



Universidad
de Navarra

DATAI
INSTITUTO DE CIENCIA DE LOS
DATOS E INTELIGENCIA ARTIFICIAL

Optimización de la Programación de Producción en una Fábrica de Motores mediante Algoritmos de Optimización.

**Trabajo Fin de Máster
Máster Universitario en Big Data Science**

Curso académico 2023-2024

AUTORES: Manuel Alejandro Suárez Calle e Inés Teresa Hernández Pastor

TUTOR ACADÉMICO: Montserrat Ana Miranda Galcerán

TUTOR DE EMPRESA: Aitor Facio Valero y Juan Julio González Paredes

Madrid a 22 de agosto de 2024

RESUMEN

En el contexto actual del mercado, la inteligencia artificial y los algoritmos de Machine Learning destacan como áreas innovadoras y de gran impacto. Con la finalidad de orientar a Navantia en la toma de decisiones informada, el presente Trabajo de Fin de Máster se centra en la optimización de la programación de la producción en su fábrica de motores ubicada en Cartagena.

A través de la implementación de algoritmos de Machine Learning se busca mejorar la eficiencia operativa y reducir los tiempos totales del proceso de producción. Se utilizan cuatro algoritmos: el algoritmo húngaro, el genético, el de enrutamiento de máquinas y el de cuellos de botella para superar las limitaciones del módulo de *Advanced Planning and Optimization* (APO) de SAP S/4 HANA. La investigación muestra, a través de una interfaz de usuario en Streamlit, cómo estos algoritmos optimizan la asignación de recursos y mejoran la planificación y programación de la producción.

Este estudio representa un avance significativo en la gestión de la producción de Navantia, ya que la herramienta desarrollada ayuda a comprender y optimizar los procesos de fabricación de motores, lo que puede tener un gran impacto en los beneficios económicos de la empresa.

Palabras clave: Optimización de la producción, inteligencia artificial, Machine Learning, algoritmo húngaro, algoritmo genético, enrutamiento de máquinas, cuellos de botella, SAP S/4 HANA.

ABSTRACT

In the current market context, Artificial Intelligence and Machine Learning algorithms are prominent as innovative and impactful areas. With the aim of guiding Navantia in informed decision-making, this Master's Thesis focuses on optimizing production scheduling in its engine factory in Cartagena.

By implementing Machine Learning algorithms, , the aim is to improve operational efficiency and reduce the overall production process times. Four algorithms are studied: the Hungarian Algorithm, the Genetic Algorithm, the Machine Routing Algorithm, and the Bottleneck Algorithm, to overcome the limitations of the Advanced Planning and Optimization (APO) module of SAP S/4 HANA. The research demonstrates, through a user interface in Streamlit, how these algorithms optimize resource allocation and improve production planning and scheduling.

This study represents a significant improvement in Navantia's production management, as the developed tool helps to understand and optimize engine manufacturing processes, which can have a great impact on the company's economic benefits.

Keywords: Production optimization, Artificial Intelligence, Machine Learning, Hungarian Algorithm, Genetic Algorithm, Machine Routing, Bottlenecks, SAP S/4 HANA.

ÍNDICE DE CONTENIDOS

RESUMEN.....	3
ABSTRACT.....	4
ÍNDICE DE FIGURAS	7
ÍNDICE DE TABLAS.....	8
CAPÍTULO 1: INTRODUCCIÓN	9
CAPÍTULO 2: DESCRIPCIÓN DEL PROYECTO.....	11
2.1. INTRODUCCIÓN AL PROYECTO Y MOTIVACIÓN	11
2.2. OBJETIVOS	12
2.3. MATERIALES EMPLEADOS.....	13
2.3.1. <i>¿Qué es un ERP?</i>	13
2.3.2. <i>¿Qué es SAP?</i>	15
2.3.3. <i>SAP S/4 HANA</i>	15
2.3.4. <i>Planificación de la Producción</i>	16
2.3.5. <i>Composición de una orden de fabricación</i>	17
2.3.6. <i>Diagrama de Gantt</i>	17
2.4. ESTADO DEL ARTE	18
CAPÍTULO 3: MATERIALES Y MÉTODOS	21
3.1. SIMULACIÓN DE DATOS	21
3.2. ANÁLISIS Y ESTRUCTURA DE LOS DATOS	22
3.2.1. <i>Hoja Ordenes</i>	23
3.2.2. <i>Hoja Operaciones</i>	24
3.2.3. <i>Hoja Recursos</i>	24
3.3. DESCRIPCIÓN DE LOS ALGORITMOS USADOS	25
CAPÍTULO 4: RESULTADOS	34
4.1. DATOS GENERADOS.....	34
4.2. ADAPTACIÓN DE LOS ALGORITMOS AL CÓDIGO EN GOOGLE COLAB	34
4.2.1. <i>Algoritmo de Munkres</i>	36
4.2.2. <i>Algoritmo genético</i>	37
4.2.3. <i>Algoritmo de enrutamiento de máquinas</i>	40
4.2.4. <i>Algoritmo de cuellos de botella</i>	41

4.3.	INTERFAZ DE USUARIO	42
4.4.	ANÁLISIS DE LOS RESULTADOS OBTENIDOS.....	46
CAPÍTULO 5: CONCLUSIONES		49
CAPÍTULO 6: TRABAJO FUTURO.....		51
CAPÍTULO 7: GLOSARIO		53
CAPÍTULO 8: BIBLIOGRAFÍA		55
ANEXOS.....		57

ÍNDICE DE FIGURAS

Figura 1. Beneficios de la implementación de un ERP	14
Figura 2. Principales módulos de un sistema ERP	14
Figura 3. Módulos de SAP S/4 HANA	16
Figura 4. Función generar_datos_sinteticos	21
Figura 5. Inputs solicitados al usuario.	22
Figura 6. Identificación de operaciones y recursos.	27
Figura 7. Definición y relleno matriz de costos.....	27
Figura 8. Resta del menor valor de cada fila.....	27
Figura 9. Resta del menor valor de cada columna.....	28
Figura 10. Cubrimiento de ceros con el mínimo número de líneas.	28
Figura 11. Ajuste de la matriz de costos.	28
Figura 12. Finalización del algoritmo.....	29
Figura 13. Asignación final.	29
Figura 14. Asignación de tareas y máquinas.	31
Figura 15. Identificación del proceso de producción	32
Figura 16. Función evaluar_fitness.	35
Figura 17. Función generar_matriz_costos.	36
Figura 18. Función algoritmo_munkres.	37
Figura 19. Función crear_cromosoma.....	38
Figura 20. Función seleccionar_padres.	38
Figura 21. Función cruzar.....	39
Figura 22. Función mutar.	39
Figura 23. Función algoritmo_genetico.	40
Figura 25. Función identificar_cuellos_de_botella.	41
Figura 26. Gráfico de barras identificación cuellos de botella.	42
Figura 27. Interfaz de usuario inicial de la aplicación.	43
Figura 28. Interfaz de usuario final de la aplicación.	43
Figura 29. Ejemplo de interfaz con 3 órdenes generadas y 3 horas de holgura.	44
Figura 30. Salida en pantalla de los datos generados.	45
Figura 31. Visualización Gantt de las órdenes de fabricación generadas.....	46

ÍNDICE DE TABLAS

Tabla 1. Estructura de la Hoja Ordenes	23
Tabla 2. Estructura de la Hoja Operaciones.....	24
Tabla 3. Estructura de la Hoja Recursos	25
Tabla 4. Valor de fitness de los algoritmos.....	47
Tabla 5. Asignaciones óptimas para cada algoritmo.....	47

CAPÍTULO 1: INTRODUCCIÓN

La inteligencia artificial y los algoritmos de *Machine Learning* son uno de los campos más prometedores y revolucionarios en la actualidad, con aplicaciones que abarcan desde el sector de los seguros hasta la industria manufacturera. En concreto, en el ámbito de la producción industrial, la optimización de la producción se ha establecido como una tarea imprescindible para las empresas que buscan posicionarse por delante de la competencia y mantener su competitividad en un mercado en constante cambio y expansión global. Esta optimización se centra en la capacidad de, manteniendo la calidad del producto, agilizar y mejorar continuamente los procesos, al mismo tiempo que se reducen los costos operativos.

Ante la creciente importancia de estos procesos, han ido surgiendo a lo largo de los años diferentes sistemas de planificación de recursos empresariales (ERP), de entre los que destaca SAP S/4 HANA. Este en particular se caracteriza por ofrecer un conjunto de herramientas innovadoras, diseñadas para facilitar una gestión y planificación eficiente de la producción. Las herramientas proporcionadas ayudan a las empresas a adaptarse y prosperar en entornos de producción cada vez más dinámicos.

El presente Trabajo de Fin de Máster se centra en llevar a cabo una **optimización del proceso de producción** de la planta de fabricación de motores de Navantia en Cartagena. En concreto, se investiga cómo los **algoritmos de inteligencia artificial**, el Algoritmo Húngaro (Munkres), el algoritmo genético, el de enrutamiento de máquinas y algoritmos para identificar y optimizar cuellos de botella, pueden mejorar la planificación de la producción. Estos ofrecen una alternativa al módulo de *Advanced Planned and Optimization* (APO) de SAP S/4 HANA (SAP, s.f.). Los métodos mencionados no solo se han aplicado en la industria naval, sino también en otros sectores como la manufactura de componentes electrónicos y la logística, demostrando su eficacia en la optimización de recursos y tiempos de producción.

La decisión de adoptar este enfoque está motivada por dos factores principales: la posibilidad de mejorar la eficiencia de la optimización y la capacidad de integrar elementos adicionales en la planificación, tales como el consumo energético de las máquinas, que no están contemplados por los algoritmos heurísticos de SAP APO.

Además, se ha implementado una **interfaz de usuario** para facilitar la interacción y permitir una visualización más clara e intuitiva de la optimización de los procesos de producción. Esta interfaz permitirá a los usuarios observar de manera visual los beneficios y las mejoras logradas a través de la puesta en práctica de los algoritmos.

Este estudio busca contribuir al campo de la gestión de la producción empresarial mediante algoritmos de inteligencia artificial, ofreciendo nuevas perspectivas que pueden ser adoptadas por la empresa Navantia para así mejorar su eficiencia productiva.

CAPÍTULO 2: DESCRIPCIÓN DEL PROYECTO

En el presente apartado se introduce el proyecto junto con la motivación que lo impulsa. Acto seguido, se describen los objetivos generales y específicos, los materiales empleados en su desarrollo y los estudios previos en los que se fundamenta.

2.1. Introducción al proyecto y motivación

Se tiene como propósito planificar y optimizar el proceso de producción en el ámbito empresarial, específicamente en el módulo de *Advanced Planned and Optimization* (APO) de SAP S/4 HANA. Pese a que este sistema ofrece herramientas para la gestión óptima de la producción, existen áreas donde su eficacia no es óptima y, por tanto, es susceptible de mejora.

En particular, se ha identificado la necesidad de mejorar la optimización en la programación de la producción, buscando una **minimización del tiempo total de producción**. Aunque el módulo de SAP APO ofrece capacidades en este ámbito, se ha considerado que la introducción de algoritmos de inteligencia artificial puede proporcionar una mejora en la precisión y velocidad de las tareas.

Se ha decidido estudiar cuatro algoritmos: el algoritmo de asignación óptima de Munkres, el algoritmo genético, el de enrutamiento de máquinas y el de cuellos de botella.

La decisión de incorporar algoritmos de inteligencia artificial se basa en su capacidad demostrada con anterioridad, tanto en este sector como en otros, para ofrecer soluciones mejores que las que ofrecen los sistemas heurísticos tradicionales. Esta decisión se apoya en estudios y fuentes que se presentan a continuación en el apartado 2.4. Estado del arte donde se muestra cómo se han aplicado estas técnicas con éxito en contextos similares, lo que justifica su adopción en el presente proyecto.

Además, la programación óptima con cada uno de estos algoritmos se implementará a través de una interfaz de usuario de Streamlit fácil de usar. Esta interfaz ofrece un resultado muy práctico para la industria, ya que no solo es interactiva, sino que también permite visualizar diferentes escenarios en la planificación de la producción. Asimismo, esta facilita la identificación de cuellos de botella y la evaluación de posibles ajustes en la programación para mejorar la eficiencia operativa.

Por todo lo comentado anteriormente, la motivación de este proyecto se podría decir que radica en la mejora y expansión de las capacidades de planificación de la producción empresarial, empleando algoritmos de inteligencia artificial e integrando nuevos criterios. Como se ofrece una alternativa más eficiente y completa al módulo existente APO de SAP, se espera que esta investigación contribuya no solo a la optimización de los procesos de fabricación de motores, sino también al avance hacia prácticas empresariales más sostenibles y responsables.

