de resolver el programador cuando se enfrenta a la programación de la producción dentro de un taller de flujo híbrido. Esta configuración productiva está presente en multitud de industrias en la actualidad como por ejemplo la textil y cerámica.

En un taller de flujo o “flow shop” tenemos un conjunto n de trabajos que debe ser procesado en un conjunto m de máquinas. Normalmente, cada trabajo debe procesarse en todas y cada una de las máquinas, siendo el orden el mismo para todos los trabajos. Esto implica una relación de precedencia en las operaciones, de este modo, un trabajo i debe sufrir una serie de operaciones $O_{1i,}O_{2i,}\dots,O_{mi}$.

La representación esquemática de un taller de flujo puede verse en la figura 3-1:

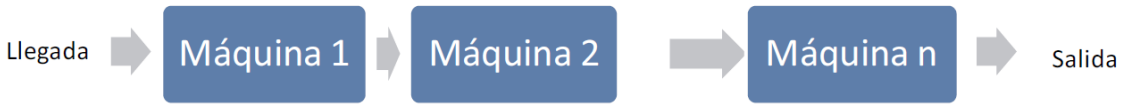

Figura 3 - 4. Representación de un Taller de Flujo con n Máquinas [Gómez Gasquet, 2010]

El problema de la programación de un taller de flujo es un problema combinatorio bastante complejo que requeriría varios años de cálculo cuando nos enfrentamos a un problema de grandes dimensiones.

Un Taller de Flujo Híbrido se define como una configuración de máquinas organizadas en una serie de Etapas donde se procesan una serie de Trabajos. Las etapas están formadas por un conjunto de máquinas o recursos equivalentes en cuanto a su funcionamiento, aunque pueden no serlo en cuanto a su eficiencia, provocando esto una variación en la duración de una operación dependiendo de la máquina elegida. Además las máquinas solo pueden ejecutar una operación a la vez y cada Trabajo sufre solamente una operación en cada etapa. En la figura 3-5 aparece la representación de un taller de flujo híbrido de forma esquemática.

Normalmente el flujo de los trabajos es unidireccional y debe pasar por todas las etapas existentes antes de la salida de la planta. Sin embargo, existen también procesos con flujos reentrantes en los que los trabajos deben visitar una misma etapa más de una vez antes de abandonar la planta.

A la hora de realizar la programación de un taller de flujo híbrido nos encontramos ante un problema que depende de los objetivos de optimización que se persigan conseguir. Normalmente el criterio que marca la programación es reducir el tiempo de procesado o makespan pero pueden existir gran variedad de objetivos que se definirán en apartados sucesivos.

Aunque la gran mayoría de la teoría existente no considera la programación multiobjetivo, es decir, realizar un programa que satisfaga más de un criterio de optimización, cada vez es mayor el interés sobre este tipo de problemas.

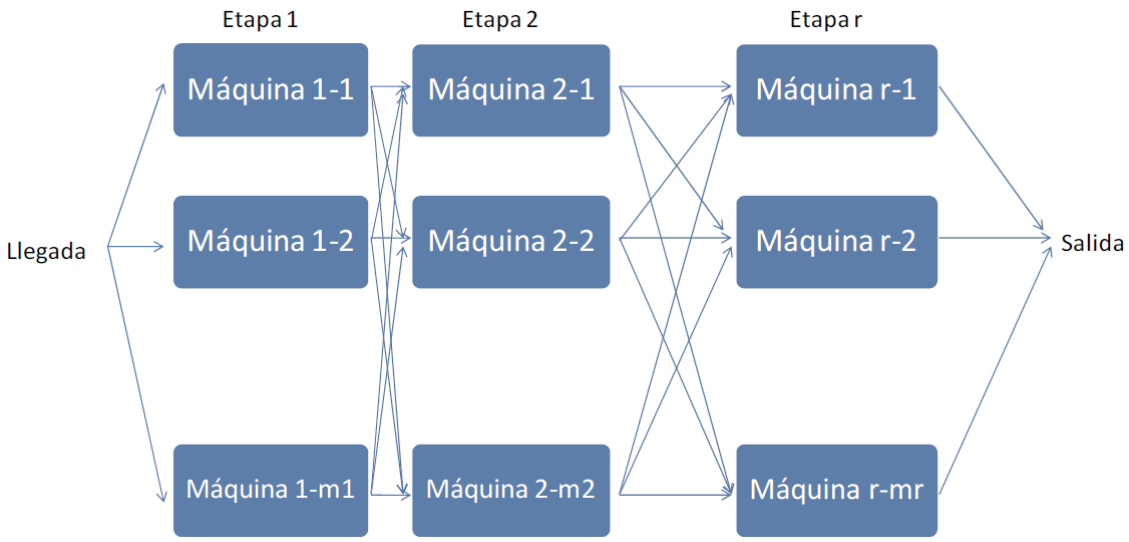

Figura 3 - 5. Representación de un Taller de Flujo Híbrido con r etapas [Gómez Gasquet, 2010].

3.7. Características de los problemas. Nomenclatura.

Debido a la gran cantidad de tipos de problemas existentes sobre secuenciación de trabajos en máquinas y programación de la producción surge la necesidad de clasificar éstos. Diferentes características de los trabajos, de las máquinas y los criterios de optimización elegidos originan una gran cantidad de problemas con diferentes características que deben ser clasificados de alguna forma.

Una de las formas más difundidas y ampliamente aceptadas es la desarrollada por (Graham, 1979). Esta clasificación consta de tres parámetros, α|β|γ, que identifican el tipo de problema al que se enfrenta el programador. El campo α define el ambiente de la máquina y contiene un único elemento. El campo β contiene detalles sobre las restricciones y características del procesamiento. Este campo puede contener varios elementos correspondientes a las diferentes restricciones y características. Por último, el campo γ hace referencia al criterio de optimización que se persigue al realizar la programación. Generalmente, este campo contiene un único elemento cuando se trata de una programación con un objetivo único pero en la actualidad existen problemas de optimización multiobjetivo generando varios elementos del campo γ.

Para el primer campo α, se pueden distinguir entre problemas en los que se dispone de una única máquina (α=1) de aquellos en los que se disponen de varias máquinas (α>1). De esta manera, en una configuración productiva tipo “flow shop” el campo α será igual al número de máquinas de la planta.

Dentro del segundo campo β, se encuentran las características de los trabajos como pueden ser trabajos independientes, con restricciones, con relaciones de precedencia, trabajos sin esperas entre máquinas sucesivas, etc.

El tercer campo γ, contiene el o los criterios de optimalidad dependiendo de si la programación es multiobjetivo o con un objetivo único. Estos criterios se verán con mayor detalle en el apartado siguiente.

3.8. Criterios de optimización

Habitualmente, es complejo definir el criterio de optimización que fija los objetivos a la hora de buscar soluciones en un problema de programación de la producción. Esto se debe a la gran cantidad de criterios existentes, que generalmente se encuentran en conflicto. Estos criterios permiten clasificar los problemas según el objetivo perseguido a la hora de programar y además cuantificar la bondad de un programa, pudiéndose establecer si un programa es mejor que otro. A continuación se expondrán una serie de criterios utilizados en la industria para dirigir la búsqueda del programa que mejor se ajuste a los objetivos de la empresa. Estos criterios están clasificados según el tipo de medida a la que afectan pudiendo ser criterios basados en los instantes de finalización, criterios basados en las fechas de entrega u otros criterios.

3.8.1. Criterios basados en instantes de finalización

- **Finalización de todos los trabajos** (**makespan**): se trata de optimizar el tiempo de finalización de la última tarea. Es sin duda la versión más estudiada en la literatura por lo que existen numerosas referencias que resuelven el problema de diferentes maneras.

$$C_{max} = \max\{C_i\}$$

- **Flujo total** (flow time): se trata de optimizar la suma de los tiempos de finalización de todos los trabajos. Aunque es una versión muy parecida del problema anterior es más difícil de resolver.

$$F_{max} = \sum C_i$$

- **Tiempo ocioso de las máquinas** (idle time): se trata de minimizar el tiempo ocioso de cada una de las máquinas, es decir los periodos de inactividad de las máquinas que intervienen en el proceso. Minimizar este objetivo equivale a un máximo aprovechamiento de las máquinas.

$$Idle = \sum_{i=1}^{m} Ii$$

Donde:

Ii: tiempo ocioso de la máquina i.

m: número de máquinas.

3.8.2. Criterios basados en la fecha de entrega

- **Retraso** (laeness): se trata de minimizar el retraso de un trabajo. Este objetivo es apropiado si existe un incentivo por entregar el trabajo antes de tiempo. Un retraso positivo significa que el trabajo no ha cumplido correctamente los plazos mientras que un retraso negativo indica que la entrega del trabajo se ha realizado antes de su fecha de vencimiento.

$$L_i = C_i - d_i$$

- **Tardanza** (tardiness): se trata de minimizar la máxima tardanza de todos los trabajos. Este criterio es útil cuando existen penalizaciones por entregar los trabajos fuera de plazo.

$$T_i = \max\{C_i - d_i, 0\}$$

- **Número de trabajos adelantados** (early Jobs): se trata de maximizar el número de trabajos que se finalizan antes de la fecha de vencimiento. Este criterio es útil cuando se premia la entrega de los trabajos antes de tiempo.

$$NEi = \sum_{i=1}^{n} Fi$$

Donde:

Fi: trabajo adelantado i.

n: número de trabajos.

- **Número de trabajos retrasados** (tardy Jobs): se trata de minimizar el número de trabajos que se finalizan después de la fecha de vencimiento. Este criterio es útil cuando se penaliza la entrega de los trabajos fuera de plazo.

$$NUi = \sum_{i=1}^{n} Gi$$

Donde:

Gi: trabajo retrasado i.

n: número de trabajos.

3.8.3. Otros criterios

Los criterios expuestos anteriormente son los más comunes en las empresas fabriles, siendo el más extendido la minimización del tiempo de finalización de todos los trabajos. Sin embargo, dependiendo del tipo de industria y de las prioridades estratégicas de la compañía en la que se realiza la programación de la producción existen multitud de criterios.

Por ejemplo, en numerosas industrias el grueso del coste de operación es debido al consumo de energía, esto deriva en se han desarrollado criterios que minimizan el consumo de energía del conjunto de máquinas de la planta, conocido el consumo de cada máquina y sus tiempos de utilización.

Por otra parte, se han desarrollado criterios basados en costes de inventario y de utilización de los recursos que buscan minimizar el inventario o maximizar la utilización de las máquinas.

Si incluimos los costes de utilización de cada recurso, las penalizaciones o incentivos en unidades monetarias por no cumplir los plazos o entregar los trabajos antes de su fecha de vencimiento, se pueden desarrollar criterios de optimización económica con el fin de maximizar el beneficio de la compañía. Muchos de estos costes no varían linealmente con el tiempo, así que los problemas de programación de la producción basados en criterios económicos no lineales se complican enormemente.

3.9. Cálculo del tiempo de ejecución o makespan

Como hemos visto en el apartado anterior hay diversos criterios de optimización siendo uno de los más interesantes y demandados por la industria la reducción del tiempo de finalización o makespan. En este apartado se calculará el valor del tiempo de finalización para un problema ejemplo dentro de una configuración productiva tipo flowshop flexible. (JIMÉNEZ MORALES, 2012)

Para nuestro problema ejemplo esta configuración dispone de tres etapas con varias máquinas en algunas de ellas como se muestra en la figura 3-6:

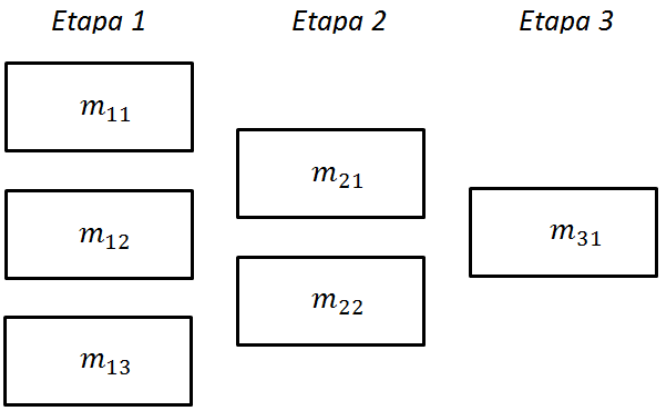

Figura 3 - 6. Configuración productiva "flow-shop" flexible [elaboración propia].

Además se consideran 4 tareas que deben ser procesadas en cada una de las 3 etapas. Cada tarea tiene un tiempo de procesamiento diferente para las distintas etapas. Estos tiempos se muestran en la tabla 3-1 que aparece a continuación.

Etapa / Trabajo	1	2	3
1	10	8	5
2	5	10	10
3	5	5	8
4	8	6	8

Tabla 3 - 1. Tiempos de procesamiento $(\mathbf{p}_{ij})$ de los trabajos en cada etapa (elaboración propia).

Conocidos los datos, el problema de programación de la producción consiste en *encontrar la secuencia óptima de los trabajos que minimicen el makespan o tiempo total de ejecución*.

Como puede verse el problema es de naturaleza combinatoria ya que existen 3 factorial combinaciones, es decir, $3! = 6$. Este hecho hace que la resolución de problemas de programación de la producción sea extremadamente compleja.

Para ilustrar el cálculo del makespan hemos elegido la secuencia S = 3241. En los siguientes diagramas se muestra como van sucediendo las operaciones sobre cada uno de los trabajos.

Al inicio del programa todas las máquinas están desocupadas por lo tanto se asignan los trabajos disponibles en las máquinas disponibles siguiendo la secuencia propuesta. Como tenemos tres máquinas en la etapa uno, hay un trabajo que no puede comenzar hasta que no acabe uno de los tres primeros y se desocupe una de las máquinas de la primera etapa (figura 3 - 7).

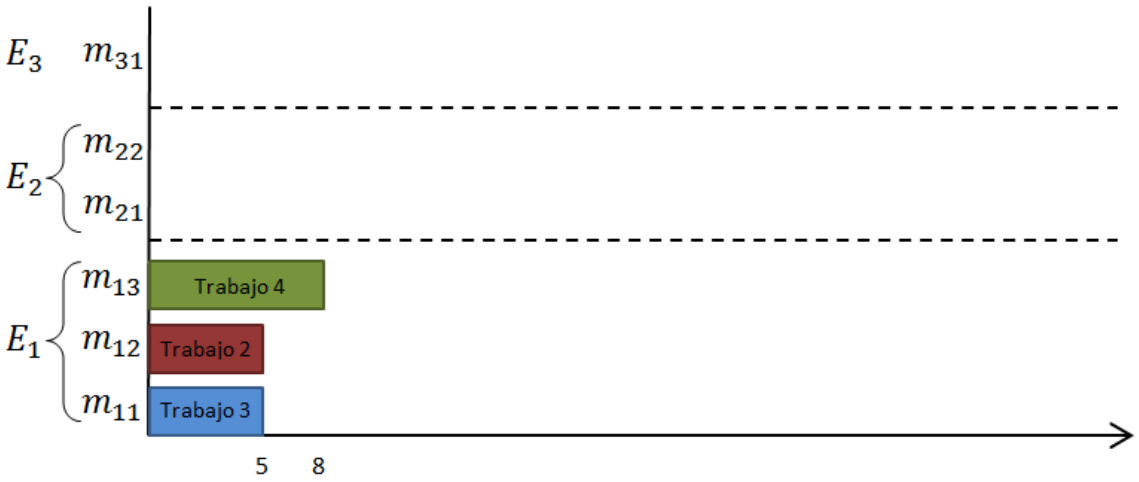

Figura 3 - 7. Fase 1 para el cálculo del Makespan [elaboración propia].

