UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

INTERPRETABILIDAD EN LA INDUSTRIA 5.0: ANÁLISIS DE UN PROBLEMA JOB SHOP FLEXIBLE

POR

Miguel Ignacio Oliva Moraga

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

Patricio Antonio Sáez Bustos

Profesor Co-Guía

Sebastián Astroza Tagle

Julio 2025

Concepción (Chile)

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que formaron parte de este proceso universitario.

A mis compañeros y amigos, por acompañarme en este camino y convertirlo en una experiencia profundamente significativa y memorable. Gracias por cada conversación, desafío compartido y momento de alegría.

A mi familia, por brindarme siempre su respaldo incondicional en cada una de las decisiones que he tomado, tanto en lo académico como en lo personal. En especial, a mi madre, quien ha sido el pilar más importante en mi vida. Gracias por entregarme siempre lo mejor de ti, por tu amor incansable y por asegurarte de que nunca me faltara nada. Este logro es fruto de tu esfuerzo, tu ejemplo y tu infinita dedicación. El profesional y la persona que soy, te lo debo a ti.

Sumario

En el contexto emergente de la quinta revolución industrial, conocida como Industria 5.0, la que promueve una colaboración estrecha y sinérgica entre humanos y sistemas inteligentes, se vuelve cada vez más crucial la incorporación de modelos de inteligencia artificial que no solo sean eficientes, sino también comprensibles y confiables. El presente estudio aborda dicha necesidad mediante la aplicación de un modelo híbrido al problema de programación flexible de talleres de trabajo (FJSSP), una de las problemáticas más desafiantes en la planificación de la producción industrial debido a su complejidad. Para ello, se desarrolla un modelo híbrido que combina una red neuronal artificial, entrenada a partir de datos generados por un modelo de programación lineal entera mixta (MILP) con un algoritmo genético orientados a predecir y optimizar la secuenciación de operaciones. Además, se incorpora la metodología SHapley Additive exPlanations (SHAP) como herramienta de interpretación para analizar la influencia de cada variable de entrada en las decisiones y comportamiento del modelo. Los resultados obtenidos muestran que el modelo hibrido propuesto es capaz de entregar soluciones de buena calidad en tiempos de cómputo considerablemente menores que los métodos exactos, especialmente en instancias complejas. Asimismo, el análisis de interpretabilidad revela que el modelo es capaz de aprender patrones coherentes con criterios operacionales, lo que refuerza su validez como sistema de apoyo a la toma de decisiones. En conjunto, este trabajo constituye un aporte metodológico al desarrollo de soluciones inteligentes alineadas con los principios de la Industria 5.0, favoreciendo la transparencia, la confianza y la integración efectiva entre personas y tecnologías avanzadas.

Summary

In the emerging context of the fifth industrial revolution, known as Industry 5.0, which promotes a close and synergistic collaboration between humans and intelligent systems, the incorporation of artificial intelligence models that are not only efficient but also understandable and trustworthy becomes increasingly crucial. This study addresses that need through the application of a hybrid model to the Flexible Job Shop Scheduling Problem (FJSSP), one of the most challenging issues in industrial production planning due to its inherent complexity. A hybrid approach is developed by combining an artificial neural network, trained on data generated by a Mixed Integer Linear Programming (MILP) model, with a genetic algorithm aimed at predicting machine assignments and optimizing the sequencing of operations. Additionally, the SHapley Additive exPlanations (SHAP) methodology is incorporated as an interpretability tool to analyze the influence of each input variable on the model's decisions and behavior. The results show that the proposed hybrid model can produce high quality solutions in significantly less computation time than exact methods, particularly in complex scenarios. Moreover, the interpretability analysis reveals that the model learns patterns aligned with operational logic, reinforcing its validity as a decision support system. Overall, this work provides a methodological contribution to the development of intelligent solutions aligned with the principles of Industry 5.0, promoting transparency, trust, and effective integration between humans and advanced technologies.

Tabla de Contenidos

Lista de Tablas

Página

Lista de Figuras

1. Introducción

A lo largo de la historia de la humanidad, el avance tecnológico y la mejora de procesos ha sido el motor fundamental en la transformación y mejora de los sistemas productivos. Desde la primera revolución industrial, hasta la automatización digital de la Industria 4.0, cada etapa ha representado un salto significativo en la eficiencia y complejidad de los procesos industriales. Sin embargo, el paradigma actual comienza a experimentar una nueva evolución: la Industria 5.0. (Leng et al., 2022).

La Industria 5.0 propone no solo una continuidad en el desarrollo tecnológico, sino también un replanteamiento sobre la forma en que los seres humanos interactúan con la tecnología dentro de los sistemas de producción, promoviendo una fuerte colaboración sinérgica entre personas, máquinas inteligentes y sistemas ciberfísicos (CPS). A diferencia de su predecesora, la Industria 4.0, cuyo enfoque privilegia la automatización y la conectividad, este nuevo paradigma busca integrar la Inteligencia Artificial (IA) y la robótica con la creatividad y las habilidades humanas, centrándose en factores decisivos en relación con dimensiones sociales y ambientales, con especial foco en las consideraciones humanas, la sostenibilidad, la resiliencia y particularmente, la interpretabilidad de los sistemas de IA (Nikiforidis et al., 2025). Esta última característica resulta crítica ya que busca que los modelos no solo sean eficientes, sino que también comprensibles y transparentes, permitiendo una mayor confianza y control por parte de los operadores humanos. Es en este contexto, donde la Inteligencia Artificial Explicable (XAI) emerge como un campo clave para enfrentar los desafíos asociados a la adopción de tecnologías complejas en entornos productivos.

Uno de los problemas más representativos y desafiantes en la gestión de operaciones industriales es el problema de programación flexible de talleres de trabajo o Flexible Job Shop Scheduling Problem (FJSSP), una variante del clásico Job Shop Scheduling Problem (JSSP). El FJSSP consiste en asignar operaciones de manera eficiente a un conjunto de máquinas con capacidades y tiempos de procesamiento variables, considerando múltiples rutas alternativas de procesamiento, lo que aumenta su flexibilidad, pero también la complejidad del problema. Debido a esto, este problema es clasificado como NP-hard, lo cual implica que su resolución requiere estrategias avanzadas para encontrar soluciones óptimas en tiempos razonables, siendo necesario aplicar enfoques avanzados de optimización, tales como programación entera, heurísticas, metaheurísticas, métodos híbridos y modelos de Machine Learning (Destouet et al., 2023; Aribi et al., 2023).

En ese mismo contexto, los modelos de Machine Learning, y particularmente, las redes neuronales artificiales (ANN) han demostrado ser herramientas poderosas para enfrentar problemas complejos de planificación y optimización (Fonseca & Navaresse, 2002; Smit et al., 2025). No obstante, un problema recurrente en el uso de este tipo de modelos es su carácter de "caja negra", lo que dificulta la comprensión de los procesos de decisión y limita su adopción en sistemas productivos que requieren una mayor transparencia y capacidad de justificación. Es aquí, donde las técnicas de interpretabilidad como SHapley Additive exPlanations (SHAP), desempeñan un papel crucial al permitir descomponer las predicciones de los modelos en contribuciones específicas atribuibles a cada variable de entrada, lo que facilita la comprensión y validación de las decisiones por parte de los operadores (Erlenbusch & Stricker, 2025).

Por consiguiente, la presente investigación se enmarca en el estudio de aplicabilidad de técnicas de XAI para abordar el problema FJSSP en el contexto de la Industria 5.0. Se plantea como objetivo principal evaluar y comprender la interpretabilidad de modelos de redes neuronales aplicados a la resolución del problema de programación flexible de talleres, utilizando SHAP como herramienta de análisis, lo que responde a la necesidad creciente de desarrollar modelos que no solo entreguen buenos resultados, sino que también permitan explicar sus decisiones de manera transparente y confiable.

Para ello, se desarrolla una red neuronal entrenada con datos generados por un modelo de programación lineal entera mixta (MILP), con el objetivo de predecir asignaciones de máquinas a operaciones en función del estado del sistema. Posteriormente, estas predicciones son integradas en un modelo híbrido que incorpora un algoritmo genético encargado de optimizar la secuenciación de operaciones. Finalmente, sobre el modelo entrenado, se aplica la metodología SHAP para interpretar el impacto de las variables de entrada en cada predicción, permitiendo analizar la lógica interna del modelo y su coherencia con principios operacionales reales.

La relevancia de este estudio radica en los siguientes aspectos:

- Aporta al desarrollo incipiente de la Industria 5.0, un campo aún poco explorado desde una perspectiva operativa e ingenieril.
- Contribuye a la discusión sobre el rol del ser humano en entornos productivos altamente automatizados, promoviendo la creación de herramientas que refuercen la confianza, la comprensión y la capacidad de intervención en decisiones críticas.

- Proporciona información y evidencias sobre el desarrollo de ANN, uno de los modelos más investigados para la resolución de los FJSSP.
- Integra metodologías cuantitativas y experimentales, combinando técnicas de programación matemática con aprendizaje automático, lo que permite realizar un análisis riguroso sobre el comportamiento, eficiencia y la explicabilidad de los modelos desarrollados.

En los siguientes capítulos se presenta la estructura del estudio desarrollado. En el capítulo 2, se expone el marco teórico que aborda los conceptos fundamentales relacionados con la Industria 5.0, el FJSSP, las redes neuronales y las técnicas de interpretabilidad de modelos de inteligencia artificial como SHAP. En el capítulo 3, se describe en detalle la metodología abordada, incluyendo la formulación matemática del FJSSP mediante MILP, la construcción del dataset, el diseño y entrenamiento del modelo de red neuronal, la implementación del modelo híbrido con algoritmo genético, y la aplicación de SHAP como técnica XAI. En el capítulo 4, se presentan los resultados obtenidos por cada uno de los métodos implementados: MILP, modelo híbrido y análisis explicativo con SHAP. Finalmente, el capítulo 5 presenta la discusión de los hallazgos y expone las conclusiones generales del estudio, sus implicancias en el ámbito de la Industria 5.0 y posibles líneas de investigación futura.