2.2. Objetivos

El presente proyecto tiene como objetivo principal: **desarrollar algoritmos de optimización en la producción de motores que permitan completar y mejorar el módulo de Advanced Planned and Optimization (APO) de SAP S/4 HANA.**

Con el enfoque puesto en mejorar la eficiencia y la efectividad en la programación de la producción, se tiene en cuenta información sobre las operaciones, los puestos de trabajo y los recursos disponibles en la planta de fabricación.

Los objetivos específicos del proyecto se detallan a continuación:

- Adquirir los conocimientos relacionados con el módulo de *Advanced Planned and Optimization* (APO) de SAP S/4 HANA.
- **Simular los datos** del escenario simple (base) siguiendo las especificaciones de Navantia.
- **Estudiar el escenario simple** con los cuatro modelos de optimización especificados y evaluación de los resultados sobre el diagrama de Gantt juntamente con la métrica de *fitness* definida: mínimo tiempo total.
- Profundizar en el estudio de los algoritmos en Google Colab.
- Implementar los algoritmos en una **interfaz de usuario** en Streamlit para visualizar los resultados de la optimización sobre el diagrama de Gantt para los algoritmos desarrollados.

2.3. Materiales empleados

Para el correcto desarrollo del proyecto, y tal y como se ha detallado en los objetivos específicos, en primer lugar, se ha iniciado una fase de adquisición de los conocimientos pertinentes. Para ello, se ha seguido un proceso de aprendizaje lógico y estructurado, similar al que se aplicaría a perfiles junior en la empresa, con el objetivo de comprender y dominar estos sistemas.

A continuación, se detalla la secuencia y el progreso de la investigación llevada a cabo, centrándose en la evolución y funcionalidad de los sistemas ERP, con especial énfasis en SAP S/4 HANA. Se explora detalladamente su arquitectura, módulos y aplicaciones específicas. Este análisis busca establecer una base sólida para la introducción de algoritmos de inteligencia artificial destinados a optimizar los procesos de fabricación.

2.3.1. ¿Qué es un ERP?

Los sistemas de Planificación de Recursos Empresariales, conocidos como ERP (*Enterprise Resource Planning*), son herramientas de *software* centralizadas diseñadas para gestionar de forma eficiente y coordinada todos los procesos centrales de un negocio (Castro-Vázquez et al., 2021). Estos son esenciales para facilitar la toma de decisiones de forma unificada entre los distintos departamentos y se han convertido en imprescindibles de cara a mejorar la competitividad en el mercado empresarial (Yen & Sheu, 2004). Más allá de ser una herramienta de *software*, es una nueva filosofía en la gestión empresarial, representando una forma de “hacer negocios” (Jacobs & Whybark, 2000).

La Figura 1 muestra los beneficios que un negocio puede obtener mediante la implementación de un ERP.

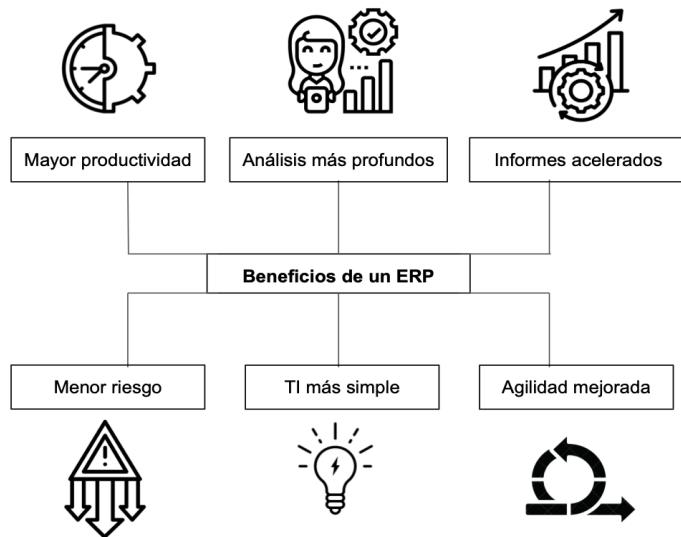


Figura 1. Beneficios de la implementación de un ERP

Los sistemas ERP se componen de módulos integrados que se especializan en distintas áreas del negocio y comparten una base de datos centralizada. Esta configuración permite personalizar el sistema seleccionando los módulos necesarios y escalar en función de las necesidades y requerimientos del negocio. Los módulos disponibles se detallan en la Figura 2. En cuanto a su adquisición, pueden ser obtenidos a través de un modelo de suscripción en la nube (*software como servicio*) o mediante licencias tradicionales para ser implementados localmente en las instalaciones de la empresa, lo que se conoce como *on-premise* (Gessa, Jiménez & Sancha, 2023).

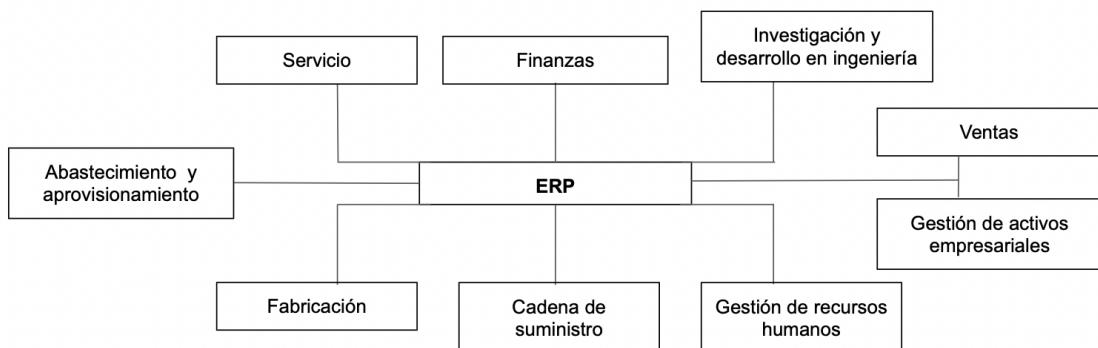


Figura 2. Principales módulos de un sistema ERP

Sin embargo, a pesar de las grandes expectativas y beneficios potenciales de los ERP, no necesariamente cumplen con las expectativas de las empresas. Como señala Davenport, "varios casos de proyectos fallidos deben ser motivo de preocupación" (Davenport, 1998). La presente investigación se centra en explorar y desarrollar distintos algoritmos para escenarios diversos, proporcionando así soluciones más efectivas.

2.3.2. ¿Qué es SAP?

SAP (*System Application and Products in Data Processing*) es la empresa referente de sistemas de ERP en Europa. Ofrece herramientas de gestión y tecnología de la información diseñadas para optimizar la planificación y ejecución de operaciones empresariales de manera más eficiente y efectiva (SAP, s.f.). Investigaciones recientes plasman que SAP mantiene su posición líder en el mercado de ERP, diferenciándose de la competencia por la alta satisfacción de sus usuarios, especialmente en las pequeñas y medianas empresas (Gessa, Jiménez & Sancha, 2023).

Desde sus inicios, SAP ha evolucionado considerablemente, pasando de ser un *software* de gestión de inventarios a un ERP integral con capacidades de procesamiento en tiempo real y análisis de datos. Esta evolución ha sido posible gracias a la integración de SAP HANA, una base de datos multimodelo revolucionaria ya que almacena datos en memoria, lo que permite ejecutar analíticas avanzadas y transacciones de alta velocidad en un único sistema.

Continuando con la evolución mencionada, SAP ha ido lanzando numerosas versiones de su sistema ERP a lo largo de los años. En particular, este estudio se centra en SAP S/4 HANA 2023, la versión más reciente y actualmente implementada en Navantia. Esta versión destaca por ofrecer nuevas funcionalidades y una capacidad mejorada para gestionar procesos empresariales complejos en tiempo real, reflejando los últimos avances en tecnología ERP (SAP, 2023).

2.3.3. SAP S/4 HANA

SAP S/4 HANA emerge como la solución integral más avanzada creada por la empresa alemana, siendo la actualización más reciente en el ámbito del ERP. Su introducción implica la sustitución de soluciones previas como SAP R/3, que fue implementada anteriormente en Navantia, y SAP ERP.

La integración de inteligencia artificial y su adaptabilidad a entornos híbridos, en la nube y en local, la convierten en una opción completa para mejorar procesos empresariales. Proporciona un único punto de referencia para obtener análisis detallados en áreas clave, que se ilustran en la Figura 3.

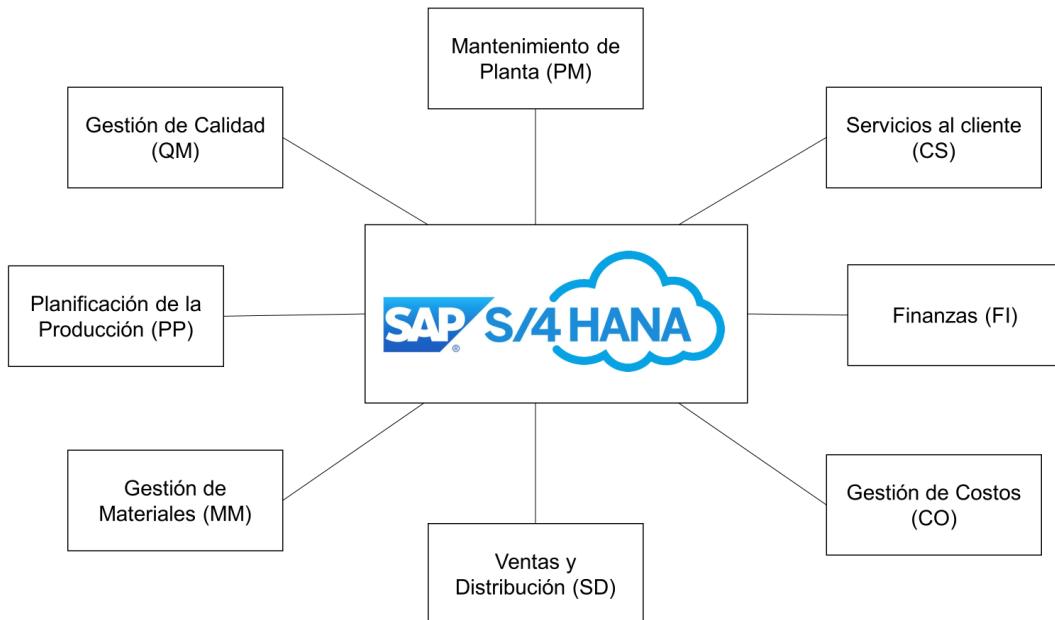


Figura 3. Módulos de SAP S/4 HANA

2.3.4. Planificación de la Producción

En relación con las necesidades específicas del proyecto, se ha decidido poner el foco en el módulo de Planificación de la Producción (PP) de SAP S/4 HANA. Según la documentación oficial de SAP, este módulo proporciona las herramientas necesarias para garantizar la disponibilidad de materiales y asegurar una planificación adecuada para cubrir los requisitos de producción. A su vez, permite planificar y programar de manera más eficaz los centros de trabajo, especialmente de aquellos en los que se dan situaciones de cuello de botella.

Dentro del módulo de Planificación de la Producción cabe destacar los siguientes dos submódulos:

- **Planificación de Necesidades de Materiales (MRP).** Garantiza la disponibilidad adecuada de materiales para uso interno y distribución, equilibrando la optimización, la minimización de costos y satisfaciendo la demanda. Supervisa los

niveles de inventario y genera propuestas de adquisición automáticas para compras y producción. La figura del controlador de MRP es la responsable de especificar requisitos, calcular cantidades necesarias y determinar cuándo crear propuestas de pedido, utilizando información detallada sobre el inventario, plazos de entrega y tiempos de aprovisionamiento.

- **Órdenes de Fabricación (SFC).** Tienen como objetivo optimizar este proceso y garantizar la calidad del producto final. En este submódulo se crean, programan y monitorizan las órdenes de fabricación. Asimismo, se lleva a cabo el control de procesos, la asignación de recursos, el registro de datos de producción y la supervisión en tiempo real.

