UNIVERSIDAD DE CONCEPCIÓN - CHILE

FACULTAD DE INGENIERIA

DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

Diseño de sistema inteligente de manufactura para la minimización de la tardanza en el problema de Job Shop Scheduling

por

“Hernán Patricio Leiva Torres”

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniero Civil Industrial

Profesor Guía:

“Carlos Enrique Herrera López”

Profesor Co-guía:

“Patricio Antonio Saez Bustos”

Concepción, julio de 2022

© 2022 Hernán Patricio Leiva Torres

Se autoriza la reproducción total o parcial, con fines académicos, por cualquier medio o procedimiento, incluyendo la cita bibliográfica del documento

SUMARIO

El presente estudio se realiza en el marco de uno de los problemas más clásicos de la optimización combinatoria y el área de la administración de operaciones, el Job Shop Scheduling Problem. Este problema también es considerado uno de los más complejos, teniendo una denominación NP-Hard en la teoría de la complejidad computacional. La resolución de este problema representa un gran desafío para las organizaciones manufactureras, debido a los acotados tiempos de respuesta del mercado actual. Se diseñó el sistema controlado por productos inteligentes (PDS) basado en una programación multiagente capaz de resolver el problema, de minimizar la tardanza máxima, y probado con distintas instancias disponibles en la literatura. Se comparan los resultados y tiempos de respuestas con la resolución del problema mediante la programación lineal entera. El PDS propuesto obtuvo mejores resultados en un 11% de las instancias pequeñas, en un 38% de las instancias medianas y en un 100% de las instancias grandes. Además, se tiene un tiempo de ejecución coherente con el funcionamiento del sistema, donde en promedio se tiene un 1.34 , 2.64 y 5.28 segundos para problemas de pequeña, mediana y gran escala respectivamente.

Palabras Claves: Job shop Scheduling, Programacion lineal entera, Producto inteligente, Sistema multiagentes. Sistemas impulsados por el producto.

ABSTRAC

The present study is carried out in the framework of one of the most classical problems of combinatorial optimization and the area of operations management, the Job Shop Scheduling Problem. This problem is considered one of the most complex, having an NP-Hard denomination in the theory of computational complexity. The resolution of this problem represents a great challenge for organizations with the presence of a productive system, due to the limited response times provided by the current market. For this purpose, an intelligent product-driven system (PDS) was designed based on a multi-agent programming capable of solving the problem, minimizing the maximum delay, and being tested through different instances studied in the literature. The results and response times in solving the problem are compared with optimal methodologies such as integer programming. The proposed PDS obtained better results in 11% of the small instances, in 38% of the medium instances and in 100% of the large instances. In addition, the execution time is consistent with the system performance, with an average of 1.34, 2.64 and 5.28 seconds for small, medium, and large-scale problems, respectively.

Keywords: Job shop Scheduling, Integer linear programming, Intelligent Product, Multi-agent system, Product Driven System.

TABLA DE CONTENIDOS

Capítulo 1: Introducción y Objetivos *8*

1.1 Introducción 8

1.2 Objetivos del estudio 9

Capítulo 2: Marco Teórico *11*

2.1. Sistema de Planificación de la producción 11

2.2. Reglas de secuenciación 12

2.3. Programación Entera 13

2.3. Sistema Inteligentes de Manufactura 14

2.4. Productos Inteligentes (Intelligent producto - IP) 16

2.5. Sistema de Agentes 18

Capítulo 3: Revisión Bibliográfica *20*

3.1. Resolución del problema Job Shop minimizando la Tardanza 20

3.2. Sistemas inteligentes de Manufacturas 22

3.3. Product Driven control systems (PDCS) 22

Capítulo 4: Material y Métodos *24*

4.1. Job Shop Scheduling Problem 24

4.2. Representación de Datos 25

4.3. Estimación de las fechas de vencimiento (Due dates) 26

4.4. Desarrollo del Algoritmo de Programación Entera 26

4.4.1. Algoritmo de Lectura 28

4.4.2. Modelamiento del problema 29

4.5. Desarrollo del Sistema Multiagentes 30

4.5.1. Proceso: Lectura e interpretación de Datos 31

4.5.2. Proceso: Operación entre agentes 36

Capítulo 5: Resultados y Análisis *40*

5.1. Experimentos 40

5.2. Resultados 41

5.2.1. Resultados de instancias: pequeña escala 41

5.2.2. Resultados de instancias: médium escala 42

5.2.3. Resultados de instancias: gran escala 43

5.3. Análisis de Resultados 44

Capítulo 6: Conclusión *45*

Bibliografías *46*

Anexos *49*

ÍNDICE DE FIGURAS

Figura 2.1 Sistemas de control tradicionales 15
Figura 2.2 Enfoque tradicional y distribuido 16
Figura 2.3 Ejemplo Producto inteligente – Tarro de salsa de espaguetis 17
Figura 2.4 Ejemplo Producto inteligente – Tarro de salsa de espaguetis 18
Figura 2.5 Ejemplo Plataforma basada en agentes 19
Figura 4.1 Ejemplo de grafo disyuntivo JSSP 25
Figura 4.2 Diagrama de Flujo del sistema multiagente 31
Figura 4.3 Variable de entrada instancia 32
Figura 4.4 Creación de tortuga 33
Figura 4.5 Inspección de parcela (0,0) 35
Figura 4.6 Agentes creados – Ejemplo instancia ft06 36
Figura 4.7 Diagrama de secuencia Sistema Multiagente 39
Figura 5.1 Grafico de instancias de pequeña escala 42
Figura 5.2 Grafico de instancias de mediana escala 43
Figura 5.3 Grafico de instancias de gran escala 44

ÍNDICE DE TABLAS

Tabla 2.1: Medidas de desempeño 12
Tabla 4.1: Ejemplo de instancia 25
Tabla 4.2: Pseudocodigo procedimiento lectura 29
Tabla 4.3: Pseudocodigo creación de modelo 30
Tabla 4.4: Pseudocodigo Netlogo - procedimiento lectura 32
Tabla 4.5: Pseudocodigo Netlogo - Guardado de datos 34
Tabla 4.6: Pseudocodigo Netlogo - Calculo due dates 37
Tabla 4.7 Pseudocodigo Netlogo - Desplazamiento de tortugas 38
Tabla 4.8: Pseudocodigo NetLogo - Operar 39
Tabla 5.1: Instancias estudiadas 40
Tabla 5.2: Resultados Instancias pequeñas 41
Tabla 5.3: Tiempos promedios de ejecución – pequeña escala 41
Tabla 5.4: Resultados Instancias medianas 42
Tabla 5.5: Tiempos promedios de ejecución – mediana escala 43
Tabla 5.6: Resultados Instancias grandes 43
Tabla 5.7: Tiempos promedios de ejecución – mediana escala 44

Capítulo 1: Introducción y Objetivos

1.1 Introducción

La gestión de operaciones de una organización es la responsable de la producción y la entrega de bienes o servicios. Esta gestión debe tomar decisiones para administrar el proceso de transformación, es decir, cuando los recursos productivos pasan a ser productos o servicios finalizados. Estos productos están enfocados en el logro de distintos objetivos, tales como: satisfacción del cliente, control de inventarios y utilización de recursos. Por esto se han desarrollado distintas técnicas que permiten obtener una transformación de los recursos, direccionándolos al objetivo deseado (Roger R. Schroeder, Susan Meyer G. y M. Johnny Rungtusanatham, 2011).

Existen distintos sistemas de producción dependiendo del flujo de los productos o procesos. Entre los sistemas de producción más conocidos se destacan: los sistemas continuos de fabricación, los sistemas "Flow shop" o fabricación en línea y los sistemas "Job shop" o taller de tareas. Específicamente, en los sistemas Job Shop se fabrican lotes de una variedad de productos, donde cada producto presenta una secuencia de operaciones y un flujo a través del taller. Generalmente estos sistemas productivos trabajan bajo un esquema make to order (MTO) o fabricación a partir del pedido de cliente, lo cual conlleva a planificar las tareas con una fecha de entrega máxima (Josefa Mula, Raúl Poler y José P. García, 2006).

Actualmente, los nuevos desafíos y exigencias en el mercado manufacturero, sumado al contexto climático y al desarrollo tecnológico de la sociedad, han generado un desajuste en la oferta y demanda de productos (Yin et al., 2018). Es por esto, que las empresas productivas se han visto obligadas a adaptar sus procesos productivos, implementando de nuevas tecnologías y técnicas que permitan generar una fabricación inteligente (Fu Yaping et al, 2021) y productos inteligentes capaces de controlar, planificar y automatizar estos procesos (Tharumarajah, 2003).

Con la llegada de la era del Big data, se han almacenado muchos datos en todo el ciclo de vida del producto. Los datos de productos recopilados contienen una gran cantidad de información que brinda nuevas oportunidades para mejorar la eficiencia de la producción y la competitividad (Fu Yaping et al, 2021). Son utilizados en la fabricación inteligente para la generación de productos inteligentes (IP), entidad física tangible y con características únicas, capaces de tomar decisiones y rastreables durante el proceso de producción (Wu et al., 2019) (Meyer et al., 2009).

Los sistemas basados en agentes (sistemas multiagentes) útiles para desarrollar productos inteligentes, pues comparten definiciones y conceptos similares (I. Kovalenko et al, 2019). Permiten generar soluciones robustas y flexibles, a partir de las interacciones de agentes-productos (entidades virtuales con representación fisica), donde su comportamiento esta alineado a los objetivos definidos, además de entregar seguridad y confiabilidad en tiempo real ante disrupciones (Maturana et al, 1999).

En este estudio, se resuelve el Job Shop Scheduling problem (JSSP) a través de dos métodos, ambos con el objetivo de minimizar la tardanza. El primero es la resolución del problema mediante la programación lineal entera (PLE), técnica de optimización tradicional que busca el óptimo a través del análisis de todas las combinaciones de variables (H. P. Williams, 1975). El segundo método corresponde al diseño y simulación de un sistema controlado por productos (PDCS). En estos sistemas, los productos son quienes "impulsan" su propia fabricación, transformándose en agentes activos en el proceso de toma de decisiones (Herrera et al., 2014). El objetivo principal es comprobar el rendimiento de los sistemas impulsados por productos (PDCS) mediante la resolución de instancias de distintas escalas, obtenidas de la literatura. Contrastando resultados y tiempos de ejecución con el método de PLE.

En el capítulo 2 se presenta el marco teórico de los conceptos que rodean al objetivo de la investigación. Capitulo 3 Seguido del análisis de la bibliográfica asociada a los dos métodos de resolución presentados y a la minimización de la tardanza. En el capítulo 4 se presentan los materiales y métodos requeridos para el desarrollo y resolución de ambas metodologías, y se explica exhaustivamente el diseño de estos. Finalmente, en el capítulo 5 se presentan las instancias seleccionadas junto con los resultados y tiempos promedios de ejecución de PLE y PDS. Se contrastan ambos resultados, analizando y concluyendo el rendimiento de los sistemas impulsados por productos en problemas Job Shop de distinta complejidad.

1.2 Objetivos del estudio

Objetivo general: Proponer un diseño de una metodología basada en control por el producto para la minimización de la tardanza en sistemas tipo Job Shop

Objetivos específicos:

• Estudiar y comprender el problema Job Shop.

• Investigar y analizar los distintos métodos utilizados en la literatura para resolver el problema.

• Investigar métodos y herramientas para el diseño de un sistema inteligente de manufactura.

• Analizar las distintas instancias utilizadas en la literatura.

• Estudiar el funcionamiento del software Netlogo y experimentar con distintas estrategias mediante agentes.

• Comparar los resultados y tiempos computacionales obtenidos por los PDS, con métodos tradicionales.

Capítulo 2: Marco Teórico

2.1. Sistema de Planificación de la producción

El principal desafío actual en las empresas manufactureras es el correcto funcionamiento de la gestión de operaciones para el logro de los objetivos propuestos. Para que la programación (o gestión) de operaciones funcione eficientemente, requiere de un sistema de planificación y control, el cual sea capaz de planear las tareas eficazmente y corregir a medida que se requiera (control). Debe ser capaz de responder necesidades como: fechas de entrega de los productos, identificación de cuellos de botella en la producción, fecha de inicio de cada actividad o tarea en particular y garantizar la finalización de los trabajos a tiempo (Roger R. Schroeder et al, 2011). El correcto funcionamiento de los sistemas de planificación de la producción para una organización manufacturera implica una mejora en la capacidad productiva, optimiza la utilización de materias primas o insumos, reduce los tiempos donde no se genera producción (tiempos muertos) y controla los inventarios de productos (Alemão et al., 2021).

Dentro de los sistemas de planificación se encuentra la programación de actividades, donde resolver la asignación de actividades o recursos, equivale a resolver el conocido problema de programación JSSP (Job Shop Sheduling Problem). Este problema se define como la asignación de recursos limitados a tareas a lo largo del tiempo y tiene como finalidad la optimización de uno o más objetivos definidos por el ente administrador (Phanden et al., 2011).