1.1. Objetivos

1.1.1. Objetivo general

El objetivo general de este estudio consiste en evaluar y comprender la interpretabilidad de un modelo de redes neuronales aplicado a la resolución del FJSSP mediante la reinterpretación de valores SHAP, con el fin de analizar la influencia de las variables de entrada en las decisiones del modelo y así mejorar su transparencia y comprensibilidad, al mismo tiempo que se reducen los tiempos de cómputo requeridos para resolver instancias de gran escala. Todo lo anterior, en línea con los principios de la Industria 5.0 orientados a la colaboración humano-máquina y la toma de decisiones informadas.

1.1.2. Objetivos específicos

Los objetivos específicos que se desprenden de este estudio son los siguientes:

- Modelar y resolver un FJSSP mediante un modelo MILP.
- Diseñar y entrenar una red neuronal capaz de predecir asignaciones de máquinas a operaciones en un FJSSP.

- Desarrollar un modelo híbrido que combine las predicciones de la red neuronal con un algoritmo genético para optimizar la secuenciación de operaciones y mejorar los tiempos de resolución.
- Evaluar la eficiencia y precisión del modelo híbrido en comparación con el modelo MILP.
- Aplicar e interpretar técnicas XAI, en particular SHAP para explicar y analizar el impacto de las variables en las decisiones del modelo de redes neuronales.

2. Marco teórico

2.1. Industria 5.0: Paradigma de la reintegración humana

La historia ha visto a la humanidad pasar por varias revoluciones industriales a un ritmo acelerado, impulsando desarrollos transformadores en todos los subsistemas de la sociedad. La primera revolución industrial introdujo la mecanización mediante la energía a vapor, marcando el paso desde una economía agraria a una industrial. Posteriormente, la segunda revolución introdujo la producción en masa y el uso de electricidad como principal fuente de energía. La tercera revolución industrial, también conocida como Revolución Digital trajo consigo transformaciones económicas y tecnológicas basadas en el desarrollo de la informática, la tecnología digital, la automatización industrial, internet y la biotecnología. Luego, la cuarta revolución industrial, denominada Industria 4.0 incorpora un nuevo modelo de producción basado en tecnologías digitales para mejorar la eficiencia y la productividad, integrando conectividad avanzada, el Internet de las Cosas (IoT), Big Data e IA para optimizar la producción mediante CPS (Leng et al., 2022).

Si bien es cierto que cada revolución ha presentado un salto enorme en términos de eficiencia y productividad, también ha generado ciertas tensiones y desafíos en la sociedad producto de barreras sociales, desafíos tecnológicos, privacidad y seguridad, entre otros (Leng et al., 2022). En este contexto, la Industria 5.0 emerge como una evolución lógica de la humanidad, que replantea el rol del ser humano dentro de los sistemas industriales. A diferencia de la lógica de automatización extrema promovida por la Industria 4.0, este nuevo paradigma se centra en restituir la participación humana como un componente primordial del proceso productivo, poniendo especial atención en la centralidad humana y las necesidades sociales. Es por esta razón que la visión de la Industria 5.0 enfatiza una colaboración sinérgica entre humanos y máquinas inteligentes, teniendo como principios rectores la centralidad humana, sostenibilidad y resiliencia (Murtaza et al., 2024). En otras palabras, se busca que los sistemas de producción sean no solo eficientes e inteligentes, sino también socialmente responsables, ecológicamente sostenibles y capaces de adaptarse a cambios e interrupciones imprevistas. En la Figura 2.1 se muestra una ilustración que representa los principios rectores de la Industria 5.0.

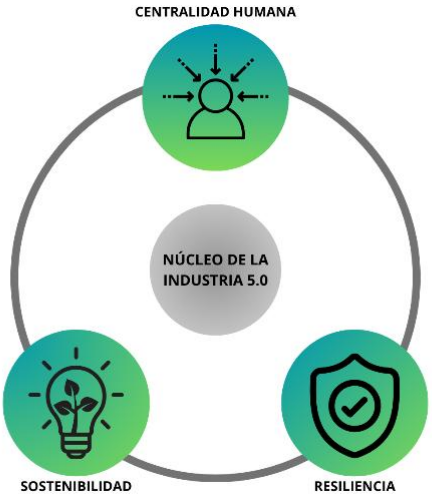

Figura 2.1. Núcleo de la Industria 5.0.

Fuente: Elaboración propia a partir de Murtaza et al., 2024

Uno de los conceptos clave que presenta esta reintegración del ser humano es el de Operador 5.0, definido por Destouet et al. (2023) como un "operador inteligente y calificado que utiliza la creatividad humana, el ingenio e innovación, ayudado por la información y la tecnología para superar obstáculos en el camino de desarrollar nuevas soluciones rentables que garanticen la sostenibilidad a largo plazo de las operaciones de fabricación y el bienestar de los trabajadores ante condiciones difíciles o inesperadas". Esta lógica de reintegración humana también se relaciona con los principios de la Sociedad 5.0 promovida inicialmente en Japón y desarrollada por Leng et al. (2022) como un modelo donde los datos, la IA y la robótica se ponen al servicio del bienestar humano, impulsando una sociedad inclusiva, segura y sostenible. Así, la Industria 5.0 adopta estos principios incorporándolos en el núcleo de la organización industrial y desplazando el foco desde la producción centrada exclusivamente en eficiencia hacia un modelo más ético y humano.

Se hace necesario resaltar que el concepto centrado en el ser humano no solo mejora la colaboración entre humanos y máquinas, sino que también destaca el potencial de las habilidades humanas para mejorar las soluciones de planificación y control de producción (Ma et al., 2025). La Industria 5.0 es capaz de suplir ciertas limitaciones de las máquinas como puede ser la flexibilidad, la adaptabilidad a situaciones extraordinarias o la percepción global en la toma de decisiones, a la vez que mejora el rendimiento de los usuarios y los sistemas (Murtaza et al., 2024). Asimismo, la colaboración entre humanos y máquinas permite a las personas abordar tareas más complejas y destinar a las máquinas inteligentes aquellas tareas que resultan repetitivas en las estaciones de trabajo.

Según Sharma et al. (2025), el rol de la Industria 5.0 es esencial en la transición global hacia sistemas energéticos sostenibles, no solo mediante la adopción de energías limpias, sino también a través de la capacitación de una fuerza laboral capaz de adaptarse a los nuevos desafíos socioeconómicos que esto implica. Este proceso representa un cambio fundamental en las operaciones industriales, ya que incentiva a las empresas a reducir su dependencia de fuentes de energía no renovables y a incorporar soluciones tecnológicas que promuevan la eficiencia y descarbonización de los procesos productivos. Los autores destacan casos de implementación de principios de sostenibilidad en sectores manufactureros mediante la incorporación de procesos circulares, reducción de residuos y estrategias energéticas basadas en IA colaborativa, lo cual demuestra que la sostenibilidad resulta ser un pilar operativo dentro de este nuevo paradigma. Sin embargo, existe la necesidad de contar con estructuras de gobernanza sólidas que prioricen la innovación ética y sostenible para asegurar que los avances tecnológicos estén alineados con los valores sociales y medioambientales.

Asimismo, la Comisión Europea (2021) destaca la relevancia de otro pilar de la Industria 5.0: la resiliencia, la cual es entendida no solo como la capacidad de recuperación de un sistema frente a interrupciones, sino también como una cualidad que implica anticipar, adaptarse y transformarse frente a crisis y perturbaciones sistémicas. Este enfoque abarca la habilidad de rediseñar procesos productivos, asegurando la continuidad operativa en entornos dinámicos para crear cadenas de suministro más robustas y diversificadas. Sin embargo, la resiliencia no solo se limita a responder a crisis puntuales, sino que también busca integrarse proactivamente en el diseño de los sistemas industriales, fomentando la capacidad de adaptarse a transformaciones tecnológicas, socioeconómicas y ambientales a largo plazo. Esta visión transformadora se ha visto materializada en proyectos piloto europeos que integran tecnologías digitales con principios de economía circular y justicia social, validando el potencial de la Industria 5.0 como un vehículo para impulsar modelos de producción sostenibles, resilientes y centrados en el valor humano, capaces de enfrentar las demandas sociales y ambientales del presente y del futuro.

2.2. Flexible Job Shop Scheduling Problem (FJSSP)

Los problemas de programación de la producción se han estudiado ampliamente en la literatura en todos los sectores industriales. Uno de los modelos más estudiados en este ámbito es el JSSP, el cual representa un caso clásico. En este problema, se nos da un conjunto de n trabajos $J = \{J_1, J_2, \dots, J_n\}$ y un conjunto de m máquinas $M = \{M_1, M_2, \dots, M_m\}$. Cada trabajo J_i consiste en n_i operaciones $O_i = \{O_{i1}, O_{i2}, \dots, O_{in_i}\}$, donde O_{ij} representa la operación j del trabajo i, que debe procesarse en una

secuencia única y en una máquina predefinida, donde todas las máquinas son diferentes y sus tiempos de procesamientos son conocidos y constantes. El objetivo del JSSP es encontrar una asignación óptima de trabajos sobre un conjunto de máquinas para minimizar o maximizar alguno de los criterios de rendimiento deseados, como pueden ser: el tiempo total de finalización (makespan), el tiempo de espera o retraso total, el número de trabajos tardíos, el costo de procesamiento, los ingresos de producción total, entre otros (Destouet et al., 2023).