2.3.5. Composición de una orden de fabricación

Una orden de fabricación está compuesta por:

1. **Número de orden:** identificador único para la orden de fabricación.
2. **Material:** producto que se va a fabricar.
3. **Cantidad:** cantidad del producto que se va a fabricar.
4. **Fecha de inicio:** fecha en que se debe de comenzar la producción.
5. **Fecha de finalización:** fecha en que se debe completar la producción.
6. **Planta:** planta donde se va a fabricar el producto.
7. **Centro de trabajo:** centro de trabajo donde se fabricará el producto.
8. **Unidad de medida:** unidad de medida para la cantidad del producto.
9. **Estatus de la orden:** estado actual de la orden de fabricación (liberada, en ejecución, completada).

2.3.6. Diagrama de Gantt

Un diagrama de Gantt es una herramienta esencial para la planificación y gestión de proyectos, extensamente utilizada para representar las tareas y principales hitos de un proyecto de forma práctica y visual. Se trata de un conjunto de barras horizontales que representan la duración de cada una de las actividades individuales que componen el conjunto de un proyecto o plan. Estas barras se extienden a lo largo de un eje temporal, mostrando la fecha de inicio y la fecha de finalización previstas de cada actividad.

Se decide incorporarlo en el presente trabajo con la finalidad de mostrar de manera clara y comprensible los plazos y evolución de cada operación de fabricación. En concreto, se utilizan dos tipos específicos de diagramas de Gantt: uno que muestra la secuencia temporal de las operaciones para cada orden, y otro que ilustra la secuencia temporal de las operaciones para cada recurso. Esta herramienta no solo facilita la comprensión del flujo de trabajo, sino que asegura que las operaciones se desarrollen de manera continua y eficiente, minimizando los tiempos de inactividad y maximizando la utilización de los recursos. Además, el uso del diagrama es particularmente beneficioso porque destaca los retrasos y facilita la adaptabilidad y respuesta a cambios, mostrando de forma instantánea el impacto de los cambios en un proyecto.

2.4. Estado del arte

En el presente apartado se revisa la bibliografía que se ha considerado relevante, proporcionando una base sólida que justifica cada una de las herramientas y técnicas empleadas, así como la elección y adaptación de los algoritmos en el contexto de la optimización del proceso de producción de motores. Se pone el foco en los avances de la industria relacionados con la **optimización del tiempo total de producción** mediante los cuatro algoritmos de *Machine Learning* seleccionados: el algoritmo de Munkres, el algoritmo genético, el de enrutamiento de máquinas y el de cuellos de botella.

Para el **estudio, introducción e implementación** de SAP y los sistemas ERP, como libro de referencia, *Production Planning with SAP S/4HANA* proporciona una guía completa para la configuración y uso de SAP S/4HANA en la planificación de la producción empresarial (Akhtar, 2021). Además, se ha consultado la página web oficial de SAP (SAP, s.f.) y se ha considerado el estudio de Gessa, Jiménez y Sancha (2023), que explora la adopción de sistemas ERP en tiempos desafiantes, pese a que la implementación se realice en R en lugar de en Python.

El uso de **datos sintéticos** es una práctica extendida en investigaciones donde la disponibilidad de datos reales es nula, limitada, o se requiere una mayor flexibilidad para explorar diferentes escenarios de fabricación (Smith et al., 2020). Los datos se generan de manera que simulan condiciones realistas del entorno de producción, permitiendo una evaluación más completa de los algoritmos propuestos.

El proceso que se debe seguir a la hora de generar los datos es: comprender la estructura del proceso de fabricación, generar las clases para modelar los elementos fundamentales de la gestión de la planta y, finalmente, implementar una función para generar los datos. En este caso en particular se han generado datos sintéticos de órdenes, operaciones y recursos, utilizando valores aleatorios dentro de rangos específicos para simular diferentes escenarios de producción. Asimismo, se ha implementado un sistema automatizado que, al introducir el usuario las tres variables que se solicitan en pantalla, genera el escenario gracias al código Python implementado.

Tal y como se ha comentado anteriormente, para optimizar un proceso de producción el uso de algoritmos ha demostrado ser una herramienta vital para mejorar la eficiencia y la precisión. La implementación de los algoritmos seleccionados ha sido objeto de numerosos estudios y aplicaciones prácticas que afirman su efectividad en diversos contextos. Para justificar el uso de estos algoritmos en el proyecto, se hace referencia a trabajos anteriores que han explorado y aplicado estas técnicas en la optimización de procesos.

Al tratarse de un problema de asignación, el algoritmo de Munkres es el que primero se introduce. Esta técnica de optimización se ha utilizado con anterioridad para resolver múltiples problemas de asignación lineal. En el estudio de Rusdiana et al. (2019) se muestra el éxito de haber aplicado este algoritmo en una industria artesanal de fabricación de bolsas. En concreto, se programa la asignación de los empleados y se consigue mejorar la eficiencia operativa, minimizando el tiempo total de producción y produciendo un aumento de los ingresos.

Por otro lado, los algoritmos genéticos se caracterizan por su capacidad para manejar grandes espacios de búsqueda y encontrar soluciones cercanas a la óptima. Dadas sus características, son una herramienta ampliamente utilizada en la optimización de sistemas de manufactura. La investigación de Suesca et al. (2023) presenta un enfoque innovador sobre la optimización de sistemas de manufactura celular mediante el uso de algoritmos genéticos. Este estudio presenta la implementación de un algoritmo genético que integra la formación de celdas y el balanceo de cargas de trabajo, obteniendo resultados muy cercanos al valor óptimo. La configuración de estas soluciones, pese a no ser la óptima, proporcionó mejoras significativas, originando una adecuada agrupación de las celdas y un flujo continuo de las piezas, reduciendo significativamente cuellos de botella.

En cuanto al algoritmo de enrutamiento de máquinas, en el contexto de la producción industrial, se busca la minimización del tiempo de fabricación determinando la secuencia más eficiente de los procesos en las máquinas disponibles. La investigación de Sheng-Chun Kao et. al (2019), sobre el enrutamiento en redes de chips analiza cómo los algoritmos de aprendizaje por refuerzo pueden aprender y seleccionar la secuencia óptima de rutas para maximizar el rendimiento de la red. Este enfoque puede relacionarse directamente con los algoritmos de enrutamiento de máquinas utilizados en la optimización de la producción, ya que, en ambos casos, el objetivo es determinar la secuencia óptima de operaciones para minimizar un objetivo específico.

Finalmente, de acuerdo con el artículo de Britto Agudelo et al. (2007), la programación de la producción en sistemas de manufactura tipo taller se puede mejorar significativamente mediante el uso del algoritmo de cuellos de botella, donde se proporciona una solución que mejora sucesivamente, incluyendo nuevos criterios para la elección de máquinas críticas y estrategias de diversificación e intensificación. Las evaluaciones realizadas con problemas clásicos demuestran la competitividad del algoritmo de cuello de botella en términos de calidad de soluciones y eficiencia computacional.

Como plataforma para el desarrollo y práctica de los algoritmos se ha decidido utilizar Google Colab, ya que se basa en un entorno de programación en la nube que facilita el trabajo colaborativo. A su vez, esta fomenta el compromiso en la colaboración y el trabajo en equipo (Werth et al. ,2022), lo cual se ha establecido como prioridad para el presente proyecto ya que requiere múltiples contribuciones y cambios.

Para la implementación de la interfaz de usuario interactiva se ha decidido utilizar la librería Streamlit, que permite crear aplicaciones web con visualizaciones interactivas de forma rápida y personalizable (Raghavendra, S., 2023).

CAPÍTULO 3: MATERIALES Y MÉTODOS

En este capítulo se detalla la simulación y estructura de los datos empleados en el proyecto.

3.1. Simulación de datos

Para la simulación de los datos se crea la función *generar_datos_sinteticos*, en la que se consideran las suposiciones especificadas por Navantia bajo unas características idílicas en su planta de producción.

En primer lugar, cada recurso tiene una capacidad unitaria, lo que significa que solo puede trabajar en una operación a la vez, asegurando que no haya sobrecarga de trabajo y que cada recurso esté completamente dedicado a una sola tarea. La planta opera continuamente en un turno de 24 horas, permitiendo programar las operaciones en cualquier momento del día, maximizando el uso del tiempo disponible y eliminando las limitaciones de horarios para otorgar una mayor flexibilidad en la planificación.

Cada orden de producción consta exactamente de seis operaciones, independientemente del tipo de motor o las especificaciones del pedido, lo cual simplifica la planificación y la secuenciación de las tareas. Se tienen tres tipos de materiales y seis tipos de recursos, tal y como se puede observar en la Figura 4. No se considera el tiempo de preparación entre operaciones (*setup*), por lo que no existen los tiempos de inactividad, permitiendo optimizar más fácilmente la secuencia de operaciones y minimizar las interrupciones.

```
# Función para generar los datos sintéticos de órdenes, operaciones y recursos
def generar_datos_sinteticos(num_ordenenes, holgura):
    num_operaciones = 6
    # Tenemos 3 tipos de materiales (motores)
    materiales = ['Motor MTU 2000', 'Motor MTU 4000', 'Motor MTU 8000']
    # Tenemos 6 tipos de recursos (máquinas)
    recursos = [
        Recurso(1, 'Torno CNC', 'Mecanizado', 1),
        Recurso(2, 'Fresadora CNC', 'Mecanizado', 1),
        Recurso(3, 'Montaje', 'Montaje', 1),
        Recurso(4, 'Pintura', 'Acabado', 1),
        Recurso(5, 'Robot de Ensamblaje', 'Montaje Avanzado', 1),
        Recurso(6, 'Estación de Pruebas', 'Pruebas', 1)
    ]
    # Tenemos 6 tipos de operaciones
    operaciones_mtu = ['Mecanizado base', 'Mecanizado bloque', 'Montaje pistones', 'Montaje cigüeñal',
                       'Ensamblaje motor', 'Pruebas finales']
```

Figura 4. Función *generar_datos_sinteticos*

Se añade un margen de holgura para proporcionar flexibilidad en la programación, permitiendo ajustar la planificación ante imprevistos o variaciones en el proceso de producción, asegurando que las operaciones se completen dentro del plazo establecido. Las operaciones tienen una duración realista que varía entre 8 y 72 horas (hasta 3 días), asegurando que las tareas se completen en un marco temporal razonable. La fecha de inicio se establece de manera aleatoria, utilizando un rango entre 24 y 240 horas (de 1 a 10 días) desde el momento de la planificación, sin tener en cuenta días festivos, lo que permite una mayor flexibilidad en la programación inicial y evita conflictos con posibles interrupciones externas. Se asume que la producción comienza en la fecha de inicio tardía y continúa ininterrumpidamente hasta completar todas las operaciones, asegurando que no haya pausas prolongadas en el proceso, lo que contribuye a la eficiencia y la optimización del tiempo total de producción.

En la generación de los datos también participa el usuario, pues como se puede ver en la Figura 5, se solicita en pantalla (*input*) el número de órdenes y la holgura.

```
# Se especifica que el número de operaciones es 6
num_operaciones = 6

# Solicitar entradas del usuario
num_ordenes = int(input("Ingrese el número de órdenes a generar: "))
holgura = int(input("Ingrese la holgura en horas: "))

# Ejecutar el proceso de generación de datos
df_ordenes, df_operaciones, df_recursos = generar_datos_sinteticos(num_ordenes, holgura)
```

Figura 5. Inputs solicitados al usuario.

3.2. Análisis y estructura de los datos

Los datos utilizados en el desarrollo del proyecto, tal y como se ha explicado en el apartado anterior (3.1), se han simulado siguiendo las especificaciones de la planta de producción proporcionadas por Navantia. Esta informa que los datos provenientes de las máquinas de la fábrica se obtienen ya estructurados de manera que no requieren procesos adicionales de limpieza o preprocesamiento. Utilizar datos simulados permite no solo examinar la situación real de la planta, sino también explorar otras circunstancias y casuísticas potenciales en el proceso de producción. Obtener diversos escenarios de producción asegura una mejor comprensión del proceso y aplicaciones más flexibles de las soluciones desarrolladas.

Se decide estructurar los datos inicialmente en un archivo Excel para poder entender el funcionamiento del proceso. Se organiza la información en tres hojas: órdenes de fabricación (Ordenes), operaciones (Operaciones) y recursos (Recursos). A continuación, se detalla y se muestra la cabecera de la estructura de cada hoja de los datos iniciales que, en este caso particular, se caracterizan por tener un tiempo de preparación de las máquinas (*setup*) igual a 0 y una capacidad diaria (cantidad) igual a 1.

3.2.1. Hoja Ordenes

Esta hoja contiene información sobre las órdenes de fabricación, es decir, sobre los motores que se van a producir en el proceso. Es importante señalar que en el proceso de fabricación se trabaja con tres tipos de órdenes: Motor MTU 2000, Motor MTU 4000 y Motor MTU 8000.