Las tareas 3 y 4 que son procesadas por las máquinas de la primera etapa terminan su procesamiento en t=5 y en este instante pasan a la etapa 2 dado que las máquinas de esta etapa se encuentran disponibles. Además al liberarse las dos máquinas de la etapa 1 entra el trabajo 1 (figura 3 - 8).

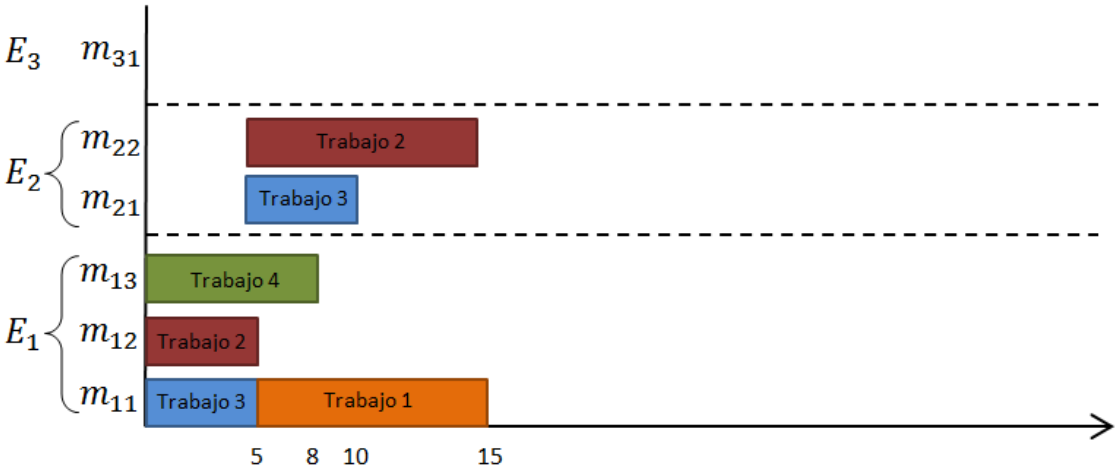

Figura 3 - 8. Fase 2 para el cálculo del Makespan [elaboración propia].

A continuación el trabajo 3 termina su procesamiento en la etapa 2 e inmediatamente pasa a la etapa 3, liberando una de las máquinas de la segunda etapa. Entonces el trabajo 4 que estaba esperando en la etapa 2 pasa a la etapa 3 (figura 3 – 9).

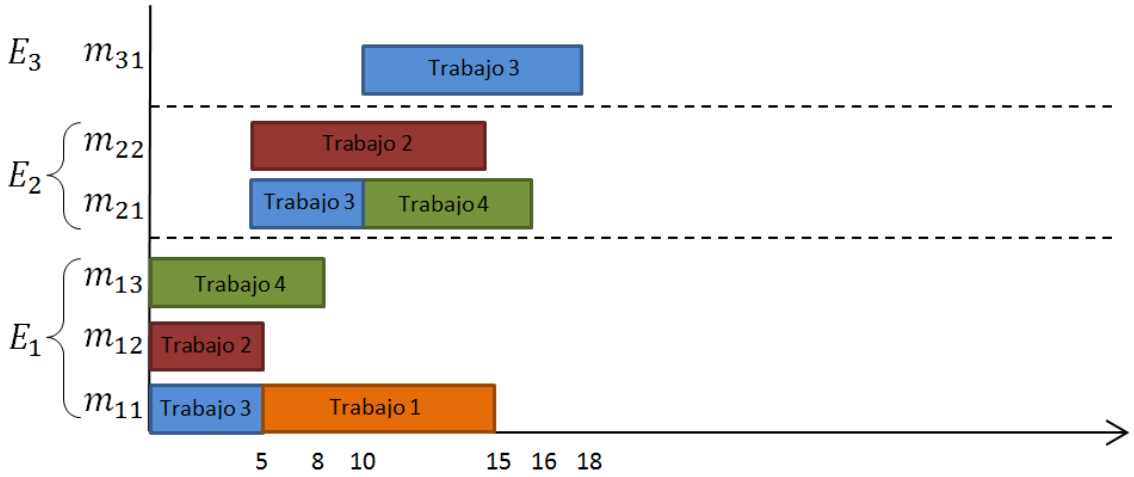

Figura 3 - 9. Fase 3 para el cálculo del Makespan [elaboración propia].

El trabajo 3 termina sus operaciones en la etapa 3 liberando la única máquina existente en esta etapa e inmediatamente el trabajo 2 pasa a la etapa 3. A consecuencia de esto el trabajo 1 puede continuar sus operaciones en la etapa 2 (figura 3 – 10).

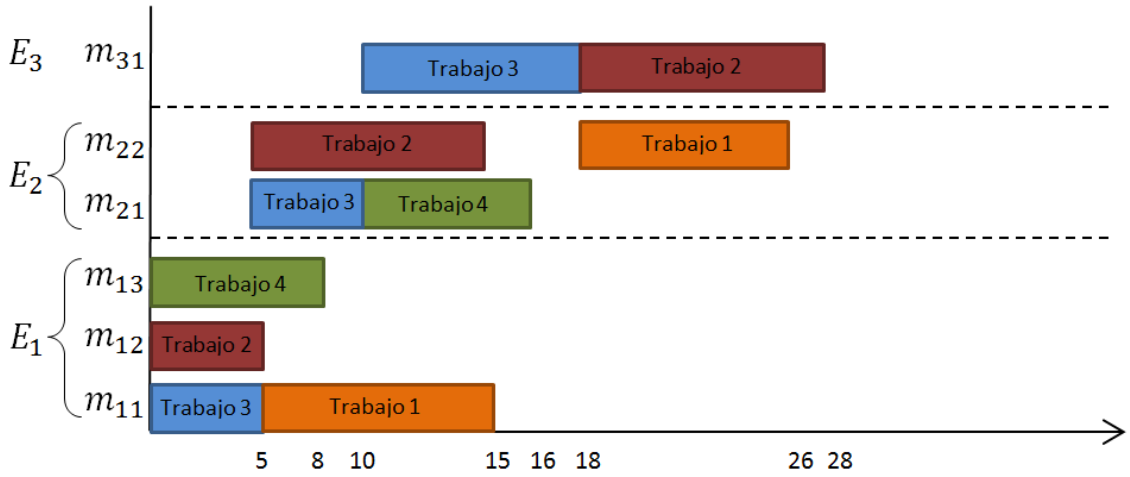

Figura 3 - 10. Fase 4 para el cálculo del Makespan [elaboración propia].

Dado que la configuración productiva con la que estamos trabajando solo dispone de una máquina en la etapa 3 los trabajos se procesarán uno tras otro en el orden establecido.

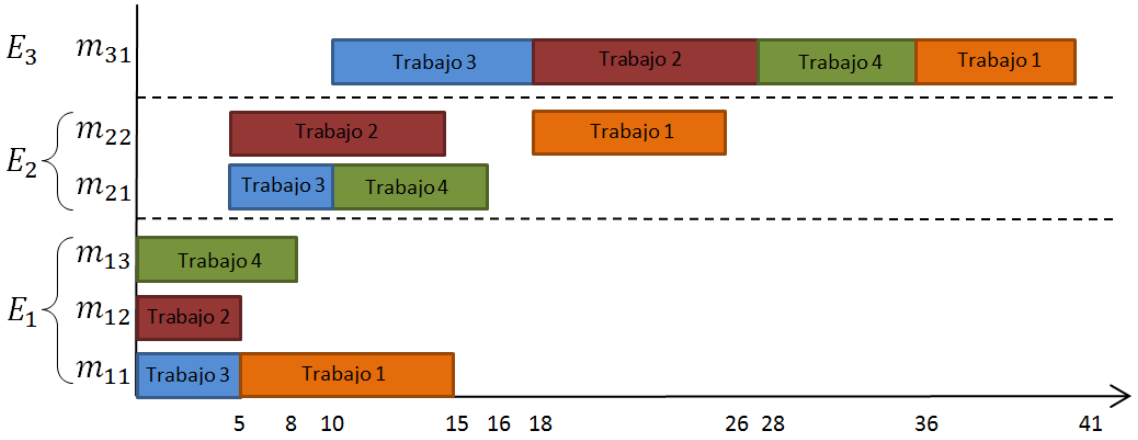

Figura 3 - 11. Fase 5 para el cálculo del Makespan [elaboración propia].

De esta manera obtenemos un tiempo de ejecución o makespan de 41 unidades de tiempo. El objetivo de este trabajo es obtener la secuencia de operaciones para un taller tipo flowshop flexible que minimiza el makespan.

3.10. Carga computacional

La teoría de la complejidad se basa en el estudio de la dificultad intrínseca de resolución de los problemas de optimización y los problemas de decisión (Garey, M. R. & Johnson, D. S. 1979; Lenstra, J. K., Rinnooykan, A. H., & Brucker, P. 1977).

Un problema de optimización es aquel en el que la solución al mismo implica un valor óptimo de la función objetivo, en nuestro caso el criterio de optimización. Cualquiera de estos problemas se reduce a un problema de decisión. Según esta teoría la eficiencia de un algoritmo se mide por el máximo número de pasos de ordenador para obtener la solución óptima.

De esta manera, se pueden clasificar los problemas de optimización en dos clases:

- **Clase P:** contiene los problemas para los que existe un algoritmo de resolución y el número de pasos de cálculo se puede expresar mediante una función polinomial en función del tamaño del problema.

- **Clase NP:** contiene los problemas para los que se puede verificar la factibilidad de una solución determinada mediante un algoritmo en el que el número de los pasos de cálculo se puede expresar mediante una función polinomial en función del tamaño del problema.

Además se dice que un problema es NP-duro si cualquier problema de la clase NP puede transformarse en dicho problema. Por otra parte, un problema es NP-Completo si es de clase NP-duro y además pertenece al conjunto NP.

Los problemas de programación de la producción con restricciones son problemas de optimización combinatoria y pertenecen a la clase NP-Completo y NP-Duro, siendo estos los más difíciles de resolver.

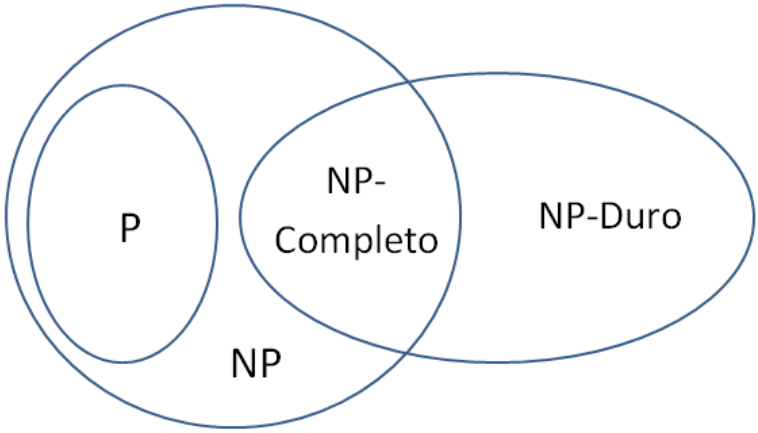

Figura 3 - 12. Clasificación de los tipos de problemas en función de su dificultad [elaboración propia].

En este tipo de problemas, la complejidad de resolverlo aumenta de forma exponencial a medida que lo hace el tamaño del problema, de esta manera cuando la complejidad aumenta exponencialmente todos los algoritmos conocidos tardarían años en resolver el problema. Una característica común a estos problemas es que suelen ser fáciles de plantear pero difíciles de resolver.

3.11. Conclusiones

Resolver un problema de programación de la producción es resolver un problema de optimización combinatoria. La complejidad de resolución de este tipo de problemas es función de una serie de factores como la configuración productiva adoptada, el criterio de optimización elegido o el tamaño del problema.

Existen multitud de configuraciones productivas así como numerosos objetivos de optimización dependiendo de la industria en la que se realice la programación. Esto origina diversos problemas que tienen que ser clasificados de alguna manera, por ejemplo la expuesta en este capítulo α|β|γ.

Este trabajo se centra en la resolución del problema de programación de la producción en un sistema productivo tipo flowshop flexible con el objetivo de optimizar el tiempo de ejecución o makespan.

Debido a la naturaleza combinatoria del problema de programación de la producción, cuando nos enfrentamos a un problema de escala real el coste computacional asociado a este problema es enorme y no admisible en numerosos casos. Por esto la tendencia

actual es buscar métodos y algoritmos que no exploren todo el conjunto de soluciones sino que utilicen mecanismos para dirigir la búsqueda hacia el óptimo.

CAPÍTULO 4:

ESTADO DEL ARTE: MÉTODOS Y ALGORITMOS DE OPTIMIZACIÓN PARA LA GESTIÓN DE RECURSOS

4.1. Introducción

El objetivo que persigue este capítulo es exponer los métodos de resolución de un problema de programación de la producción. Debido a la gran cantidad de métodos propuestos en la literatura consultada es fundamental clasificar éstos de algún modo.

Al comenzar el capítulo se clasificarán los métodos según la solución que obtienen, pudiendo ser métodos exactos o métodos aproximados. Además en este primer apartado se dará una breve descripción de los grupos donde se ubican cada uno de los métodos. Una vez clasificados los métodos de resolución se describirá cada uno de una forma más rigurosa, exponiendo ventajas e inconvenientes de cada uno de ellos. En primer lugar se repasarán los métodos exactos, que son aquellos que arrojan una solución óptima del problema. Dentro de estos métodos existen dos categorías, analíticos y no analíticos. Después describiremos los métodos aproximados, que no garantizan la obtención de una solución óptima del problema pero en numerosos casos obtienen soluciones aceptables de una manera rápida, característica fundamental en muchas industrias. Dentro de los métodos aproximados nos encontramos con tres subclases, las reglas de despacho, los algoritmos heurísticos y los metaheurísticos.

4.2. Clasificación de los métodos

Cualquier problema de programación de la producción trata de alcanzar la mejor solución posible para maximizar o minimizar un criterio determinado, teniendo en cuenta las restricciones impuestas por numerosos factores. Existen numerosas formas de abordar este tipo de problemas. En este apartado se expondrá una clasificación de estas. En el siguiente esquema (figura 4-1) se muestra una clasificación de los métodos de resolución existentes actualmente:

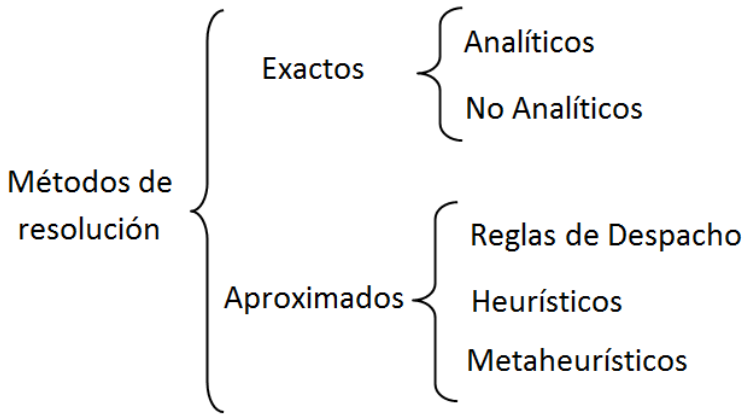

Figura 4 - 1. Clasificación de los métodos de resolución [elaboración propia].