A pesar de que el JSSP ha sido ampliamente abordado como un caso académico típico, la verdad es que su aplicabilidad directa a escenarios industriales reales es limitada debido a su estructura rígida, ya que asume que cada operación solo puede ser procesada por una única máquina específica, lo cual no se ajusta completamente a los entornos productivos modernos donde la flexibilidad es esencial para responder a la variabilidad en la demanda, a las fallas imprevistas o a la disponibilidad de recursos. Por ello, para representar de forma más realista los entornos productivos actuales, se ha desarrollado la variante del FJSSP, la cual introduce un nivel adicional de complejidad y flexibilidad. Se dice que un taller de trabajo es flexible cuando permite que algunas o todas las operaciones sean procesadas por más de una máquina alternativa, cada una con diferentes tiempos de procesamiento. Esto permite adaptarse a variaciones en la disponibilidad de recursos, cambios en la demanda o detención del proceso productivo por algún evento inesperado como puede ser el fallo de alguna de las máquinas (Gong et al., 2018).

La Figura 2.2 ilustra la diferencia entre el JSSP y el FJSSP. En esta figura se muestran dos conjuntos de trabajos $J = \{J_1, J_2\}$ y tres máquinas $M = \{M_1, M_2, M_3\}$, cada trabajo consta de tres operaciones denotadas por O_{ij}, donde i indica el número del trabajo y j el número de la operación de la secuencia. En la parte superior, correspondiente al JSSP, cada operación tiene asignada una máquina específica para su ejecución y la ruta de trabajo está claramente definida por flechas sólidas, lo que significa que la máquina para la siguiente operación es fija. Por otro lado, en la parte inferior correspondiente al FJSSP, se introduce flexibilidad al permitir que una misma operación pueda ser procesada por diferentes máquinas, lo cual se representa mediante líneas punteadas, las cuales muestran las posibles asignaciones para cada operación. Por ejemplo, para el trabajo J_1, la operación O_{12} puede ser realizada en la máquina M_2 o M_3, cada una con distintos tiempos de procesamiento. Esta flexibilidad en la asignación de máquinas permite mayor cantidad de combinaciones para la planificación, pero también incrementa la complejidad del problema.

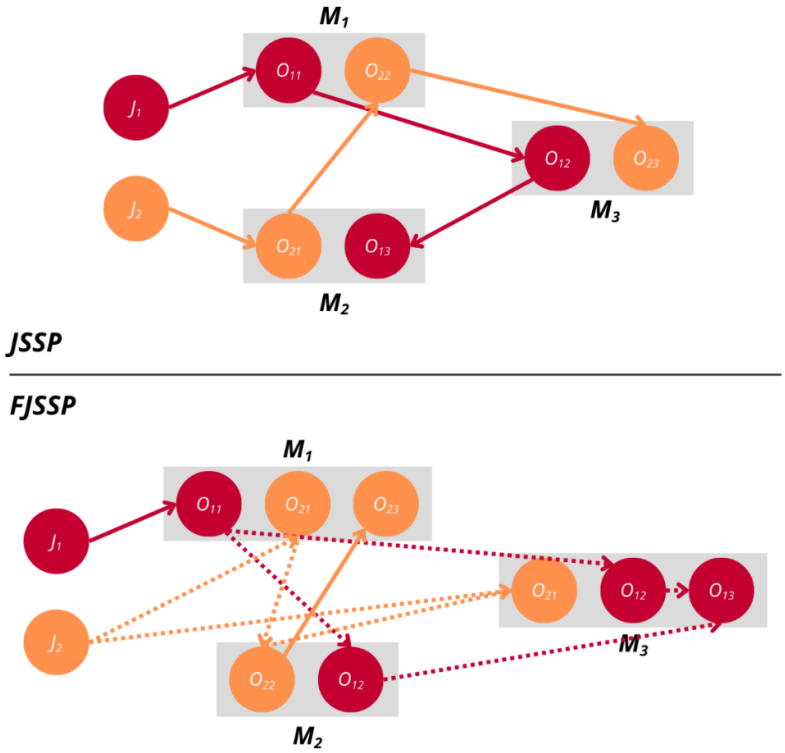

Figura 2.2. Comparación entre JSSP y FJSSP.

Fuente: Elaboración propia a partir de Destouet et al., 2023.

El FJSSP al igual que otros problemas más complejos es clasificado como un problema NP-hard, lo cual implica que no se conocen algoritmos exactos que sean eficientes y capaces de generar una solución óptima en un tiempo polinomial. Por esta razón, se han desarrollado varias estrategias de resolución para tratar este problema. Por un lado, se encuentran los enfoques exactos como los modelos de programación lineal entera mixta (MILP), que permiten formular el problema de forma rigurosa, pero que son computacionalmente costosos y aplicables solo a problemas de pequeña escala (Fattahi et al., 2007). Por otra parte, se han desarrollado heurísticas y metaheurísticas como recocido simulado, búsqueda tabú y, principalmente, algoritmos genéticos, que permiten explorar soluciones cercanas al óptimo en menores tiempos de cómputo. En este contexto, Gong et al. (2018) logran aplicar con éxito un algoritmo genético híbrido (NHGA) para reducir el makespan en un Double Flexible Job Shop Scheduling Problem (DFJSSP), un problema con asignaciones flexibles de máquinas y trabajadores, demostrando mayor eficiencia y precisión en problemas de gran escala. Asimismo, Aribi et al. (2023) presentan una estrategia dinámica basada en un algoritmo genético mejorado (EGA) para resolver un FJSSP capaz de adaptarse a condiciones variables, integrando

además restricciones de fatiga y consumo energético, lo que representa un avance hacia modelos más sostenibles y realistas.

En entornos productivos dinámicos, Jimenez et al. (2017) desarrollan “Pollux” una arquitectura hibrida de control dinámico (D-HCA) para el FJSSP que permite la conmutación entre diferentes modos operativos en entornos flexibles, utilizando algoritmos genéticos como mecanismos de respuesta ante eventos inesperados. Esta arquitectura demostró mejorar la adaptabilidad operativa del sistema en pruebas aplicadas a entornos industriales reales, mostrando mejores resultados frente a arquitecturas estáticas. Por otra parte, distintos estudios han explorado el uso de modelos de aprendizaje automático, destacándose las ANN como herramientas capaces de aprender patrones complejos de asignación y secuenciación. Fonseca & Navaresse (2002) y Smit et al., (2025) muestran que las ANN pueden generar soluciones viables y competitivas frente a enfoques clásicos, especialmente cuando se dispones de grandes volúmenes de datos simulados o históricos.

No obstante, la mayoría de los trabajos presentados en la literatura se han enfocado principalmente en la dimensión técnica del problema, poniendo especial énfasis en los recursos de la máquina y dejando de lado factores humanos y medioambientales (o incorporando solo uno de ellos), los cuales son fundamentales en el contexto de la Industria 5.0. Destouet et al. (2024) sostienen que, en el contexto actual de crecimiento demográfico y agotamiento progresivo de los recursos naturales, resulta fundamental reevaluar el uso de los recursos en los sistemas de fabricación. Los autores argumentan que la sostenibilidad ambiental, las condiciones humanas y la eficiencia económica no deben abordarse de manera aislada, ya que son una parte integral del proceso de programación que no debiesen ser pasadas por alto. Sus hallazgos evidencian que la integración de dichos factores en modelos como el FJSSP no solo es viable, sino también necesarias para garantizar una alta calidad, adaptabilidad y responsabilidad social de los sistemas modernos. Por lo tanto, para lograr una industria más sostenible, resiliente y basada en el valor humano, se debe explorar el FJSSP en estas direcciones.

En contraste con los enfoques existentes, la presente investigación propone avanzar en la aplicación de modelos de redes neuronales para resolver el FJSSP, pero integrando técnicas de explicabilidad, específicamente SHAP, con el fin de reinterpretar y comprender las decisiones del modelo desde una perspectiva centrada en el ser humano. Esto busca no solo optimizar la asignación y secuenciación de operaciones, sino también facilitar la interpretación y validación de los resultados, promoviendo su adopción en contextos industriales reales bajo los principios de la Industria 5.0.

2.3. Machine Learning: Redes Neuronales en la resolución del FJSSP

Dada la naturaleza NP-hard del FJSSP, las técnicas exactas como MILP tienden a ser ineficientes en términos de tiempo de compilación y memoria consumida, especialmente cuando se escala la complejidad del sistema o se busca una resolución en tiempo real. Por esta razón, han cobrado relevancia el uso de enfoques alternativos que, si bien no garantizan una solución óptima, permiten obtener resultados de buena calidad en tiempos computacionales razonables. Entre estos enfoques se encuentran las metaheurísticas, los modelos híbridos y, más recientemente, las técnicas de aprendizaje automático, las cuales son capaces de aprender patrones de asignación y secuenciación a partir de datos históricos o simulados.

Dentro del ámbito del aprendizaje automático se encuentran las ANN, las cuales constituyen un modelo computacional inspirado en la estructura del cerebro humano. Estas redes están compuestas por capas de nodos (también llamadas neuronas artificiales), los cuales están interconectados mediante pesos ajustables que permiten transformar las entradas de datos en salidas a través de funciones de activación. Esta arquitectura permite modelar relaciones no lineales complejas entre variables, lo cual es particularmente útil en contextos donde las reglas explícitas son difíciles de definir.