Existen variantes para los problemas de programación de una industria manufacturera, las cuales se relacionan con la secuenciación de operaciones u ordenamiento de actividades, denominadas patrón de flujo. Donde, si todos los trabajos presentes en la programación presentan una misma secuencia de ordenación de las operaciones, el problema se conoce como flow-shop. Por otro lado, si existe un ordenamiento determinado para cada uno de los trabajos, y a su vez existe un orden de uso de cada uno de los recursos (o maquinas) por parte de las operaciones, el problema se denomina Job-shop. En caso de no existir una restricción de ordenamiento, el problema pasa a llamarse open-shop (Peña V. & Zumelzu L., 2006).

En la literatura se han propuesto distintos objetivos a lograr en la secuenciación de máquinas. Donde los objetivos basados en el tiempo son los más frecuentes. En la siguiente tabla se muestran las

medidas de desempeño más comunes propuestas por distintos investigadores (Jaime A., Carlos A. & Fabian, 2013) (Ver Tabla 2.1).

Medida	Significado	Formula
Tiempo total de flujo	Tiempo total requerido para terminar la ejecución de los n trabajos	$\sum C_i$
Atraso del trabajo j	Tiempo de atraso del trabajo respecto a su fecha de entrega	$l_i = C_i - d_i$
Tardanza Máxima	El máximo tiempo de retraso	$\max(l_i, \dots, l_n)$
Atraso total	Tiempo de retraso de los n trabajos	$\sum l_i$
Tiempo de flujo del trabajo j	Es la suma de los tiempos de movimiento entre operaciones, tiempos de espera por no disponibilidad de las maquinas, tiempos de proceso (incluyendo tiempos de alistamiento) y retrasos resultantes de fallos en los recursos	$Tf_i = c_i - r_i$
Inventario en proceso (WIP)	Cualquier trabajo en una línea de espera, en movimiento o en proceso considerado WIP	$\sum Trabajos\ en\ proceso$
Tasa de utilización de las maquinas	Porcentaje de tiempo que las maquinas permanecen ocupadas	$U = \frac{tiempo\ de\ uso}{tiempo\ trab.disponible}$

Tabla 2.1: Medidas de desempeño

Fuente: Jaime A., Carlos A. & Fabian, 2013

2.2. Reglas de secuenciación

En las empresas cuyo sistema productivo corresponde al Job Shop, se aplican distintas reglas de secuenciación para la producción. Estas reglas se basan en un criterio, el cual permite elegir los trabajos según el índice más favorable de acuerdo con la definición del criterio, y procesarlos en las respectivas maquinas.

En la literatura se ha estudiado y analizado la aplicación de varias técnicas, destacando el estudio reportado por Ang. Et. Al (2011), donde se experimenta varias reglas de secuenciación mediante la simulación. Entre las reglas utilizadas destacan:

- SPT (shortest processing time): Procesamientos de trabajos según menor tiempo de proceso

- LPT (Largest processing time): Procesamientos de trabajos según mayor tiempo de procesamiento
- SST (Shortest setup time): Procesamientos de trabajos según menor tiempo de preparación
- EDD (earliest due date): Procesamientos de trabajos según fecha de entrega más cercana
- FIFO (Firts come first served): Procesamientos de trabajos según orden de llegada a las maquinas

Comparando y midiendo el desempeño de cada regla bajo diferentes funciones objetivos como, tiempo medio de flujo, atraso promedio, tardanza máxima, etc. Similar es el análisis realizado por Tavakkoli-Moghaddam y Daneshmand-Mehr (2005), donde se utiliza un modelamiento de redes que minimizan el makespan, donde se evalúan las reglas de desempeño FIFO (firts come first served), LIFO (last input first output), HVF (Highest Value First), LVF (Lowest Value First), entre otros.

2.3. Programación Entera

Un modelo de programación Entera es aquel cuya solución óptima tiene sentido solamente si todas las variables de decisión toman valores enteros no negativos, permitiendo incorporar en el modelamiento matemático algunos aspectos que quedan fuera del alcance de los modelos de programación lineal. En situaciones reales, el análisis contempla "decisiones sí o no", las que pueden representarse con variables denominadas binarias. En este sentido los algoritmos de resolución de los modelos de Programación entera difieren a los utilizados en los modelos de Programación Lineal, destacándose entre ellos el Algoritmo de Ramificación y Acotamiento, Branch and bound, Branch and cut, entre otros (H. P. Williams, 1975). Los Modelos de programación entera se puede clasificar en 2 grandes áreas, programación entera mixta y programación entera pura:

- Programación Entera Mixta: Cuando sólo es necesario que algunas de las variables sean enteras y el resto continuas, el modelo recibe el nombre de problema de Programación Lineal Entera Mixta. Esta clasificación incluye modelos que además de tener variables enteras no negativas y variables continuas, tienen también variables binarias (Hillier y Lieberman, 2002). A esta categoría pertenecen aquellos problemas de optimización que consideran variables de decisión enteras o binarias, pero no de forma exclusiva. De esta forma, la programación entera

mixta puede considerarse como un hibrido entre distintas categorías de modelamiento, siendo un caso típico aquel que considera la mezcla de variables enteras y variables continuas.

- Programación Entera Pura: En esta categoría encontraremos aquellos modelos de programación Entera que consideran exclusivamente variables de decisión que adoptan valores enteros o binarios. Un ejemplo son las siguientes aplicaciones: Problema de asignación, problema de la mochila, etc. Notar que en los problemas anteriores el conjunto de las soluciones factibles es finito. Esto ocurrirá generalmente con los problemas de programación entera (H. P. Williams, 1975).

2.3. Sistema Inteligentes de Manufactura

Un sistema de manufactura es un conjunto de actividades que interactúan con un conjunto de recursos para obtener un producto. Las actividades son los procesos de fabricación: maquinación, manejo de materiales y procesos de información; que son necesarios para la producción. El objetivo de este sistema es refinar las operaciones de fabricación, reciclaje y ahorro de recursos para creación de productos, y transferir conocimiento a futuras generaciones para generar un avance significativo en el profesionalismo en la manufactura (McFarlane, 1995).

Los sistemas de manufactura o industrias manufactureras presentan una gran relevancia en la historia moderna. La primera revolución industrial, industria manufacturera, fue una división entre "el mundo antiguo y el nuevo". Desde entonces, la industria manufacturera ha generado riqueza, empleo y calidad de vida, al mismo tiempo que promueve y sostiene los servicios, la educación, la investigación y el desarrollo. Actualmente, los desafíos a los que se enfrentan los sistemas manufactureros están enfocados a la competitividad y a las fluctuaciones del mercado, provocados por el avance de la sociedad, la destrucción del medioambiente a escala global y disminuyendo los recursos naturales utilizados como materias primas (Yoshikawa, 1994).

El enfoque tradicional de los sistemas de control de fabricación basados en estructuras de control centralizadas o jerárquicas presenta buenas características en términos de productividad, esencialmente debido a sus capacidades intrínsecas de optimización. Sin embargo, no soportan de

manera eficiente los requisitos actuales impuestos a las exigencias actuales en los sistemas de fabricación, como la flexibilidad, expansibilidad, agilidad y Re-configurabilidad (Leitao P., 2008) (Ver figura 2.1).

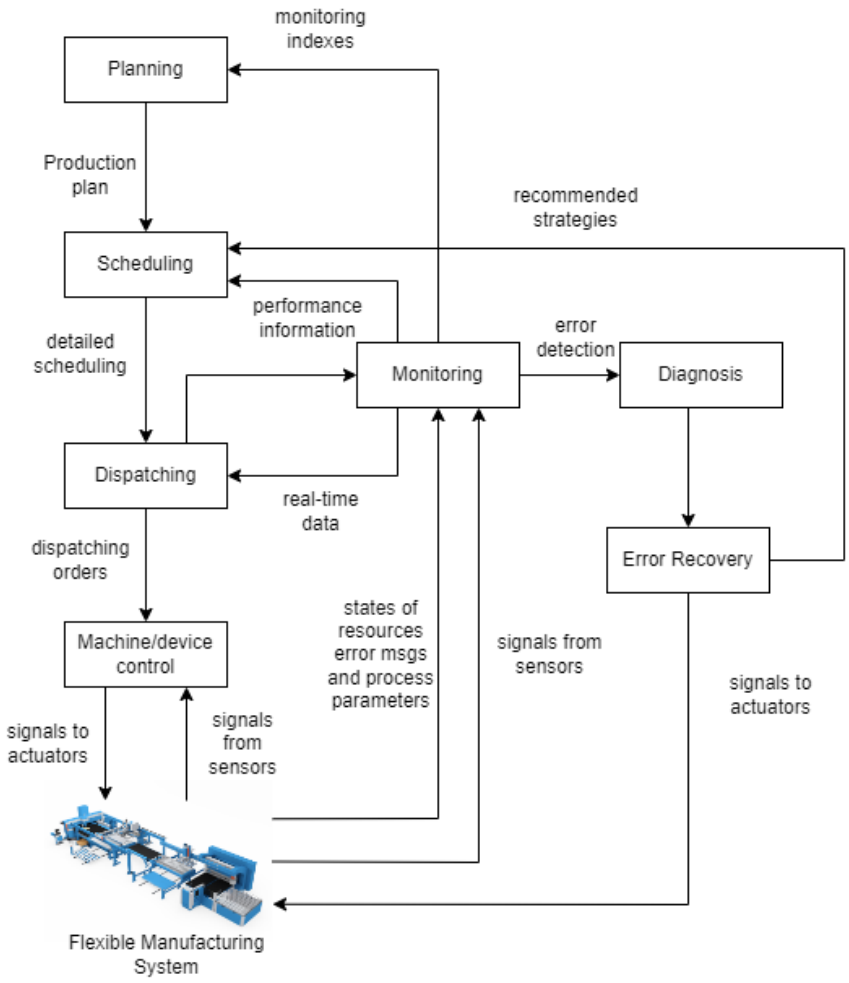

Figura 2.1 Sistemas de control tradicionales

Fuente: Leitao P., 2008

Respecto a lo anterior, se requieren de una nueva clase de sistemas de control de fabricación inteligente y distribuidos para llenar el vacío dejado por los enfoques centralizados, en los que, usando un enfoque distribuido, un problema complejo se puede dividir en varios problemas pequeños, cada uno mapeado una unidad de control. Cada unidad de control es autónoma teniendo sus propios objetivos, conocimientos y habilidades, y encapsulando funciones inteligentes. Las decisiones de control global son determinadas por la interacción entre unidades. Estas unidades deben exhibir varias

características importantes, como la Re-configurabilidad, la solidez, la capacidad de conexión, el aprendizaje y la reutilización (Leitao P., 2008).

Un sistema de control de fabricación que satisfaga los requisitos anteriores, se le denomina fabricación inteligente, y funciona de forma totalmente diferente en comparación con los sistemas de control centralizados tradicionales. El cambio del enfoque centralizado tradicional al nuevo enfoque distribuido e inteligente se ilustra en la siguiente figura (Leitao P., 2008). (Ver Figura 2.2):

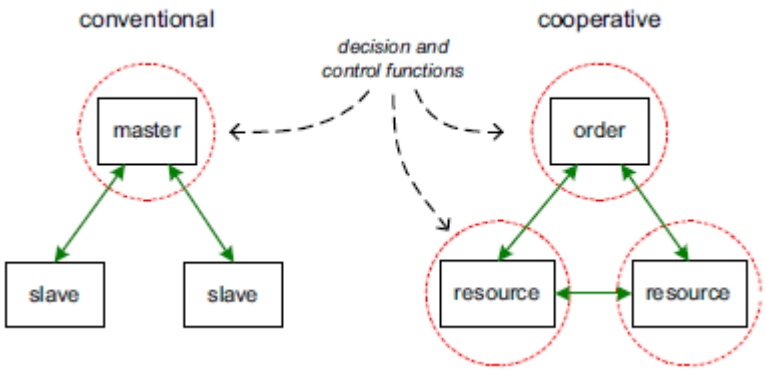

Figura 2.2 Enfoque tradicional y distribuido

Fuente: Leitao P., 2008

2.4. Productos Inteligentes (Intelligent producto - IP)

El desarrollo de la tecnología en el área de la inteligencia artificial (IA) , ha permitido instalar distintos elementos inteligentes en las industrias manufactureras, logrando obtener un proceso de fabricación inteligente y flexibilidad en respuestas a las demandas dinámicas del mercado (Yixiang Feng et al, 2020). La implementación de estas tecnologías en los sistemas productivos, también han permitido dotar de inteligencia a los recursos y/o productos (Liang et al., 2018). Estos productos son representados como entidades del sistema rastreables, con características únicas asociadas, capaces de interpretar y tomar decisiones (Meyer et al., 2009).