La primera gran división de los métodos y algoritmos de resolución de un problema de programación de la producción se realiza entre métodos exactos y métodos

aproximados. Los métodos exactos son aquellos que garantizan una de las mejores soluciones posibles dado un criterio de optimización, a diferencia de los métodos aproximados que no garantizan que la solución encontrada sea óptima.

Dentro de los métodos exactos no existe una clasificación ampliamente aceptada, pudiendo ser una de ellas la propuesta en el esquema anterior atendiendo a si éstos son analíticos o no. Los métodos analíticos se basan en el análisis de las variables que intervienen en el problema y sus relaciones.

Dentro de los métodos aproximados, una división aceptada, es la que se realiza en base a la arquitectura del método. De esta manera tenemos métodos basados en Reglas de Despacho, métodos Heurísticos y métodos Metaheurísticos. Las Reglas de Despacho permiten definir prioridades entre los trabajos que se encuentran en un taller. No constituyen un método en si ya que solo sirven para definir un criterio de prioridad. Los métodos Heurísticos son procedimientos que proporcionan soluciones factibles aunque no necesariamente óptimas. Por último, los métodos Metaheurísticos utilizan la mezcla de varios procedimientos como Heurísticos y Reglas de Despacho para encontrar una solución viable.

4.3. Métodos Exactos

Los métodos exactos garantizan encontrar una solución óptima para cualquier instancia del problema en un tiempo acotado. Una solución es óptima solo si es la mejor de todas las soluciones posibles, de esta manera, se podría pensar que los métodos exactos son la mejor opción para abordar este tipo de problemas, pero en realidad solo son aplicables a instancias de tamaño muy reducido.

La problemática asociada a estos métodos tiene que ver con el tiempo necesario para alcanzar una solución. Al clasificarse los problemas de Programación de la Producción como NP-Duros el tiempo necesario para resolver el problema crece exponencialmente con el tamaño del mismo, provocando en muchos casos que abordar un problema de estas características sea inviable, dado que el tiempo necesario para encontrar una solución puede ser cientos de años con la capacidad de cálculo actual.

Dentro de los métodos exactos se encuentran la Programación Lineal, la Programación Entera, La programación Dinámica, los métodos de Enumeración Total y los métodos de Ramificación y Poda que pasaremos a describir a continuación.

4.3.1. Programación Lineal

La Programación Lineal es un método relativamente reciente (siglo XX), que consiste en una serie de procedimientos que permiten resolver problemas de optimización donde se exige maximizar o minimizar funciones que se encuentran sujetas a determinadas limitaciones o restricciones. (Mateo & Lahoz, 2009)

De forma más rigurosa, la Programación Lineal consiste en optimizar (maximizar o minimizar) una función llamada función objetivo, que es una función lineal de varias variables. Además la optimización está sujeta a una serie de restricciones expresadas en forma de inecuaciones lineales.

Cuando en los problemas intervienen más de dos variables, su resolución se complica y se resuelven por el llamado método Simplex, ideado por el matemático estadounidense (Danzig, G.B., 1951). Este método consiste en un algoritmo iterativo que secuencialmente se va acercando a la solución óptima.

Los modelos de Programación Lineal constituyen la forma tradicional de abordar los problemas de programación de la producción. Son varios los autores que repasan el problema de programación en máquinas mediante este método.

4.3.2. Programación Entera

La Programación Entera es aquella donde la totalidad o un subconjunto de las variables de decisión toman valores enteros. Al igual que en la programación lineal existe una función objetivo que se desea optimizar (maximizar o minimizar) sujeta a una serie de restricciones. Estos métodos nos permiten modelar muchas más situaciones que la programación lineal, pero a cambio, la resolución de este tipo de problemas es más compleja y conlleva mayor coste computacional.

Dentro de la Programación Entera podemos encontrarnos diferentes tipos de problemas atendiendo al tipo de variables del problema. De esta manera tenemos:

- Problemas enteros puros: todas las variables únicamente pueden tomar valores enteros.

- Problemas mixtos: algunas variables toman valores enteros y al mismo tiempo existen variables continuas.

- Problemas binarios: las variables solo pueden tomar dos valores cero o uno.

Es habitual encontrar autores que utilizan la programación entera para resolver un problema de programación en máquinas.

4.3.3. Programación Dinámica

La Programación Dinámica trata de reducir el tiempo de cálculo mediante la división del problema en distintos subproblemas que se resuelven por separado. Estos subproblemas se resuelven una sola vez y sus soluciones se guardan para su posterior utilización. (Guerequeta & Vallecillo, 2000)

En la Programación Dinámica primero se calculan las soluciones óptimas para los diferentes subproblemas de tamaño pequeño y luego utilizando dichas soluciones, se encuentra la solución al problema completo. Para evitar calcular dos veces la solución a un subproblema, ésta se almacena y antes de resolver un subproblema se comprueba si la solución ha sido obtenida previamente.

Es habitual utilizar la Programación Dinámica para resolver distintos problemas dentro de la industria como son la gestión de inventarios y la programación de la producción.

4.3.4. Enumeración Total

La Enumeración Total es uno de los métodos más sencillos para abordar un problema de programación de la producción. Este método consiste en enumerar todas las soluciones posibles al problema y evaluarlas. Es un método poco eficiente ya que para una instancia del problema de un tamaño considerable la complejidad computacional asciende a niveles no permisibles debido al tiempo de cómputo necesario para resolverlo.

4.3.5. Ramificación y Poda

El método exacto más relevante es el algoritmo de Ramificación y Poda (Branch and Boud). Este método fue propuesto por Bruker en 1994 (Bruker, 1994) desarrollado a partir de conceptos y técnicas propuestas por otros investigadores.

En estos algoritmos el espacio de las soluciones se divide en ramas (fase de Ramificación) y cada rama se evalúa, investigando solamente las ramas susceptibles de llegar a una solución óptima. Aunque por el camino se van descartando soluciones, el orden de estos métodos es exponencial lo que deriva en un tiempo de cálculo excesivo cuando el problema es de un tamaño considerable.

En un algoritmo de Ramificación y Poda se realizan tres etapas:

- Selección: se extrae un nodo del conjunto. La forma de escogerlo dependerá de la estrategia de búsqueda que escojamos para el algoritmo.

- Ramificación: se construyen los posibles nodos hijos derivados del nodo que se escogió en el paso anterior.

- Poda: se eliminan algunos de los nodos creados en la etapa anterior ya que éstos no llevarán a soluciones óptimas. Esto contribuye a disminuir el espacio de búsqueda y así atenuar la complejidad de estos algoritmos.

Una de las características de valor de esta técnica es la posibilidad de disponer de distintas formas de exploración del árbol definidas por el usuario y acotar la búsqueda de la solución, aportando eficiencia al algoritmo.

Además una de las ventajas de este algoritmo es la posibilidad de ejecutar varios en paralelo, teniendo varios procesos extrayendo nodos del conjunto, ejecutando su ramificación y luego su poda. Esta posibilidad es muy importante a la hora de resolver un problema de programación de la producción en un tiempo razonable debido a la complejidad intrínseca del problema.

4.4. Métodos Aproximados

Los métodos aproximados sacrifican la garantía de encontrar una solución óptima a cambio de encontrar una solución de calidad en un tiempo razonable. Estos métodos están recibiendo una atención cada vez mayor dentro de la comunidad científica internacional en los últimos años.

Dentro de los métodos aproximados la primera y gran división se realiza en función de la arquitectura del método en sí. De esta manera, nos encontramos con tres clases: las Reglas de Despacho, los métodos Heurísticos y los métodos Metaheurísticos.

4.4.1. Reglas de Despacho

Las Reglas de Despacho no constituyen un método en sí mismas ya que solo sirven para establecer unas reglas de prioridad. Estas reglas suelen ser utilizadas dentro de un procedimiento más complejo, como puede ser un método Heurístico o Metaheurístico, así que conviene conocerlas.

En el problema concreto de programación de la producción estas reglas tiene una importancia vital, ya que la mayoría de algoritmos tienen una fase encargada de seleccionar el trabajo que luego será secuenciado. Es en la fase de selección de un trabajo o un recurso donde estas reglas juegan un papel importante.

Como se ha dicho, las Reglas de Despacho establecen un criterio de prioridad, estas reglas pueden ser sencillas, basadas en un atributo del trabajo o más complejas, basadas en un cálculo como la holgura.

Algunas Reglas de Despacho comunes son:

- **FIFO** (First In First Out): primero en entrar, primero en atender. Se emplea a menudo, por ejemplo cuando tratamos con productos perecederos.

- **LIFO** (Last In First Out): último en entrar, primero en atender. No es una regla muy común pero en ocasiones es utilizada, por ejemplo, cuando el material ocupa grandes superficies.

- **SIRO** (Service In Random Order): bajo esta regla no se sigue ningún criterio de prioridad.

- **STP** (Shortest Processing Time): Ordena los trabajos de menor a mayor tiempo de procesamiento, es decir, ejecuta primero el trabajo más corto. Es una de las reglas más utilizada en la industria. De esta manera y con la notación definida en el capítulo anterior los trabajos quedan ordenados de la siguiente forma:

$$p_1 \leq p_2 \leq \cdots \leq p_n$$

- **LTP** (Longest Processing Time): Ordena los trabajos de mayor a menor tiempo de procesamiento, es decir, ejecuta primero en trabajo más largo. Los trabajos quedan ordenados de la siguiente forma:

$$p_1 \geq p_2 \geq \cdots \geq p_n$$

- **EDD** (Earliest Due Date): Ordena los trabajos en función de la fecha de vencimiento de forma creciente, es decir, se ejecuta primero el trabajo con menor fecha de entrega o trabajo más urgente. Los trabajos quedan ordenados de la siguiente forma:

$$d_1 \leq d_2 \leq \cdots \leq d_n$$

- **ECT** (Earliest Completion Time): Ordena los trabajos en función de la fecha de terminación de forma creciente, es decir, se ejecuta primero los trabajos que terminan antes. Los trabajos quedan ordenados de la siguiente forma:

$$C_1 \leq C_2 \leq \cdots \leq C_n$$

- **Holgura Mínima:** Ordena los trabajos en función de su holgura de forma creciente, es decir, se ejecutan primero los trabajos con mayores posibilidades de retrasarse. Los trabajos quedan ordenados de la siguiente forma:

$$H_1 \leq H_2 \leq \cdots \leq H_n$$

4.4.2. Métodos Heurísticos

Los métodos Heurísticos proporcionan soluciones factibles aunque no necesariamente óptimas. Estos algoritmos buscan alcanzar una solución de calidad en el menor tiempo posible y es común su aplicación cuando los problemas son de alta complejidad. Su estrategia consiste en no considerar algunos aspectos del problema de forma deliberada para así reducir el espacio de soluciones y por lo tanto el tiempo de resolución.

Al principio, estos métodos no fueron bien vistos en determinados círculos académicos, debido fundamentalmente a su escaso rigor matemático, sin embargo, gracias a su interés práctico para solucionar problemas reales, normalmente complejos, estos métodos fueron poco a poco aceptados.

Dentro del campo de la programación de la producción los métodos Heurísticos han encontrado un amplio campo de aplicación. Estas son algunas de las circunstancias que han impulsado su desarrollo:

- No se puede aplicar un método exacto a la resolución de un problema, ya sea porque no existe o porque requiere demasiados recursos como tiempo de cálculo o memoria.

- El problema no requiere una solución exacta, sino solo una aproximación de calidad.

- Los datos disponibles son poco fiables o sujetos a incertidumbre.

- El método Heurístico es un paso intermedio en la aplicación de otros algoritmos, siendo la solución de este método la entrada de otro procedimiento.

Son varios los autores que han propuesto una clasificación de los métodos Heurísticos aunque estas clasificaciones difieren dependiendo de la bibliografía consultada. Una de ellas es la siguiente:

- **Métodos de descomposición:** consisten en dividir el problema en subproblemas más pequeños siendo la salida de uno la entrada de otro. La solución global se obtiene al resolver todos los subproblemas.

- **Métodos de reducción:** consisten en simplificar el problema identificando alguna característica que deba poseer la solución.

- **Métodos de manipulación del modelo:** consisten en modificar la estructura del modelo con el fin de simplificarlo. Una vez se obtiene la solución al problema modificado es posible inferir la solución del problema real.

- **Métodos de búsqueda local:** consisten en partir de una solución inicial y mediante alteraciones en esta van iterando hasta conseguir una solución que cumpla una determinada condición de parada. Estos métodos pueden ofrecer una medida de la bondad de la solución obtenida y tienen la ventaja de que en un tiempo relativamente pequeño pueden obtener soluciones suficientemente buenas.

- **Métodos basados en cuellos de botella**: consisten en relajar un problema de N máquinas en N problemas de 1 máquina y resolver cada subproblema de forma iterativa. Cada una de las soluciones se compara con las demás y se ordenan las máquinas dependiendo de su solución, siendo la máquina con mayor solución la que provoca el cuello de botella.

Normalmente estos algoritmos están diseñados para resolver un tipo de problema muy concreto y por lo tanto no tienen aplicación genérica ya que pueden proporcionar soluciones no factibles para determinadas instancias del problema.

4.4.3. Métodos Metaheurísticos

Los métodos Metaheurísticos son algoritmos de alto nivel que se utilizan para cohesionar un conjunto de acciones mezcla de otros procedimientos como Heurísticos, Reglas de Despacho, métodos exactos aplicados a partes del problema, etc. Estos métodos son menos específicos que los métodos Heurísticos y normalmente tienen el objetivo de resolver un tipo específico de problemas.

Los Métodos Metaheurísticos proporcionan unos pasos que el usuario debe seguir para resolver el problema. Además estos métodos disponen de una serie de parámetros para ajustar la aplicación de los métodos a un problema concreto, así como espacios donde el usuario puede insertar sus propuestas, normalmente Heurísticas.

Existen múltiples propuestas, muchas de ellas inspiradas en la naturaleza y muchas otras en técnicas de Inteligencia Artificial. A continuación se describirán algunos de los métodos a los que se ha tenido acceso y que han sido aplicados al problema de Programación de la Producción.

4.4.3.1. Recocido Simulado

Los algoritmos de Recocido Simulado se basan en una analogía con el proceso de recocido de los metales que fue modelado por (Metropolis, Rosenbluth, 1953). El recocido implica el calentamiento del metal a altas temperaturas para luego dejarlo enfriar gradualmente hasta que éste alcance un estado de mínima energía. A partir de este procedimiento otros autores resuelven un problema de optimización combinatoria que aparece en el diseño de circuitos impresos (Kirkpatrick, Gelatt, Vecchi, 1983). Es uno de los métodos más aplicados en la resolución de problemas de optimización combinatoria.