Las ANN han sido exploradas como una herramienta potente y flexible, capaz de modelar de forma no lineal y adaptativa la estructura del problema de programación para enfrentar la complejidad del FJSSP. Fonseca y Navaresse (2002) demostraron que las redes neuronales pueden utilizarse para inferir reglas de prioridad en problemas que requieren una planificación más compleja, abriendo la puerta a estudios más sofisticados. Mas recientemente, estudios como los realizados por Smit et al. (2025) y Liu et al. (2025) han demostrado la eficacia de las redes neuronales para abordar variantes dinámicas del problema JSSP, incorporando incertidumbre, variabilidad y restricciones heterogéneas propias del entorno industrial.

Morinaga et al. (2023) proponen una mejora metodológica para el JSSP basada en la combinación de programación matemática y redes neuronales. En su estudio, se utiliza un método mejorado para encontrar una solución inicial que satisfaga las restricciones del modelo de programación entera mixta (MIP), logrando reducir significativamente los tiempos de solución sin comprometer la calidad de los resultados. Posteriormente, Morinaga et al. (2024) profundizan sobre la misma línea de investigación, proponiendo un nuevo modelo de reducción de características de entrada, logrando optimizar tanto el

tiempo de entrenamiento como la precisión del modelo al eliminar información irrelevante del espacio de decisiones.

Un enfoque más avanzado es el propuesto por Liu et al. (2025), el cual emplea redes neuronales de grafos heterogéneos (HGNN) en conjunto con aprendizaje de refuerzo profundo (DRL) para abordar el problema de programación dinámico de talleres de trabajo (DJSSP). La arquitectura propuesta logra abordar la complejidad estructural y la heterogeneidad de relaciones en entornos dinámicos de manufactura, donde las condiciones cambian contantemente debido a eventos inesperados como pueden ser fallos en las máquinas o la llegada de nuevos trabajadores. Además, el modelo resulta ser adaptable en tiempo real e independiente del tamaño del problema, demostrando altos niveles de generalización y eficiencia en entornos de manufactura inteligentes propios de la Industria 5.0.

En base a lo anterior, es posible notar que el FJSSP plantea desafíos únicos, lo que implica la necesidad de métodos de programación robustos, adaptables y capaces de prosperar en entornos dinámicos. Las estrategias basadas en Machine Learning han mostrado beneficios considerables en comparación con los enfoques convencionales para abordar este tipo de problemas. La capacidad de estos métodos para gestionar datos complejos y de alta dimensión los posiciona como una de las mejores opciones para abordar desafíos de programación a gran escala debido a su velocidad de procesamiento en comparación con los métodos exactos o heurísticos. Sin embargo, a pesar del alto rendimiento que pueden ofrecer este tipo de modelos, una de las principales limitaciones radica en su escasa interpretabilidad. La mayoría de los modelos de ANN operan como cajas negras, donde las decisiones se generan sin una explicación clara de los factores que llevaron a la IA a tomar ciertas decisiones. Por ello, para garantizar la confianza y aceptación de los modelos por parte de los usuarios, especialmente en entornos industriales colaborativos como los que propone la Industria 5.0, se vuelve indispensable incorporar técnicas de interpretabilidad.

2.4. Interpretabilidad de modelos de IA

Si bien los modelos de redes neuronales han demostrado un alto rendimiento en la resolución de problemas complejos como lo es el FJSSP, su principal limitación radica en su escasa interpretabilidad. Estos modelos suelen operar como cajas negras, generando predicciones sin proporcionar una explicación clara sobre los factores que motivan sus decisiones. Esta falta de transparencia representa un obstáculo crítico para su adopción en entornos industriales reales, donde las decisiones deben ser comprensibles, auditables y confiables por parte de los operadores humanos.

Por esta razón, ha surgido una creciente demanda por técnicas que permitan explicar y justificar el comportamiento y las decisiones de los modelos de inteligencia artificial, especialmente en entornos colaborativos como los propuestos por la Industria 5.0, donde la sinergia entre humanos y máquinas se vuelve esencial.

En respuesta a esta necesidad aparece el concepto de XAI, el cual ofrece un conjunto de metodologías que permiten comprender, auditar y justificar las decisiones tomadas por modelos complejos como pueden ser las ANN. Herrera (2025) propone una distinción entre dos conceptos: RED XAI y BLUE XAI. El primero está orientado al diseño técnico de los modelos y su validación estructural, mientras que el segundo se enfoca en la comprensión por parte del usuario, la utilidad de las explicaciones y la gobernanza algorítmica. Lo anterior resulta especialmente relevante en el contexto de la Industria 5.0, donde el ser humano debe desempeñar un rol activo en la toma de decisiones apoyadas por la IA y no limitarse a ser un mero receptor de recomendaciones.

Es necesario destacar que cuando los operadores comprenden las limitaciones del sistema de IA, su desempeño conjunto mejora significativamente, lo cual destaca la importancia de no solo explicar los aciertos del modelo, sino también hacer visibles sus debilidades para facilitar una colaboración transparente e informada entre operadores y máquinas (Rieger et al., 2025). Por otra parte, y desde una perspectiva organizacional, XAI tiene un rol clave en los procesos de innovación y gestión del conocimiento, ya que un sistema explicable permite que el conocimiento generado por la IA sea comprendido y transferido por diferentes usuarios dentro de una misma organización, fortaleciendo así la capacidad adaptativa y la toma de decisiones en ambientes dinámicos.

Se hace necesario resaltar que en áreas como la salud y la manufactura aditiva (impresión 3D) ya existen algunas aplicaciones de XAI, como las mostradas por Yahata et al. (2025) y Ukwaththa et al. (2024). Sin embargo, en el ámbito de la manufactura discreta, y en particular en problemas de planificación como el FJSSP, la incorporación de técnicas de interpretabilidad es nula. Esto representa una brecha en la literatura y una oportunidad para generar investigaciones que busquen integrar explicabilidad en los modelos de programación.

Bajo este contexto, técnicas como SHAP ofrecen una solución robusta al problema de caja negra en modelos de Machine Learning. SHAP es un método de explicabilidad basado en la teoría de juegos Shapley, el cual permite descomponer la predicción de un modelo en contribuciones individuales de cada variable de entrada, proporcionado tanto explicaciones locales (para instancias específicas) como

globales (para todo el modelo), lo cual ayuda a entender no solo que decisiones se toman, sino también el por qué.

En síntesis, la revisión bibliográfica evidencia que los avances recientes en redes neuronales aplicadas al FJSSP han permitido abordar con éxito problemas de planificación complejos en entornos caracterizados por incertidumbre, variabilidad y restricciones multidimensionales. Sin embargo, también se confirma que gran parte de estos desarrollos se enfocan exclusivamente en mejorar el rendimiento computacional, sin considerar el componente de la explicabilidad. A pesar de los importantes aportes de XAI en áreas como la medicina, finanzas, justicia o la manufactura aditiva (Herrera, 2025), su aplicación en problemas de programación de la producción como el FJSSP, es prácticamente inexistente, lo que limita la transparencia y confiabilidad de los modelos, dificultando su integración en entornos colaborativos como los que son promovidos por la Industria 5.0, donde la interacción humano y máquina exige modelos comprensibles y auditables.

En este escenario, se identifica una brecha clara en la literatura, la necesidad de incorporar técnicas de interpretabilidad en modelos de redes neuronales aplicados al FJSSP. Esta incorporación no solo permitiría comprender el comportamiento del modelo y sus decisiones, sino que también facilitaría la aceptación por parte de los usuarios. Por esto, la presente investigación busca responder a esta necesidad, evaluando el potencial explicativo de la metodología SHAP en la reinterpretación de decisiones tomadas por el modelo de redes neuronales en la planificación del FJSSP, contribuyendo así a cerrar la brecha entre precisión algorítmica y comprensión humana en el contexto de la Industria 5.0.

3. Materiales y métodos

La metodología desarrollada en este estudio consta de cinco etapas principales, las cuales están orientadas al desarrollo, entrenamiento e interpretación de un modelo de red neuronal aplicado a la resolución del FJSSP, integrando herramientas de optimización exacta, aprendizaje automático y técnicas de inteligencia artificial explicable.

En primer lugar, se modela y resuelve el FJSSP utilizando un modelo MILP. Para ello, se utilizan 20 instancias propuestas por Fattahi et al. (2007), las cuales son extraídas del repositorio estandarizado de Behnke & Geiger (2012) (ver Anexos 1 y 2 para detalles de las instancias y su estructura). Estas instancias se dividen en 10 pequeñas, que consideran entre 2 a 4 trabajos con 1 a 5 máquinas, y 10 medianas, las cuales incluyen entre 5 a 12 trabajos con 6 a 8 máquinas. El objetivo de esta primera etapa consiste en obtener soluciones óptimas o cercanas al óptimo que permitan generar un conjunto de datos confiables y representativos del proceso de asignación de operaciones a máquinas para el entrenamiento supervisado de la red neuronal desarrollada posteriormente. Estos datos incluyen: información específica sobre la operación (identificador, posición en el flujo de trabajo, tiempo de procesamiento), el estado del sistema al momento de la decisión (disponibilidad y carga de las máquinas), y la asignación efectiva realizada por el modelo MILP.

A partir de las soluciones entregadas por el modelo MILP, las cuales definen la asignación óptima (o casi óptima) de cada operación a una máquina específica dentro de una instancia FJSSP, se extrajo la información necesaria para construir un dataset estructurado. Este conjunto de datos fue utilizado con fines de entrenamiento supervisado de una red neuronal, integrando tanto el resultado de las asignaciones como el contexto del sistema productivo en el momento de cada decisión. En total, se generaron 347 muestras, donde cada una representa una decisión individual de asignación de máquina a una operación y contiene un total de 43 atributos, los cuales son descritos a continuación:

- $\boldsymbol{job}$**:** Número identificador del trabajo al que pertenece la operación.
- $\boldsymbol{op}$**:** Número de la operación dentro del flujo de trabajo.
- $\boldsymbol{m_i}$**:** Variables binarias que indican si la máquina i (con i = 1, 2, ...,8) está habilitada para llevar a cabo la operación, donde: 1 indica que la máquina está habilitada y 0 indica que no lo está.
- $\boldsymbol{t_{m_i}}$**:** Tiempo de procesamiento requerido por la operación si se ejecuta en la máquina i (si la máquina no esta habilitada, el valor es 0).