El autor Mc. Farlane (Wong et al., 2002) define un producto inteligente como una representación física y basada en información de un producto. El autor enseña en la siguiente figura un ejemplo de

Inteligent Product (IP), donde el tarro o frasco de salsa de espagueti es el producto físico, la representación del producto basada en la información se almacena en la base de datos, y la inteligencia la proporciona el agente que toma las decisiones (Meyer et al., 2009) (Ver Figura 2.3).

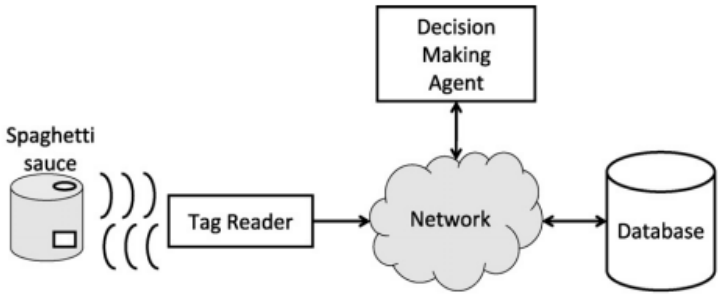

Figura 2.3 Ejemplo Producto inteligente – Tarro de salsa de espaguetis

Fuente: Wong et al, 2002

Según Mc. Farlane, las principales características que resumen a un producto inteligente (IP) son (Wong et al., 2002):

- Poseer una identidad única
- Tener capacidad de comunicarse eficazmente con su entorno
- Retener y guardar datos sobre sí mismo
- Desplegar un lenguaje para mostrar sus características, requisitos de producción, etc.
- Ser capaz de participar o tomar decisiones relevantes para su propio destino.

Con respecto a estas características, se define una clasificación de inteligente de dos niveles para los productos. Donde si el IP solamente presenta identidad única y es almacena datos sobre sí mismo, este está orientado a la información y se denomina producto con inteligencia de producto nivel 1. Un IP de nivel 2 cubre todos los puntos anteriormente mencionados, y se denomina orientado a decisiones. Esta clasificación de niveles se puede resumir en el siguiente diagrama ortogonal tridimensional (Meyer et al., 2009) (Ver Figura 2.4):

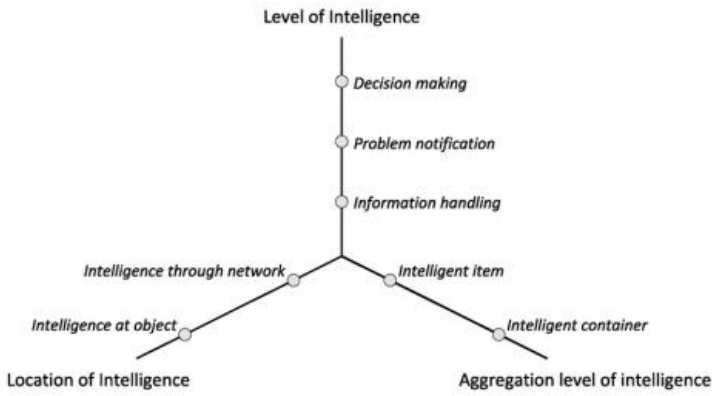

Figura 2.4 Ejemplo Producto inteligente – Tarro de salsa de espaguetis

Fuente: Meyer et al., 2009

La aplicación de los productos inteligentes entrega información en tiempo real de su estado y su entorno, que, a través de su seguimiento, permiten la identificación de cuellos de botella o fallas en algún punto de la producción. Lo anterior contribuye al funcionamiento del proceso de control y planificación producción, ya que entrega datos que permiten el correcto funcionamiento del sistema y rápidas respuestas ante distintos tipos de emergencia que se relacionen con la producción a planta (Wong et al., 2002).

2.5. Sistema de Agentes

Un sistema multiagente (MAS) o inteligencia artificial, es un sistema compuesto por agentes inteligentes, los cuales interactúan mediante protocolos de comunicación. Estos agentes, se interpretan como entidades autónomas con capacidad de resolver problemas computacionales y operar de forma efectiva en ambientes dinámicos y abiertos, en busca del objetivo para el cual fueron diseñados.

Estos sistemas requieren de plataformas que permita el desarrollo adecuado de agentes. El uso de este tipo de sistemas es ampliamente utilizado para problemas complejos que no pueden ser resueltos por un sistema monolítico o centralizado, permitiendo obtener resultados robustos, en un ambiente que requiera adaptaciones al sistema como extensiones, cambios rápidos o reconfiguraciones (Marik, 2005).

Las soluciones obtenidas mediante un a sistemas multiagentes presentan una gran serie de beneficios, destacando la robustes de sus resultados, permitiendo mejorar el desempeño del sistema ante una gran variedad de circunstancias o fallas, logrando reorganizarse y responder ante estas disrupciones. Otro beneficio, es su capacidad de reconfiguración, es decir, el sistema es capaz de cambiar, agregar o remover módulos de software en el momento requerido.

Los sistemas basados en agentes son programados e implementados en arquitecturas de software especializadas que proveen al programador un conjunto de funciones que permite que el sistema trabaje como una aplicación representada por Agentes. El objetivo de los softwares es proveer un marco de trabajo que asegure: la administración de Agentes, envío de mensajes y comunicaciones administración del sistema, además de la entrega de flexibilidad para las demandas dinámicas del mercado (Yixiang Feng et al., 2022).

El uso de tipo de plataformas o sistemas para productos inteligentes es beneficioso. En primer lugar, cuando existe una gran cantidad de productos, el proceso de control de productos es más complejo. Este proceso se puede desarrollar de mejor forma, logrando que los productos sean autónomos. De esta forma, los productos inteligentes pueden desarrollar la mayoría de sus taras repetitivas de forma automatizada, gracias a sus capacidades de conocimiento y razonamiento. En segundo lugar, los productos inteligentes deberían poder detectar y reaccionar ante cambios en el entorno, donde los agentes pueden ayudar de forma proactiva al producto y tratar de lograr los objetivos dado el cambio del entorno. Con respecto al ambiente, los agentes pueden descubrir información mediante la comunicación con otros agentes de otros productos. Lo anterior deja en claro que la utilización de agentes presenta características similares y deseables a los que requiere los productos inteligentes.

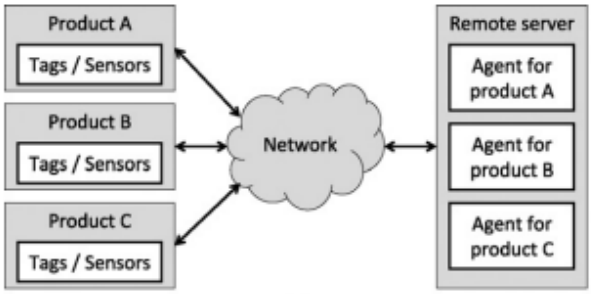

Figura 2.5 Ejemplo Plataforma basada en agentes

Fuente: Meyer et al., 2009

Capítulo 3: Revisión Bibliográfica

3.1. Resolución del problema Job Shop minimizando la Tardanza

La investigación y resolución del problema Job Shop ha venido desarrollándose en los últimos años, y ha sido uno de los objetos de estudio más relevantes en la literatura relacionada con la secuenciación de tareas. Las técnicas utilizadas en la literatura van desde: la utilización de distintas reglas de despacho hasta algoritmos paralelos de ramificación y poda altamente sofisticados, heurísticas y metaheurísticas basadas en "cuellos de botella" (bottleneck based heuristics), y algoritmos genéticos paralelos. además, dichas técnicas han sido formuladas desde un amplio espectro de investigadores, desde científicos de gestión hasta expertos de la producción (Peña V. & Zumelzu L., 2006).

Con respecto a los objetivos de la secuenciación de operaciones destaca el análisis del tiempo de flujo (o makespan), donde se busca minimizar el tiempo final de todos los trabajos en un sistema productivo. Dentro de estos objetivos, se encuentra el estudio de la minimización de la tardanza y/o tardanza total. Esta medida de desempeño es analizada por los autores para responder a problemáticas como el cumplimiento de fechas de entrega de los productos, y la capacidad de respuesta de los sistemas productivos ante emergencias relacionada con el aumento o modificación de la demanda (Pan et al,1997).

Dentro del análisis de la tardanza destaca el trabajo realizado por Víctor Peña y Lillo Zumelzu (2006), en el cual se presenta un algoritmo RFL (really full lookahead) para la minimización de la tardanza máxima del problema de Job Shop, el cual, mediante distintas instancias realiza una comparativa de rendimientos (brensmarck). El algoritmo genera una solución inicial a partir del trabajo inicial ingresado (como parámetro), para luego utilizar una función recursiva encargada de ir generando soluciones factibles mediante una combinatoria de las operaciones. Una vez que se ha encontrado una solución se procede a asignar los tiempos mediante RFL, es decir, cuando se asigna el tiempo de inicio y de término de una operación también se asignan los tiempos de inicio para todas las operaciones siguientes de ese trabajo, para luego asignar también los tiempos de inicio de todas las operaciones no instanciadas que ocupen la misma máquina. Luego, se evalúa la solución buscando minimizar la tardanza máxima (Peña V. & Zumelzu L., 2006).

N. Zribi et al en su estudio de *"Minimizing the total tardiness in a flexible job-shop"* propuso un enfoque jerárquico basado en la idea de descomponer el problema en subproblemas para la reducción de complejidad, estos subproblemas representan la asignación de máquinas y la secuenciación de operaciones en las maquinas elegidas. Para el primer subproblema correspondiente a la asignación, el autor propone la solución mediante la heurística Tabú Search la cual se basa en la minimización del criterio propuesto por Kacen (2002), el cual calcula las cargas de trabajo de las maquinas. Para el segundo subproblema de secuenciación, se aplica un algoritmo hibrido con dos técnicas: algoritmos genéticos y Tabú Search. Las ventajas obtenidas por el autor en este tipo de enfoque consisten en la resolución del problema de asignación, donde en tiempos computacionales aceptables, se obtienen buenos resultados. Además, al estudiar instancias en P. Brandimarte (1993), se define un criterio de generación de fechas de vencimiento y liberación, estas relacionadas con la media de los teimpos de procesamientos y tiempos de fin de flujo de los trabajos (N. Zribi et al, 2006).

Chen et al (2012) busca desarrollar una regla de despacho que minimice heurísticamente la tardanza de los trabajos pesados en un Flexible Job Shop, para problemas de una empresa de fabricación pesada representada como un taller de fabricación a pedido de gran escala. La regla desarrollada RR modifica con sesgo de peso (WBME) es flexible a la minimización de tardanza en varios niveles de prioridad especificados y resistente a diversos grados de congestión a nivel de sistema y a la estrechez de las fechas de vencimiento. Esta regla se comprueba con heurísticas de programación para minimizar la tardanza de los trabajos con otras reglas de despacho conocidas en la literatura. Donde se comprueba el rendimiento robusto a las variaciones de asignación y la tendencia central de los tiempos de procesamiento, y se mantiene a través de los niveles de truncamiento en la métrica de tardanza. Además de la integración de un parámetro de polarización agrega una característica adicional a la regla de despacho, esta permite a un gerente de producción cambiar rápidamente el cronograma para cumplir con los objetivos de nivel de producción para el rendimiento de tardanzas. La regla WBMR aprovecha los puntos fuertes de la programación basada en el tiempo de procesamiento y la fecha de vencimiento de acuerdo con los niveles de utilización del sistema. Tiene una complejidad lineal y, como tal, es adecuado para horarios móviles. En general, la regla WBMR es una combinación lineal de otras reglas de despacho que se multiplica por una función que sesga el cronograma en función de las ponderaciones del trabajo para lograr el rendimiento de tardanzas objetivo (Chen et al, 2012).