El proceso general de un algoritmo basado en el Recocido Simulado es el siguiente:

- **Etapa de inicio:** en el comienzo deben establecerse una serie de parámetros como son:

    - Una solución inicial
    - Una temperatura inicial
    - Un número de iteraciones
    - Un índice de enfriamiento

    La temperatura controla la posibilidad de aceptación de una solución, de esta manera, la temperatura inicial debe ser lo suficientemente alta para que todos los estados sean susceptibles de ser visitados.

- **Generación de una solución alternativa vecina:** la vecindad puede ser definida como el conjunto de soluciones que pueden crearse a partir de la modificación o cambio de algún elemento de la estructura de la solución actual. De esta manera se genera una solución alternativa modificando levemente la solución inicial.
- **Evaluación de la nueva solución:** siempre que el valor de la función objetivo mejore la solución se acepta y pasa a ser la solución actual. En caso de no mejorarse la función objetivo, podrá aceptarse la solución en función de una probabilidad de transición y la temperatura actual.

    En las primeras etapas del método la mayoría de los movimientos son aceptados puesto que el proceso comienza a una temperatura alta, pero a medida que avanza la búsqueda la temperatura se reduce y también la posibilidad de aceptar nuevas soluciones.
- **Actualización:** una vez evaluada la solución se reduce la temperatura del sistema conforme al índice de enfriamiento introducido por el usuario. Un enfriamiento lento permite una exploración más intensa del espacio de soluciones.
- **Cierre:** el criterio de parada es establecido por el usuario y puede establecerse teóricamente cuando la temperatura converge a cero. Es típico detener el algoritmo cuando la función objetivo no mejora en ciertas iteraciones.

Como vemos este algoritmo es controlado por el proceso de enfriamiento, además la bondad de la solución obtenida y el coste computacional es función de los parámetros iniciales del algoritmo.

Una de las características llamativas de este método es que permite el movimiento a soluciones de peor calidad lo que le da al procedimiento la posibilidad de salir de óptimos locales y poder encontrar una mejor solución en una etapa posterior.

4.4.3.2. Búsqueda Tabú

La búsqueda tabú es un método Metaheurístico de resolución de problemas de optimización propuesto por (Glover, 1980). Este algoritmo se considera una técnica de búsqueda local ya que explora la vecindad encontrando óptimos locales.

El procedimiento comienza con una solución inicial, obtenida por otros métodos, normalmente heurísticos, y busca en el vecindario de esta solución una que presente mejor rendimiento. Cuando esta solución aparece la búsqueda se mueve a la vecindad de la nueva solución y se repite el proceso de forma iterativa hasta que alguna condición de parada se satisfaga.

Una de las claves para que este algoritmo funcione correctamente es implementar mecanismos que impidan que la búsqueda se quede atrapada en un óptimo local. De esta manera se introducen en el algoritmo algunos movimientos prohibidos que no podrán aplicarse en un momento dado. Además es fundamental determinar un tamaño de la vecindad adecuado ya que mientras más grande sea ésta más complejo será obtener el óptimo local.

La búsqueda tabú considera prohibidos todos los movimientos hacia configuraciones que contengan atributos seleccionados en el pasado reciente, de esta manera todas las configuraciones que posean alguno de los atributos prohibidos (tabú) son excluidas de la formación de la nueva vecindad. Así el algoritmo evita volver a configuraciones ya visitadas y ampliar la búsqueda del algoritmo.

La búsqueda tabú hace uso de distintos tipos de memoria empleados para seleccionar el mejor movimiento en cada iteración. Normalmente esta memoria hace referencia a cuatro dimensiones o propiedades: ser recientes, frecuencia, calidad e influencia. Las memorias relativas a lo reciente y la frecuencia son utilizadas para diversificar la búsqueda. La dimensión calidad hace referencia a la capacidad del algoritmo para diferenciar la bondad de dos soluciones. Por último la influencia tiene que ver con el impacto del movimiento en el algoritmo.

La principal ventaja de este algoritmo es que tiene la característica de utilizar procedimientos deterministas y no aleatorios a la hora de moverse a un nuevo óptimo local. Estos procedimientos deterministas hacen uso de la memoria anteriormente descrita para decidir cuál es el movimiento más adecuado en cada iteración.

El uso de memoria y procedimientos deterministas son las principales diferencias con otros de los Metaheurísticos expuestos en este trabajo como el recocido simulado.

4.4.3.3. Colonia de Hormigas

Este procedimiento Metaheurístico está basado en el comportamiento real de las hormigas para encontrar el camino más corto de su fuente de alimento hasta el

hormiguero. Cuando las hormigas buscan alimento segregan una feromona que dejan en sus senderos. Cuanto mayor sea el número de hormigas que caminen por un sendero mayor será la cantidad de feromonas que éste contiene. Debido a que la siguiente hormiga escogerá el camino con una probabilidad proporcional a las feromonas que contiene se desarrollará un camino común desde la fuente de alimento al hormiguero. (SALAZAR PINTO, 2009)

Para modelar el comportamiento de las hormigas en un computador deben definirse los siguientes componentes del algoritmo:

- Representación adecuada de la feromona.
- Mecanismo de actualización de la cantidad de caminos.
- Función específica encargada de informar del problema específico.

La selección adecuada de estos componentes es crucial para el buen funcionamiento del sistema ya que influyen en la toma de decisiones y afectan directamente al rendimiento del sistema.

Al inicio del procedimiento toda la colonia de hormigas se sitúa en el nodo de origen (hormiguero) definiéndose un número arbitrario de hormigas en la colonia. En cada iteración se usan una cantidad concreta de hormigas para construir la solución. Una vez construida ésta se aplican mecanismos de actualización como la evaporación de las feromonas y la cantidad de feromona añadida al camino.

Esta Metaheurística es una de las más aplicadas a problemas de optimización y está siendo mejorada y extendida. El algoritmo se ha usado recientemente para resolver problemas de programación de la producción con éxito.

4.4.3.4. Algoritmos Voraces

Los Algoritmos Voraces o GRASP, siglas de Greedy Randomized Adaptative Search Procedures, son una técnica que permite encontrar soluciones de buena calidad a problemas de optimización con origen en los años 80. Este procedimiento concentra sus esfuerzos en obtener soluciones de alta calidad que posteriormente son procesadas para obtener otras aun mejores. (MARQUEZ DELGADO, 2012)

El algoritmo es de tipo iterativo en los que cada iteración incluye dos fases, en la primera se construye una solución inicial y en la segunda se optimiza la solución generada en la primera fase.

Mientras no se satisfaga el criterio de parada, en cada iteración, se construye una solución greedy aleatoria, posteriormente se aplica un procedimiento de búsqueda local para mejorar la solución generada en el paso anterior y por último se actualiza la solución mejorada para comenzar la siguiente iteración.

Para lo construcción de soluciones greedy aleatorias se utiliza una lista de candidatos restringida RLC (Restricted Candidate List) en la que se incluyen los candidatos a formar parte de la solución. De esta lista se escoge uno aleatoriamente.

4.4.3.5. Algoritmos Evolutivos

Los Algoritmos Evolutivos simulan la evolución natural y constituyen un enfoque alternativo para resolver problemas complejos de optimización mediante modelos computacionales de procesos evolutivos. La simulación de procesos de evolución natural de las especies tiene como resultado una técnica de optimización estocástica donde se ubican los Algoritmos Evolutivos.

Estos algoritmos trabajan con una población de individuos que representan posibles soluciones al problema. Esta población es sometida a ciertas transformaciones y después a un proceso de selección que favorece a los mejores. De cada iteración surge una nueva generación de individuos que vuelve a ser transformada y seleccionada. Después de un cierto número de generaciones se espera que alguno de los individuos de la generación final esté cerca de la solución óptima.

Los Algoritmos Evolutivos combinan la búsqueda aleatoria, dada por las transformaciones de los individuos con una búsqueda dirigida, dada por los mecanismos de selección. (MARQUEZ DELGADO, 2012)

Los principales componentes de un Algoritmo Evolutivo son:

- Población de Individuos: representación de las posibles soluciones.
- Procedimiento de selección: basado en la calidad o aptitud de los individuos para resolver el problema.
- Procedimiento de transformación: proceso de construcción de nuevos individuos a partir de los anteriores.

Un Algoritmo Evolutivo parte de un conjunto de soluciones iniciales que va siendo transformada por un conjunto de operadores de búsqueda que refinan la población hasta llegar a la solución final. Para refinar el conjunto de soluciones se utilizan desde técnicas clásicas como el seguimiento del gradiente hasta técnicas inspiradas en la biología.

4.4.3.6. Algoritmos Genéticos

Los Algoritmos Genéticos son una técnica de optimización estocástica inspirada en la naturaleza que emplea los conceptos de la selección natural para refinar el conjunto de soluciones. (Arranz de la Peña & Parra Truyol, 2007)

Estos algoritmos trabajan sobre un conjunto de potenciales soluciones denominado *población*. Dicha población está compuesta por una serie de soluciones denominadas *individuos* que a su vez están compuestos por una serie de posiciones, que representan las variables que intervienen en el problema, denominadas *cromosomas*. Estos cromosomas están compuestos por una cadena de números que normalmente es representada en números binarios.

Las estrategias de evolución actúan sobre los individuos que representan las posibles soluciones del problema. Estos individuos *evolucionan* a través de *generaciones*. Dentro de la población los individuos se diferencian de acuerdo a su nivel de *aptitud* que es obtenido usando algunas medidas de acuerdo al problema a resolver. Para la creación de nuevas poblaciones se crean nuevos individuos llamados *hijos* de acuerdo a varios mecanismos de evolución como son el cruce y la mutación.

Una vez que la población ha evolucionado se aplican mecanismos de selección de acuerdo a la aptitud de los individuos frente al problema. Existen numerosos métodos de selección que tienen en cuenta la aptitud como la selección proporcional a la aptitud, selección por ruleta, etc.

La aptitud media de la población se incrementa en cada generación y por tanto iterando un número adecuado de veces pueden obtenerse soluciones factibles y de alta calidad. Dicho de otro modo, tras varias iteraciones el algoritmo converge al individuo con mejor valor de aptitud, siendo éste el óptimo o subóptimo del problema.

4.5. Conclusiones

Como se ha podido constatar existen multitud de métodos y algoritmos capaces de resolver el problema de optimización ligado a la programación de la producción. Estos métodos pueden ser clasificados como exactos y aproximados.

Los métodos exactos obtienen una solución óptima del problema en un tiempo acotado pero dada la naturaleza combinatoria del problema exigen tiempos inasumibles por la industria cuando nos enfrentamos a un problema real y no a una simplificación del mismo.

Los métodos aproximados obtienen soluciones de alta calidad en tiempos razonables, considerablemente menores que los métodos exactos. Estos métodos aproximados

hacen uso de heurísticas o reglas para reducir tanto el problema como el espacio de las posibles soluciones a este.

Por último existen procedimientos metaheurísticos que combinan lo mejor de otros procedimientos como métodos exactos y heurísticas para llegar a una solución óptima del problema. Dentro de los métodos metaheurísticos tienen gran importancia los algoritmos inspirados en la naturaleza o bio-inspirados como son los algoritmos genéticos. Estos algoritmos implementan en un computador procedimientos similares a los que vemos en los seres vivos.

Los algoritmos genéticos tienen un gran potencial debido a que combinan la búsqueda aleatoria de la solución con la búsqueda dirigida, haciendo uso de varios mecanismos inspirados en la selección natural. Este hecho hace que se obtengan soluciones de alta calidad en tiempos relativamente cortos a cualquier problema de optimización combinatoria como es el de programación de la producción. Es por todo ello que se ha escogido este método para la realización de este Trabajo de Fin de Grado.

CAPÍTULO 5:

FUNDAMENTOS TEÓRICOS DE LOS ALGORITMOS GENÉTICOS

5.1. Introducción

Este capítulo tiene como principal objetivo introducir la teoría sobre la que se asientan los Algoritmos Genéticos. Un Algoritmo Genético es una Metaheurística inspirada en la naturaleza utilizada para resolver problemas de optimización, como por ejemplo, el problema de programación de la producción tratado en este trabajo.

En primer lugar se definirá formalmente un Algoritmo Genético y se expondrán los principios básicos, inspirados en la naturaleza, sobre los que funcionan estos algoritmos. Una vez dada esta definición se expondrá una terminología común a esta metaheurística que será usada a lo largo de este trabajo. Después se repasarán las posibles codificaciones del problema y finalmente se describirá el proceso evolutivo que rige estos algoritmos formado por diferentes etapas como evaluación, selección, cruce y mutación.

5.2. Definición y principios de los Algoritmos Genéticos

Un Algoritmo Genético es un algoritmo comúnmente usado para resolver problemas de optimización combinatoria de forma secuencial o iterativa basado en el principio de la selección natural enunciado por Darwin. (Arranz de la Peña & Parra Truyol, 2007)

Según la teoría de Darwin solamente los individuos más aptos, en nuestro caso soluciones al problema, prosperan y tienen la oportunidad de transmitir algunas de sus características a sus descendientes. Después de varias generaciones la aptitud de los individuos que forman la población es mayor que la de las generaciones antiguas.

Algunos de los principios que rigen la selección natural de las especies y en los que se basan este tipo de algoritmos son los siguientes:

- La **evolución** se da en los cromosomas y no en los individuos.
- La **selección natural** es el proceso por el cual los cromosomas con buena aptitud se reproducen más que otros.
- La **reproducción** se lleva a cabo intercambiando cromosomas procedentes de los progenitores.
- Existen **mutaciones** aleatorias que modifican el material genético de una generación a otra.
- En la **evolución** biológica los cromosomas no tienen memoria, es decir solo se consideran la información de la generación anterior.

Utilizando estas premisas observadas en la naturaleza y replicándolas mediante algoritmos en un ordenador se consiguen resolver problemas de optimización de una manera eficiente en tiempos razonables.

5.3. Terminología

En apartado se definirán una serie de términos utilizados comúnmente cuando se trabaja con Algoritmos Genéticos. Estos términos tienen su origen en la biología y en nuestro caso representan una parte del problema a resolver. De esta manera, algunos de los términos utilizados son los siguientes:

- **Población:** conjunto de potenciales soluciones sobre las que trabaja el algoritmo genético.

- **Individuo:** una de las soluciones al problema. La población está compuesta por un conjunto de individuos.

- **Cromosoma:** posiciones que conforman un individuo y que representan alguna de las variables involucradas en el problema de optimización.

- **Aptitud:** puntuación de adaptación dependiendo de la calidad de respuesta al problema que pretende resolver el algoritmo.

- **Generación:** conjunto de individuos presentes en cada iteración del algoritmo.

- **Padres:** individuos de los que desciende la nueva generación transmitiendo sus características a sus descendientes.

- **Hijo:** nuevo individuo obtenido por mecanismos evolutivos descendiente de dos individuos de la generación anterior.

Una vez dadas estas definiciones se puede decir que una población de individuos evoluciona a través de generaciones. Para la obtención de las siguientes generaciones se crean nuevos individuos (hijos) utilizando mecanismos evolutivos. Dentro de la población cada individuo es diferenciado del resto por su valor de aptitud.

5.4. Codificación del problema

Los cromosomas de alguna manera deben contener información del problema que representan. Esta información puede almacenarse de varias maneras siendo una de las más utilizadas, aunque no la única, la codificación binaria. También pueden utilizarse codificaciones basadas en números enteros o incluso cadenas de caracteres.