- $\boldsymbol{disp_{m_i}}$**:** Refleja la primera unidad de tiempo en que la máquina i queda libre para iniciar una nueva operación.
- $\boldsymbol{op_{cola}}$**:** Número de operaciones restantes en cola del mismo trabajo.
- $\boldsymbol{carga_{m_i}}$**:** Carga actual de trabajo asignado a la máquina i.
- $\boldsymbol{assigned_{m_i}}$**:** Codificación one-hot de la máquina asignada para llevar a cabo la operación. Solo una de las ocho variables posee el valor 1, indicando la máquina seleccionada para procesar la operación, el resto de las máquinas tienen valor 0.

La limitación de 8 máquinas responde a la estructura de las instancias seleccionadas para el estudio, ya que todas ellas consideran como máximo 8 máquinas en sus configuraciones. Esto permite normalizar la dimensión de entrada y salida del modelo, facilitando su entrenamiento sin la necesidad de ajustar dinámicamente su arquitectura. No obstante, la metodología es completamente extensible a problemas con un mayor o menor número de máquinas, modificando en consecuencia la cantidad de variables de entrada y de salida utilizadas. De esta forma, el resultado final es un dataset estructurado con 43 columnas por fila: 35 correspondientes a las variables de entrada y 8 correspondientes a la variable categórica objetivo codificado mediante one-hot encoding ($assigned_{m_i}$). Así, el dataset generado sirve como base para la siguiente etapa del estudio, en la que se desarrolla un modelo de red neuronal artificial cuyo objetivo es predecir la máquina más adecuada para asignar una operación, en función del estado del sistema y las características de la operación en cuestión.

La tercera etapa consiste en el desarrollo de un modelo hibrido que integra las asignaciones predichas por la red neuronal con un algoritmo genético encargado de optimizar la secuenciación de las operaciones. Es importante destacar que la red neuronal solo predice la máquina a la que debe asignarse cada operación, pero no define el orden en que estas deben ser ejecutadas. Por ello, el algoritmo genético parte de una población inicial basada en la estructura entregada por la red neuronal y evoluciona durante 100 generaciones aplicando procesos de selección, cruce y mutación, donde cada individuo es evaluado mediante simulación del cronograma completo y cálculo del tiempo total de procesamiento, descartando soluciones inviables mediante penalización y entregando aquella que posee el menor makespan.

En la cuarta etapa se incorpora la metodología SHAP como herramienta de interpretación post hoc, con el objetivo de analizar la lógica interna del modelo de red neuronal y aportar transparencia a sus decisiones. Una vez que el modelo es entrenado, este puede ser explicado por cualquier instancia de

entrada gracias a la metodología SHAP. Sin embargo, para efectos de análisis y presentación de resultados, se seleccionan 10 instancias del conjunto de prueba, ya que estas no son utilizadas durante el entrenamiento del modelo. De esta forma, para cada instancia se obtuvieron los valores SHAP de las 35 variables de entrada, analizando su contribución sobre la probabilidad de asignación a cada una de las 8 máquinas.

Finalmente, en la quinta etapa se presentan los resultados obtenidos para cada una de las etapas anteriores. Se presentan los desempeños del modelo MILP, la red neuronal y el modelo híbrido, evaluando su eficiencia y calidad de solución. Además, se analiza la información entregada por SHAP para identificar los factores más influyentes en las decisiones del modelo y determinar si estas siguen una lógica coherente con los principios operativos del FJSSP.

Para facilitar la comprensión global de la metodología desarrollada, la Figura 3.1 presenta un esquema general que resumen las 5 etapas metodológicas abordadas en este estudio.

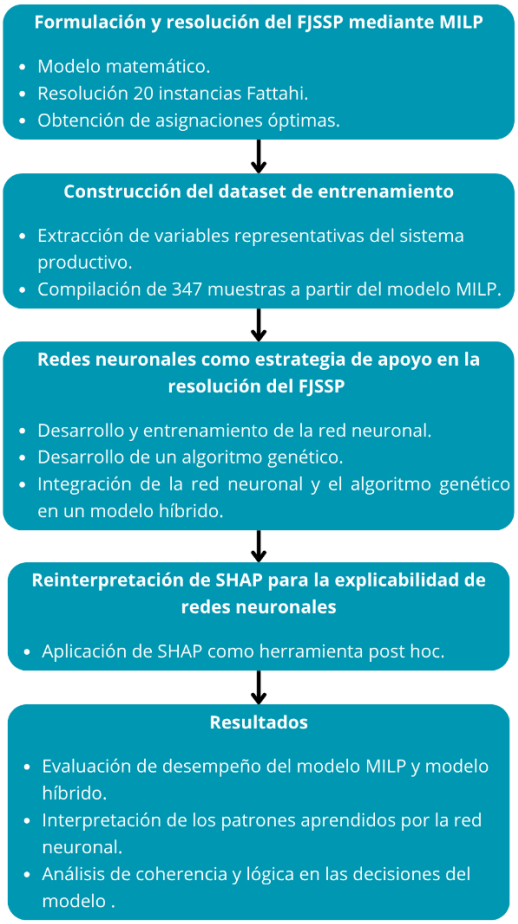

Figura 3.1. Etapas del estudio.

Fuente: Elaboración propia.

El motivo de este estudio surge como respuesta a la creciente necesidad de desarrollar sistemas inteligentes explicables en contextos industriales complejos, donde no solo se requiere precisión en la toma de decisiones, sino también transparencia y confiabilidad. Así, esta investigación aporta un enfoque metodológico para integrar XAI en problemas de planificación de la producción, contribuyendo a la transición hacia una Industria 5.0 centrada en la colaboración humano-máquina y la toma de decisiones informadas.

En las siguientes secciones de este documento se desarrolla en detalle cada una de las etapas mencionadas, presentando su fundamento técnico, implementación y los resultados obtenidos.

Todas las simulaciones y experimentos se desarrollaron en un equipo con procesador AMD Ryzen 5, 16 GB de RAM y tarjeta gráfica NVIDIA GTX 1050, utilizando el lenguaje de programación Python y Visual Studio Code como entorno de desarrollo. Las principales librerías utilizadas fueron: Pandas, NumPy, Matplotlib, Scikit-learn, PuLP, PyTorch y SHAP. En cuanto a la recopilación de información para sustentar teóricamente esta investigación, se llevó a cabo una revisión bibliográfica en bases de datos como ScienceDirect y Scielo, seleccionando artículos revisados por pares entre los años 2000 y 2025. Las búsquedas se realizaron utilizando combinaciones de los siguientes términos: "Industry 5.0", "Flexible Job Shop Scheduling Problem", "Explainable Artificial Intelligence", "Neural Networks", "SHAP".

3.1. Formulación y resolución del FJSSP mediante MILP

En esta etapa se presenta la formulación matemática y la implementación computacional del FJSSP mediante MILP. El objetivo de esta etapa es obtener una solución óptima (o cercana al óptimo) para las distintas instancias de prueba seleccionadas, con el fin de extraer las decisiones individuales de asignación máquinas a operaciones, las cuales servirán como base para construir un dataset de entrenamiento que será utilizado en etapas posteriores para entrenar una red neuronal artificial que busque replicar los patrones de decisión observados. Por lo tanto, esta etapa no solo permite resolver directamente el problema, sino que además constituye el punto de partida para el desarrollo del modelo híbrido propuesto en este estudio.

3.1.1. Modelo matemático

Para modelar y resolver el FJSSP mediante MILP se consideran los siguientes supuestos:

- Cada trabajo consta de una secuencia de operaciones que deben ser ejecutadas en un orden predefinido.
- Cada operación puede ser procesada por un subconjunto de máquinas elegibles, donde cada máquina posee diferentes tiempos de procesamiento.
- Cada máquina puede procesar solo una operación a la vez.
- No se consideran interrupciones en el procesamiento de las operaciones, es decir, una vez que comienza una operación en alguna máquina, esta debe completarse sin ser interrumpida.
- El objetivo es minimizar el tiempo total necesario para completar todos los trabajos (makespan).

Los parámetros y variables de decisión del modelo se presentan en la Tabla 3.1.

Tabla 3.1. Parámetros y variables de decisión del modelo MILP.

Notación	Descripción
J	Conjunto de trabajos
O_j	Conjunto de operaciones del trabajo j, con $j \in J$
M	Conjunto de máquinas
$M_{j,o}$	Conjunto de máquinas capaces de ejecutar la operación o del trabajo j, con $o \in O_j$
$p_{j,o,m}$	Tiempo de procesamiento de la operación o del trabajo j en la máquina m, con $m \in M$
M	Constante suficientemente grande (Big M)
$x_{j,o,m}$	$x_{j,o,m} = 1$; si la operación (j, o) se lleva a cabo en la máquina m, en otro caso, $x_{j,o,m} = 0$
$s_{j,o}$	Tiempo de inicio de la operación (j, o)
$y_{(j,o)(j',o')}$	$y_{(j,o)(j',o')} = 1$; si la operación (j, o) se procesa antes que la operación (j', o') en la misma máquina, en otro caso $y_{(j,o)(j',o')} = 0$
C_{max}	Makespan

Fuente: Elaboración propia.