Como se puede observar en la Tabla 1. Estructura de la Hoja Ordenes., las columnas incluidas son:

1. **ID Orden:** Identificador único para cada orden de fabricación.
2. **Material:** Tipo de material o producto que se está fabricando (Motor MTU 2000, Motor MTU 4000 y Motor MTU 8000).
3. **Cantidad:** Cantidad de unidades del producto que se van a fabricar a diario.
4. **Fecha Inicio Temprana:** Fecha y hora más temprana de inicio para la orden.
5. **Fecha Fin Temprana:** Fecha y hora más temprana de finalización para la orden.
6. **Fecha Inicio Tardía:** Fecha y hora más tardía de inicio permitida para la orden.
7. **Fecha Fin Tardía:** Fecha y hora más tardía de finalización permitida para la orden.

ID Orden	Material	Cantidad	Fecha Inicio Temprana	Fecha Fin Temprana	Fecha Inicio Tardía	Fecha Fin Tardía
1	Motor MTU 8000	1	2024-05-23 21:00:00	2024-06-06 05:00:00	2024-05-28 01:00:00	2024-06-10 09:00:00
2	Motor MTU 2000	1	2024-05-27 19:00:00	2024-06-08 15:00:00	2024-05-31 23:00:00	2024-06-12 19:00:00
3	Motor MTU 2000	1	2024-05-28 20:00:00	2024-06-09 10:00:00	2024-06-02 00:00:00	2024-06-13 14:00:00
4	Motor MTU 8000	1	2024-05-25 20:00:00	2024-06-10 08:00:00	2024-05-30 00:00:00	2024-06-14 12:00:00
5	Motor MTU 8000	1	2024-05-27 09:00:00	2024-06-09 17:00:00	2024-05-31 13:00:00	2024-06-13 21:00:00

Tabla 1. Estructura de la Hoja Ordenes.

3.2.2. Hoja Operaciones

Aquí se detallan las operaciones necesarias para elaborar cada orden de fabricación. Todas las órdenes del proceso están compuestas por el mismo número de operaciones, por lo tanto, siguen una estructura uniforme. En el proceso de fabricación se trabaja con seis tipos de operaciones: Mecanizado base, Mecanizado bloque, Montaje pistones, Montaje cigüeñal, Ensamblaje motor y Pruebas finales.

Como se puede observar, en la Tabla 2, las columnas incluidas son:

1. **ID Orden:** Identificador de la orden a la que pertenece la operación.
2. **ID Operación:** Identificador único para cada operación dentro de una orden.
3. **Descripción Operación:** Breve descripción de la operación.
4. **Tiempo Setup:** Tiempo de preparación (en horas) necesario antes de iniciar la operación.
5. **Duración:** Duración de la operación (en horas).
6. **ID Recurso:** Identificador del recurso (máquina o equipo) utilizado en la operación.
7. **Descripción Recurso:** Breve descripción del recurso.
8. **Tipo Recurso:** Tipo de recurso utilizado (Mecanizado, Montaje, Acabado Ensamblaje Montaje Avanzado y Pruebas).

ID Orden	ID Operación	Descripción Operación	Tiempo Setup	Duración	ID Recurso	Descripción Recurso	Tipo Recurso
1	0	Mecanizado base	0	9	1	Torno CNC	Mecanizado
1	1	Mecanizado bloque	0	49	2	Fresadora CNC	Mecanizado
1	2	Montaje pistones	0	53	3	Montaje	Montaje
1	3	Montaje cigüeñal	0	39	4	Pintura	Acabado
1	4	Ensamblaje motor	0	35	5	Robot de Ensamblaje	Montaje Avanzado

Tabla 2. Estructura de la Hoja Operaciones.

3.2.3. Hoja Recursos

Esta hoja contiene información sobre los recursos disponibles para cada una de las operaciones. Los recursos disponibles son: Torno CNC, Fresadora CNC, Montaje, Pintura, Robot de Ensamblaje y Estación de Pruebas.

Como se puede observar en la Tabla 3, las columnas incluidas son:

- ID Recurso:** Identificador único para cada recurso.
- Descripción:** Breve descripción del recurso (Torno CNC, Fresadora CNC, Montaje, Pintura, Robot de Ensamblaje y Estación de Pruebas)
- Tipo:** Tipo de recurso.
- Capacidad Diaria:** Capacidad diaria del recurso. Indica cuántas unidades puede procesar en un día.

ID Recurso	Descripción	Tipo	Capacidad Diaria
1	Torno CNC	Mecanizado	1
2	Fresadora CNC	Mecanizado	1
3	Montaje	Montaje	1
4	Pintura	Acabado	1
5	Robot de Ensamblaje	Montaje Avanzado	1

Tabla 3. Estructura de la Hoja Recursos.

Esta estructuración de los datos facilita la planificación y el seguimiento de cada paso en el proceso de producción. Asimismo, mejora la visibilidad y control del flujo de trabajo, lo cual es esencial para optimizar los tiempos de producción.

3.3. Descripción de los algoritmos usados

Tras la generación de los datos, la investigación comienza con una fase exploratoria en la que se identifican y evalúan los siguientes algoritmos potenciales:

- **Planificación y programación lineal.** Utiliza modelos matemáticos para asignar recursos de forma óptima, minimizando los costos de producción como consecuencia de reducir los tiempos de inactividad de los operarios.
- **Algoritmos Genéticos.** Utilizan principios de evolución biológica y se aplican para encontrar soluciones óptimas o cercanas a la óptima modelando la producción como un conjunto de genes, mejorando iterativamente las soluciones propuestas.
- **Algoritmo de asignación óptima.** Se centra en la distribución equilibrada de la carga de trabajo de cada máquina, teniendo en cuenta los tiempos de cambio de herramientas y la complejidad de las operaciones.

- **Algoritmo de enrutamiento de máquinas.** Establece la secuencia más eficiente de procesos en las máquinas, considerando la minimización del tiempo total de producción.
- **Algoritmo de cuello de botella.** Identifica y examina los puntos críticos en la línea de producción que limitan la capacidad productiva total del sistema.

Después de un minucioso estudio y análisis de las posibles opciones, se decide poner el foco en aquellos que muestran un mayor potencial para abordar de manera integral los distintos aspectos de la producción de motores.

Los algoritmos seleccionados han sido los siguientes:

- **Asignación Óptima: Algoritmo de Munkres (Húngaro).** Empleado para resolver problemas de asignación, tradicionalmente con el objetivo de minimizar los costos asociados a la asignación de tareas (TopCoder, 2021). En el presente proyecto se adapta para centrarse en la minimización del objetivo, es decir, en el tiempo total de fabricación. Se ha decidido seleccionar este algoritmo ya que es particularmente útil en situaciones como la asignación de trabajadores a tareas o máquinas a trabajos, donde cada asignación tiene un costo o beneficio asociado y se desea optimizar el resultado global.

El funcionamiento del algoritmo, según se detalla (TopCoder, 2021), se basa en una serie de pasos repetitivos que ajustan la matriz de costos hasta encontrar la solución óptima. Se puede simplificar en los siguientes pasos:

1. **Identificación de operaciones y recursos.** Se identifican las operaciones involucradas en el proceso de fabricación de motores y las máquinas o recursos disponibles en la planta de producción.

Operaciones	Recursos
Mecanizado Base	Torno CNC
Mecanizado bloque	Fresadora CNC
Montaje pistones	Montaje
Montaje cigüeñal	Pintura
Ensamblaje motor	Robot de Ensamblaje
Pruebas finales	Estación de Pruebas

Figura 6. Identificación de operaciones y recursos.

2. **Definición y relleno de la matriz de costos.** Se formula la matriz de costos o beneficios, donde cada fila representa una máquina de fabricación y cada columna una tarea específica de producción. En este caso, el ‘costo’ es el tiempo total que cada máquina tarda en completar una tarea.

	Mecanizado base	Montaje pistones	Ensamblaje motor
Torno CNC	10	9	5
Montaje	9	8	3
Pintura	6	4	7

Figura 7. Definición y relleno matriz de costos.

3. **Reducción de la matriz.** Se resta a cada elemento el menor valor de su fila (Figura 8) y luego el menor valor de su columna (Figura 9). Este proceso transforma la matriz para que los ceros representen las asignaciones potenciales de menor tiempo de operación.

	Mecanizado base	Montaje pistones	Ensamblaje motor
Torno CNC	10 - 5 = 5	9 - 5 = 4	5 - 5 = 0
Montaje	9 - 3 = 6	8 - 3 = 5	3 - 3 = 0
Pintura	6 - 4 = 2	4 - 4 = 0	7 - 4 = 3

Figura 8. Resta del menor valor de cada fila.

	Mecanizado base	Montaje pistones	Ensamblaje motor
Torno CNC	$5 - 2 = 3$	$4 - 0 = 4$	$0 - 0 = 0$
Montaje	$6 - 2 = 4$	$5 - 0 = 5$	$0 - 0 = 0$
Pintura	$2 - 2 = 0$	$0 - 0 = 0$	$3 - 0 = 3$

Figura 9. Resta del menor valor de cada columna.

4. **Cubrimiento de ceros con el mínimo número de líneas.** Utilizando el mínimo número de líneas horizontales o verticales, se cubren todos los ceros en la matriz (Figura 10). En caso de que el número de líneas sea igual al número de filas o columnas, se tendrá la solución óptima. En este caso aún no se ha obtenido la solución óptima.

	Mecanizado base	Montaje pistones	Ensamblaje motor
Torno CNC	3	4	0
Montaje	4	5	0
Pintura	0	0	3

Figura 10. Cubrimiento de ceros con el mínimo número de líneas.

5. **Ajuste de la matriz.** Si la solución óptima aún no se ha obtenido, se ajusta la matriz de modo que se selecciona el menor de los valores no cubiertos y se resta a los elementos no cubiertos de cada fila. Esto prepara la matriz para una nueva iteración de reducción y cubrimiento de ceros.

	Mecanizado base	Montaje pistones	Ensamblaje motor
Torno CNC	$3 - 3 = 0$	$4 - 3 = 1$	0
Montaje	$4 - 3 = 1$	$5 - 3 = 2$	0
Pintura	0	0	3

Figura 11. Ajuste de la matriz de costos.

	Mecanizado base	Montaje pistones	Ensamblaje motor
Torno CNC	0	1	0
Montaje	1	2	0
Pintura	0	0	3

Figura 12. Finalización del algoritmo.

6. **Asignación final.** Una vez todos los ceros están cubiertos con el número mínimo de líneas correspondiente al número de filas o columnas, las posiciones de estos ceros indican la asignación óptima. Se hace de modo que se inicia asignando la fila con menor número de ceros con la columna en la que se asignó (tiene valor cero) y así sucesivamente.

	Mecanizado base	Montaje pistones	Ensamblaje motor
Torno CNC	0	1	0
Montaje	1	2	0
Pintura	0	0	3

Figura 13. Asignación final.

- **Algoritmo Genético.** Este algoritmo parte del proceso genético de los organismos y de cómo interactúan entre ellos, lo que provoca cambios en sus características de generación a generación (Grefenstette, 1986). Como se ha comentado anteriormente, es particularmente útil cuando se manejan grandes volúmenes de datos y la carga computacional es elevada, lo que hace muy costoso alcanzar una solución óptima exacta. Minimiza el tiempo de simulación y aporta una aproximación de lo que sería la mejor combinación proporcionando una aproximación eficiente y efectiva, ofreciendo resultados viables en tiempos razonables. Se ha optado por implementar este tipo de algoritmo ya que es robusto, puede hacer frente a una gran variedad de tipos de problemas y trabaja con funciones no lineales de una manera eficiente.

El funcionamiento del algoritmo se puede simplificar en los siguientes pasos:

1. **Función de aptitud (fitness).** Se calcula la eficacia de las asignaciones realizadas sobre el cromosoma en el algoritmo. Esta se basa en los tiempos

totales asociados a cada operación, que están compuestos por el total necesario para completar todas las operaciones asignadas y el tiempo muerto (setup) asociado. Un valor de fitness más alto indica una asignación más eficiente.