Elegir una codificación u otra dependerá del tipo de problema que queramos resolver, dado que elegir una mala codificación complicará la resolución del problema enormemente. Es por esto que ante cualquier problema que se quiera resolver mediante Algoritmos Genéticos se debe estudiar la codificación más óptima. A continuación se expondrán las codificaciones habituales.

5.4.1. Codificación Binaria

Como hemos dicho anteriormente, la codificación binaria es la más extendida dado que los primeros algoritmos genéticos utilizaron este tipo de codificación. En este caso el cromosoma está formado por una cadena de bits (0 o 1).

0	0	1	0	1	1	0	1	0	1

Tabla 5 - 1. Codificación Binaria [elaboración propia].

Aunque la más extendida, esta opción no es la adecuada cuando nos enfrentamos a numerosos problemas y en algunas ocasiones es necesario hacer correcciones después de los mecanismos evolutivos como la reproducción o la mutación.

5.4.2. Codificación Entera

Este tipo de codificación utiliza cadenas de números enteros que representan una solución al problema. Es muy útil cuando nos enfrentamos a problemas donde se requiere ordenar algo, como por ejemplo en la resolución del problema de programación de la producción que se trata en este trabajo.

Al igual que en el caso anterior pueden darse situaciones donde después de alguno de los mecanismos evolutivos haya que realizar correcciones a la cadena generada.

8	2	3	5	7	4	0	1	6	9

Tabla 5 - 2. Codificación Entera [elaboración propia].

5.4.3. Codificación por Valor Directo

Esta codificación es utilizada en el caso de problemas en los que se requiera el uso de valores de cifrado complicados, como por ejemplo números reales, cuya codificación de forma binaria resultaría muy compleja.

En la codificación por valor directo cada cromosoma es una cadena de valores relacionados con el problema a estudiar pudiendo ser desde una cadena de números reales con decimales, una cadena de caracteres o una mezcla de ellos.

La aplicación de esta codificación es muy ventajosa en problemas concretos pero por el contrario han de desarrollarse nuevos métodos de reproducción y mutación para cada problema.

0.21	2.54	3.16	5.62	7.95	4.01	0.52	1.98	6.17	9.50

Tabla 5 - 3. Codificación por Valor Directo con números reales [elaboración propia].

C	D	H	A	J	E	H	G	B	F

Tabla 5 - 4. Codificación por Valor Directo con caracteres [elaboración propia].

5.5. Obtención de la población inicial

La primera tarea del algoritmo es generar una población inicial de individuos, codificados de alguna de las formas vistas en el apartado anterior, sobre las que operaran los distintos mecanismos evolutivos.

Existen diferentes técnicas para generar esta población que van desde la generación aleatoria hasta la utilización de alguna heurística específica del problema que queremos resolver. La manera más simple e intuitiva de generar esta población es seleccionar cada individuo de manera aleatoria hasta completar toda la población.

La diversidad de la población inicial puede afectar de manera significativa al comportamiento del algoritmo, así que deberá tenerse especial cuidado en el inicio. Además la población debe ser lo suficientemente grande como para que el algoritmo opere eficazmente sobre la población y se mantenga la diversidad.

5.6. Proceso Evolutivo

En este apartado se expondrá cuáles son los mecanismos que rigen un Algoritmo Genético así como las diferentes posibilidades de implementación que existen para estos mecanismos evolutivos.

Una vez decidida la codificación más adecuada para representar el problema que queremos resolver y generada la población inicial el algoritmo trabaja sobre esta población con los siguientes mecanismos:

1. **Proceso de Evaluación:** consiste en calcular la aptitud de cada uno de los individuos frente al problema.
2. **Proceso de Selección:** consiste en seleccionar los individuos más aptos que serán los que tengan mayor probabilidad de reproducirse.
3. **Proceso de Reproducción:** consiste en generar una descendencia que lleve consigo las características de los individuos seleccionados.
4. **Proceso de Mutación:** consiste en mutar un cromosoma del nuevo individuo generado con cierta probabilidad.

Cada uno de estos procesos serán detallados en los siguientes apartados.

5.6.1. Evaluación

El proceso de evaluación consiste en calcular la función objetivo o aptitud del individuo frente al problema para saber la calidad de la solución representada. Como se puede observar esta función depende del criterio de optimización elegido.

Este proceso, una vez evaluadas las soluciones, asigna valores de aptitud frente al problema a los diferentes individuos. Estos valores deben ser veraces para discriminar correctamente los individuos. Si los valores de aptitud fuesen asignados erróneamente donde el valor de la aptitud de una solución buena es muy cercano al valor de aptitud de una solución mala sería necesario mucho tiempo para que por medio de la reproducción los descendientes con buenas características influyan en los malos.

Al diseñar una función de evaluación se debe tener especial cuidado en determinar las restricciones del problema y como se penalizan a los individuos que violan éstas para evitar la generación de individuos no factibles.

5.6.2. Selección

Una vez calculada la aptitud de los individuos frente al problema es necesario seleccionar los individuos más capacitados para que éstos sean los que se reproduzcan con mayor probabilidad. De acuerdo a la teoría de Darwin los individuos más capacitados o aptos son los que sobreviven y crean una mejor descendencia.

De esta manera la descendencia que se crea a partir de los individuos seleccionados está mejor preparada frente al problema que queremos optimizar.

Para realizar la selección existen varios métodos que serán expuestos a continuación.

5.1.1.1. Selección por Rueda de Ruleta

En este método se crea una ruleta con los individuos presentes en una generación. Cada individuo tendrá una parte mayor o menor de esa ruleta en función de la aptitud que tenga cada uno. Se hace girar la ruleta y se selecciona el individuo en el que se pare la ruleta. Obviamente los individuos con mayor aptitud tendrán una mayor probabilidad de salir.

En el caso de que las probabilidades sean muy diferentes este método no es adecuado dado que si un individuo ocupa la mayor parte de la ruleta, el resto apenas será seleccionado lo que conlleva la reducción de la diversidad y la posibilidad de quedar atrapado en óptimos locales.

5.1.1.2. Selección por Rango

Este método consiste en asignar un rango numérico basado en su aptitud a los individuos y la selección se hace en base a ese rango. Así pues un individuo con mayor aptitud tendrá un rango mayor que uno con menor aptitud.

De esta manera se soluciona el problema presente en la selección por rueda de ruleta produciéndose una variedad de individuos mucho más rica. Por otra parte, uno de los problemas asociados a este método es que la convergencia puede ser más lenta, ya que dependiendo la manera de asignar los rangos, no existe tanta diferencia entre el mejor individuo y el resto.

5.1.1.3. Selección Elitista

Con los métodos anteriores puede ocurrir que perdamos el individuo con mejor adaptación al problema. Mediante la selección elitista el mejor individuo es copiado a la nueva población y el resto se selecciona mediante una de las formas vistas anteriormente.

La selección elitista puede mejorar los algoritmos genéticos evitando que se pierda la mejor solución y mejorando la convergencia. Existe una variación de este método en la cual el mejor individuo solo se copia a la siguiente generación en caso de que tras la reproducción y mutación no se haya generado un individuo con mejor aptitud.

5.1.1.4. Selección por Torneo

Mediante este método de selección se escogen un número predeterminado de individuos de la población que son evaluados y el que mayor puntuación tiene se reproduce, descartándose la descendencia del que tiene menor puntuación.

5.1.1.5. Selección Jerárquica

En esta selección los individuos presentes en la población atraviesan varias rondas de selección en cada generación. Las primeras evaluaciones son más rápidas y menos discriminatorias mientras que los que sobreviven hasta las últimas rondas son evaluados más rigurosamente.

La ventaja de este método es que agiliza el algoritmo dado que al utilizar una evaluación más rápida y menos rigurosa en las primeras rondas la mayoría de los individuos con bajas aptitudes son descartados para después someter a una evaluación más rigurosa y por lo tanto computacionalmente más costosa solo a los que han sobrevivido a las rondas previas.

5.1.1.6. Otros métodos de Selección

Existen otros métodos de selección aplicados a problemas concretos. Por ejemplo la selección por estado estacionario donde los individuos seleccionados en cada generación vuelven a la población preexistente, sustituyendo a algunos de los miembros menos aptos. En la selección por prueba de aptitud los miembros con mayor puntuación tienen más posibilidades de ser elegidos pero no la certeza.

Además dependiendo del problema al que nos enfrentemos puede ser ventajoso diseñara algún nuevo método de selección adecuado al problema específico.

5.6.3. Reproducción

El siguiente paso, una vez realizada la selección de los individuos más aptos de la población es realizar la reproducción o cruce entre dos de estos. Esta reproducción, más concretamente, consiste en el intercambio de material genético entre dos individuos para formar un nuevo. El objetivo de este intercambio es conseguir un nuevo individuo, denominado hijo, que mejore la aptitud de sus padres.

Para poder realizar la reproducción han de haberse seleccionado dos individuos de la población con una de las técnicas expuestas en los apartados anteriores, además puede elegirse el mismo padre un descendiente. Esto no constituye ningún problema sino que asegura la perpetuación del individuo dominante. Sin embargo, si este cruce se realiza con mucha frecuencia puede tener consecuencias negativas si el individuo presenta genes no deseados.

Existen diferentes formas de realizar los cruces entre dos individuos dependiendo de la codificación utilizada, siendo algunas de ellas válidas para cualquier codificación empleada. A continuación se expondrán algunas de ellas.

5.1.1.7. Cruce por un punto

Esta técnica consiste en cortar los dos individuos por un punto concreto y copiar la información genética de uno de los padres desde el inicio hasta el punto de cruce y el resto se copia del otro padre. Esta es una de las formas clásicas de reproducción utilizada en los algoritmos genéticos.

En la figura adjunta puede verse un ejemplo gráfico de este tipo de cruce.

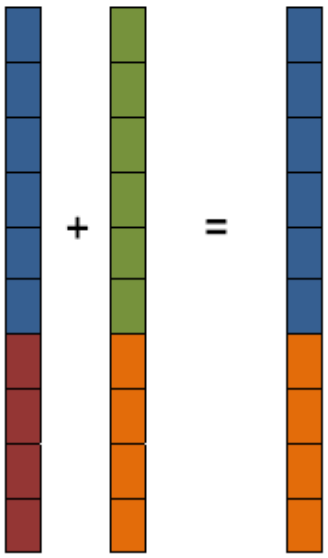

Figura 5 - 1. Cruce por un punto [elaboración propia].

5.1.1.8. Cruce por dos puntos

Este mecanismo de reproducción es similar al visto en el caso anterior con la salvedad de que los progenitores se cortan por dos puntos en lugar de uno. De esta manera se copiará al hijo el material genético comprendido entre los dos puntos de corte de un progenitor y el material genético comprendido entre el principio y el primer punto de corte y del segundo punto de corte hasta el final del otro progenitor.

En la figura adjunta puede verse un ejemplo gráfico de este tipo de reproducción.

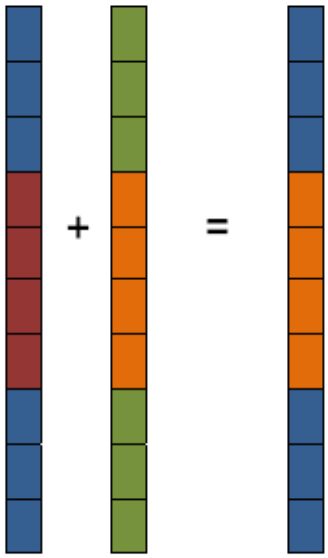

Figura 5 - 2. Cruce por dos puntos [elaboración propia]

5.1.1.9. Cruce Aritmético

Utilizando este proceso de reproducción la descendencia es generada al aplicar un operador aritmético sobre los padres. Generalmente este mecanismo de reproducción es usado cuando se trabaja con individuos codificados mediante cadenas binarias. De esta manera es posible aplicar sobre los progenitores de manera sencilla distintos operadores que componen alguna función lógica que controle el proceso de reproducción.

En la siguiente tabla se muestra un ejemplo de reproducción aritmética sobre individuos codificados binariamente utilizando el operador suma lógica.

0	1	1	0	1	0	1
0	0	1	0	1	1	0

AND >

0	0	1	0	1	0	0

Tabla 5 - 5. Cruce Aritmético utilizando el operador AND [elaboración propia].

5.6.4. Mutación

Una vez realizado el proceso de reproducción algunos de los individuos sufren transformaciones de forma extraordinaria. Estas transformaciones, aplicadas sobre un pequeño porcentaje de la población, contribuyen a conservar la diversidad genética evitando que el algoritmo converja hacia un óptimo local.

De forma más rigurosa, el proceso de mutación cosiste en modificar alguno de los genes de ciertos individuos de manera aleatoria. La cantidad de individuos afectados por este proceso depende de una probabilidad fijada con anterioridad, siendo esta probabilidad un parámetro variable que deberá sintonizarse de manera adecuada para el correcto funcionamiento del algoritmo.

En el caso de trabajar con una población codificada binariamente, la mutación cosiste en invertir el bit correspondiente al gen mutado. De forma similar, cuando se trabaja con una codificación numérica la mutación intercambia un número por otro generado de manera aleatoria.

En la siguiente tabla aparece una mutación sobre el gen señalado de un individuo codificado de manera binaria.

0	1	0	0	1	1	0

→

0	1	1	0	1	1	0

Tabla 5 - 6. Mutación aleatoria sobre individuo codificado binariamente [elaboración propia].

5.7. Parámetros de los Algoritmos Genéticos

Una vez vistos los procedimientos que se llevan a cabo sobre la población en un algoritmo genético se definirán algunos parámetros que rigen estos procedimientos y cómo influyen en la obtención de la solución.

5.7.1. Tamaño de la población

Este parámetro define el número de individuos presentes en la población de una generación determinada. Si este parámetro es insuficiente el algoritmo tendrá pocas posibilidades de encontrar el óptimo dado que se reduce la diversidad de la población. Por el contrario, si la población es excesiva penalizaremos la agilidad del algoritmo haciéndolo excesivamente lento.

Existe un límite a partir del cual un incremento en el tamaño de la población no aporta ninguna ventaja al algoritmo dado que no se consigue una velocidad mayor de resolución del problema.

5.7.2. Probabilidad de cruce

La probabilidad de cruce indica la frecuencia con la que se realiza la reproducción entre dos padres. En el caso de que esta probabilidad sea del 100% los hijos siempre se generarán mediante cruce entre dos padres. En el caso contrario, si no existe esta probabilidad los hijos serán copias exactas de los progenitores.

5.7.3. Probabilidad de mutación

Este parámetro hace referencia a la frecuencia con la que los genes de un individuo sufren transformaciones o mutaciones. Si esta probabilidad no existe, los descendientes son idénticos a los que había después de realizar el cruce entre dos progenitores. De la misma manera, si la probabilidad de mutación es del 100% se producirán cambios aleatorios en la totalidad de los individuos hijos procedentes del cruce entre dos padres.

5.8. Ventajas de los Algoritmos Genéticos

Una de las principales ventajas de los Algoritmos Genéticos es que son **algoritmos paralelos**, es decir, operan con varias soluciones simultáneamente en cada iteración. Esta característica diferencia a los Algoritmos Genéticos de las técnicas clásicas.