3.2. Sistemas inteligentes de Manufacturas

El rápido desarrollo de las tecnologías conduce a una creciente complejidad en el desarrollo de productos, lo que provoca que las empresas tengan que lidiar con información cada vez mayor. Lo que consecuencia, vuelve cada vez más difícil para las empresas manufactureras completar las transacciones relacionadas con el desarrollo de productos de forma independiente dentro del límite de tiempo prescrito. Es por lo que Yu-Guodong et al (2015) propone la solución de un sistema de programación de operaciones mediante un enfoque multiobjetivo, donde enfoca su algoritmo al diseño colaborativo de productos, considerando emergencias. El autor se basa en uno de los principales objetivos de los sistemas de manufactura inteligente, que es garantizar que todas las tareas tengan éxito con respecto a su fecha de vencimiento, además de lograr asegurar y garantizar el costo del diseño y la satisfacción del cliente. En este estudio, se supone una organización que presenta diseñadores y productos que se descomponen en distintas tareas, uno de los objetivos del algoritmo es asignar los diseñadores de manera que se minimice el makespan de las tareas. El otro objetico, se basa en la sinergia multidisciplinaria de los sistemas distribuidos, donde se asumen pedidos urgentes o cambios en los requisitos de los clientes, generando complejidad e incertidumbre en el procedimiento de las tareas. Es por esto que se propone un objetivo que minimice la tardanza máxima, para que la producción tenga en cuenta las emergencias posibles y el impacto de estas en el desarrollo y producción de los productos. El problema de programación PDC (productos colaborativos) entrega resultados que indican la reducción de la penalización de la tardanza ante accidentes en la programación de operaciones, además de mantener relativa estabilidad del sistema (Yu-Goudong et al, 2015)

3.3. Product Driven control systems (PDCS)

Si bien la utilización de Product driven systems son cada vez mas considerados y estudiados por distintos autores para el análisis de los sistemas productivos. No se logró encontrar la implementación de PDCS para la minimización de la tardanza máxima. Sin embargo, en la literatura se encuentran metodologías PDCS basadas en agentes, con el objetivo de la minimización del flujo total o makespan, donde uno de los estudios a destacar es Herrera et al (2014) , donde se proponen simulaciones de los procesos de planificación y control, para analizar la coordinación de los objetivos multinivel, además

de simular la coordinación entre los niveles táctico y operativo. El objetivo es coordinar las decisiones a diferentes niveles utilizando métodos centralizados y distribuidos, estos se modelan como holones. En cuanto a los resultados, indican los autores la coordinación entre las decisiones centrales y locales para un enfoque basado en PDCS es eficiente. En cuanto a los resultados de Cmax , se observa robustes en los diferentes tipos de decisiones. son especialmente interesantes, ya que revelan una ganancia neta sin caída de la estabilidad, y demuestran que se puede alcanzar la "robustez" en diferentes tipos de decisiones.

Capítulo 4: Material y Métodos

4.1. Job Shop Scheduling Problem

El Job Shop Scheduling es un problema de optimización tipo combinatorio categorizado en la clase NP-hard (Van Dyke Parunak, 1991), enfocado en la planificación de tareas, donde uno de los objetivos principales es la disminución de tiempos en la planificación o makespan. El problema consta en planificar una cantidad "n" de trabajos, con $n \in J/\{J_1, \dots, J_n\}$, sobre "m" maquinas o recursos, con $m \in R/\{R_1, \dots, R_m\}$. Cada trabajo contiene un conjunto O de operaciones o tareas $\{\theta_{i1}, \dots, \theta_{im}\}$, las cuales deben ser realizadas en forma secuencial. Cada operación θ_{ij} presenta un tiempo de procesamiento du_{ij}, el cual es ininterrumpido y requiere una sola máquina para su ejecución. Las operaciones de un mismo trabajo deben ser procesadas en un orden determinado, de forma que la operación $\theta_{(i+1)j}$, no puede procesarse antes que θ_{ij} complete su tiempo (Udomsakdigool & Kachitvichyanukul, 2008).

Las restricciones del problema se pueden resumir de la siguiente manera:

- Una Tarea no puede visitar una misma máquina dos veces
- No hay restricciones de precedencia entre operaciones de distintas tareas
- Las operaciones no se pueden interrumpir
- Cada máquina puede procesar sólo una tarea a la vez

El principal objetivo es encontrar una planificación factible para cada una de las tareas, que minimice el makespan (Cmax) o tiempo de finalización de la última tarea. El Job Shop Scheduling problema ser representado como un grafo disyuntivo. En la siguiente figura se muestra una representación gráfica de un problema con 3 trabajos, cada uno con 3 tareas, donde el tiempo mínimo de inicio es 0 para todos ellos y el tiempo máximo de fin es 15. Los arcos están etiquetados con el costo de la tarea de origen (Florez et al., 2013) (Ver Figura 4.1).

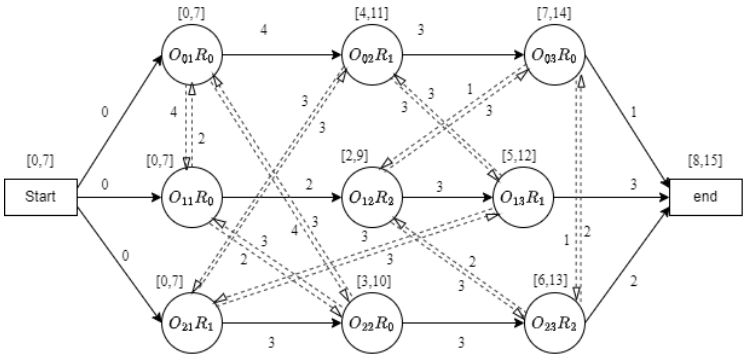

Figura 4.1 Ejemplo de grafo disyuntivo JSSP

Fuente: Víctor Peña & Lillo Zumelzu, 2006

4.2. Representación de Datos

Los datos de las instancias se presentan en un orden o especificación estándar, donde funciona de la siguiente manera:

En la primera línea hay dos números, el primero indica la cantidad de trabajos, y el segundo la cantidad de máquinas. A continuación, hay una línea para cada trabajo, donde cada una de esas líneas contiene la orden de visita a las máquinas junto con el tiempo de procesamiento correspondiente. Estas líneas comienzan del primer trabajo, seguidas de los demás trabajos en orden numérico hasta llegar al último. Cabe destacar que la numeración de los trabajos y las maquinas comienzan en 0.

Por ejemplo, una instancia que presenta un trabajo que debe procesarse por tres máquinas, donde el tiempo de procesamiento es 8 en la máquina 1, 4 en la máquina 2 y 3 en la máquina 3 y el orden en que ese trabajo debe visitar las máquinas es 2,3,1. La instancia se presentaría como (Ver Tabla 4.1):

1 3
1 4 2 3 0 8

Tabla 4.1: Ejemplo de instancia

Fuente: Elaboración propia

4.3. Estimación de las fechas de vencimiento (Due dates)

Para el estudio del Job Shop Scheduling Problem (JSSP) con el objetivo de minimizar la tardanza, se requiere establecer las fechas de vencimientos de cada uno de los trabajos. Lo anterior debido a que la gran mayoría de las instancias estudiadas en la literatura no presentan due dates. En diversas investigaciones, autores han propuestos distintos métodos de estimación para la obtención de las fechas de entrega de los trabajos, basándose a partir de la información de cada trabajo utilizando herramientas estadísticas o ponderaciones entre tiempos de procesamiento.

En este estudio se utilizará la siguiente formula observada en Victor P & Lillo Z, 2006 , donde el cálculo de los due dates para cada trabajo se obtiene partir de: la suma total de los tiempos de procesamiento multiplicados por un parámetro el cual toma el valor de 0,9. Donde este parámetro se encarga forzar tardanzas positivas para el correcto funcionamiento de las técnicas a utilizar:

$$Fecha\ de\ entrega\ del\ trabajo\ \text{i}\ (d_i): \left(\sum_{k=1}^{n} \tau_{ik}\right) * 0.9$$

Donde:

$$\tau_{ik}: Tiempo\ de\ procesamiento\ de\ la\ operacion\ \text{k}\ del\ trabajo\ \text{i}$$

4.4. Desarrollo del Algoritmo de Programación Entera

La programación lineal entera mixta es un método de resolución de problemas capaz de obtener el valor óptimo de la función objetivo debido a su forma de operar, la cual es recorrer todas las opciones combinatorias, siendo así, un buen método de comparación con nuevas estrategias de resolución. Los modelos de programación entera son considerados como una manera ineficiente de resolución.

El objetivo principal del estudio es minimizar la tardanza máxima, la cual se define como el máximo tiempo de atraso de los trabajos, donde el atraso es calculado mediante la resta del tiempo de finalización del trabajo (c_i) y su fecha de entrega o vencimiento (d_i).

Definición Formal:

- $Atraso\ del\ trabajo\ \text{i}\ (L_i)$:

$$L_i = c_i - d_i \quad , \forall i \in J$$

- $Tardanza\ del\ trabajo$ i (T_i):

$$T_i = \max(0, L_i) \vee T_i = \max(0, c_i - d_i) \quad , \forall i \in J$$

- $Tardanza\ máxima$:

$$T_{max} = \max(T_i, \dots, T_n) \quad , n \in J / \{J_1, \dots, J_n\}$$

Con respecto a lo anterior, el modelo se representa de la siguiente manera:

t_{ik}: $Tiempo\ de\ inicio\ de\ la\ operación$ k $del\ trabajo$ i

τ_{ik}: $Tiempo\ de\ procesamiento\ de\ la\ operacion$ k $del\ trabajo$ i

O_{ik}: $operación\ del\ trabajo$ i $que\ debe\ ser\ procesado\ en\ la\ maquina$ m

$por\ un\ período\ ininterrumpido\ de\ tiempo\ \tau_{ik}$

d_i: $fecha\ de\ entrega\ del\ trabajo$ i

$$\text{Minimizar } T_{max} = max\ (t_{ik} + \tau_{ik} - d_i)$$

Sujeto a:

(1) $t_{ik} \geq 0$ $\quad \forall\ i, p \in J\ /\ \forall\ k, h \in M$

(2) $t_{ik} - t_{ih} \geq \tau_{ih}$ $\quad si\ O_{ih}\ precede\ a\ O_{ik}$

(3) $t_{ik} - t_{pk} + K(1 - y_{ipk}) \geq \tau_{ik}$ $\quad y_{ipk} = 1, \quad si\ O_{ik}\ precede\ a\ O_{pk}$

(4) $t_{ik} - t_{pk} + K(y_{ipk}) \geq \tau_{pk}$ $\quad y_{ipk} = 0, \quad en\ otro\ caso$

$\quad donde\ K > \sum_{i=1}^{n} \sum_{k=1}^{m} (\tau_{pk} - \min(\tau_{pk}))$

(5) $t_{ik} + \tau_{pk} \geq d_i$ $\quad \forall\ i, p \in J\ /\ \forall\ k, h \in M$

La restricción en la ecuación (1) va enfocada a los tiempos de inicio de las tareas, y permite que las tareas no tengan inicios negativos. La segunda restricción (2) indica que, si dos operaciones de un mismo trabajo son consecutivas, el tiempo de inicio de la operación sucesora, debe ser mayor a los

tiempos de inicio de la operación antecesora sumado al tiempo de procesamiento de esta. Por otro lado (3) y (4), indican que ninguna maquina puede realizar más de una operación y ninguna operación pueda ser realizada en más de una maquina al mismo tiempo. Para que la secuencia de trabajos presente tardanza, se debe cumplir que, el tiempo de finalización de los trabajos sea mayor a su correspondiente fecha de entrega, permitiendo que no existan tardanzas iguales o menores a 0 (5).

Para poder modelar y resolver el problema se utilizará un programa en Python, donde se requiere importar la librería "Pulp", el cual es un paquete que permite el uso de herramientas de programación matemática. El desarrollo del código se centró en dos partes fundamentales para el correcto funcionamiento, la lectura de archivos y el modelamiento del problema.

4.4.1. Algoritmo de Lectura

Esta sección del código se encarga de leer los datos del archivo de entrada (formato txt). Se guardan en distintas variables, la cantidad de trabajos y de máquinas que presenta la instancia. Para los tiempos de procesamientos y secuencia de máquinas de cada trabajo se guardan en dos matrices, ambas de manera matriz[i][j], donde i corresponde al trabajo y j el numero de la operación. Además, se inserta un valor del tiempo máximo de ejecución (Ver Tabla 4.2).

Job Shop Min Tardanza: Lectura de datos

Entrada: archivo, tiempo_maximo, import Pulp

Comienzo

```
t ← time max
Dat ← Load text ( instance, dtype = int)
Job , machine = int (dat[0][0] , dat[0][1])
Dat ← Delete (dat[0][0] , dat[0][1])
#Guardado de Secuencias (sec) y  tiempos de procesamiento (tp)
Sec ,tp = [ ], [ ]
for ( i := 1 to len(dat) )
   for ( j := 1 to len(dat[0]) )
      if ( j es impar)
         agregar elemento dat[i][j] en lista sec
      endif
      if (j es par)
```

agregar elemento dat[i][j] en lista tp
endif
Fin

Tabla 4.2: Pseudocodigo procedimiento lectura

Fuente: Elaboración propia

4.4.2. Modelamiento del problema

Esta parte es la más relevante del programa, ya que contiene la definición del problema y las distintas restricciones que deben ser agregadas para el correcto funcionamiento. Antes del modelado, es necesario la recopilación de toda la información (variables) entregada por la instancia, y la calculada a partir de ella, por ejemplo, el cálculo de los due dates de cada trabajo. Una vez completado lo anterior, el código se encarga de ir generando soluciones factibles mediante una combinatoria de las operaciones, es decir, revisando el espacio de búsqueda, teniendo en cuenta un tiempo máximo de ejecución. Una vez finalizado el análisis, el programa despliega la mejor planificación de trabajos encontrada, que minimiza la tardanza máxima (Ver Tabla 4.3).