En base a la descripción del problema y la definición de las variables anteriores, la función objetivo del modelo matemático se establece de la siguiente manera:

$$Min\ C_{max} \tag{1}$$

Sujeto a:

$$\sum_{m \in M_{j,o}} x_{j,o,m} = 1\ ;\ \forall (j,o) \tag{2}$$

$$s_{j,o+1} \geq s_{j,o} + \sum_{m \in M_{j,o}} p_{j,o,m} \times x_{j,o,m}\ ;\ \forall j, \forall o < \left|O_j\right| \tag{3}$$

$$s_{j,o} + p_{j,o,m} \leq s_{j',o'} + \mathbf{M}\left(1 - x_{j,o,m} + 1 - x_{j',o',m} + 1 - y_{(j,o)(j',o')}\right)\ ; \quad \forall (j,o) \neq (j',o'), \forall m \in M_{j,o} \cap M_{j',o'} \tag{4}$$

$$s_{j',o'} + p_{j',o',m} \leq s_{j,o} + \mathbf{M}\left(1 - x_{j,o,m} + 1 - x_{j',o',m} + y_{(j,o)(j',o')}\right)\ ; \quad \forall (j,o) \neq (j',o'), \forall m \in M_{j,o} \cap M_{j',o'} \tag{5}$$

$$C_{max} \geq s_{j,o} + \sum_{m \in M_{j,o}} p_{j,o,m} \times x_{j,o,m}\ ;\ \forall (j,o) \tag{6}$$

La ecuación (1) establece el objetivo de optimización de este problema, el cual busca minimizar el makespan del sistema. La ecuación (2) nos asegura que cada operación debe ser asignada exactamente a una única máquina elegible dentro de su conjunto de alternativas. La ecuación (3) garantiza el cumplimiento de la secuencia de operaciones dentro de cada trabajo, estableciendo que cada operación comience solo después de que la operación anterior del mismo trabajo haya finalizado, respetando así su tiempo de procesamiento. Por su parte, las ecuaciones (4) y (5) imponen las restricciones de no solapamiento entre operaciones que comparten una misma máquina. La ecuación (4) establece que, si dos operaciones comparten una misma máquina, una de ellas debe terminar antes de que la otra pueda empezar. De igual manera, la ecuación (5) complementa esta lógica. Por ejemplo, si $y_{(j,o)(j',o')} = 0$, entonces la operación (j', o') debe terminar antes que (j, o). En ambas restricciones se utiliza la constante Big M para desactivar las restricciones si las operaciones no son asignadas a la misma máquina. De esta manera, se garantiza que no se produzcan conflictos de asignaciones temporales

entre máquinas. Finalmente, la ecuación (6) nos asegura que el makespan debe ser mayor o igual al tiempo de finalización de las operaciones del sistema.

Las instancias utilizadas para ser resueltas mediante el modelo matemático corresponden a un conjunto de 20 instancias de FJSSP propuestas por Fattahi et al (2007), las cuales son ampliamente utilizadas en la literatura para validar metodologías de programación debido a su variabilidad en tamaño, flexibilidad y complejidad.

3.2. Construcción del dataset de entrenamiento

Uno de los mayores desafíos al trabajar con modelos de aprendizaje automático radica en la necesidad de contar con grandes volúmenes de datos relevantes y estructurados para su entrenamiento. En el caso del FJSSP, si bien existen instancias ampliamente utilizadas para evaluar algoritmos de resolución, estas no incluyen la información detallada que se requiere para entrenar modelos predictivos, como las condiciones del sistema productivo en el momento de la decisión o los atributos específicos asociados a cada asignación de máquinas, por lo que fue necesario generar un conjunto de datos adaptado a los objetivos del estudio que permita a la red neuronal aprender patrones de asignación de máquinas en función del entorno y características de cada operación

Para este propósito, se construyó un dataset a partir de las 20 instancias propuestas por Fattahi et al. (2007), las mismas que fueron resueltas mediante el modelo MILP presentado en la sección anterior. A partir de las soluciones obtenidas, se extrajeron y estructuraron datos representativos de cada asignación, generando un total de 347 muestras individuales, correspondientes a decisiones individuales de asignación de máquina para cada operación específica.

Cada registro del dataset representa características relevantes tanto del trabajo como del estado del sistema. Las variables de entrada consideradas fueron:

- $\boldsymbol{job}$**:** Número identificador del trabajo al que pertenece la operación.
- $\boldsymbol{op}$**:** Número de la operación dentro del flujo de trabajo.
- $\boldsymbol{m_i}$**:** Variables binarias que indican si la máquina i (con i = 1, 2, ...,8) está habilitada para llevar a cabo la operación, donde: 1 indica que la máquina está habilitada y 0 indica que no lo está.
- $\boldsymbol{t_{m_i}}$**:** Tiempo de procesamiento requerido por la operación si se ejecuta en la máquina i (si la máquina no esta habilitada, el valor es 0).

- $\boldsymbol{disp_{m_i}}$**:** Refleja la primera unidad de tiempo en que la máquina i queda libre para iniciar una nueva operación.
- $\boldsymbol{op_{cola}}$**:** Número de operaciones restantes en cola del mismo trabajo.
- $\boldsymbol{carga_{m_i}}$**:** Carga actual de trabajo asignado a la máquina i.
- $\boldsymbol{assigned_{m_i}}$**:** Codificación one-hot de la máquina asignada para llevar a cabo la operación. Solo una de las ocho variables posee el valor 1, indicando la máquina seleccionada para procesar la operación, el resto de las máquinas tienen valor 0.

Cabe destacar que el modelo desarrollado está limitado a problemas con un máximo de 8 máquinas, tal como lo definen las instancias seleccionadas. Sin embargo, esta estructura es extensible a problemas de mayor escala ajustando las columnas involucradas. De esta forma, el resultado final es un dataset estructurado con 43 columnas por fila: 35 correspondientes a las variables de entrada y 8 correspondientes a la codificación one-hot de la variable dependiente ($assigned_{m_i}$). Este conjunto de datos permite el entrenamiento de una red neuronal supervisada para predecir, a partir del contexto operativo, cual es la máquina más adecuada para ejecutar una operación dada. En la Tabla 3.2 se presenta un ejemplo de una fila del dataset de entrada junto con la descripción correspondiente de cada una de sus columnas, lo que permite visualizar como se estructuran los datos para el entrenamiento del modelo.

Tabla 3.2. Ejemplo de una fila del dataset de entrenamiento y descripción de sus componentes.

Entrada: 1,2,1,1,0,0,0,0,0,0,0,32,24,0,0,0,0,0,0,0,45,37,0,0,0,0,0,0,0,1,45,37,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0	
(1,2)	Operación 2 del trabajo 1.
(1,1,0,0,0,0,0,0)	Conjunto de máquinas habilitadas para esta operación (máquina 1 y 2).
(32,24,0,0,0,0,0,0)	Tiempos de procesamiento por cada máquina.
(45,37,0,0,0,0,0,0)	Disponibilidad actual de las máquinas.
(1)	Operaciones en cola para el trabajo actual.
(45,37,0,0,0,0,0,0)	Carga actual de trabajo de cada máquina.
(0,1,0,0,0,0,0,0)	Máquina 2 asignada a la operación.

Fuente: Elaboración propia.

Este tipo de representación permite encapsular tanto la información estructural del problema como el estado del entorno, lo que resulta clave para que la red neuronal pueda identificar patrones

significativos y generalizables para la asignación de máquinas. En la siguiente sección se detalla la arquitectura de la red neuronal y el proceso de entrenamiento llevado a cabo.

3.3. Redes neuronales como estrategia de apoyo en la resolución del FJSSP

En esta etapa, se presenta el desarrollo e implementación de un modelo híbrido compuesto por una red neuronal artificial y un algoritmo genético. A partir del dataset generado anteriormente, se busca entrenar una red neuronal con el fin de aprender patrones eficientes de asignación de máquinas a operaciones en función del estado del sistema. Posteriormente, las asignaciones generadas por la red se integran en un esquema de resolución híbrido que combina las predicciones del modelo con un algoritmo genético simple, el cual aprovecha estas asignaciones como punto de partida para construir soluciones iniciales prometedoras y guiar la búsqueda hacia zonas factibles del espacio de solución.

La estrategia propuesta tiene por objetivo mejorar la eficiencia computacional al reducir los tiempos de resolución en instancias de mayor complejidad, facilitando además la toma de decisiones locales mediante decisiones rápidas y basadas en experiencia previa. De este modo, es posible avanzar hacia un sistema de planificación más ágil y colaborativo, en sintonía con los principios de la Industria 5.0.

A continuación, se describen las etapas relacionadas con el diseño y entrenamiento del modelo de redes neuronales y la integración del modelo híbrido.

3.3.1. Arquitectura y entrenamiento de la red neuronal

El modelo utilizado fue desarrollado utilizando la biblioteca Pytorch, donde la red neuronal implementada consiste en una arquitectura secuencial de tipo feedforward, también conocida como red neuronal multicapa (Multilayer Perceptron, MLP), es decir, una red neuronal simple, donde la información se mueve en una sola dirección: desde la capa de entrada, pasando por capas ocultas, hasta la capa de salida, sin retroalimentaciones internas ni bucles. Este tipo de arquitectura resulta adecuada para tareas de clasificación supervisada como la asignación de máquinas a operaciones en el contexto del FJSSP (Goodfellow et al., 2016, cap. 6).