Mientras las técnicas tradicionales solo exploran el espacio de soluciones en una dirección, los Algoritmos Genéticos exploran el espacio de soluciones en varias direcciones simultáneamente. De esta manera donde una técnica tradicional se

quedaría atascada y debería comenzar de nuevo, el algoritmo genético desecha la solución y continúa con otros individuos de la población.

Debido a la diversidad de la población, es decir, la diversidad de soluciones con las que trabaja el algoritmo, los algoritmos genéticos resultan **menos vulnerables a los óptimos locales** que las técnicas clásicas. Muchos algoritmos quedan atrapados en óptimos locales dado que en las cercanías de éstos no existe ninguna solución mejor y concluyen con que han alcanzado la mejor de las soluciones.

Cuando nos enfrentamos a problemas de **optimización multiobjetivo**, los algoritmos genéticos son especialmente útiles dado que pueden modificar varias características de la solución en cada iteración.

Por último, a la hora de implementar un algoritmo genético sobre algún problema en concreto, el programador **no necesita conocimiento específico del problema** dado que los cambios que se realizan en las soluciones son aleatorios para luego utilizar las funciones de aptitud sobre ellas.

5.9. Desventajas de los Algoritmos Genéticos

La principal desventaja de los algoritmos genéticos es la dificultad de **codificar correctamente el problema**. Esta codificación debe permitir modificaciones aleatorias de la solución sin que se produzcan continuamente resultados carentes de sentido.

Por otro lado, los algoritmos genéticos son extremadamente **sensibles a los parámetros** utilizados, como el tamaño de la población o las tasas de mutación y cruce, pudiéndose dar casos donde el algoritmo tarde mucho en converger o no lo haga de ninguna forma.

Por ultimo si los mecanismos evolutivos que rigen el proceso no son los adecuados el algoritmo puede **converger prematuramente** debido a que uno de los individuos es más apto que todos sus competidores.

5.10. Conclusiones

En este capítulo se han expuesto los fundamentos teóricos de los algoritmos genéticos y como se ha podido comprobar, existen multitud de variantes de los mecanismos evolutivos en los que se basan estos algoritmos.

Es tarea del programador elegir adecuadamente estos mecanismos dependiendo del problema que se quiera resolver. Pero sin lugar a dudas, es un método robusto que permite su aplicación sobre el problema objeto de este trabajo.

CAPÍTULO 6:

ALGORITMO GENÉTICO APLICADO AL PROBLEMA DE PROGRAMACIÓN DE LA PRODUCCIÓN

6.1. Introducción

El objetivo que persigue este capítulo es exponer las decisiones y métodos empleados para la elaboración del algoritmo genético que se ha desarrollado para resolver el problema de programación de la producción en una configuración productiva tipo flow shop flexible con el fin de minimizar el tiempo de producción o makespan.

En primer lugar, se hará una descripción general del funcionamiento del algoritmo para después tratar cada uno de los elementos de este por separado comenzando por la codificación del problema, argumentando porqué se ha elegido la codificación entera y no otra. Una vez elegida una codificación adecuada se tratará la forma en la que se genera la población inicial. Después se explicará el procedimiento empleado para el cálculo de la función objetivo o criterio de optimización. Por último se expondrán los diversos mecanismos evolutivos concretos en los que se basa el algoritmo desarrollado como son la selección por torneo, la reproducción por cruce de dos puntos y la mutación aleatoria.

6.2. Descripción general del algoritmo genético

El Algoritmo Genético desarrollado está basado en el desarrollado por [Chu – Beasley, 1997]. Dicho algoritmo ha ganado especial importancia en numerosos sectores en los últimos años debido a la calidad de las respuestas obtenidas haciendo uso de una alta carga computacional.

La principal característica de este algoritmo es que mantiene el tamaño de la población constante, remplazando solamente uno de los individuos en cada generación. De esta manera se conserva la diversidad de soluciones, evitando que el algoritmo quede atrapado en óptimos locales.

El individuo solamente es remplazado si el nuevo individuo mejora la aptitud frente a la mejor solución encontrada hasta el momento, denominada incumbente. Esta característica del algoritmo lo hace elitista o dicho de otro modo, la solución solo puede mejorarse, en ningún caso empeorar.

El proceso evolutivo del algoritmo se realizar de forma clásica haciendo uso de los procedimientos de selección, reproducción y mutación, comunes a cualquier algoritmo genético. Además existe una etapa de aceptación en la que se determina si el nuevo individuo ingresará a la población de la nueva generación.

En el siguiente esquema se muestra un diagrama de flujo con los mecanismos que rigen el algoritmo genético implementado.

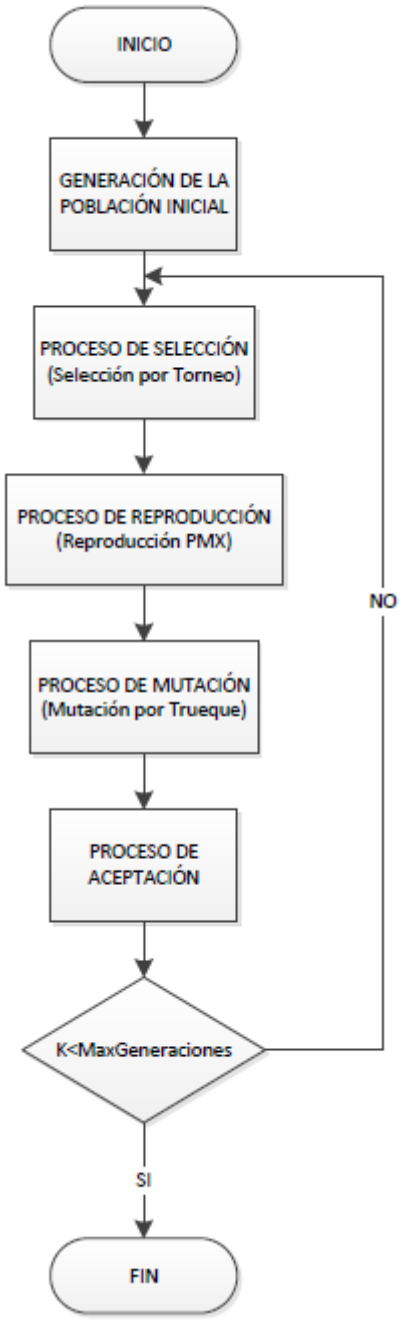

Figura 6 - 1. Diagrama de flujo del Algoritmo Genético implementado [elaboración propia].

El primer paso del algoritmo consiste en generar la población inicial de individuos sobre los que se realizará el proceso evolutivo. Para ello, una vez determinado el tamaño de la población se genera ésta de manera aleatoria teniendo especial cuidado de que no existan gemelos, es decir, dos soluciones iguales dentro de la población inicial.

El segundo paso, ya dentro del proceso evolutivo, consiste en la selección de dos progenitores, que serán los encargados de pasar sus genes a los descendientes. Este proceso se realizar por torneo..

El tercer paso consiste en el proceso de reproducción mediante el cual los dos progenitores seleccionados en la etapa anterior se combinan para crear un individuo hijo. Esta recombinación se realiza mediante una modificación del cruce por dos puntos llamada PMX (Partially Matched Crossover).

El cuarto paso hace referencia al proceso de mutación, en el cual, en función de una tasa de mutación ajustable, se realiza la modificación de manera aleatoria de uno de los genes del individuo hijo.

Por último, en la quinta etapa, se realizar el proceso de aceptación en el que se decide si el nuevo individuo puede formar parte de la nueva generación en función de que mejore el criterio de optimización con el que trabaja el algoritmo. Además en esta etapa se comprueba la diversidad de la población, asegurándose de que el nuevo individuo que se va a insertar no existe en la población.

El proceso finaliza cuando se han realizado un cierto número de iteraciones ajustables por el usuario. Este número deberá ser determinado en función del tamaño del problema que se pretende resolver. Una vez finalizado el proceso se muestran los resultados obtenidos así como la evolución del algoritmo de forma gráfica.

6.3. Codificación del problema

Dado que resolver un problema de programación de la producción significa encontrar la secuencia óptima de los trabajos que optimice un criterio dado, la codificación elegida es una codificación directa por números enteros que representan la secuencia de trabajos.

De esta manera la codificación será un vector de 1xm siendo m el número de trabajos que deseamos optimizar. Como ejemplo, aparece a continuación la codificación de un problema con 10 trabajos.

8	2	3	5	7	4	0	1	6	9

Tabla 6 - 1. Codificación entera empleada en el algoritmo desarrollado [elaboración propia].

6.4. Generación de la población inicial

Una vez definida la codificación del problema, debemos generar la población inicial de individuos que formarán la primera generación. Para ello existen diversas posibilidades, desde utilizar alguna heurística para generar un conjunto de soluciones factibles hasta crear la población inicial de manera aleatoria.

En el algoritmo desarrollado en este proyecto se genera la población inicial de manera totalmente aleatoria, sin la utilización de heurísticas o reglas que nos aproximen a la solución.

Crear un individuo de manera aleatoria, consiste en generar un vector de longitud igual al número de tareas que se han de programar, cuyos elementos serán números enteros aleatorios no repetidos. Este vector representa una posible secuencia de programación.

Para construir la población inicial se repite el proceso hasta completar el tamaño máximo de la población, parámetro ajustable por el usuario antes de la ejecución del algoritmo.

Es de vital importancia generar una población inicial diversa para el correcto funcionamiento del algoritmo. Por esto, se comprueba, antes de insertar el individuo generado de manera aleatoria a la población inicial, que no tenga ya un gemelo dentro de esta población.

Cuando el proceso de generación de la población inicial se ha completado, se obtiene una matriz de individuos donde cada fila representa a un individuo, es decir, una secuencia o solución al problema.

6.5. Obtención de la función objetivo

Cuando ya tenemos los individuos sobre los que trabajar debemos evaluar la aptitud de estos frente al problema. Como se expuso en el capítulo 3, existen multitud de criterios de optimización o formas de medir la calidad de la programación de los trabajos sobre una configuración tipo flowshop flexible.

El algoritmo desarrollado, trata de optimizar el tiempo de ejecución de todos los trabajos o makespan dado que éste es un criterio ampliamente utilizado en la industria. Además, el algoritmo, se ha desarrollado de manera modular para poder sustituir fácilmente este criterio de optimización por otro con solo cambiar el cálculo de este criterio contenido dentro de una función.

En el capítulo 3, se explicó mediante un ejemplo el cálculo del makespan o tiempo de ejecución para un conjunto de tareas. El proceso que se realizó manualmente ha sido implementado en MatLab para evaluar la aptitud de los individuos frente al problema.

Los datos de entrada para el cálculo del tiempo de ejecución son la secuencia u orden de ejecución de las tareas, la matriz de tiempos de ejecución de cada tarea en cada etapa y el número de máquinas por etapa.

Con estos datos, el primer paso es ordenar la matriz de tiempos de ejecución de acuerdo a la secuencia propuesta. Una vez ordenada la matriz, se procede a la asignación de tareas en las máquinas libres de la primera etapa para calcular el tiempo necesario para que las tareas pasen a la siguiente etapa. Este proceso se realiza de forma iterativa hasta que se completan todas las etapas, garantizando que la secuencia propuesta sea respetada y las máquinas a las que se asigna cada tarea estén libres.

Cuando el proceso iterativo llega a su fin, el valor del makespan corresponde al tiempo de terminación de la última tarea dada por la secuencia, es decir, al mayor valor de los tiempos de terminación de todas las tareas.

6.6. Proceso evolutivo

En este apartado se describirán los procesos evolutivos del algoritmo genético desarrollado para resolver el problema de programación de la producción. Como todo algoritmo genético consta de los tres procesos principales de selección, reproducción y mutación además de contar con una cuarta etapa de aceptación donde se determina si el nuevo individuo generado es lo suficientemente bueno para pasar a formar parte de la nueva población.

6.6.1. Selección

El proceso de selección consiste en elegir dos individuos de la población existente con el fin de que pasen sus genes al individuo hijo. Como se vio en el capítulo anterior existen multitud de formas de realizar esta selección, siendo una de las más comunes la selección aleatoria o por torneo.

En la selección por torneo se escoge el mejor de una serie de individuos seleccionados de manera aleatoria. En nuestro caso se seleccionan dos individuos que posteriormente se comparar en relación a su aptitud frente al problema. Este proceso se realiza dos veces para obtener dos progenitores que posteriormente serán los que transmitan sus características al nuevo individuo.

El diagrama de flujo del proceso de selección por torneo implementado aparece en la siguiente figura.

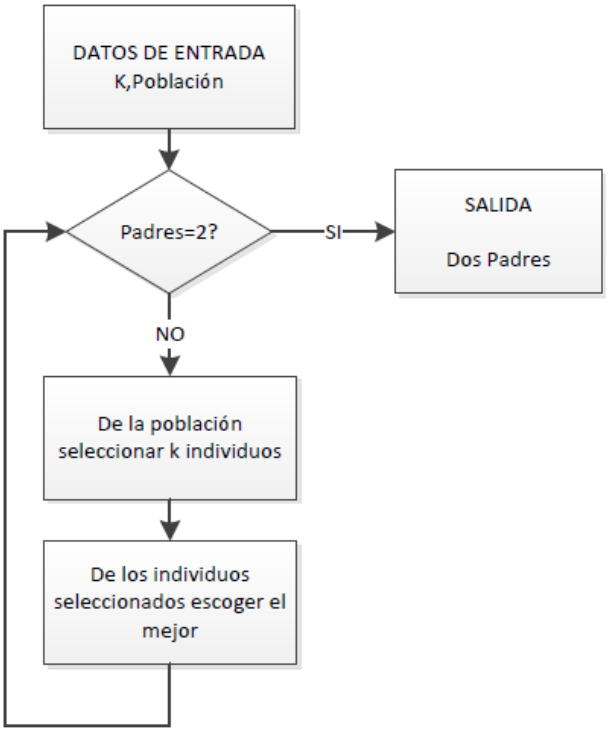

Figura 6 - 2. Proceso de selección por torneo [elaboración propia].

6.6.2. Reproducción

Cuando se ha completado con éxito el proceso de selección se puede llevar a cabo el proceso de cruce o reproducción. Mediante este mecanismo, los dos progenitores seleccionados en la etapa anterior pasarán sus genes al individuo hijo. En el algoritmo implementado se utiliza una modificación del cruce por dos puntos denominada PMX (Partially Matched Crossover).

El primer paso para realizar el cruce mediante este procedimiento es determinar los dos puntos de corte por donde se cruzarán los progenitores. Estos puntos son obtenidos de manera aleatoria. A partir de estos dos puntos de corte se obtienen dos segmentos, uno en cada padre.

Después se intercambian estos dos segmentos en los dos hijos que se generan y el resto de la cadena se completa haciendo mapeos entre los dos padres, dependiendo si el valor a mapear está contenido en el segmento o no.

Como ejemplo, tenemos dos progenitores formados por las secuencias P1=[1,2,4,6,3,7,5,8] y P2=[5,4,1,7,2,6,8,3]. Además los dos puntos de corte, seleccionados de manera aleatoria, establecen que el segmento a insertar de P1 en P2 es [4,6,3] que corresponde con el segmento [1,7,2] de P2.

De esta manera, se genera un hijo partiendo de P2 siendo H2=[5,4,4,6,3,6,8,3] que eliminando las posiciones repetidas quedaría H2=[5,*,4,6,3,*,8,*] donde los asteriscos

representan las posiciones que serán ocupadas por el segmento de P2, H2=[5,1,4,6,3,7,8,2].