Job Shop Min Tardanza: Creación de modelo
Entrada: Secuencia de máquinas (sec), tiempo de procesamiento (tp), tiempo maximo
Comienzo
Due date trabajo “i” ← Suma Tp trabajo “i” * 0,9
#Creacion del modelo a través de las funciones de librería PULP
Modelo LpMinimize ← definición y creación modelo de minimización
#Definicion de variables de decisión para cada trabajo
Variables ← Tiempos de inicio, Tiempo de finalización, precedencia entre operaciones, tardanza, Tardanza máxima
#Definicion de función objetivo
Modelo += Tmax ← Minimizar Tardanza máxima
#Definicion de restricciones
Restricciones
Tiempos inicios no negativos
Maquinas solo pueden procesar un trabajo a la vez
Operación i de los trabajos solo pueden ser procesados por una maquina
Tardanza igual a release time – due dates
Tardanza no negativa

Tardanza máxima ← max (Ti)
Fin-Restricciones
#Generar solución
Solve.modelo (max second = t) ← solucionar modelo con un tiempo limite
Desplegar valor funcion objetivo y variables
Fin

Tabla 4.3: Pseudocodigo creación de modelo

Fuente: Elaboración propia

4.5. Desarrollo del Sistema Multiagentes

Los sistemas multiagente son uno de los enfoques más prometedores para construir una solución compleja, robusta y rentable. Esto se debe a su autonomía, naturaleza distribuida, robustez contra fallas, y su capacidad para programar dinámica y flexiblemente los procesos de fabricación.

El desarrollo y programación de este sistema requiere de un software capaz de generar el ambiente propicio para que los agentes subsistan. Para ello se utilizará la plataforma NetLogo, el cual presenta cuatro tipos de agentes: tortugas, parcelas, enlaces y observador. El mundo donde interactúan los agentes es bidimensional y se divide en una grilla de parcelas. Cada parcela es una pieza cuadrada de "tierra" sobre la cual las tortugas se pueden mover y relacionarse con otras tortugas o con la misma parcela mediante enlaces. Por otro lado, el observador no tiene una ubicación en el "mapa", y su función es observar el mundo y dar instrucciones a los otros agentes.

En cuanto a la resolución del Job Shop Scheduling Problem, se utilizarán dos agentes, las tortugas (turtles), que adquirirán la identidad de los trabajos, y las parcelas (patches) que representarán a las maquinas. Como el mundo de NetLogo inicia sin tortugas, es necesario la lectura de la instancia para, crear las tortugas y seleccionar las parcelas a utilizar. Posteriormente, se le asignan características únicas a cada agente, para las tortugas destacan la secuencia de máquinas y el tiempo de procesamiento, y para las parcelas, el tiempo de utilización de la máquina.

Para la resolución del problema, las tortugas se van desplazan a las parcelas (maquinas) que indican su secuencia. En este proceso de interacción, se analizan distintas variables entre agentes, destacando la regla de prioridad de procesamientos de tortugas (trabajos) con menor fecha de entrega. En el procesamiento se analizan y actualizan, el tiempo de utilización de la parcela y el tiempo de flujo de

la tortuga. Una vez finalizados todos los desplazamientos de las tortugas, se observan las tardanzas y el tiempo de flujo total de cada una, y se despliegan los máximos valores obtenidos. En la siguiente figura se presenta un diagrama de flujo del funcionamiento general del sistema (Ver Figura 4.2):

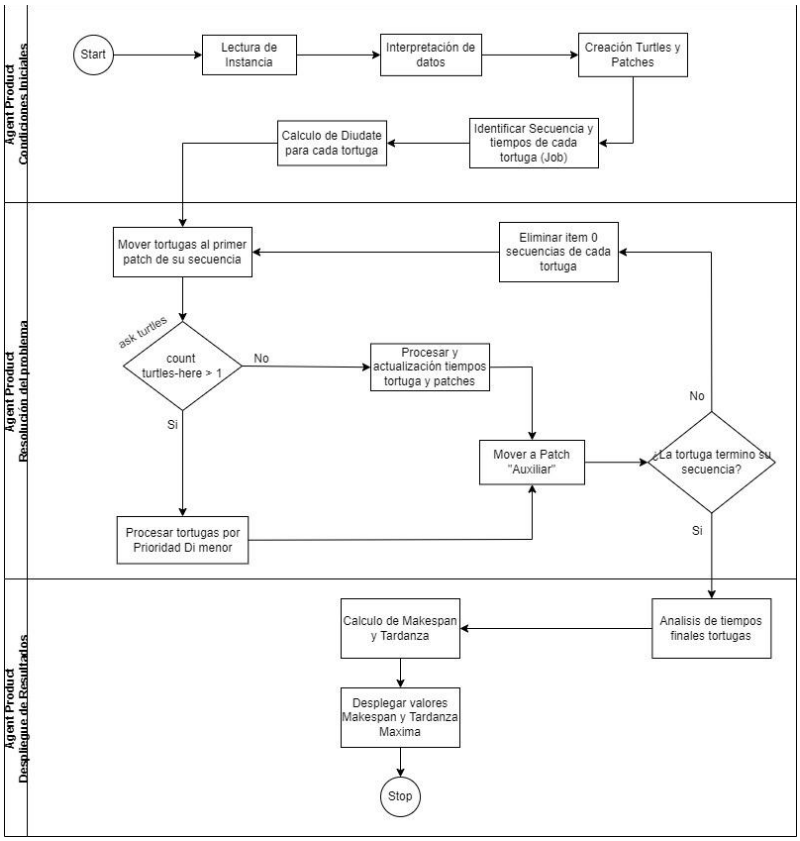

Figura 4.2 Diagrama de Flujo del sistema multiagente

Fuente: Elaboración propia

4.5.1. Proceso: Lectura e interpretación de Datos

Para el desarrollo del sistema multiagentes, es necesario la creación de funciones que se encarguen de leer e interpretar los datos de las instancias cargadas. A través de las funciones internas del software *file-open* y *file-read* se cargan y leen archivo por completo. Posteriormente, se crea una variable tipo lista con el nombre "data", la cual guarda todos los datos cargados de la instancia, permitiendo un mejor manejo de la información (Ver Tabla 4.4).

Extracto de Código: Procedimiento Lectura
Comienzo:

File-close-all → Cerrar todos los archivos

File-open “Texto” → Abrir la instancia seleccionada en formato texto *ej: abz5.txt*

Data = [ ] → Creación de lista vacía

While ¿Es el final del archivo?

  Asignar a la lista “data” los valores del archivo

End-While

File-close → Cerrar la instancia

Fin

Tabla 4.4: Pseudocodigo Netlogo - procedimiento lectura

Fuente: Elaboración propia

Para facilitar la carga de distintas instancias, se establece una variable de entrada tipo texto la cual es modificada por el observador, esta admite el nombre de la instancia que se quiere analizar, y se debe ingresar el nombre del archivo con su formato. A continuación, se muestra un ejemplo del ingreso correcto de los datos (Ver Figura 4.3):

Figura 4.3 Variable de entrada instancia

Fuente: Elaboración propia a partir de Netlogo

Como se explica en la “forma de las instancias”, los primeros dos números de cada instancia indican la cantidad de trabajos y maquinas presentes, por lo que, se guardan los dos primeros elementos de la lista “data” en dos nuevas variables “fila” y “columna”, donde la primera guarda la cantidad de trabajos, y la segunda la cantidad de máquinas. Estas nuevas variables permiten ir segmentando la información que entrega el archivo cargado.

Una vez asignado el valor de fila y columna, se generan las tortugas y los patches. Para las tortugas, se crean la cantidad equivalente al valor que guarda “fila”, y para mejorar su visualización se les asigna el color amarillo. Como la función de las tortugas es representar a los trabajos, es necesario entregarles características para la realización de sus operaciones, es decir, la asignación de variables

únicas que contengan la secuencia de máquinas por las que debe pasar y los tiempos de cada una de sus tareas (Ver figura 4.4).

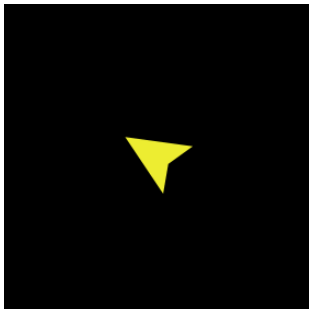

Figura 4.4 Creación de tortuga

Fuente: Elaboración propia a partir de Netlogo

De los datos restantes, se separan en dos variables, “maquinas” y “tiempos”, donde gracias a la configuración de los datos se sabe que los números que se encuentren en posiciones pares (incluyendo el 0) son las maquinas por donde deben pasar las tortugas, y los datos que se encuentran en posiciones impares son tiempos de procesamiento de cada operación. Basta con recorrer la lista donde se encuentra almacenada la información, y asignar los elementos en posiciones pares a la variable que guarda el recorrido de todas las maquinas, y los elementos en posiciones impares a la variable que guarda los tiempos de procesamientos de los trabajos.

Posteriormente, se seleccionan las tortugas una por una, y se les asignan la secuencia de máquinas que deben recorrer, y los tiempos de procesamiento en cada máquina, estas asignaciones permitirán facilitar las instrucciones que permitan manejar y enseñar a los agentes (Ver Tabla 4.5).

Extracto de Código: Guardado de Datos
Entrada: data (variable que guarda los datos de la instancia, sin trabajos y maquinas) **Comienzo:** **Ciclo foreach [x] range (0:2) in data** ← recorre desde el elemento 0, hasta el último, cada 2 elementos de data #recorre elementos pares y 0 maquinas ← guardar elemento x **endciclo** **Ciclo foreach x range (1:2) in data** ← recorre desde (desde el elemento 1, hasta el último, cada 2 elementos de data

```
    #recorre elementos impares
    tiempos ← guardar elemento x
  endciclo
  Seleccionar tortugas
    secuencias ← Asignar a cada tortuga su secuencia de maquinas
    tiempos_p ← Asignar a cada tortuga sus tiempos de procesamiento de operaciones
  Fin de selección
Fin
```

Tabla 4.5: Pseudocodigo Netlogo - Guardado de datos

Fuente: Elaboración propia

En cuanto a las maquinas, se seleccionan aleatoriamente los patches que representaran a las maquinas, a los cuales se les asignan distintas características propias para poder diferenciarlos y darle la capacidad de interactuar y aprender. Dentro de estas características destacan:

- **Color:** Tanto para la visualización como para la interacción con las tortugas (trabajos), es necesario el cambio del color de las parcelas que representan a las maquinas. Ya que esta característica puede ser utilizada por el observador para dar instrucciones a parcelas o tortugas.
- **Etiqueta:** Para mejorar la identificación de las parcelas, se le etiqueta a cada una de ellas con el numero de la maquina a la cual representa. Al igual que el color, esta variable es utilizada para dar instrucciones a los agentes.
- **Tiempo de utilización:** Esta característica permite capturar constantemente el tiempo de utilización que presenta la máquina y es relevante a la hora de ir procesando los trabajos, permitiendo el procesamiento de trabajos siempre y cuando la maquina se encuentre desocupada.

En la siguiente figura se muestra la inspección de la parcela en las coordenadas (0 , 0) antes y después de agregar las características (Ver Figura 4.5):

Figura 4.5 Inspección de parcela (0,0)

Fuente: Elaboración propia a partir de Netlogo

Una vez finalizada la etapa de lectura, con las tortugas creadas y las parcelas seleccionadas. El ambiente es propicio para el desarrollo del problema, por lo que corresponde desarrollar el proceso de operación e interacción entre los agentes-trabajos (tortugas) y los agentes-maquinas (parcelas). Para ello es necesario definir los criterios y características de cada agente, para que su evolución esté relacionada con el objetivo principal propuesto. En la siguiente figura se muestra el mundo de netlogo una vez creados los agentes de las instancias seleccionadas. Se observan que todas las tortugas creadas sin características de posición se ubican en el punto (0,0), y las parcelas que representan las maquinas son aleatorias (Ver Figura 4.6).