La red neuronal cuenta con tres capas ocultas, compuestas por 128, 64 y 32 neuronas respectivamente. Cada capa oculta utiliza la función de activación LeakyReLU, una variante de ReLU (Rectified Linear Unit) popularmente utilizada. A diferencia de ReLu, que anula completamente los valores negativos, LeakyReLU permite un pequeño gradiente cuando la entrada es negativa, lo que ayuda a evitar que las neuronas “mueran” permanentemente, mejorando así el proceso de aprendizaje de la red.

Adicionalmente, para prevenir el sobreajuste dentro del modelo, se aplicó la técnica de regularización mediante Dropout con una tasa del 20% en las dos primeras capas ocultas. Esta técnica permite desactivar aleatoriamente un porcentaje de neuronas durante el entrenamiento, lo cual obliga a la red a no depender excesivamente de ciertas conexiones y a generalizar de mejor manera (Goodfellow et al., 2016, caps. 6-7). Cabe destacar que para la tercera capa oculta se omite el uso de Dropout para no interferir con la estabilización de las salidas que se dirigen a la capa final de predicción. La arquitectura completa del modelo se resume en la Figura 3.2.

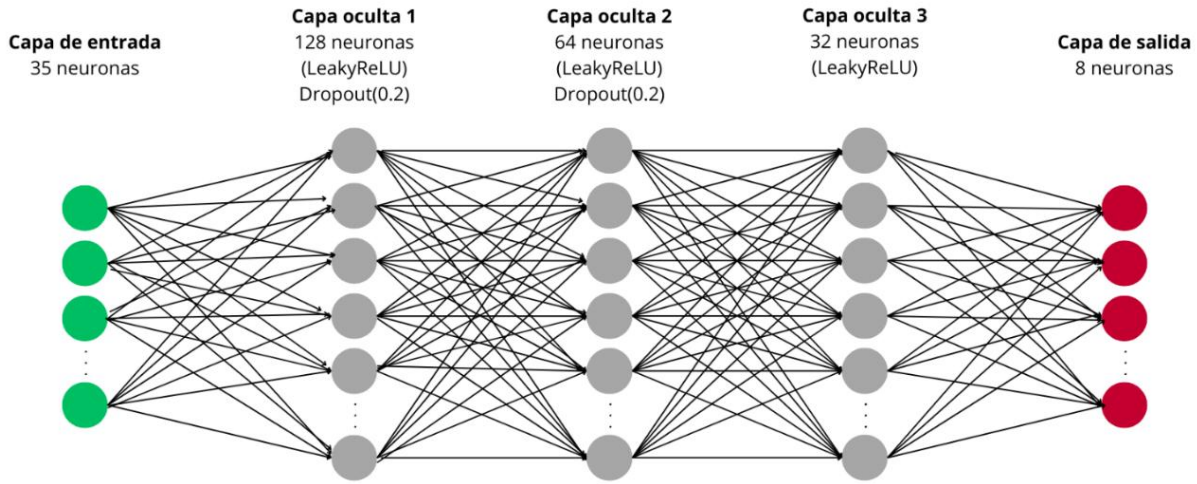

Figura 3.2. Arquitectura de la red neuronal.

Fuente: Elaboración propia.

Para el entrenamiento del modelo se utilizó la función de perdida CrossEntropyLoss, la cual es ampliamente empleada para problemas de clasificación multiclase. Esta función compara la probabilidad predicha para cada clase con la clase real (codificada como índice entero) y penaliza fuertemente las predicciones incorrectas con alta confianza. Como optimizador se utilizó Adam (Adaptive Moment Estimation) con una tasa de aprendizaje de 0.001, valor seleccionado por su capacidad de converger de forma estable y rápida en redes de tamaño intermedio. El optimizador Adam combina las ventajas de dos métodos: AdaGrad y RMSProp, adaptando la tasa de aprendizaje para cada parámetro de forma individual en función de las estimaciones de primer y segundo momento del gradiente (Goodfellow et al., 2016, caps. 5,6 y 8).

Antes de entrenar la red, y para mejorar su rendimiento durante el entrenamiento, se aplicó un escalado estándar (Z-score) a las variables de entrada mediante la función StandardScaler de Scikit-learn. Este proceso resulta fundamental para redes neuronales, ya que mejora la estabilidad numérica, acelera la

convergencia y asegura que las neuronas reciban entradas en un rango comparable, lo que evita que algunas características dominen el proceso de aprendizaje simplemente por tener un mayor rango numérico (Scikit-learn Developers, n.d.). Posteriormente, el conjunto de datos se dividió en un 80% para entrenamiento y un 20% para validación, permitiendo medir la capacidad del modelo para realizar predicciones sobre datos no vistos. El entrenamiento se llevó a cabo durante 500 épocas, monitoreando la función de pérdida y la precisión de clasificación sobre el conjunto de entrenamiento.

Al finalizar el proceso de entrenamiento, el modelo alcanzó una función de pérdida de 0.0814 y una precisión del 75.71% sobre el conjunto de validación, lo que indica que, si bien el modelo logró captar de forma significativa los patrones en los datos, aún existe un margen de mejora en la capacidad de predicción. Sin embargo, considerando la naturaleza compleja del FJSSP, la precisión del modelo representa un desempeño razonable, especialmente si se considera que la red fue entrenada sobre un conjunto relativamente reducido de muestras. En este contexto, el modelo se considera lo suficientemente competente como para ser utilizado como una estrategia de apoyo dentro de un modelo híbrido.

3.3.2. Modelo híbrido: Integración con Algoritmo Genético

Con el objetivo de mejorar la resolución de instancias complejas del FJSSP, se desarrolló un modelo híbrido que combina las predicciones generadas por la red neuronal con un algoritmo genético (AG). En este modelo, se utiliza la red neuronal para predecir asignaciones de máquinas, mientras que el AG cumple la función de optimizar la secuenciación de las operaciones sobre las máquinas asignadas. Esto permite reducir el espacio de búsqueda y construir soluciones viables en tiempos razonables, incluso cuando el modelo MILP no converge debido a la complejidad del problema.

El proceso comienza con la asignación de máquinas a operaciones mediante la red neuronal previamente entrenada. Para ello, se recorren las distintas operaciones de una instancia, prediciendo la mejor máquina disponible en función del estado del sistema. A medida que se van realizando asignaciones, se actualizan las variables de entrada para reflejar la evolución del entorno. Una vez se asigna una máquina a cada operación, la información se transforma en una estructura interna que agrupa los trabajos y operaciones con su respectiva máquina y tiempo de proceso, conformando la base de datos para la población inicial del algoritmo genético.

El AG parte de una población de tamaño fijo, definida como 50 individuos, donde cada uno representa una posible planificación. Para construir cada individuo, se distribuyen las operaciones asignadas a

cada máquina (según lo predicho por la red) en un orden aleatorio, manteniendo la asignación, pero variando la secuencia, lo que permite diversificar y explorar múltiples configuraciones viables sin perder la coherencia estructural proporcionada por la red neuronal.

Posteriormente, se realiza la decodificación y simulación de cada individuo, etapa clave para evaluar la factibilidad y calidad de la solución. Para cada solución candidata, se simula la ejecución de las operaciones, asegurándose de: respetar la precedencia entre operaciones de un mismo trabajo, calcular los tiempos de inicio y finalización de cada operación en función de la disponibilidad de las máquinas y del trabajo, y finalmente, registrar el cronograma y el makespan obtenido para la instancia. Cabe destacar que, para evitar bloqueos o ciclos infinitos, se contempla un mecanismo de control mediante contadores de espera que interrumpen la simulación en caso de no detectar progreso.

Con el objetivo de utilizar las mejores soluciones, cada individuo de la población es evaluado mediante el makespan del cronograma simulado, donde, las soluciones invalidas o bloqueadas reciben una penalización alta, lo que las descarta del proceso evolutivo. De esta forma, solo se privilegian soluciones factibles y eficientes en el proceso de selección.

La selección de padres se realiza mediante un método de selección por truncamiento, donde se ordena la población de forma creciente por el valor del makespan y se elige la mitad superior como base para la próxima generación, lo que asegura la conservación de los mejores individuos mientras que elimina aquellos con bajo rendimiento.

Durante el cruce (o crossover), se generan nuevos individuos a partir de dos padres. Para cada máquina, se selecciona de forma aleatoria si se hereda la secuencia de un padre u otro. Posteriormente, se aplica una fase de mutación con una probabilidad del 10%, la cual intercambia aleatoriamente la posición de dos operaciones dentro de la misma máquina, manteniendo la validez de la solución y permitiendo explorar nuevas configuraciones. De esta forma se introduce variabilidad y se previene la convergencia prematura del modelo.

El ciclo evolutivo se ejecuta por un número definido de 100 generaciones, donde en cada generación, la nueva población se forma a partir de los mejores individuos y sus descendientes cruzados y mutados, reemplazando completamente a la población anterior. Al finalizar el proceso, se retorna el mejor individuo encontrado junto a su cronograma y makespan. A continuación, en la Figura 3.3, se resume gráficamente el proceso completo de este modelo híbrido.