En la siguiente figura se puede ver el proceso de reproducción mediante PMX de forma esquemática:

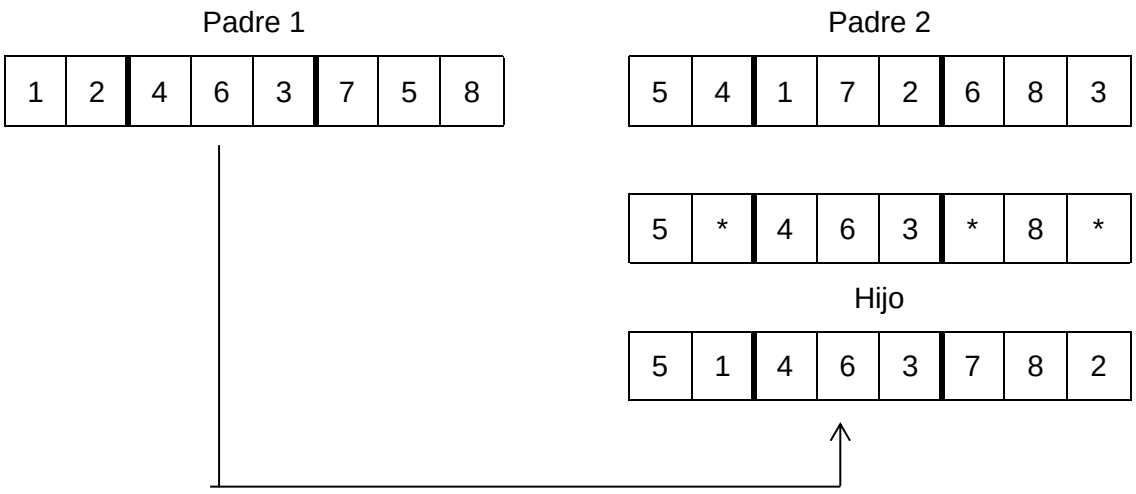

Figura 6 - 3. Reproducción mediante PMX [elaboración propia].

6.6.3. Mutación

En el algoritmo desarrollado, el individuo hijo obtenido tras el proceso de reproducción, es sometido al proceso de mutación, por el cual se intercambian dos de sus posiciones de manera aleatoria.

Que se realice el proceso de mutación, depende de una probabilidad determinada por el usuario del algoritmo denominada tasa de mutación. De esta manera, la mutación no se lleva a cabo en todos los individuos sino solamente en un porcentaje de ellos.

En el diagrama de la figura 6-5 puede verse el proceso de mutación de forma esquemática.

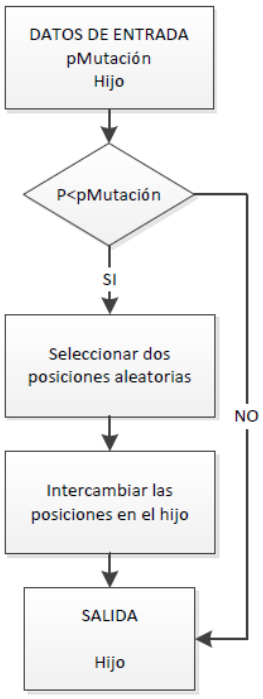

Figura 6 - 4. Proceso de Mutación [elaboración propia].

Mediante este proceso, se consigue aumentar la diversidad de la población y obtener nuevas configuraciones de manera aleatoria, explorando así un mayor número de propuestas.

6.6.4. Aceptación

Una vez completado el proceso evolutivo se evalúa el nuevo individuo generado para determinar si formará parte de la nueva población. El criterio que determina si se aceptará el nuevo individuo o será rechazado es que la función objetivo de éste mejore la del individuo de peor calidad presente en la población.

Si el nuevo individuo mejora la función objetivo del individuo de peor calidad, el individuo generado reemplaza al individuo de peor calidad. Además de este criterio, se comprueba que se mantenga la diversidad, es decir, que el nuevo individuo no tenga un gemelo dentro de la población.

6.7. Conclusiones

En este capítulo se ha expuesto un mecanismo evolutivo concreto para la resolución del problema de programación de la producción. El algoritmo desarrollado consta de varias etapas, comunes a cualquier algoritmo genético, que pueden ser mejoradas o modificadas de acuerdo al tipo de problema al que nos enfrentemos.

CAPÍTULO 7:

ANÁLISIS DE RESULTADOS

7.1. Introducción

En este capítulo se expondrán los resultados obtenidos al aplicar el algoritmo descrito en el capítulo anterior a diversos problemas de programación de la producción. Para ello, se utilizarán problemas tipo de los que se conoce su solución óptima.

En primer lugar, se describirán los casos de prueba utilizados y después se utilizará el algoritmo desarrollado para abordar dichos problemas.

7.2. Casos de prueba

Muchos autores han probado sus algoritmos usando casos de prueba generados aleatoriamente, sin embargo estos no han sido publicados. En algunos artículos especializados sobre la programación en máquinas aparecen los casos de prueba publicados por la universidad Koç University de Estambul, Turquía.

Estos casos están formados por problemas de distinta dificultad con diferente número de tareas y diferente número de máquinas para procesarlas. En nuestro caso utilizaremos algunos de estos casos, partiendo de problemas de baja dificultad para acabar con problemas de una dificultad superior.

Los casos más sencillos constan de 5 tareas que pueden ser procesados por 2 o 5 etapas. Los casos de dificultad intermedia constan de 20 tareas procesables en 2 o 5 etapas. Por último, los casos más complejos constan de 50 y 100 tareas procesables en 2 o 5 etapas.

A modo de resumen, en la siguiente tabla aparecen los casos de prueba con algunas de sus características:

CASO	TAREAS	ETAPAS	Cmax
T5E2	5	2	267
T5E5	5	5	304
T20E2	20	2	1288
T20E5	20	5	1172
T50E2	50	2	2483
T100E5	100	5	4985

Tabla 7 – 1. Presentación de los casos de prueba [elaboración propia].

7.3. Resultados

En este apartado se probará el algoritmo frente a los casos de prueba comentados en el apartado anterior. Se realizarán varias ejecuciones del algoritmo sobre los casos presentados anteriormente y luego se compararan los resultados.

Para cada caso de prueba, en primer lugar aparece una tabla que resume las características del caso estudiado. Estas características hacen referencia al número de tarea a programar, el número de etapas para procesar dichas tareas y el mejor tiempo de ejecución del caso de prueba.

Además de las características, aparece el tamaño de la población utilizado para cada caso, siendo ésta mayor para los problemas de mayor complejidad. Este tamaño ha sido determinado de una manera empírica, con el objetivo de obtener un número de iteraciones razonables sin perjudicar el tiempo empleado para resolver el problema.

Además del resultado obtenido, se muestra de forma gráfica la evolución de la función objetivo, en nuestro caso el tiempo de ejecución o makespan, frente a las iteraciones del algoritmo, para una pasada representativa de cada caso de prueba.

7.3.1. Caso T5E2

Descripción del caso T5E2	
Número de Tareas	5
Número de Etapas	2
Tiempo de ejecución (Cmax)	267
Tamaño de la Población	10

Tabla 7 - 2. Descripción del caso T5E2 [elaboración propia].

Resultados del caso T5E2			
Nº Ejecución	**Cmax**	**Iteraciones**	**Tiempo**
Ejecución 1	267	11	0.2188
Ejecución 2	267	117	0.1250
Ejecución 3	267	2	0.1314
Ejecución 4	267	8	0.1094
Ejecución 5	267	1	0.2656

Tabla 7 - 3. Resultados del caso T5E2 [elaboración propia].

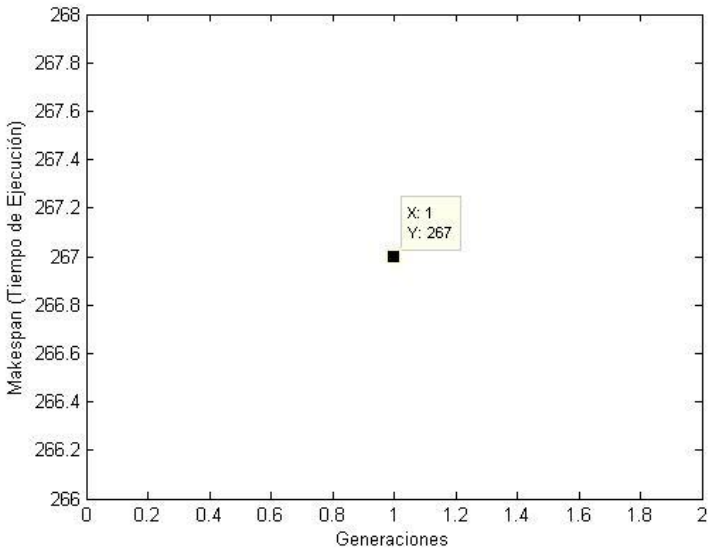

Figura 7 - 1. Evolución de la Ejecución 5 [elaboración propia].

7.3.2. Caso T5E5

Descripción del caso T5E5	
Número de Tareas	5
Número de Etapas	5
Tiempo de ejecución (Cmax)	304
Tamaño de la Población	10

Tabla 7 - 4. Descripción del caso T5E5 [elaboración propia].

Resultados del caso T5E5			
Nº Ejecución	**Cmax**	**Iteraciones**	**Tiempo**
Ejecución 1	304	33	0.1250
Ejecución 2	304	10	0.0938
Ejecución 3	304	26	0.1094
Ejecución 4	304	367	0.1406
Ejecución 5	304	4	0.0313

Tabla 7 - 5. Resultados del caso T5E5 [elaboración propia].

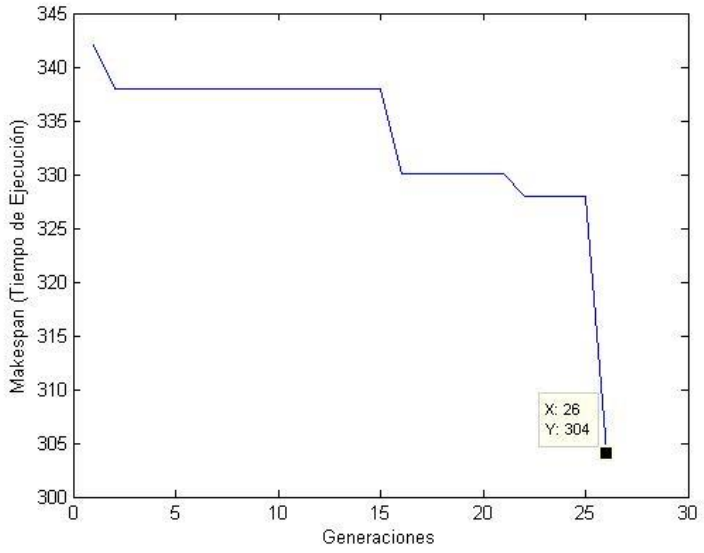

Figura 7 - 2. Evolución de la Ejecución 3 [elaboración propia].

7.3.3. Caso T20E2

Descripción del caso T20E2	
Número de Tareas	20
Número de Etapas	2
Tiempo de ejecución (Cmax)	1288
Tamaño de la Población	10

Tabla 7 - 6. Descripción del caso T20E2 [elaboración propia].

Resultados del caso T20E2			
Nº Ejecución	**Cmax**	**Iteraciones**	**Tiempo**
Ejecución 1	1288	845	0.5313
Ejecución 2	1288	79	0.1250
Ejecución 3	1288	1603	0.9844
Ejecución 4	1288	280	0.2188
Ejecución 5	1288	1155	0.7500

Tabla 7 - 7. Resultados del caso T20E2 [elaboración propia].

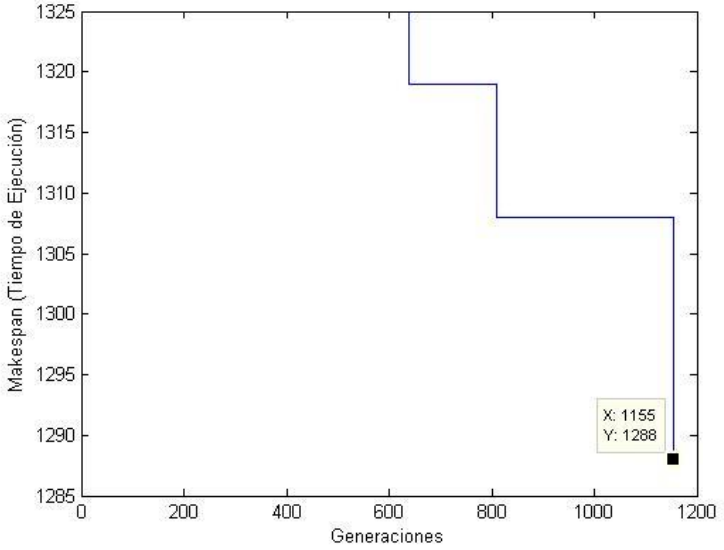

Figura 7 - 3. Evolución de la Ejecución 5 [elaboración propia].

7.3.4. Caso T20E5

Descripción del caso T20E5	
Número de Tareas	20
Número de Etapas	5
Tiempo de ejecución (Cmax)	1172
Tamaño de la Población	10

Tabla 7 - 8. Descripción del caso T20E5 [elaboración propia].

Resultados del caso T20E5			
Nº Ejecución	**Cmax**	**Iteraciones**	**Tiempo**
Ejecución 1	1170	1111	1.0938
Ejecución 2	1170	108	0.1719
Ejecución 3	1170	532	0.5469
Ejecución 4	1170	1159	1.0313
Ejecución 5	1170	368	0.3906

Tabla 7 - 9. Resultados del caso T20E5 [elaboración propia].

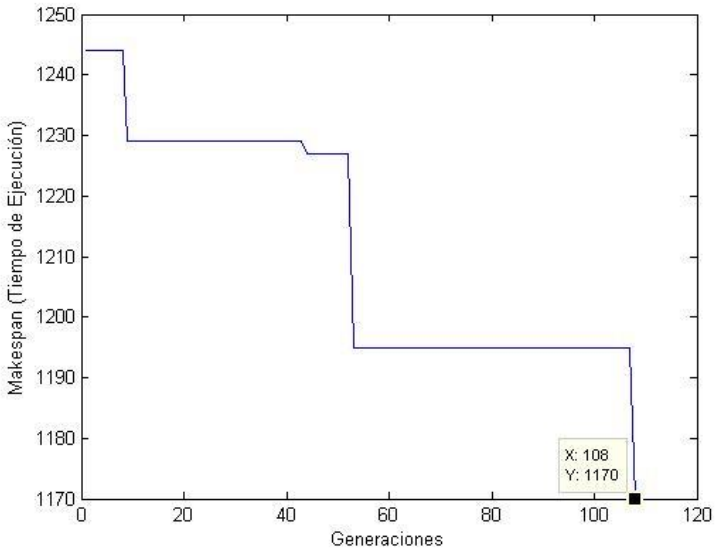

Figura 7 - 4. Evolución de la Ejecución 2 [elaboración propia].

7.3.5. Caso T50E2

Descripción del caso T100E2	
Número de Tareas	50
Número de Etapas	2
Tiempo de ejecución (Cmax)	2483
Tamaño de la Población	20

Tabla 7 - 10. Descripción del caso T100E2 [elaboración propia].