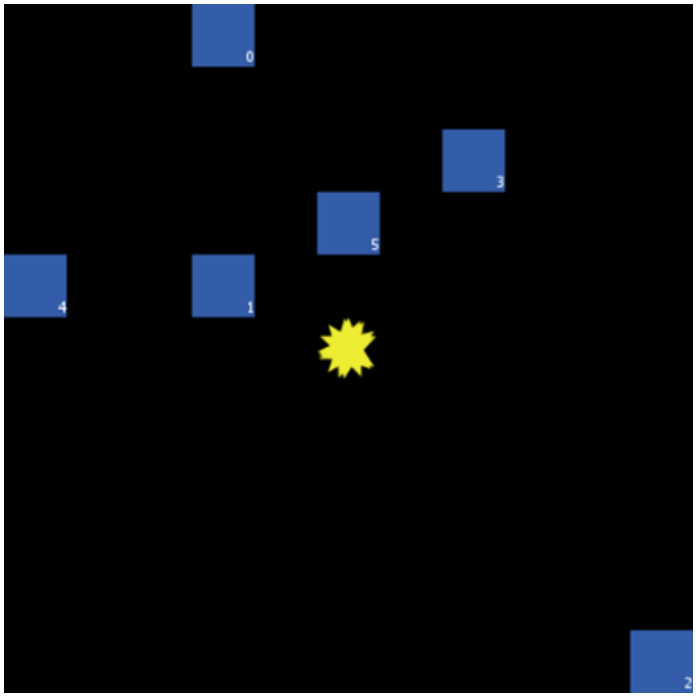

Figura 4.6 Agentes creados – Ejemplo instancia ft06

Fuente: Elaboración propia a partir de Netlogo

4.5.2. Proceso: Operación entre agentes

El razonamiento utilizado para el desarrollo del sistema se basa en la regla de secuenciación E.D.D. (earliest due date). La cual prioriza procesamientos de trabajos que presenten fecha de entrega más cercana. En este caso, las tortugas situadas en una misma parcela (que represente a las maquinas), deberán ser operadas por esta regla. Cabe destacar que, si la tortuga se encuentra sola, esta simplemente se procesará.

Es necesario crear otra variable única de las tortugas que una característica explique el due dates de cada una. Como las tortugas son creadas en el proceso de lectura, se les formula el cálculo de las fechas según sus tiempos de procesamiento, y se guarda en esta nueva característica (Ver Tabla 4.6).

Extracto de Código: Calculo due dates (Di)
Entrada: tortugas [sec tiempos_p tp] (tortugas con las variables únicas ya asignadas)
Comienzo:
Seleccionar tortugas

#Calculo due date para cada tortuga

Due date ← sum (tiempos_p) * 0,9 (suma de los tiempos de procesamiento multiplicados por factor)

Fin de selección

Fin

Tabla 4.6: Pseudocodigo Netlogo - Calculo due dates

Fuente: Elaboración propia

Para resolver el problema y calcular la tardanza máxima, cada tortuga debe recorrer todas las parcelas (maquinas) que indica su secuencia. El cumplimiento de esta se consigue ingresando una instrucción que seleccione a todas las tortugas, y les indica desplazarse a las parcelas que tengan la etiqueta igual al número de la primera máquina de su secuencia. Una vez finalizado la iteración, se elimina el primer elemento de la secuencia, y se repite la instrucción descrita hasta que ya no quede parcelas que recorrer.

En cada desplazamiento o "parada" de las tortugas en las parcelas, estas deben operar y procesarse siguiendo la regla EDD. Para ello, las parcelas analizan las tortugas situadas en ellas, y consultan su cantidad. Si hay más de una tortuga, las parcelas procesan la tortuga con menor fecha de entrega, actualiza sus tiempos de usos y vuelve a consultar la cantidad de tortugas. Por otro lado, si al consultar solo se encuentra con una tortuga, simplemente la procesa. Una vez procesadas en una parcela, las tortugas son movilizadas a una parcela "auxiliar", a la espera de que todas estén procesadas, para luego, si es necesario, continuar con su secuencia (Ver Tabla 4.7).

Extracto de Código: Desplazamiento de tortugas

Entrada: tortugas, parcelas (cada agente con sus características únicas)

#Operar ← procesar la tortuga en parcelas, actualización de tiempo de flujo y uso de parcela)

Comienzo:

While fin de la secuencia? (ciclo que termina al momento de que todas las tortugas terminen su secuencia)

Mover tortuga a parcela que indica el primer elemento de su secuencia

Eliminar primer elemento de su secuencia

Preguntar parcela ← consulta cantidad de tortugas en una parcela

Si Cantidad tortugas = 1: Operar tortuga

Si Cantidad tortugas > 1: Operar tortuga en desde menor due date a mayor due date

Fin pregunta

Fin While

Fin

Tabla 4.7 Pseudocodigo Netlogo - Desplazamiento de tortugas

Fuente: Elaboración propia

Al momento de procesar las tortugas es importante definir el funcionamiento de los tiempos de flujo de las tortugas y el tiempo de utilización de las parcelas. La lógica del problema indica que, si una maquina está siendo utilizada, la siguiente tarea debe esperar. En el sistema multiagente, esta lógica se codifica de la siguiente manera:

- Si la tortuga presenta un tiempo de flujo menor al tiempo de uso de la parcela en la que se debe procesar, la tortuga "espera" e iguala su tiempo de flujo al tiempo de uso de la parcela, y luego se procesa la tarea.
- Si la tortuga presenta un tiempo de flujo mayor al tiempo de uso de la parcela en la que se debe procesar, la parcela iguala su tiempo de uso al tiempo de flujo de la parcela, y luego se procesa la tarea.
- Si el tiempo de flujo de la parcela es igual al tiempo de uso de la parcela, se procesa la tarea.

Donde "procesar la tarea" se entiende como la suma del tiempo de procesamiento de la tarea a procesar, con el tiempo de uso de la máquina. Este resultado se actualiza y se guarda en el tiempo de uso de la maquina y en el tiempo de flujo de la tortuga (Ver Tabla 4.8).

Extracto de Código: Operar

Entrada: tiempos_p/tp/tm (tiempos de procesamientos tortugas/tiempo de flujo tortuga/tiempo de uso parcelas)

Comienzo:

```
Si tp > tm (tiempo de flujo de tortuga menor a tiempo de uso parcela)
    tp = tp + tiempos_p[0] (tp nuevo = tp actual + primer tiempo de procesamiento)
    tm = tp (tiempo de uso = tiempo de flujo nuevo)
Si tp < tm (tiempo de flujo de tortuga mayor a tiempo de uso parcela)
    tp = tm + tiempos_p[0] (tp nuevo = tiempo de uso actual + primer tiempo de procesamiento)
    tm = tp (tiempo de uso = tiempo de flujo nuevo)
Si tp == tm:
```

tp = tm + tiempos_p[0] (tp nuevo = tiempo de uso actual + primer tiempo de procesamiento)

tm = tp (tiempo de uso = tiempo de flujo nuevo)

Fin

Tabla 4.8: Pseudocodigo NetLogo - Operar

Fuente: Elaboración propia

Al finalizar todas las secuencias de las tortugas, se calcula la de cada una de ellas, la cual se obtiene restando el tiempo de flujo total, con los due dates. Donde el máximo de los valores obtenidos equivale a la tardanza máxima, la cual se despliega junto con el tiemplo de flujo máximo (Makespan). A continuación, se enseña un diagrama de secuencia del completo desarrollo del sistema multiagente (Ver Figura 4.7):

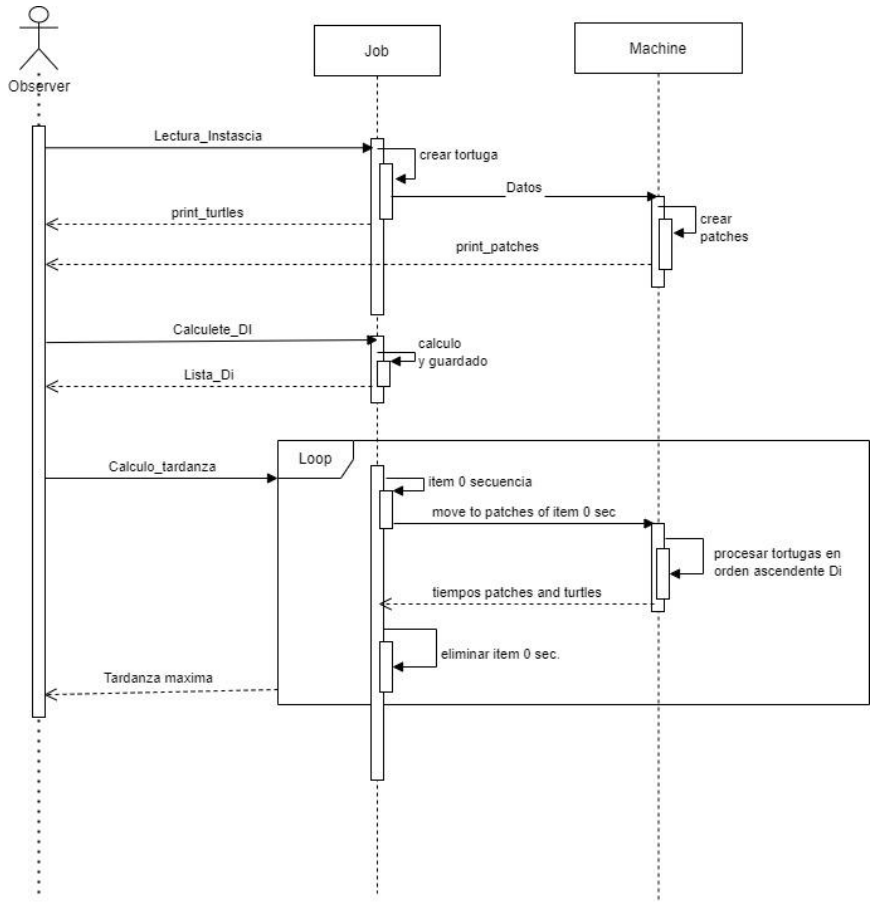

Figura 4.7 Diagrama de secuencia Sistema Multiagente

Fuente: Elaboración propia

Capítulo 5: Resultados y Análisis

5.1. Experimentos

Para realizar las pruebas en ambos métodos, se definió un tiempo máximo de ejecución de aproximadamente 1 hora, ya que se consideró un tiempo prudente para la obtención de buenos resultados mediante el método de programación lineal entera, teniendo en cuenta el gran tamaño de variables de decisión que presenta el problema Job Shop, y más aún cuando aumenta el tamaño de la instancia. Se utilizaron un total de 100 instancias ampliamente utilizadas en la literatura para la experimentación, donde dependiendo de la cantidad su dificultad, se categorizan como pequeña (Small Scale), mediana (Medium Scale) y gran escala (Large Scale). En la siguiente tabla se muestran las instancias seleccionadas (Ver Tabla 5.1):

Small Scale	Medium Scale	Large Scale
ft06, ft10, ft20, abz5, abz6, la01, la02, la03, la04, la05, la06, la07, la08, la09, la10, la11, la12, la13, la14, la15, la16, la17, la18, la19, la20 orb01, orb02, orb03, orb04, orb05, orb06, orb07, orb08, orb09, orb10,	la21, la22, la23, la24, la25, la26, la27, la28, la29, la30, swv01, swv02, swv03, swv04, swv05, la36, la37, la38, la39, la40, swv06, swv07, swv08, swv09, swv10, abz7, abz8, abz9, la31, la32, la33, la34, la35, yn01, yn02, yn03, yn04.	swv11, swv12, swv13, swv14, swv15, swv16, swv17, swv18, swv19, swv20, ta61, ta62, ta63, ta64, ta65, ta66, ta67, ta68, ta69, ta70, ta71, ta72, ta73, ta74, ta75, ta76, ta78, ta80.

Tabla 5.1: Instancias estudiadas

Fuente: Elaboración propia a partir de la literatura (Ver Anexo 1)

Cabe destacar que las condiciones de ejecución de los métodos fueron:

- Google Colab: Cuenta con guardado automático en la nube y compatibilidad con Google drive, proporciona 12,68 GM de memoria RAM y 107.72 GB de Disco
- Netlogo: Las pruebas en este software se llevaron a cabo en un computador Dell Latitude 3400 con las siguientes características relevantes: Procesador Intel® Core™ i7-8565U de 1.80 Ghz, 16,0 GB de memoria RAM, Sistema operativo de 64 bits, procesador x64, Windows 11 Pro 21H2.