2. **Creación de un cromosoma aleatorio.** Se crea un cromosoma, que en este caso es una lista de asignaciones de recursos a operaciones, de manera aleatoria. Cada asignación está representada por una tupla que contiene el ID de la orden, el ID de la operación y el ID del recurso asignado. La función itera sobre todas las operaciones y asigna un recurso aleatorio a cada una.
 3. **Selección de los progenitores.** Se realiza la selección de los progenitores del algoritmo genético y se decide utilizar el método de selección por torneo, que selecciona varios aspirantes al azar de la población y elige el mejor (el de mayor *fitness*) como padre. Se repite este proceso hasta obtener una lista de padres seleccionados del mismo tamaño que la población original.
 4. **Cruce de los progenitores.** Se realiza el cruce de dos cromosomas (padre1 y padre2) para generar otros dos nuevos (hijo1 e hijo2). Se realiza para combinar la información genética de los padres y así crear descendencia que tenga *características de ambos*.
 5. **Mutación.** Se introducen mutaciones en un cromosoma basado en una tasa de mutación especificada. Esto se realiza con el objetivo de mantener la diversidad genética en la población, lo que ayuda a evitar el estancamiento en óptimos locales.
 6. **Sustitución.** Se sustituye la población original por la nueva población generada.
- **Algoritmo de enrutamiento de máquinas.** En este proceso se busca la mejor secuencia de máquinas para procesar un conjunto de tareas, minimizando un objetivo específico, como el tiempo total de procesamiento, el tiempo de espera máximo o el uso de la máquina (Akturk, M. S., & Zhang, X. 2005). Este algoritmo se ha elegido ya que se desea minimizar los tiempos de inactividad de las máquinas, un factor crucial para mejorar la eficiencia y reducir los tiempos de fabricación en el proceso de producción de motores.

El funcionamiento del algoritmo se puede simplificar en los siguientes pasos:

1. **Identificación de operaciones y recursos.** Se realiza de igual forma que en el Algoritmo Húngaro (*Figura 6*).
2. **Asignación basada en el costo mínimo.** Se busca identificar qué máquina incurre en el menor costo para cada una de las operaciones. Para cada fila de la matriz, se selecciona la columna (máquina) que tiene el valor más bajo.
3. **Asignación de tareas a máquinas.** Se determina la secuencia más eficiente de modo que cada tarea sea procesada por las diferentes máquinas disponibles.

Tareas	Recursos	Tipo de Proceso
Mecanizado Base	Torno CNC	Mecanizado
Mecanizado bloque	Fresadora CNC	Mecanizado
Montaje pistones	Montaje	Montaje
Montaje cigüeñal	Montaje	Montaje
Ensamblaje motor	Robot de Ensamblaje	Montaje Avanzado
Pruebas finales	Estación de Pruebas	Pruebas

Figura 14. Asignación de tareas y máquinas.

4. **Generación de la secuencia óptima.** Una vez se poseen las asignaciones, se establece una secuencia óptima de tareas que minimiza el costo total de producción.
 5. **Minimización del tiempo de fabricación.** Al asignar cada tarea a la máquina que tiene el menor costo se obtiene una minimización del tiempo total de fabricación.
- **Algoritmo de cuellos de botella.** Algoritmo diseñado para identificar y optimizar los puntos críticos más significativos dentro de un proceso. Estos puntos críticos, también conocidos como cuellos de botella, son aquellos recursos o etapas del proceso que limitan la capacidad total del sistema (Ventura Silva, G., et al., 2021).

Se ha seleccionado este algoritmo porque puede ayudar a identificar las etapas del proceso de fabricación de motores que hacen el proceso de producción más lento, lo que permite dirigir los esfuerzos de optimización a maximizar la capacidad y el rendimiento del sistema en general.

El funcionamiento del algoritmo se puede simplificar en los siguientes pasos:

1. **Identificación del proceso de producción de motores.** Se analiza y comprende en detalle el proceso completo de producción de motores, desde el mecanizado de las piezas hasta las pruebas finales.

Motores	Recursos	Tipo de Proceso
Motor MTU 2000	Torno CNC	Mecanizado
Motor MTU 4000	Fresadora CNC	Mecanizado
Motor MTU 8000	Montaje	Montaje
	Montaje	Montaje
	Robot de Ensamblaje	Montaje Avanzado
	Estación de Pruebas	Pruebas

Figura 15. Identificación del proceso de producción

2. **Recopilación de datos.** Se recopilan datos sobre tiempos de procesamiento, capacidad de recursos (máquinas), y otros factores relevantes.
3. **Inicialización de tiempos de procesamiento por recurso.** Se establece un diccionario donde se almacenará el tiempo total de procesamiento acumulado para cada recurso.
4. **Cálculo de los tiempos totales de procesamiento.** Se itera sobre cada operación registrada en los datos recopilados, sumando la duración de cada una de las operaciones y el tiempo total de procesamiento del recurso correspondiente. Esto ayuda a identificar cuánto tiempo está ocupada cada máquina.

5. **Visualización de cuellos de botella.** Se genera un gráfico de barras que muestra los recursos identificados como cuellos de botella, con sus tiempos totales de procesamiento.

La elección de estos algoritmos se fundamenta en su capacidad para mejorar algunos aspectos clave de la producción óptima de motores: desde la asignación eficiente de recursos hasta la identificación y optimización de cuellos de botella, permitiendo así una mejora integral en la eficiencia y el rendimiento del proceso de fabricación. Estas estrategias no solo promueven una mejora en la eficiencia operativa y el rendimiento de la planta, sino que también proporcionan una base sólida para futuras investigaciones y aplicaciones prácticas en el campo de la optimización de la producción industrial.

CAPÍTULO 4: RESULTADOS

Se muestran los resultados obtenidos mediante un caso de estudio concreto. De esta forma, se evalúan los algoritmos implementados y se obtienen mejoras en el rendimiento del proceso de producción de la planta.

4.1. Datos generados

Para generar los datos se establecen 3 órdenes de producción (`num_ordenes = 3`) y una holgura de 3 horas (`holgura= 3`). Estas suposiciones permiten estructurar el proceso de producción de manera simple, facilitando la comprensión inicial de los algoritmos de optimización.

4.2. Adaptación de los algoritmos al código en Google Colab

A la hora de implementar los algoritmos en Python, en primer lugar, se decide realizar el desarrollo en un cuaderno en Google Colab. Se crean las clases `Recurso`, `Operacion` y `OrdenFabricacion`, con el objetivo de modelar los elementos fundamentales en la gestión de la planta de fabricación. Cada una de estas representa un componente específico del proceso de producción, facilitando la organización, planificación y ejecución de las operaciones en la planta. La clase `Recurso` representa los recursos físicos o humanos disponibles en la planta que se utilizan para realizar operaciones específicas, la clase `Operacion` modela las distintas tareas o actividades que deben realizarse en la fabricación de un motor y la clase `OrdenFabricacion` representa un pedido específico o una tarea de fabricación en la planta, conteniendo todos los detalles necesarios para su ejecución.

Acto seguido se crea la función `evaluar_fitness`, que es la encargada de determinar el algoritmo óptimo, pues calcula la eficiencia de las asignaciones bajo las condiciones dadas.

```

# Función evaluar_fitness: evaluar la bondad de las asignaciones
def evaluar_fitness(resultado_optimo, operaciones, recursos):
    # Inicializamos las variables para el tiempo total y el tiempo muerto
    tiempo_total = 0
    tiempo_muerto = 0

    # Creamos un diccionario el uso de cada máquina (recurso)
    uso_maquinas = {recurso: 0 for recurso in recursos['ID Recurso'].unique()}

    # Iteramos sobre cada tarea en el resultado_optimo
    for tarea in resultado_optimo:
        id_orden, id_operacion, id_recurso = tarea

        # Verificamos si el id_recurso está en el diccionario uso_maquinas (para evitar acceder a un recurso
        # inexistente)
        if id_recurso in uso_maquinas:
            # Obtenemos la operación correspondiente a la tarea actual
            operacion = operaciones[(operaciones['ID Orden'] == id_orden) &
                                      (operaciones['ID Operación'] == id_operacion)].iloc[0]
            duracion = int(operacion['Duración'])
            tiempo_setup = int(operacion['Tiempo Setup'])

            # Calculamos el tiempo de inicio de la operación
            tiempo_inicio = uso_maquinas[id_recurso]
            # Calculamos la duración total (duracion + tiempo_setup)
            uso_maquinas[id_recurso] += duracion + tiempo_setup
            tiempo_fin = uso_maquinas[id_recurso]

            # Actualizamos el tiempo total y el tiempo muerto (suma de los tiempos de setup)
            tiempo_total = max(tiempo_total, tiempo_fin)
            tiempo_muerto += tiempo_setup
        else:
            print(f"El recurso {id_recurso} no está en el diccionario.")

    # Calculamos el fitness como el inverso de la suma del tiempo total y el tiempo muerto
    fitness = 1 / (tiempo_total + tiempo_muerto) if tiempo_total + tiempo_muerto > 0 else 0

    # Devolvemos el fitness (eficacia de las asignaciones)
    return fitness

```

Figura 16. Función evaluar_fitness.

Esta calcula el *fitness* basado en los tiempos totales asociados a cada operación, que incluyen el tiempo necesario para completar todas las operaciones asignadas (duración) y el tiempo muerto (suma de *setup*) asociado. Un valor de fitness más alto indica una asignación más eficiente. La fórmula empleada: “*fitness = 1 / (tiempo_total + tiempo_muerto)*” penaliza las asignaciones que tienen tiempos totales elevados, incentivando la minimización tanto del tiempo de procesamiento como del tiempo muerto. Esta aproximación es común en problemas de optimización de programación debido a su simplicidad y efectividad, como se menciona en estudios previos sobre optimización de programación flexible con tiempos de preparación dependientes de la secuencia (Song & Liu, 2022).

Tras establecer la forma de evaluación de los algoritmos, se procede a implementarlos.

4.2.1. Algoritmo de Munkres

El algoritmo de Munkres en Python puede implementarse utilizando las librerías SciPy, OR-Tools o con el módulo Munkres que proporciona una implementación del algoritmo. Se ha decidido implementar el módulo Munkres por tener una mayor simplicidad y especialización en este algoritmo en concreto.

Para implementar este algoritmo se definen dos funciones.

La primera función, *generar_matriz_costos*, crea una matriz cuadrada donde cada valor representa la duración de asignar una operación a un recurso.

En caso de que los tipos de recursos y las operaciones coincidan, se asigna la duración asociada. En caso contrario se asigna el valor 10000, un tiempo elevado que nunca se va a dar y que sirve para indicar una asignación inviable.

Cabe destacar que esta matriz también se utiliza en el algoritmo de enrutamiento de máquinas y para ambos se obtiene con la finalidad de optimizar la asignación de operaciones a recursos.

```
# Función para generar la matriz de costos (tiempos)
def generar_matriz_costos(operaciones, recursos):

    # Obtenemos el número de operaciones y recursos
    num_operaciones = operaciones.shape[0]
    num_recursos = recursos.shape[0]

    # Determinamos la dimensión máxima entre operaciones y recursos (para que la matriz después sea cuadrada)
    max_dim = max(num_operaciones, num_recursos)

    # Inicializamos la matriz de costos cuadrada rellena con un valor muy alto (10000)
    matriz_costos = np.full((max_dim, max_dim), 10000)

    # Iteramos sobre cada operación y cada recurso
    for i, op in operaciones.iterrows():
        for j, recurso in recursos.iterrows():
            if op['Tipo Recurso'] == recurso['Tipo']:
                # Si el tipo de recurso de la operación coincide con el tipo de recurso disponible asignamos
                # la duración de la operación como costo
                matriz_costos[i, j] = op['Duración']

    return matriz_costos
```

Figura 17. Función *generar_matriz_costos*.

A continuación, se crea la función *algoritmo_munkres*, que incluye en su definición la generación de la matriz de costos para obtener los tríos de órdenes, operaciones y recursos óptimos. Se asignan las operaciones a los recursos filtrando por las asignaciones, asegurándose de que se respeten las restricciones de cantidad de operaciones por orden.