Resultados del caso T100E2			
Nº Ejecución	**Cmax**	**Iteraciones**	**Tiempo**
Ejecución 1	2483	9883	28.3594
Ejecución 2	2483	5106	13.7969
Ejecución 3	2483	2364	6.7969
Ejecución 4	2483	1386	4.1875
Ejecución 5	2483	6679	18.2031

Tabla 7 - 11. Resultados del caso T100E2 [elaboración propia].

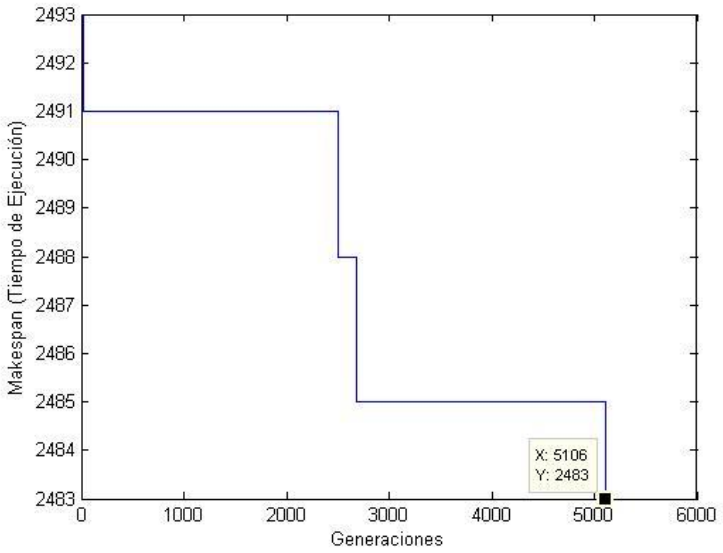

Figura 7 - 5. Evolución de la Ejecución 1 [elaboración propia].

7.3.6. Caso T100E5

Descripción del caso T100E5	
Número de Tareas	100
Número de Etapas	5
Tiempo de ejecución (Cmax)	4985
Tamaño de la Población	20

Tabla 7 - 12. Descripción del caso T100E5 [elaboración propia].

Resultados del caso T100E5			
Nº Ejecución	**Cmax**	**Iteraciones**	**Tiempo**
Ejecución 1	4985	9398	143.5938
Ejecución 2	4985	42893	583.9844
Ejecución 3	4985	10312	150.7656
Ejecución 4	4985	11758	168.1719
Ejecución 5	4985	3231	49.3281

Tabla 7 - 13. Resultados del caso T100E5 [elaboración propia].

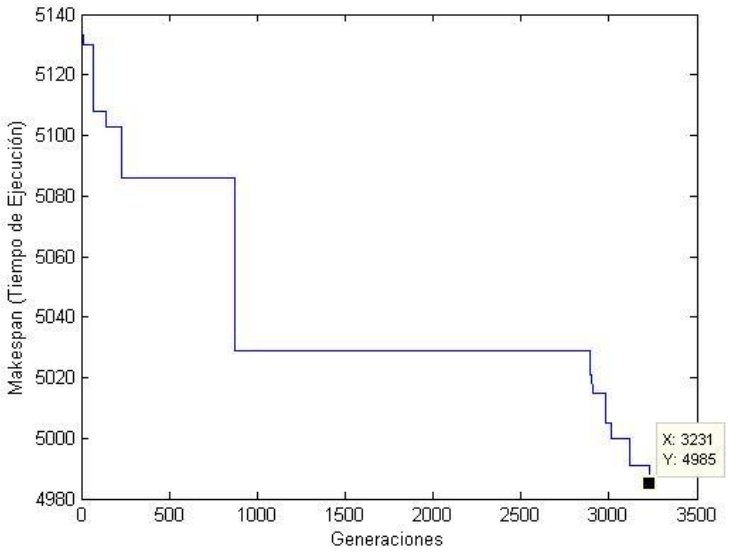

Figura 7 - 6. Evolución de la Ejecución 5 [elaboración propia].

7.4. Conclusiones

Como se puede ver en los resultados obtenidos tras el uso del algoritmo sobre los diversos casos de prueba expuestos, en todos los casos se alcanza el óptimo tras un número de iteraciones adecuado. Esta característica es de vital importancia dado que lo que buscamos es alcanzar esta configuración óptima para cualquier caso.

El buen funcionamiento del algoritmo, depende en gran medida de la buena calibración de sus parámetros, siendo uno de los más importantes el tamaño de la población. Este parámetro se configurará dependiendo de la complejidad del problema abordado.

En nuestro caso para los problemas de gran complejidad, es decir de 100 tareas, hemos duplicado el tamaño de la población para que exista una diversidad considerable, que permita resolver el problema en un tiempo razonable.

Por otra parte, se ve que la población inicial, generada de manera aleatoria en nuestro caso, influye de manera decisiva sobre el tiempo que tarda el algoritmo en converger. De esta manera una posible mejora, será inicializar la población utilizando alguna heurística que obtenga una población inicial de mayor calidad.

Una de las ventajas del algoritmo genético desarrollado, es la posibilidad de procesar en paralelo diferentes alternativas de solución. De este modo, aunque se llegue a soluciones con el mismo makespan, la secuencia de operaciones que arroja dicho makespan suele ser diferente de una ejecución a otra.

CAPÍTULO 8:

CONCLUSIONES Y LINEAS FUTURAS

8.1. Conclusiones

Resolver un problema de programación de la producción es resolver un problema de optimización combinatoria. Esto implica que las soluciones posibles al problema crecen de manera exponencial al crecer el tamaño del problema.

Enfrentarnos a un problema cuya complejidad crece de manera exponencial es complejo y más aún si queremos resolverlo de manera exacta. Por ello, los métodos aproximados están cobrando cada vez mayor importancia.

Dentro de los métodos aproximados, los métodos inspirados en la naturaleza tienen un gran potencial ya que implementan reglas o mecanismos vistos en los seres vivos en un computador, con el fin de aprovechar el conocimiento intrínseco de los seres vivos.

Uno de los algoritmos aproximados bio-inspirados son los algoritmos genéticos que utilizan mecanismos evolutivos para refinar la solución partiendo de un conjunto de soluciones iniciales. Y además ofrecen soluciones viables de forma menos compleja.

Estos algoritmos, aplicados al problema estudiado en este trabajo, obtienen resultados de alta calidad en tiempos razonables. Además con esta metodología es posible encontrar numerosas soluciones que arrojen resultados óptimos, característica propia de los algoritmos genéticos.

8.2. Líneas futuras

Existen multitud de algoritmos y propuestas capaces de resolver problemas de optimización combinatoria como el tratado en este trabajo. Sería interesante evaluar varias propuestas para determinar cuál es la más adecuada, en términos de calidad de solución y coste computacional para tratar los problemas de programación de la producción.

El algoritmo desarrollado, es un algoritmo de optimización con un objetivo único, en nuestro caso, minimizar el makespan o tiempo de ejecución de todos los trabajos.

Sin embargo existen diversos criterios de optimización que pueden ser buscados por las empresas de forma paralela. Esto da como resultado la optimización multiobjetivo, donde se busca optimizar varias variables a la vez. Los algoritmos genéticos, son una buena alternativa cuando se busca optimizar varios criterios para un mismo problema.

Por otra parte, todo lo expuesto en este trabajo, asume un entorno estático y determinista donde todo es ideal. Este planteamiento, se aleja enormemente de la realidad dado que cualquier compañía está sujeta a incertidumbre.

Por esto, un campo de investigación abierto, es el desarrollo de algoritmos que tengan en cuenta esta incertidumbre o borrosidad, a la que todos los procesos en el mundo real están sujetos.

Además, es interesante incluir en los algoritmos, un conjunto de estrategias o pasos a seguir cuando sucede un evento no previsto por el programador. De esta manera, se consigue un mejor resultado y el consiguiente ahorro de recursos.

CAPÍTULO 9:

BIBLIOGRAFÍA

ANDRES, C., VICENS, E., & LARIO, F. C. (2001). ALGORITMO DE RECOCIDO SIMULADO PARA LA SECUENCIACIÓN EN TALLERES DE FLUJO HÍBRIDOS CON TIEMPOS DE CAMBIO DE PARTIDA DEPENDIENTES DE LA SECUENCIA. *Congreso nacional de estadística e investigación operativa*.

ARRANZ DE LA PEÑA, J., & PARRA TRUYOL, A. (2007). *ALGORITMOS GENÉTICOS.* Universidad Carlos III.

BALLESTEROS SILVA, P. P., BALLESTEROS RIVEROS, D. P., & BRAVO BOLÍVAR, J. E. (2013). APLICACIÓN DE UNA HEURÍSTICA CONSTRUCTIVA EN PROGRAMACIÓN SECUENCIAL PARA ASIGNACIÓN DE VARIOS TRABAJOS A VARIAS MÁQUINAS EN PARALELO. *Universidad de Pereira*.

CADENA PALAGOT, N. S. (Mayo de 2015). *EVOLUCIÓN DE LOS SISTEMAS PRODUCTIVOS*. Obtenido de http://www.gestiopolis.com/evolucion-de-los-sistemas-productivos/

CERVANTES POSADA, M. (2009). NUEVOS MÉTODOS METAHEURÍSTICOS PARA LA ASIGNACIÓN EFICIENTE, OPTIMIZADA Y ROBUSTA DE RECURSOS LIMITADOS. *Universidad Politécnica de Valencia*.

CEYDA OĞUZ, P. (s.f.). *Koç University*. Obtenido de http://home.ku.edu.tr/~coguz/Research/dataset2.zip

COMPANYS, R., & COROMINAS, A. (1996). *DIRECCIÓN DE OPERACIONES. ORGANIZACIÓN DE LA PRODUCCIÓN.*

CONWAY, R. W., MAXWELL, W. L., & MILLER, L. (1967). *THEORY OF SCHEDULING.* Addison-Wesley Publishing Company.

CRESPO FRANCO, T., & VAZQUEZ, J. (1996). SISTEMAS DE PLANIFICACIÓN Y CONTROL DE LA FABRICACIÓN: ANÁLISIS COMPARATIVO. *Universidad de Vigo*.

EGUIA SALINAS, I. (1996). PROGRAMACIÓN DE LA PRODUCCIÓN EN PROCESOS SEMICONTINUOS. MÉTODOS Y ALGORITMOS DE RESOLUCIÓN. *Universidad de Sevilla*.

GALBRAITH, J. (1973). DESIGNING COMPLEX ORGANIZATIONS. *Massachussets: Addison-Wesley Pub. Co.*

GAREY, M. R., & JOHNSON, D. S. (1979). *COMPUTERS AND INTRACTABILITY: A GUIDE TO THE THEORY OF NPCOMPLETENESS.* San Francisco: W. H. Freeman and Company.

GLOVER, F. W., & LAGUNA, M. (1997). *TABU SEARCH.* Springer.

GOMEZ GASQUET, P. (2010). PROGRAMACIÓN DE LA PRODUCCIÓN EN UN TALLERDE FLUJO HÍBRIDO SUJETO A INCERTIDUMBRE: ARQUITECTURA Y ALGORITMOS. APLICACIÓN A LA INDUSTRIA CERÁMICA. *Universidad Politéccnica de Valencia*.

GUEREQUETA, R., & VALLECILLO, A. (2000). *TÉCNICAS DE DISEÑO DE ALGORITMOS .* Universidad de Málaga.

JIMÉNEZ MORALES, Á. P. (2012). SOLUCIÓN DEL PROBLEMA DE PROGRAMACIÓN DE FLOW-SHOP. *UNIVERSIDAD TECNOLÓGICA DE PEREIRA*.

KIRKPATRICK, S. (1983). OPTIMIZATION BY SIMULATED ANNEALING. *Science*.

LEFCOVICH, M. (MAYO DE 2015). *LEAN MANAGEMENT Y FILOSOFÍA LEAN*. Obtenido de http://www.gestiopolis.com/lean-management-y-filosofia-lean/

LOPEZ VARGAS, J. C. (2013). METODOLOGÍA DE PROGRAMACIÓN DE PRODUCCIÓN EN UN FLOW SHOP HÍBRIDO FLEXIBLE CON EL USO DE ALGORITMOS GENÉTICOS PARA REDUCIR EL MAKESPAN. APLICACIÓN EN LA INDUSTRIA TEXTIL. *Universidad Nacional de Colombia*.

MACCARTHY, B. L., & LIU, J. Y. (1993). ADDRESSING THE GAP IN SCHEDULING RESEARCH - A REVIEW OF OPTIMIZATION AND HEURISTIC METHODS IN PRODUCTION SCHEDULING. *International Journal of Production Research*.

MARQUEZ DELGADO, J. (2012). OPTIMIZACIÓN DE LA PRODUCCIÓN (SCHEDULING) EN TALLERES DE MECANIZADO. *Universidad Politecnica de Madrid*.

MATEO, P., & LAHOZ, D. (2009). *PROGRAMACIÓN LINEAL ENTERA.* Universidad de Zaragoza.

METROPOLIS, N., & ROSENBLUTH, A. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal Of Chemical Physics*.

MIRLEDY TORO, E., STEVEN RESTREPO, Y., & ECHEVERRI GRANADA, M. (2006). ALGORITMO GENETICO MODIFICADO APLICADO AL PROBLEMA DE SECUENCIAMIENTO DE TAREAS EN SISTEMAS DE PRODUCCION LINEAL – FLOW SHOP. *Scientia et Technica Año XII, No 30, Mayo de 2006 UTP. ISSN 0122-1701*.

Mula Bru, J., Poler Escoto, R., & Lario Esteban, F. C. (2002). MODELOS Y MÉTODOS PARA LA PLANIFICACIÓN DE LA PRODUCCIÓN DE LA CADENA DE SUMINISTRO BAJO INCERTIDUMBRE: UNA INTRODUCCIÓN AL ESTADO DEL ARTE. *Universidad Politécnica de Valencia*.

PINEDO, M. (2012). *SCHEDULING - THEORY, ALGORITHMS, AND SYSTEMS.* Springer.

ROYO, J., LAMBÁN, M., & RIVAS, A. (2007). ESTUDIO Y ANÁLISIS DE REGLAS Y ALGORITMOS DE PROGRAMACIÓN DE LA PRODUCCIÓN. *Primer Congreso de Logística y Gestión de la Cadena de Suministro*.

SALAZAR PINTO, P. (2009). ALGORITMO HÍBRIDO AUTO CONFIGURADO PARA OPTIMIZACIÓN ESTRUCTURAL. *Universidad Industrial de Santander*.

SAN MARTÍN CONTRERAS, O. A. (2006). OPTIMIZACIÓN DE LA PRODUCCIÓN PARA PROBLEMAS DE 'FLOW-SHOP' MULTIOBJETIVO MEDIANTE LA UTILIZACIÓN DE METAHEURÍSTICAS. *Universidad del Bío-Bío*.

SERVIN OCHOA, D. (2004). EQUILIBRADO DE LINEAS DE ENSAMBLE EN A INDUSTRIA DEL VESTIDO: UN ENFOQUE MEDIANTE ALGORITMOS GENÉTICOS HÍBRIDOS. *Instituto Politecnico Nacional, Mexico*.