5.2. Resultados

Las tablas y gráficos a continuación presentan los resultados obtenidos por los métodos anteriormente seleccionados (PLE y PDS), además del tiempo promedio de ejecución de cada método en las distintas clasificaciones de instancias:

5.2.1. Resultados de instancias: pequeña escala

Small Scale

Inst.	Job/Machine	Tmax PLE	Tmax PDS	**Inst.**	Job/Machine	Tmax PLE	Tmax PDS
ft06	6 / 6	23	30	**la14**	20 / 5	1042	1042
ft10	10 / 10	471	924	**la15**	20 / 5	1138	1270
ft20	20 / 5	1430	1321	**la16**	10 / 10	379	865
abz5	10 / 10	517	896	**la17**	10 / 10	390	617
abz6	10 / 10	364	639	**la18**	10 / 10	329	589
la01	10 / 5	445	599	**la19**	10 / 10	327	584
la02	10 / 5	338	497	**la20**	10 / 10	358	701
la03	10 / 5	413	618	**orb01**	10 / 10	556	1201
la04	10 / 5	348	531	**orb02**	10 / 10	395	787
la05	10 / 5	391	410	**orb03**	10 / 10	564	1111
la06	15 / 5	682	695	**orb04**	10 / 10	566	881
la07	15 / 5	656	761	**orb05**	10 / 10	433	910
la08	15 / 5	579	712	**orb06**	10 / 10	486	923
la09	15 / 5	643	683	**orb07**	10 / 10	172	303
la10	15 / 5	689	712	**orb08**	10 / 10	537	983
la11	20 / 5	977	1024	**orb09**	10 / 10	461	799
la12	20 / 5	877	824	**orb10**	10 / 10	464	1072
la13	20 / 5	966	915				

Tabla 5.2: Resultados Instancias pequeñas

Fuente: Elaboración propia

Tiempo promedio de ejecución PLE (s)	Tiempo promedio de ejecución PDS (s)
3277,95	1,34

Tabla 5.3: Tiempos promedios de ejecución – pequeña escala

Fuente: Elaboración propia

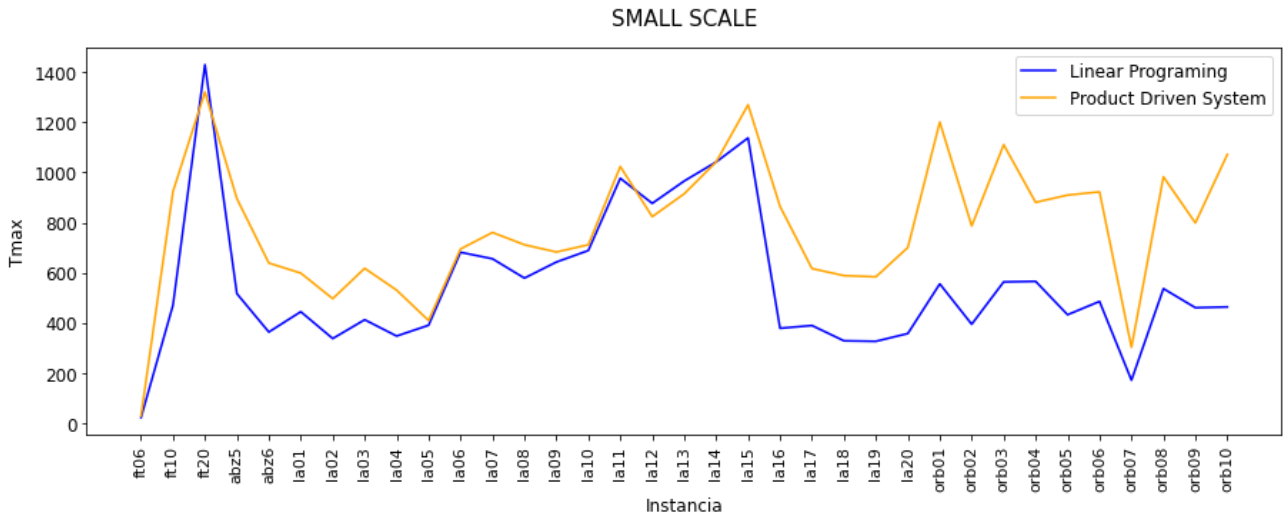

Figura 5.1 Grafico de instancias de pequeña escala

Fuente: Elaboración propia a partir de Python

5.2.2. Resultados de instancias: médium escala

Medium Scale

Inst.	Job/Machine	Tmax PLE	Tmax PDS	**Inst.**	Job/Machine	Tmax PLE	Tmax PDS
la21	15 / 10	633	829	**la40**	15 / 15	719	1142
la22	15 / 10	765	867	**swv06**	20 / 15	1436	1866
la23	15 / 10	587	844	**swv07**	20 / 15	1871	1707
la24	15 / 10	642	812	**swv08**	20 / 15	2528	2336
la25	15 / 10	703	1056	**swv09**	20 / 15	1692	2538
la26	20 / 10	1055	1121	**swv10**	20 / 15	1772	2116
la27	20 / 10	1369	1145	**abz7**	20 / 15	745	507
la28	20 / 10	1108	1125	**abz8**	20 / 15	985	679
la29	20 / 10	1276	1179	**abz9**	20 / 15	960	664
la30	20 / 10	1261	1293	**la31**	30 / 15	2875	1644
swv01	20 / 10	1574	1894	**la32**	30 / 15	1712	1803
swv02	20 / 10	2048	1817	**la33**	30 / 15	1455	1615
swv03	20 / 10	1691	1867	**la34**	30 / 15	1984	1638
swv04	20 / 10	1699	2004	**la35**	30 / 15	2432	2094
swv05	20 / 10	1602	1938	**yn01**	20 / 20	687	742
la36	15 / 15	869	1072	**yn02**	20 / 20	770	722
la37	15 / 15	817	1326	**yn03**	20 / 20	914	732
la38	15 / 15	802	1058	**yn04**	20 / 20	1201	892
la39	15 / 15	737	1092				

Tabla 5.4: Resultados Instancias medianas

Fuente: Elaboración propia

Tiempo promedio de ejecución PLE (s)	Tiempo promedio de ejecución en PDS (s)
3601,72	2,64

Tabla 5.5: Tiempos promedios de ejecución – mediana escala

Fuente: Elaboración propia

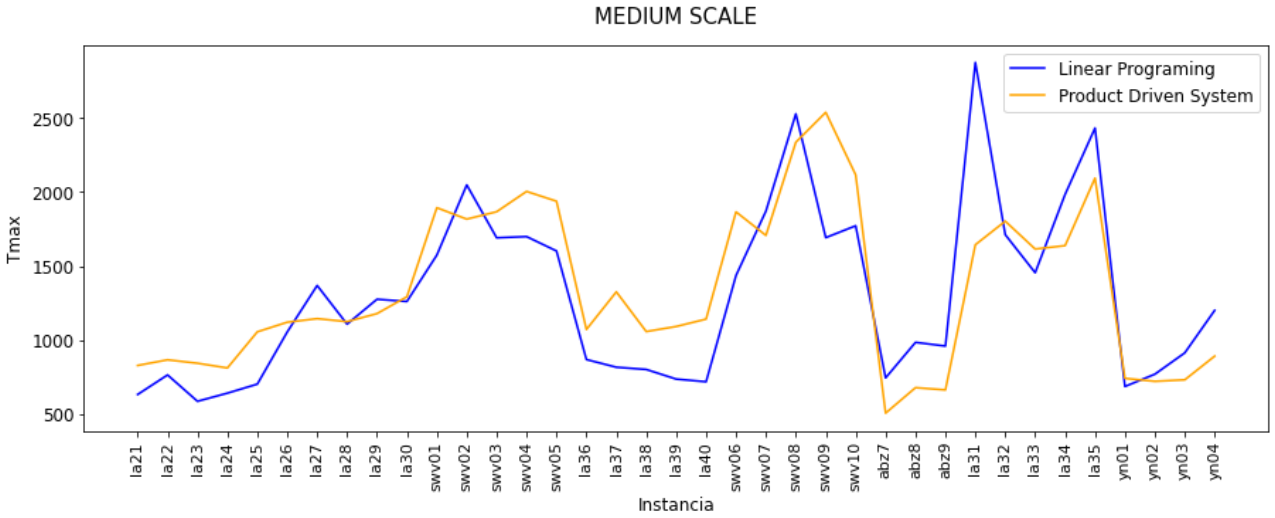

Figura 5.2 Grafico de instancias de mediana escala

Fuente: Elaboración propia a partir de Python

5.2.3. Resultados de instancias: gran escala

Large Scale

Inst.	Job/Machine	Tmax PLE	Tmax PDS	**Inst**	Job/Machine	Tmax PLE	Tmax PDS
swv11	50 / 10	9892	4620	**ta65**	50 / 20	33602	3046
swv12	50 / 10	9378	4652	**ta66**	50 / 20	25613	2948
swv13	50 / 10	10132	4885	**ta67**	50 / 20	33812	3376
swv14	50 / 10	8761	4534	**ta68**	50 / 20	38210	3148
swv15	50 / 10	9739	4970	**ta69**	50 / 20	25061	3182
swv16	50 / 10	16468	2598	**ta70**	50 / 20	39449	3142
swv17	50 / 10	15891	2689	**ta71**	100 / 20	79502	6173
swv18	50 / 10	5742	2606	**ta72**	100 / 20	76523	5725
swv19	50 / 10	3929	3072	**ta73**	100 / 20	77487	5951
swv20	50 / 10	5152	2583	**ta74**	100 / 20	79596	5455
ta61	50 / 20	38779	3194	**ta75**	100 / 20	77783	6504
ta62	50 / 20	35680	3232	**ta76**	100 / 20	80097	5706
ta63	50 / 20	35723	3059	**ta78**	100 / 20	77051	5857
ta64	50 / 20	9881	2932	**ta80**	100 / 20	76904	5483

Tabla 5.6: Resultados Instancias grandes

Fuente: Elaboración propia

Tiempo promedio de ejecución en PL (s)	Tiempo promedio de ejecución en PDS (s)
3607,68	5,28

Tabla 5.7: Tiempos promedios de ejecución – mediana escala

Fuente: Elaboración propia

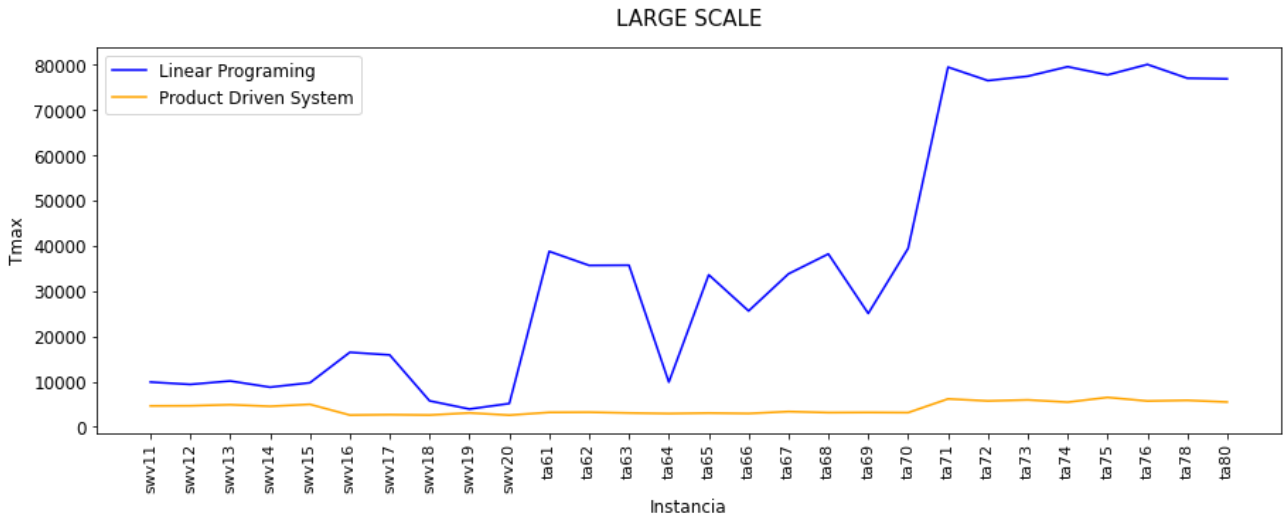

Figura 5.3 Grafico de instancias de gran escala

Fuente: Elaboración propia a partir Python

5.3. Análisis de Resultados

Con respecto a los resultados, se observa que el sistema impulsado por productos inteligentes, desarrollado en NetLogo, obtiene mejores resultados que la PLE en: 4 de 35 instancias de pequeña escala (11%), en 14 de 37 instancias de mediana escala (37%) y en 28 de 28 instancias de gran escala (100%). Lo anterior indica que, a pesar de que el tiempo promedio de ejecución PLE es ajustado al tiempo máximo establecido, los resultados obtenidos en problemas de pequeña y mediana envergadura son de mejor calidad que los obtenidos por la metodología PDS, en otras palabras, el método de programación lineal entera es capaz de recorrer suficientes combinaciones para la obtención de un buen resultado. No obstante, al momento de analizar instancias de gran complejidad o envergadura, los resultados obtenidos por PDS presentan una calidad muy superior al método tradicional, además de un tiempo promedio de ejecución de 5.28 segundos, lo cual, si lo trasladamos a problemas de la vida real, es un tiempo muy aceptable para la planificación de tareas en los sistemas productivos.

Capítulo 6: Conclusión

En este estudio se presentó el desarrollo de un sistema impulsado por el producto (PDS), desarrollado en una plataforma basada en agentes, para la resolución del Job Shop Scheduling Problem, con el objetivo de minimizar la tardanza máxima de los trabajos. Estos basan su comportamiento a una regla de secuenciación que prioriza los trabajos con menor fecha de entrega. Este sistema se comparó con los resultados obtenidos mediante la técnica de programación lineal entera, implementada en Python.

Para los PDS fue necesario la definición y el análisis del marco teórico que engloba al problema, donde destacan los sistemas de manufactura inteligente, productos inteligentes y sistemas multiagentes. Logrando captar los requerimientos y medidas de desempeño de los sistemas productivos sumado a los desafíos de la época moderna.