Esta función finalmente evalúa el fitness de la solución generada y devuelve las asignaciones óptimas junto con su valor de *fitness*.

```
# Función para optimizar con el algoritmo de Munkres
def algoritmo_munkres(matriz_costos, operaciones, recursos):
    # Creamos una instancia de la clase Munkres para aplicar el algoritmo
    m = Munkres()
    # Ejecutamos el algoritmo sobre la matriz de costos y obtenemos los índices
    indices = m.compute(matriz_costos)

    # Inicializamos una lista vacía para almacenar las asignaciones filtradas
    asignaciones_filtradas = []

    # Tenemos 6 operaciones por cada orden
    num_op_por_orden = 6

    # Obtenemos el número total de órdenes en el DataFrame de operaciones
    num_ordenes = operaciones['ID Orden'].nunique()

    # Iteramos sobre cada orden
    for id_orden in range(1, num_ordenes + 1):
        # Inicializamos un contador para llevar controlar las operaciones por orden
        contador_op = 0

        # Iteramos sobre los índices resultantes del algoritmo de Munkres (pares de id_operacion y id_recurso)
        for id_operacion, id_recurso in indices:
            # Verificamos si la operación y el recurso están dentro del rango de operaciones y recursos válidos
            if id_operacion < operaciones.shape[0] and id_recurso < recursos.shape[0]:
                # Filtramos las operaciones correspondientes a la orden actual
                operacion_actual = operaciones[operaciones['ID Orden'] == id_orden]

                # Verificamos si no hemos alcanzado el número máximo de tuplas por orden
                if contador_op < num_op_por_orden and len(operacion_actual) > 0:
                    # Obtenemos la operación correspondiente a través de la posición del contador
                    operacion = operacion_actual.iloc[contador_op]
                    id_operacion_real = operacion['ID Operación']

                    # Añadimos la asignación a la lista como una tupla con el ID de la orden,
                    # la operación y el recurso
                    asignaciones_filtradas.append((id_orden, id_operacion_real, id_recurso))

                    # Aumentamos el contador de operaciones para la orden actual
                    contador_op += 1

    # Evaluamos el fitness de las asignaciones obtenidas
    fitness_munkres = evaluar_fitness(asignaciones_filtradas, operaciones, recursos)

    # Devolvemos la lista de asignaciones y el valor de fitness
    return asignaciones_filtradas, fitness_munkres
```

Figura 18. Función *algoritmo_munkres*.

4.2.2. Algoritmo genético

Para implementar el algoritmo genético se crean cuatro funciones independientes y una quinta que integra a las otras cuatro. Las cuatro primeras las podemos observar a continuación en la Figura 19, Figura 20, Figura 21 y Figura 22.

La primera de ellas, *crear_cromosoma*, genera un cromosoma aleatorio (lista de asignaciones de recursos a operaciones), representado por tuplas que contienen el ID de

la orden, el ID de la operación y el ID del recurso asignado. Esta función itera sobre todas las operaciones y asigna un recurso aleatorio a cada una.

```
# Función para crear un cromosoma aleatorio
def crear_cromosoma(operaciones, recursos):
    # Inicializamos una lista vacía para almacenar el cromosoma
    cromosoma = []

    # Iteramos sobre cada operación en operaciones
    for _, operacion in operaciones.iterrows():
        # Obtenemos el ID de la orden y el ID de la operación
        id_orden = operacion['ID Orden']
        id_operacion = operacion['ID Operación']

        # Seleccionamos aleatoriamente un ID de recurso de la lista de recursos disponibles
        id_recurso = random.choice(recursos['ID Recurso'].values)

        # Añadimos una tupla (ID de la orden, ID de la operación, ID del recurso) al cromosoma
        cromosoma.append((id_orden, id_operacion, id_recurso))

    # Devolvemos el cromosoma generado
    return cromosoma
```

Figura 19. Función *crear_cromosoma*.

La función *seleccionar_padres* utiliza el método de la selección por torneo para elegir los progenitores, el mejor (mayor *fitness*) entre varios aspirantes aleatorios como padre. Se repite este proceso hasta obtener una lista de padres seleccionados del mismo tamaño que la población original.

```
# Función de selección de padres mediante el método de selección torneo
def seleccionar_padres(poblacion, fitness_poblacion, k=3):
    # Inicializamos una lista vacía para almacenar los padres seleccionados
    seleccionados = []

    # Repetimos el proceso de selección hasta obtener el mismo número de padres que la población original
    for _ in range(len(poblacion)):
        # Seleccionamos k aspirantes al azar de la población junto con sus valores de fitness
        aspirantes = random.sample(list(zip(poblacion, fitness_poblacion)), k)

        # De los aspirantes seleccionados, elegimos el que tenga el mayor valor de fitness
        mejor_aspirante = max(aspirantes, key=lambda x: x[1])[0]

        # Añadimos el mejor aspirante (el cromosoma) a la lista de seleccionados
        seleccionados.append(mejor_aspirante)

    # Devolvemos la lista de padres seleccionados
    return seleccionados
```

Figura 20. Función *seleccionar_padres*.

La función *cruzar* combina la información genética de dos cromosomas (padre1 y padre2) para generar otros nuevos (hijo1 e hijo2). Se realiza para combinar la información genética de los padres y así crear descendencia que tenga características de ambos.

```

# Función de cruce para generar dos hijos a partir de dos padres
def cruzar(padre1, padre2):
    # Seleccionamos un punto de cruce al azar (evitando los extremos)
    punto = random.randint(1, len(padre1) - 2)

    # Generamos el primer hijo combinando la primera parte del padre1 con la segunda parte del padre2
    hijo1 = padre1[:punto] + padre2[punto:]

    # Generamos el segundo hijo combinando la primera parte del padre2 con la segunda parte del padre1
    hijo2 = padre2[:punto] + padre1[punto:]

    # Devolvemos los dos hijos generados
    return hijo1, hijo2

```

Figura 21. Función cruzar.

Por último, la función *mutar* introduce mutaciones basadas en una tasa de mutación de 0,1. Se establece una tasa del 10% para evitar el estancamiento en óptimos locales y mantener la diversidad genética en la población sin modificar excesivamente las soluciones buenas ya encontradas.

```

# Función de mutación para alterar un cromosoma con una cierta probabilidad
def mutar(cromosoma, recursos, tasa_mutacion=0.1):
    # Iteramos sobre cada gen (tarea) en el cromosoma
    for i in range(len(cromosoma)):
        # Generamos un número aleatorio y comparamos con la tasa de mutación
        if random.random() < tasa_mutacion:
            # Si se cumple la condición de mutación, obtenemos los IDs de orden y operación
            id_orden, id_operacion, _ = cromosoma[i]
            # Seleccionamos aleatoriamente un nuevo ID de recurso de la lista de recursos disponibles
            id_recurso = random.choice(recursos['ID Recurso'].values)
            # Actualizamos el cromosoma con el nuevo recurso asignado
            cromosoma[i] = (id_orden, id_operacion, id_recurso)

    # Devolvemos el cromosoma posiblemente mutado
    return cromosoma

```

Figura 22. Función mutar.

Con estas funciones definidas, se crea la función *algoritmo_genetico* que busca la mejor asignación de recursos a operaciones para minimizar el tiempo total de operación utilizando la función de *fitness*. Este proceso incluye la inicialización de una población de soluciones, la evaluación de su calidad, y la evolución de esta población a través de selección, cruce y mutación.

Como resultado, esta función también devuelve la asignación óptima y su valor de *fitness*.

```

# Algoritmo Genético
def algoritmo_genetico(operaciones, recursos, tam_poblacion=100, num_generaciones=50, tasa_cruce=0.7,
                       tasa_mutacion=0.1):
    # Inicializar población
    # Creamos una población inicial de cromosomas aleatorios
    poblacion = [crear_cromosoma(operaciones, recursos) for _ in range(tam_poblacion)]

    # Iteramos a través de un número fijo de generaciones
    for generacion in range(num_generaciones):
        # Evaluar población
        # Calculamos el fitness de cada cromosoma en la población
        fitness_poblacion = [evaluar_fitness(cromosoma, operaciones, recursos) for cromosoma in poblacion]

        # Selección
        # Seleccionamos los padres utilizando el método de selección por torneo
        padres = seleccionar_padres(poblacion, fitness_poblacion)

        # Cruce
        # Generamos una nueva población a través del cruce de los padres seleccionados
        nueva_poblacion = []
        while len(nueva_poblacion) < tam_poblacion:
            if random.random() < tasa_cruce:
                # Seleccionamos dos padres al azar y aplicamos el cruce
                padre1 = random.choice(padres)
                padre2 = random.choice(padres)
                hijo1, hijo2 = cruzar(padre1, padre2)
                nueva_poblacion.append(hijo1)
                nueva_poblacion.append(hijo2)
            else:
                # Si no se realiza cruce, seleccionamos un parente aleatorio para la nueva población
                nueva_poblacion.append(random.choice(padres))

        # Mutación
        # Aplicamos mutaciones a la nueva población con una cierta probabilidad
        nueva_poblacion = [mutar(cromosoma, recursos, tasa_mutacion) for cromosoma in nueva_poblacion]

        # Sustitución
        # La nueva población reemplaza a la antigua
        poblacion = nueva_poblacion

    # Seleccionar mejor solución
    # Evaluamos el fitness de la población final para encontrar la mejor solución
    fitness_poblacion = [evaluar_fitness(cromosoma, operaciones, recursos) for cromosoma in poblacion]
    mejor_solucion = poblacion[np.argmax(fitness_poblacion)]
    mejor_fitness = max(fitness_poblacion)

    # Devolvemos la mejor solución encontrada y su valor de fitness
    return mejor_solucion, mejor_fitness

```

Figura 23. Función *algoritmo_genetico*.

4.2.3. Algoritmo de enrutamiento de máquinas

El algoritmo de enrutamiento de máquinas tiene como objetivo asignar de manera óptima las tareas a los recursos disponibles, minimizando los costos asociados al proceso de producción. Se crea la función *algoritmo_enrutamiento* que recibe tres entradas: la matriz de costos definida en el apartado 4.2.1., los datos de operaciones y los de recursos.

Esta función itera a través de las filas de la matriz de costos (cada fila representa una operación) y, para cada operación, selecciona el recurso con el costo más bajo usando la función *np.argmin*. Da como resultado una lista de tuplas, que igual que en el algoritmo de

Munkres, se filtran para asegurar que solo se asignan seis operaciones por orden. Finalmente, proporciona las asignaciones óptimas y evalúa el fitness de estas.

4.2.4. Algoritmo de cuellos de botella.

El proceso de implementación del algoritmo de cuellos de botella comienza con la definición de la función *identificar_cuellos_de_botella*, que recibe como entradas un conjunto de operaciones y los recursos disponibles.

La función primero inicializa un diccionario que asigna a cada recurso un tiempo de procesamiento igual a cero. Después, itera sobre cada operación en el conjunto de datos, sumando la duración de cada operación al tiempo total del recurso correspondiente. Así se acumulan los tiempos de procesamiento para cada recurso.

```
# Función para identificar los cuellos de botella
def identificar_cuellos_de_botella(operaciones, recursos):
    # Inicializamos un diccionario para almacenar el tiempo total de procesamiento para cada recurso
    tiempos_recurso = {recurso: 0 for recurso in recursos['ID Recurso']}

    # Iteramos sobre cada operación en el DataFrame de operaciones
    for _, operacion in operaciones.iterrows():
        # Obtenemos el recurso asignado y la duración de la operación
        recurso = operacion['ID Recurso']
        duracion = operacion['Duración']
        # Incrementamos el tiempo total de procesamiento del recurso con la duración de la operación actual
        tiempos_recurso[recurso] += duracion

    # Ordenamos los recursos por su tiempo total de procesamiento de mayor a menor
    cuellos_de_botella = sorted(tiempos_recurso.items(), key=lambda x: x[1], reverse=True)
    # Devolvemos la lista de recursos ordenados con sus tiempos totales de procesamiento
    return cuellos_de_botella
```

Figura 24. Función *identificar_cuellos_de_botella*.

Una vez acumulados los tiempos, los recursos se ordenan de mayor a menor según el tiempo total de procesamiento. Esta lista ordenada de recursos se almacena en la variable *cuellos_de_botella*.

Se implementa la función *visualizar_cuellos_de_botella*, para la generación de un gráfico de barras donde se visualizan los recursos junto con sus respectivos tiempos acumulados.

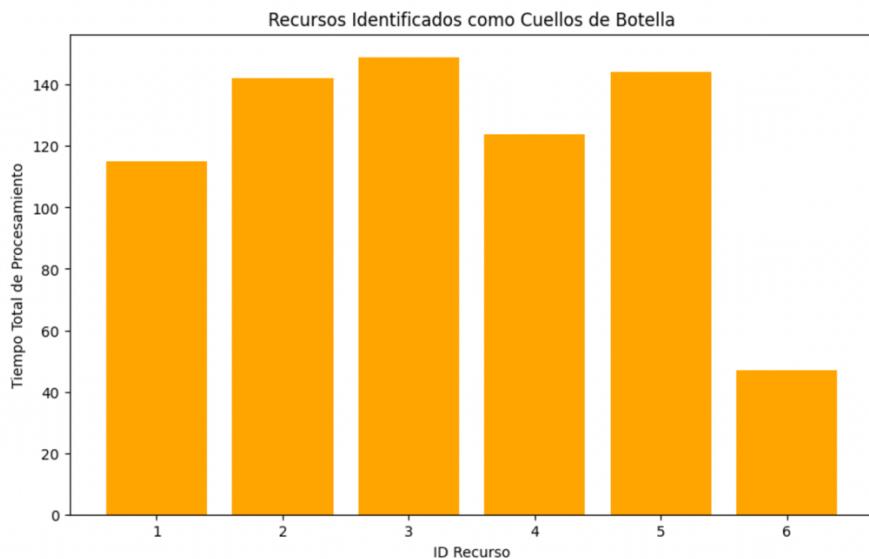


Figura 25. Gráfico de barras identificación cuellos de botella.

4.3. Interfaz de usuario

Se decide migrar el proyecto a una interfaz de usuario en un entorno local para que Navantia tenga a su disposición una herramienta que proporcionar a sus empleados encargados del proceso de fabricación. Tras considerar la implementación en dos opciones: Gradio o Streamlit, se llega a la conclusión de que Streamlit es más adecuado, ya que permite crear interfaces interactivas más complejas para modelos de Machine Learning y algoritmos.

En un primer intento, la finalidad de la aplicación es permitir a los usuarios subir un archivo Excel (estructura especificada en el CAPÍTULO 3: MATERIALES Y MÉTODOS), procesarlos y aplicar los algoritmos para encontrar la mejor asignación de tareas a recursos, mostrando los resultados en un diagrama de Gantt interactivo.

Sin embargo, al no trabajar con datos reales sino generador y tras su implementación se observa que esta no es adecuada ya que el tiempo de carga del archivo Excel es elevado. De cara a la implementación real esto supone un tiempo de carga muy elevado, lo que la convierte en una web poco funcional.

Optimización de la Producción de Motores - TFM de Manuel e Inés

Sube tu archivo Excel aquí



Drag and drop file here

Limit 200MB per file • XLSX

Browse files

Figura 26. Interfaz de usuario inicial de la aplicación.

Tras este intento fallido, se decide modificar la forma de introducción de los datos. En lugar de generar un archivo Excel con un escenario simulado e introducirlo en la web, se opta por solicitar dos inputs al usuario: el número de órdenes a generar (*num_ordenes*) y la holgura en horas (*holgura*). Luego, los datos se generan automáticamente mediante código con esta información.

Optimización de la Programación de Producción en una Fábrica de Motores mediante Algoritmos de Optimización.

Manuel Suárez Calle e Inés Hernández Pastor

Ingrese el número de órdenes a generar:

1

- +

Ingrese la holgura en horas:

0

- +

Generar datos y visualizar Gantt

Figura 27. Interfaz de usuario final de la aplicación.

En esta interfaz, una vez se introducen los inputs, se tiene que pinchar en el botón 'Generar datos y visualizar Gantt' para generar los datos, verlos en pantalla y visualizar el diagrama de Gantt.

A continuación, se realiza un ejemplo del funcionamiento. En la Figura 28, se puede ver la salida en pantalla poniendo como inputs número de órdenes = 3 y holgura = 3.

Optimización de la Programación de Producción en una Fábrica de Motores mediante Algoritmos de Optimización.

Manuel Suárez Calle e Inés Hernández Pastor

Ingrese el número de órdenes a generar:

3

- +

Ingrese la holgura en horas:

3

- +

Generar datos y visualizar Gantt

Figura 28. Ejemplo de interfaz con 3 órdenes generadas y 3 horas de holgura.

Tal y como se observa en la Figura 29 se generan los datos y se muestran en pantalla.

Órdenes

	ID Orden	Material	Cantidad	Fecha Inicio Temprana	Fecha Fin Temprana	Fecha Inicio Tardía
0	1	Motor MTU 8000	1	2024-08-24 08:00:00	2024-09-03 18:00:00	2024-08-24 11:00:00
1	2	Motor MTU 2000	1	2024-08-25 12:00:00	2024-08-31 09:00:00	2024-08-25 15:00:00
2	3	Motor MTU 2000	1	2024-08-29 09:00:00	2024-09-10 16:00:00	2024-08-29 12:00:00

Operaciones

	ID Orden	ID Operación	Descripción Operación	Tiempo Setup	Duración	ID Recurso	Descripción Re
0	1	0	Mecanizado base	0	34	1	Torno CNC
1	1	1	Mecanizado bloque	0	21	2	Fresadora CNC
2	1	2	Montaje pistones	0	65	3	Montaje
3	1	3	Montaje cigüeñal	0	25	4	Pintura
4	1	4	Ensamblaje motor	0	43	5	Robot de Ensamblaje
5	1	5	Pruebas finales	0	59	6	Estación de Pruebas
6	2	0	Mecanizado base	0	11	1	Torno CNC
7	2	1	Mecanizado bloque	0	29	2	Fresadora CNC
8	2	2	Montaje pistones	0	9	3	Montaje
9	2	3	Montaje cigüeñal	0	32	4	Pintura

Recursos

	ID Recurso	Descripción	Tipo	Capacidad Diaria
0	1	Torno CNC	Mecanizado	1
1	2	Fresadora CNC	Mecanizado	1
2	3	Montaje	Montaje	1
3	4	Pintura	Acabado	1
4	5	Robot de Ensamblaje	Montaje Avanzado	1
5	6	Estación de Pruebas	Pruebas	1

Figura 29. Salida en pantalla de los datos generados.

Posteriormente, se genera el diagrama de Gantt con la secuencia temporal de las operaciones para cada orden, además de ofrecer la posibilidad de optimizar este proceso.

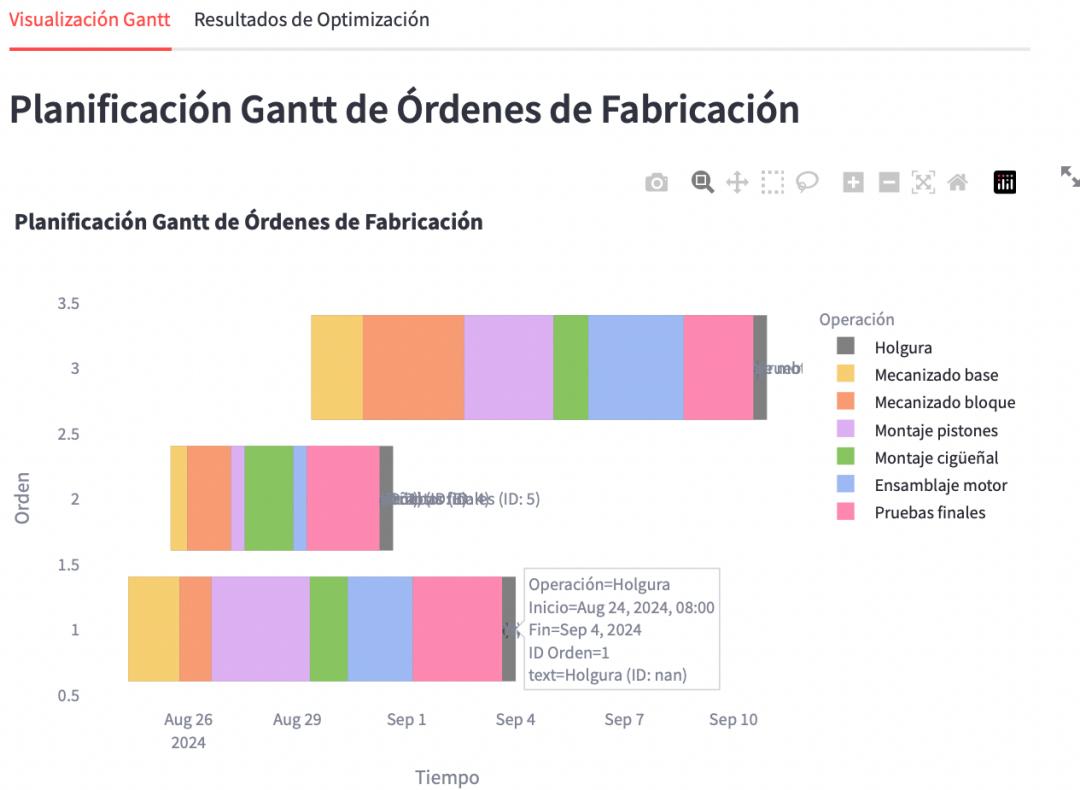


Figura 30. Visualización Gantt de las órdenes de fabricación generadas.

Si el usuario opta por realizar la optimización, deberá seleccionar la ventana ‘Resultados de Optimización’, donde, se presentará el algoritmo óptimo, es decir, el del mayor valor de *fitness* y su correspondiente diagrama de Gantt de las operaciones para cada recurso.

4.4. Análisis de los resultados obtenidos

En el ejemplo que hemos mencionado anteriormente, se han obtenido los resultados contenidos en la Tabla 4. Esta tabla presenta los valores de la métrica *fitness* obtenida por tres de los algoritmos de estudio: el algoritmo de Munkres (Húngaro), el algoritmo genético y el algoritmo de enrutamiento de máquinas. Se recuerda que, cuanto mayor es el valor de la métrica de *fitness*, mejor es el desempeño del algoritmo en términos de minimización del tiempo total de fabricación.

Algoritmo	Valor de fitness
Munkres (Húngaro)	0.008065
Genético	0.007634
Enrutamiento de máquinas	0.008064

Tabla 4. Valor de fitness de los algoritmos.

Algoritmo	Asignaciones óptimas (orden, operación, recurso)
Munkres (Húngaro)	$[(1, 0, 0), (1, 1, 5), (1, 2, 3), (1, 3, 1), (1, 4, 2), (1, 5, 4), (2, 0, 0), (2, 1, 5), (2, 2, 3), (2, 3, 1), (2, 4, 2), (2, 5, 4), (3, 0, 0), (3, 1, 5), (3, 2, 3), (3, 3, 1), (3, 4, 2), (3, 5, 4)]$
Genético	$[(1, 0, 1), (1, 1, 3), (1, 2, 3), (1, 3, 2), (1, 4, 2), (1, 5, 5), (2, 0, 6), (2, 1, 5), (2, 2, 4), (2, 3, 2), (2, 4, 1), (2, 5, 6), (3, 0, 4), (3, 1, 4), (3, 2, 2), (3, 3, 3), (3, 4, 2), (3, 5, 5)]$
Enrutamiento de máquinas	$[(1, 0, 0), (1, 1, 4), (1, 2, 5), (1, 3, 3), (1, 4, 0), (1, 5, 2), (2, 0, 0), (2, 1, 4), (2, 2, 5), (2, 3, 3), (2, 4, 0), (2, 5, 2), (3, 0, 0), (3, 1, 4), (3, 2, 5), (3, 3, 3), (3, 4, 0), (3, 5, 2)]$

Tabla 5. Asignaciones óptimas para cada algoritmo.

En este caso particular, el algoritmo de Munkres es el que ha logrado el mejor resultado, seguido de cerca por el enrutamiento de máquinas. Como podemos observar en la Tabla 4 ambos algoritmos presentan valores casi idénticos. Se podría decir que, en términos de eficiencia, ambos métodos ofrecen soluciones de optimización equivalentes dentro del marco de este estudio. Sin embargo, a pesar de la proximidad en los valores de fitness, en la Tabla 5 podemos observar que las asignaciones óptimas de cada algoritmo varían. La similitud en los valores de *fitness* obtenidos refleja el hecho de que ambos algoritmos utilizan una matriz de costos común para guiar sus decisiones de asignación, aunque sus métodos y prioridades difieren.

Por otro lado, se observa que el algoritmo genético presenta un desempeño peor que los otros dos algoritmos. No obstante, es importante destacar que el algoritmo genético está diseñado para explorar un espacio de soluciones mucho más amplio. Esto significa que, en situaciones más complejas, probablemente proporcione mejores resultados que los otros, además de soluciones más robustas o adaptativas, especialmente en escenarios con restricciones más complejas.

La implementación del algoritmo de cuellos de botella no ha sido completada, lo cual representa una oportunidad para el desarrollo de futuras investigaciones. Este algoritmo es

clave en la detección de recursos o etapas del proceso donde se limita la capacidad total del sistema, y su implementación podría revelar puntos críticos que no son detectados por los otros algoritmos. Identificar y abordar estos cuellos de botella puede ofrecer una optimización adicional, mejorando la eficiencia del proceso productivo en la fábrica de motores de Navantia.