Para modelar el sistema fue necesario la utilización de la herramienta NetLogo, la cual permite el desarrollo de sistemas complejos basados en agentes. La aplicación de este software requiere de varias sesiones de aprendizajes para el completo entendimiento tanto de su funcionamiento, como del alcance de sus herramientas internas.

La experimentación comprende el análisis de distintas instancias halladas en la literatura, divididas en pequeña, mediana y gran envergadura dependiendo de la cantidad de trabajos y maquinas que presenten. Lo anterior, con el fin de contrastar la calidad y el tiempo de ejecución de ambos métodos, con el fin de corroborar la robustes, flexibilidad y operación de los sistemas impulsados por los productos.

El análisis entrega que, para problemas pequeños y medianos, la programación entera entrega mejores resultados del problema, aunque no tan alejados que los que proporciona el PDS. En cuanto a problemas de gran complejidad, el método desarrollado en sistemas de agentes obtiene excelentes resultados en tiempos computacionales muy buenos.

Se concluye que, si bien el sistema no logro mejorar los resultados para escalas medianas o pequeñas, si logra una mejora sustancial para problemas grandes, los cuales son más ajustados a la realidad de las organizaciones. Además, este método entrega soluciones en tiempos coherentes y aceptables para la toma de decisiones en una empresa, otorgando una gran cantidad de robustez, flexibilidad y reconfiguración en caso de fallos externos o modificaciones que la programación entera no logra abarcar.

Bibliografías

Garcia-Sabater, Jose P. (2021) Programación Matemática en Python con PULP RIUNET Repositorio UPV http://hdl.handle.net/10251/158416

Ang, L., Yew W. K. y Peng W. W. *Simulation of Sequencing Rules Using Witness in a Milling Job Shop,* Communications of the IBIMA http://www.ibimapublishing.com/journals/CIBIMA/cibima.html, Vol. 2011, Article ID 402089, 6 pages DOI: 10.5171/2011.402089. (2011).

F. Jovane, H. Yoshikawa, L. Alting, C.R. Boër, E. Westkamper, D. Williams, M. Tseng, G. Seliger, A.M. Paci, The incoming global technological and industrial revolution towards competitive sustainable manufacturing, CIRP Annals, Volume 57, Issue 2, 2008, Pages 641-659, ISSN 0007-8506, https://doi.org/10.1016/j.cirp.2008.09.010.

Aydin, M. E., & Öztemel, E. (2000). Dynamic job-shop scheduling using reinforcement learning agents. In *Robotics and Autonomous Systems* (Vol. 33).

Baker, K. R., & Kanet, J. J. (1983). Job Shop Scheduling With Modified Due Dates. In *Journal of Operations Management* (Vol. 4, Issue I).

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. In *Annals of Operations Research* (Vol. 41).

Chen, B., & Matis, T. I. (2013a). A flexible dispatching rule for minimizing tardiness in job shop scheduling. *International Journal of Production Economics*, *141*(1), 360–365. https://doi.org/10.1016/j.ijpe.2012.08.019

Fu, Y., Wang, H., Wang, J., & Pu, X. (2020). Multiobjective Modeling and Optimization for Scheduling a Stochastic Hybrid Flow Shop With Maximizing Processing Quality and Minimizing Total Tardiness. *IEEE Systems Journal*, *15*(3), 4696–4707. https://doi.org/10.1109/jsyst.2020.3014093

Garcia-Sabater, J. P. (2021). *Un tutorial para comenzar a utilizar Python*. http://hdl.handle.net/10251/158416

Giraldo, J. A., Toro, C. A., & Jaramillo, F. A. (2013). Aprendiendo sobre la Secuenciación de Trabajos en un Job Shop mediante el Uso de Simulación. *Formacion Universitaria*, *6*(4), 27–38. https://doi.org/10.4067/S0718-50062013000400004

Herrera, C., Thomas, A., & Vera, V. (2013). Simulation of a hybrid product-driven system for manufacturing planning and control. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, *46*(7), 69–74. https://doi.org/10.3182/20130522-3-BR-4036.00093

Herrera, C., Thomas, A., & Parada, V. (2014). A product-driven system approach for multilevel decisions in manufacturing planning and control. Production & Manufacturing Research, 2(1), 756–766. doi:10.1080/21693277.2014.949895

Kouider, A., & Bouzouia, B. (2012). Multi-agent job shop scheduling system based on co-operative approach of idle time minimisation. *International Journal of Production Research*, *50*(2), 409–424. https://doi.org/10.1080/00207543.2010.539276

Mattfeld, D. C., & Bierwirth, C. (2004). An efficient genetic algorithm for job shop scheduling with tardiness objectives. *European Journal of Operational Research*, *155*(3), 616–630. https://doi.org/10.1016/S0377-2217(03)00016-X

Pan, C. H. (1997). A study of integer programming formulations for scheduling problems. *International Journal of Systems Science*, *28*(1), 33–41. https://doi.org/10.1080/00207729708929360

Peña, V., & Zumelzu, L. (2006). *Job Shop Scheduling Problem Minimización de Tardanza Máxima con RFL*.

Roger G. Schroeder, S. M. G. & M. J. R. (2011). *Operations Management. Comtemporary concepts and cases*.

Safarzadeh, H., & Kianfar, F. (2019). Job shop scheduling with the option of jobs outsourcing. *International Journal of Production Research*, *57*(10), 3255–3272. https://doi.org/10.1080/00207543.2019.1579934

Walker, S. S., Brennan, R. W., & Norrie, D. H. (2005). Holonic job shop scheduling using a multiagent system. In *IEEE Intelligent Systems* (Vol. 20, Issue 1, pp. 50–57). https://doi.org/10.1109/MIS.2005.9

Williams H. P. (1975). The Formulation of Mathematical Programming Models. In *Jl of Mgmt Sci* (Vol. 3, Issue 4).

Zhou, H., Cheung, W., & Leung, L. C. (2009a). Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm. *European Journal of Operational Research*, *194*(3), 637–649. https://doi.org/10.1016/j.ejor.2007.10.063

Zribi, N., Kamel, A., & Borne, P. (2006). Total Tardiness in a Flexible Job-shop. The Proceedings of the Multiconference on "Computational Engineering in Systems Applications." doi:10.1109/cesa.2006.313560

J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. Management Science, 34.3: 391-401, 1988. doi: 10.1287/mnsc.34.3.391 jstor: 2632051

H. Fisher and G. L. Thompson. Probabilistic learning combinations of local job-shop scheduling rules. In: Industrial Scheduling: 225-251. ed. by J.F. Muth and G.L. Thompson. Prentice Hall, 1963. oclc: 781815542

S. Lawrence. Resource Constrained Project Scheduling. An Experimental Investigation of Heuristic Scheduling Techniques (Supplement). Carnegie-Mellon University, 1984

D. Applegate and W. Cook. A computational study of job-shop scheduling. ORSA Journal of Computing, 3.2: 149-156, 1991. doi: 10.1287/ijoc.3.2.149

R.H. Storer, S.D. Wu and R. Vaccari. New search spaces for sequencing instances with application to job shop scheduling. Management Science, 38.10: 1495-1509, 1992. doi: 10.1287/mnsc.38.10.1495

E. D. Taillard. Benchmarks for basic scheduling problems. European Journal of Operational Research, 64.2: 278-285, 1993. doi: 10.1016/0377-2217(93)90182-M

T. Yamada and R. Nakano. A genetic algorithm applicable to large-scale job-shop instances. In: Parallel instance solving from nature II: 281-290. ed. by R. Manner and B. Manderick. Elsevier, 1992. isbn: 978-0-444-89730-5

Nicolich, M., Persi, P., Pesenti, R., & Ukovich, W. (1997). A Hierarchical Approach to an FMS Scheduling Problem. IFAC Proceedings Volumes, 30(1), 145–150. doi:10.1016/s1474-6670(17)44622-2

Leitão, P. (2009). Agent-based distributed manufacturing control: A state-of-the-art survey. Engineering Applications of Artificial Intelligence, 22(7), 979–991. doi:10.1016/j.engappai.2008.09.005

Guodong, Y., Yu, Y., Xi, Z., & Aijun, L. (2015). Multi-objective dynamic fuzzy scheduling and its algorithm in product collaborative design considering emergency. Journal of Intelligent & Fuzzy Systems, 29(4), 1355–1365. doi:10.3233/ifs-141491

Anexos

Autores	Inst.	Job / Machine	Autores	Inst.	Job / Machine
Adams, Balas and Zawack (Adams et al., 1988)	Abz5	10/10	Applegate and Cook (Applegate & Cook, 1991)	Orb01	10/10
	Abz6	10/10		Orb02	10/10
	Abz7	20/15		Orb03	10/10
	Abz8	20/15		Orb04	10/10
	Abz9	20/15		Orb05	10/10
Fisher and Thompson (Fisher & Thompson, 1963)	Ft06	6/6		Orb06	10/10
	Ft10	10/10		Orb07	10/10
	Ft20	20/5		Orb08	10/10
Lawrence (Lawrence, 1984)	La01	10/5		Orb09	10/10
	La02	10/5		Orb10	10/10
	La03	10/5	Storer, Wu and Vaccari (Storer et al., 1992)	Swv01	20/10
	La04	10/5		Swv02	20/10
	La05	10/5		Swv03	20/10
	La06	15/5		Swv04	20/10
	La07	15/5		Swv05	20/10
	La08	15/5		Swv06	20/15
	La09	15/5		Swv07	20/15
	La10	15/5		Swv08	20/15
	La11	20/5		Swv09	20/15
	La12	20/5		Swv10	20/15
	La13	20/5		Swv11	50/10
	La14	20/5		Swv12	50/10
	La15	20/5		Swv13	50/10
	La16	10/10		Swv14	50/10
	La17	10/10		Swv15	50/10
	La18	10/10		Swv16	50/10
	La19	10/10		Swv17	50/10
	La20	10/10		Swv18	50/10
	La21	15/10		Swv19	50/10
	La22	15/10		Swv20	50/10
	La23	15/10	E. D. Taillard (Taillard, 1993)	ta61	50/20
	La24	15/10		ta62	50/20
	La25	15/10		ta63	50/20
	La26	20/10		ta64	50/20
	La27	20/10		ta65	50/20
	La28	20/10		ta66	50/20
	La29	20/10		ta67	50/20
	La30	20/10		ta68	50/20
	La31	30/10		ta69	50/20
	La32	30/10		ta70	50/20
	La33	30/10		ta71	100/20

	La34	30/10		ta72	100/20
	La35	30/10		ta73	100/20
	La36	15/15		ta74	100/20
	La37	15/15		ta75	100/20
	La38	15/15		ta76	100/20
	La39	15/15		ta78	100/20
	La40	15/15		ta80	100/20
Yamada and Nakano (Yamada & Nakano, 1992)	Yn01	20/20			
	Yn02	20/20			
	Yn03	20/20			
	Yn04	20/20			

Anexo 1: Tabla de Instancias estudiadas

Fuente: Elaboración propia a partir de la literatura

Anexo: Resumen FI

UNIVERSIDAD DE CONCEPCION – FACULTAD DE INGENIERIA

RESUMEN DE MEMORIA DE TITULO

Departamento: Departamento de Ingeniería Industrial

Carrera: Ingeniería Civil Industrial

Nombre del memorista: Hernán Patricio Leiva Torres

Título de la memoria: Diseño de sistema inteligente de manufactura para el problema de Job Shop Scheduling minimizando la tardanza

Fecha de la presentación oral:

Profesor(es) Guía: Carlos Enrique Herrera López

Patricio Antonio Saez Bustos

Profesor(es) Revisor(es):

Concepto:

Calificación:

Resumen

El presente estudio se realiza en el marco de uno de los problemas más clásicos de la optimización combinatoria y el área de la administración de operaciones, el Job Shop Scheduling Problem. Este problema es considerado como muy complejos, teniendo una denominación NP-Hard en la teoría de la complejidad computacional. La resolución de este problema representa un gran desafío para las organizaciones con presencia de un sistema productivo, debido a los acotados tiempos de respuesta que entrega el mercado actual. Para esto se diseñó un sistema controlado por productos inteligentes (PDS) basado en una programación multiagente capaz de resolver el problema, minimizando la tardanza máxima, y siendo probado a través de distintas instancias estudiadas en la literatura. Se comparan los resultados y tiempos de respuestas en la resolución del problema con metodologías optimas como la programación entera. El PDS propuesto obtuvo mejores resultados en un 11% de las instancias pequeñas, en un 38% de las instancias medianas y en un 100% de las instancias grandes. Además, se tiene un tiempo de ejecución coherente con el funcionamiento del sistema, donde en promedio se tiene un 1.34 , 2.64 y 5.28 segundos para problemas de pequeña, mediana y gran escala respectivamente.