CAPÍTULO 5: CONCLUSIONES

El presente trabajo ha abordado de manera completa la **optimización del proceso de fabricación de motores** mediante la implementación y comparación de los cuatro algoritmos seleccionados: **el algoritmo de Munkres (Húngaro), el genético, el de enrutamiento de máquinas y el de cuellos de botella.**

El proyecto se ha desarrollado desde cero, partiendo sin conocimientos previos en la industria naval. Esto ha requerido una inmersión inicial en la materia, seguida de la implementación del código, culminando en la creación de una interfaz de usuario funcional que puede considerarse como un producto mínimo viable.

Durante el proceso de generación de datos a través de la **función generar_datos_sintéticos**, se ha simulado un entorno de producción que ha permitido modelar un escenario básico de fabricación de motores. Esta simulación ha sido esencial para probar y validar los algoritmos en condiciones controladas, permitiendo un análisis inicial de su comportamiento y rendimiento. Además, se ha garantizado que los datos generados son adecuados para realizar comparaciones entre los diferentes enfoques implementados.

Una parte clave de nuestro trabajo ha sido la **integración y comparación de diferentes enfoques algorítmicos** en la fabricación de motores. Esta comparación no solo nos ha permitido evaluar la efectividad de cada uno, sino también subrayar la importancia de elegir el método adecuado según las necesidades específicas del proceso productivo. Cada algoritmo tiene su punto fuerte en distintos contextos, destacando según las restricciones y características del proceso. Por lo tanto, el conocimiento y la capacidad de seleccionar el algoritmo más adecuado se convierte en un factor esencial en el proceso de mejora de la producción y así poder adaptar las soluciones a cada escenario particular.

Finalmente, la implementación de la **interfaz de usuario interactiva** ha permitido que los resultados obtenidos sean comprendidos tanto por usuarios con conocimientos técnicos como por aquellos que no los poseen, facilitando la visualización y comparación de los resultados en tiempo real. Esta interfaz puede ser utilizada como base para futuros desarrollos o mejoras en entornos productivos reales. Además, la interfaz puede llegar a

integrarse en sistemas de planificación de la producción, ayudando a optimizar los procesos de Navantia de fabricación de motores de manera más eficiente y efectiva.

El trabajo realizado sienta una base sólida para futuras investigaciones y mejoras, tanto en la ampliación de escenarios más complejos como en la implementación de nuevas técnicas de optimización.

CAPÍTULO 6: TRABAJO FUTURO

Este capítulo presenta las posibles líneas de trabajo futuro derivadas del presente estudio, con el objetivo de seguir mejorando los resultados obtenidos y abordar nuevas problemáticas y oportunidades identificadas a lo largo del proyecto. A continuación, se detallan las principales líneas de trabajo futuras que se consideran relevantes para seguir avanzando en este campo:

- Implementar el **algoritmo de cuellos de botella**, el cual no ha podido ser abordado en la fase actual del proyecto. Esta implementación permitirá obtener su valor de fitness y realizar una comparación con los demás algoritmos, proporcionando una evaluación más completa de su rendimiento en diferentes escenarios de fabricación.
- Simular **dos escenarios distintos** de la producción en línea de la fábrica de motores bajo las condiciones especificadas por Navantia. Una vez generado el primer escenario, se deben generar un segundo escenario en el que se introduce un tiempo de setup aleatorio entre 0 y 10 horas y un escenario en el que se introduce un tiempo de setup y una capacidad diaria con valores comprendidos entre 0 y 10 horas.
- **Optimización de la eficiencia energética de las máquinas:** El consumo energético es un factor crucial en la producción industrial, pues supone un alto gasto. Se propone desarrollar un sistema para monitorizar y optimizar la eficiencia energética de las máquinas utilizadas en la fabricación de motores. Este sistema podría utilizar técnicas de inteligencia artificial para ajustar automáticamente los parámetros de operación y minimizar el consumo energético sin perjudicar la productividad.
- **Monitorización en tiempo real del estado de las máquinas:** Implementar dispositivos en toda la planta de producción para poder monitorizar en tiempo real el estado de las máquinas y los procesos de producción. Esto permitirá programar mantenimientos preventivos, reduciendo los tiempos de inactividad y costos asociados a reparaciones muy costosas y prolongadas inesperadas.

- **Detección de fallos en la producción:** Implementar algoritmos de Machine Learning para analizar los datos recopilados de las máquinas e identificar patrones y anomalías en el funcionamiento de las máquinas, anticipando posibles fallos en los motores fabricados. Predecir estos problemas con antelación, permitirá tomar medidas preventivas para evitar fallos críticos, asegurando la calidad del producto final y previniendo posibles catástrofes. Este enfoque además de mejorar la seguridad y la fiabilidad de los motores optimiza el tiempo de producción y reduce los costos asociados a reparaciones y reemplazos inesperados.
- **Realidad virtual para mejorar las condiciones del puesto de trabajo:** Simular diferentes configuraciones de la planta y las condiciones laborales en un entorno virtual. Esto permitirá, entre otras cosas, ajustar las alturas de las mesas, la disposición de las herramientas y la ubicación de los equipos para reducir la fatiga y el riesgo de lesiones entre los trabajadores. Puede derivar en una mejora de la salud y el bienestar del personal y reducir el tiempo de inactividad por lesiones.

Mediante las líneas de trabajo futuras se pretende seguir en el camino de la mejora de la eficiencia y la productividad de la planta de fabricación de Navantia, adicionalmente permite avanzar hacia prácticas más sostenibles y responsables en el ámbito industrial.

CAPÍTULO 7: GLOSARIO

En este capítulo se definen y aclaran los términos y conceptos clave utilizados a lo largo de este Trabajo de Fin de Máster. Dada la naturaleza técnica y especializada del estudio, es esencial proporcionar un glosario que facilite la comprensión de los lectores sobre las tecnologías, métodos y herramientas empleados en la optimización de la producción en la fábrica de motores. Puede servir como una referencia rápida para los términos técnicos. A continuación, se presentan las definiciones de los términos más relevantes:

- **ERP (*Enterprise Resource Planning*)**: Sistema de software centralizado que integra y gestiona todos los procesos y datos de una empresa, facilitando la toma de decisiones y mejorando la eficiencia operativa.
- **Lista de materiales (BOM)**. Inventario que incluye una descripción detallada de los materiales y componentes necesarios para la fabricación de un producto, mostrando los subensamblajes, partes y materias primas. Además, especifica la cantidad de cada componente necesaria para realizar el ensamblaje final (Wang, Liu, Bai, & Xiao, 2023).
- **Hoja de ruta**. Plan estratégico que detalla los pasos y actividades a seguir que son necesarios para alcanzar los objetivos de un proyecto. Sirve como una guía para la ejecución, estableciendo metas claras, métodos de implementación e hitos críticos. Proporciona una estructura coherente y global que permite explorar, mapear y comunicar el desarrollo y la evolución de un negocio o sistema y sus componentes (Gholamzadeh Chofreh et al., 2014).
- **Interfaz de Usuario**: Plataforma a través de la cual los usuarios interactúan con un sistema o aplicación. Diseñada para ser intuitiva y facilitar la visualización y manipulación de datos.
- **Machine Learning**: Rama de la inteligencia artificial que se enfoca en el desarrollo de algoritmos que permiten a las máquinas aprender y tomar decisiones basadas en datos.
- **Orden de fabricación**. Documento que se emplea para autorizar y supervisar la producción de un producto. Este documento detalla las características del producto final, la cantidad a producir, los materiales necesarios, el plazo de entrega y cualquier otra instrucción relevante para el proceso de fabricación (Serrano-Ruiz, J. C., et al., 2021).

- **SAP S/4 HANA:** Sistema ERP avanzado desarrollado por SAP que integra capacidades de procesamiento en tiempo real y análisis de datos, diseñado para optimizar la planificación y ejecución de operaciones empresariales.
- **Python:** Lenguaje de programación característico por tener una sintaxis simple y clara, idóneo para principiantes y para programadores con gran bagaje.
- **Streamlit:** Herramienta de Python de desarrollo de aplicaciones web interactivas y visualizaciones de datos, utilizada para crear interfaces de usuario personalizables y fáciles de usar.

CAPÍTULO 8: BIBLIOGRAFÍA

Akhtar, J. (2021). *Production Planning with SAP S/4HANA*. SAP PRESS.

Britto Agudelo, R. A., Mejía Delgadillo, G., & Caballero Villalobos, J. P. (2007). Programación de la producción en sistemas de manufactura tipo taller con el algoritmo combinado cuello de botella móvil y búsqueda tabú. *Ingeniería y Universidad*, 11(2), 203-224.

Castro-Vázquez, M. M., García-Villaverde, P., & Fernández-Navarro, F. (2021). The effect of public support on SMEs' innovative performance: The mediating role of absorptive capacity. *Technological Forecasting and Social Change*, 163, 120461. <https://doi.org/10.1016/j.techfore.2020.104993>

Davenport, T. H. (1998). Putting the enterprise into the enterprise system. *Harvard Business Review*, 76(4), 121-131.

Gessa, A., Jiménez, A., & Sancha, P. (2023). Exploring ERP systems adoption in challenging times: Insights of SMEs stories. *Technological Forecasting and Social Change*, 195, 122795. <https://doi.org/10.1016/j.techfore.2023.122795>

Gholamzadeh Chofreh, A., Ariani Goni, F., Mohamed Shararoun, A., & Ismail, S. (2014). Review on Enterprise Resource Planning Implementation Roadmap: Project Management Perspective. *Sains Humanika*, 2(2), 135–138.

Jacobs, F. R., & Whybark, D. C. (2000). *Why ERP? A Primer on SAP Implementation*. McGraw-Hill/Irwin.

Mills-Tettey, G. A., Stentz, A., & Dias, M. B. (2007). The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs (CMU-RI-TR-07-27) [Technical report]. Robotics Institute, Carnegie Mellon University.

Raghavendra, S. (2023). Introduction to Streamlit. En *Beginner's Guide to Streamlit with Python*. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-8983-9_1

Rusdiana, S., Oktavia, R., & Charlie, E. (2019). Application of Hungarian Method in Optimizing The Scheduling of Employee Assignment and Profit of Home Industry

Production. *Journal of Research in Mathematics Trends and Technology*, 01(01), 24-33. <https://doi.org/10.32734/jormtt.v1i1.754>

SAP. (2023). SAP S/4 HANA 2023. <https://www.sap.com/products/s4hana-erp.html>

SAP. (s.f.). ¿Qué es SAP? Recuperado de <https://www.sap.com/spain/about/what-is-sap.html>

SAP. (s.f.). ¿Qué es SAP HANA? Recuperado de <https://www.sap.com/spain/products/technology-platform/hana/what-is-sap-hana.html>

SAP. (s.f.). ¿Qué es un ERP? Recuperado de <https://www.sap.com/spain/products/erp/what-is-erp.html>

SAP. (s.f.). SAP S/4 HANA: Advanced Planning and Optimization (APO). Recuperado de <https://www.sap.com/spain/products/scm/advanced-planning-optimization.html>

Serrano-Ruiz, J. C., Mula, J., & Poler, R. (2021). Smart manufacturing scheduling: A literature review. *Journal of Manufacturing Systems*, 61, 265-287. <https://doi.org/10.1016/j.jmsy.2021.09.011>

Smith, J., Doe, A., & Brown, P. (2020). Synthetic Data Generation for Machine Learning: Methods and Applications. *Journal of Artificial Intelligence Research*, 67, 345-375.

Suesca, J., Jiménez, M., & Sancha, F. (2023). Optimization of Manufacturing Cell Systems using Genetic Algorithms. *Journal of Manufacturing Systems*, 45, 123-135. <https://doi.org/10.1016/j.jmsy.2023.01.002>

Wang, S., Liu, X., Bai, Z., & Xiao, J. (2023). A BOM model transformation method for hierarchical production planning management process of complex products. *Advanced Engineering Informatics*, 58, 102138.

Werth, A., Oliver, K., West, C. G., & Lewandowski, H. J. (2022). Engagement in collaboration and teamwork using Google Colaboratory. National Science Foundation. <https://par.nsf.gov/servlets/purl/10454095>

Yen, H. R., & Sheu, C. (2004). Aligning ERP implementation with competitive priorities of manufacturing firms: An exploratory study. *International Journal of Production Economics*, 92(3), 207-220.

ANEXOS

El código del desarrollo del proyecto se encuentra disponible en el siguiente repositorio de GitHub: https://github.com/ihernandezp1/TFM_NAVANTIA