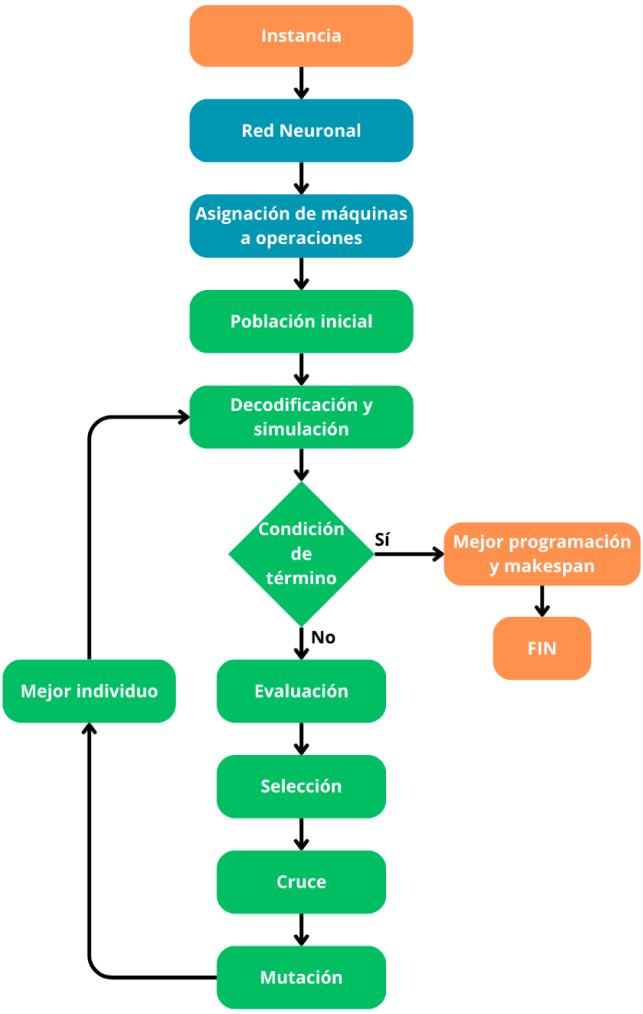

Figura 3.3. Proceso de ejecución del Modelo Híbrido.

Fuente: Elaboración propia.

En resumen, el modelo híbrido propuesto busca reducir los tiempos de resolución para instancias FJSSP de mayor tamaño. La red neuronal aporta conocimiento previamente aprendido sobre asignaciones eficientes, mientras que el algoritmo genético trabaja sobre estas decisiones explorando múltiples secuencias de ejecución.

3.4. Reinterpretación de SHAP para la explicabilidad de redes neuronales

En esta sección se aborda la aplicación de técnicas XAI para el análisis e interpretación del modelo de red neuronal desarrollado anteriormente. En particular, se utiliza SHAP, una de las metodologías más robustas y ampliamente aceptadas para explicar el comportamiento de modelos complejos de machine learning que utilizan aprendizaje por refuerzo, como lo son las redes neuronales profundas (Erlenbusch & Stricker, 2025). SHAP es una técnica de interpretación post hoc, es decir, se aplica una

vez que el modelo ha sido completamente entrenado y ha generado sus predicciones, lo que significa que no interfiere en el proceso de aprendizaje ni modifica la lógica del modelo, sino que analiza su comportamiento posterior.

Basado en la teoría de juegos cooperativos, SHAP asigna a cada variable de entrada un valor Shapley, el cual representa su contribución marginal a la salida del modelo. De este modo, es posible explicar las decisiones que ha tomado el modelo para generar una predicción específica, proporcionando justificaciones cuantitativas sobre la influencia que tuvo cada variable de entrada, lo que permite generar explicaciones tanto locales, interpretando el razonamiento del modelo para una única predicción, como globales, comprendiendo el comportamiento general del modelo sobre el conjunto de datos (Nikiforidis et al., 2025). En términos simples, SHAP indica qué tan relevante fue cada variable de entrada para que el modelo tome una determinada decisión.

La naturaleza del modelo de red neuronal desarrollado implica un comportamiento no lineal y opaco, lo que dificulta comprender cómo y por qué se obtienen determinadas predicciones. Esta falta de interpretabilidad, también conocida como carácter de caja negra, representa una barrera significativa para la adopción confiable de estos modelos por parte de los usuarios en contextos industriales reales, especialmente bajo los principios de la Industria 5.0, donde se fomenta la transparencia y la confianza.

Con el objetivo de aportar transparencia al proceso de asignación de máquinas llevado a cabo por la red neuronal, se aplicó SHAP para cuantificar el impacto de cada variable de entrada, es decir, se busca explicar en qué medida atributos como la disponibilidad de la máquina, su carga, el tiempo de procesamiento, el número de operaciones en cola o el tipo de trabajo influyen en las decisiones que toma el modelo. Este análisis busca responder algunas preguntas clave como: ¿Qué variables tienen mayor influencia sobre la asignación de máquinas? ¿Existe una lógica coherente en las decisiones del modelo?

En la siguiente sección se describe el proceso de integración de SHAP al modelo entrenado, con el fin de evaluar su coherencia dentro del contexto del FJSSP.

3.4.1. Incorporación de SHAP al análisis del modelo

Para explicar el razonamiento interno del modelo de red neuronal entrenado para predecir asignaciones de máquinas en el FJSSP, se incorporó la metodología SHAP para poder analizar el impacto individual de cada una de las variables de entrada sobre la predicción realizada por la red, favoreciendo así una mayor transparencia en sus decisiones. Las variables utilizadas corresponden a

aquellas definidas en la sección 3.2, e incluyen: el identificador del trabajo (job), el número de operación (op), el conjunto de máquinas habilitadas (m_i), los tiempos de procesamiento por máquina (t_{m_i}), la disponibilidad de las máquinas ($disp_{m_i}$), el número de operaciones en cola (op_{cola}) y la carga acumulada por máquina ($carga_{m_i}$).

Con el modelo ya entrenado y validado (con una precisión del 75,71% y una función de pérdida de 0.0814 en el conjunto de prueba), se procedió a su análisis SHAP. Dado que la red neuronal está desarrollada en PyTorch, se utilizó el método GradientExplainer, el cual está especialmente diseñado para modelos de aprendizaje profundo y problemas de clasificación multiclase. Este método estima los valores SHAP mediante el cálculo de los gradientes del modelo respecto a sus entradas, lo que permite capturar la sensibilidad de la predicción ante cambios marginales en cada variable. Para poder realizar el análisis, GradientExplainer requiere definir un conjunto de referencia (también llamado background), que representa un subconjunto del conjunto de entrenamiento utilizado para modelar el comportamiento promedio del modelo. Finalmente, requiere de una instancia específica a explicar, la cual es extraída del conjunto de prueba y corresponde a una operación concreta cuya predicción se desea analizar en detalle.

Gracias a este método, es posible descomponer la predicción del modelo en contribuciones atribuibles a cada variable de entrada, revelando qué tan influyentes fueron ciertos factores en la decisión final de asignación.

3.5. Resultados

En esta sección se presentan los resultados obtenidos a partir de la aplicación de las distintas etapas metodológicas llevadas a cabo en el estudio. En primer lugar, se analizan los resultados alcanzados mediante el modelo MILP para las distintas instancias FJSSP. Luego, se evalúa el desempeño del modelo híbrido que integra la red neuronal con el algoritmo genético, considerando tanto la calidad de las soluciones como los tiempos de cómputo alcanzados y su comparación con las soluciones alcanzadas por el modelo MILP. Finalmente, se presentan los resultados obtenidos de la aplicación de SHAP como técnica de explicabilidad, permitiendo interpretar la lógica del modelo de red neuronal y verificar si sus decisiones se alinean con criterios operativos coherentes del FJSSP.

3.5.1. Resultados MILP

Con el objetivo de evaluar la robustez y eficiencia del modelo planteado, se resuelven un total de 20 instancias del problema FJSSP (Fattahi et al., 2007), registrando para cada una de ellas el valor del makespan obtenido y los tiempos de cómputo requeridos. Gracias a este análisis, es posible medir el desempeño del modelo MILP ante distintos niveles de dificultad del problema, lo que garantiza construir un conjunto confiable de datos para el entrenamiento de la red neuronal desarrollada.

Los resultados obtenidos se presentan en la Tabla 3.3, donde se especifica para cada instancia el valor del makespan encontrado y el tiempo de cómputo asociado. Adicionalmente, con la finalidad de proporcionar una visualización estructurada de las secuencias de operaciones, se incorpora en el Anexo 3 el correspondiente diagrama de Gantt para cada instancia, los cuales permiten observar la distribución temporal de las operaciones por máquina y el uso de recursos en cada caso.

Tabla 3.3. Makespan y tiempo de cómputo de cada instancia FJSSP resuelta mediante MILP.

Instancia	Makespan (C_{max})	Tiempo [s]
SFJS1	66	0.03
SFJS2	107	0.04
SFJS3	221	0.13
SFJS4	355	0.10
SFJS5	119	0.15
SFJS6	320	0.17
SFJS7	397	0.12
SFJS8	253	0.18
SFJS9	210	0.13
SFJS10	516	0.16
MFJS1	468	1.34
MFJS2	446	1.68
MFJS3	466	4.97
MFJS4	554	26.64
MFJS5	514	12.06
MFJS6	634	91.18
MFJS7	879	887.74

MFJS8	887	3600
MFJS9	1147	3600
MFJS10	1288	3600

Fuente: Elaboración propia.

Cabe destacar que para la resolución de cada una de las instancias se utilizó la librería PuLP de Python, la cual permite modelar problemas de optimización lineal y entera de manera eficiente, ya que actúa como una interfaz flexible que conecta con distintos solvers de optimización, tales como: CBC (Coin-or Branch and Cut), HiGHS, CPLEX y Gurobi. Sin embargo, en este caso se configuró PuLP para utilizar HiGHS como solver principal debido a su disponibilidad como código abierto. Además, HiGHS ofrece tiempos de procesamiento considerablemente competitivos frente a otros solvers comerciales, lo que lo convierte en una buena alternativa para la resolución de las instancias de FJSSP consideradas. Adicionalmente, se configuró un tiempo límite de 3600 segundos para cada instancia con la finalidad de establecer un marco de comparación equitativo y evitar que los tiempos de cómputo se extiendan indefinidamente cuando aumenta la complejidad del problema.

Una vez resueltas las instancias, se llevó a cabo una segunda validación comparativa. Así, en la Tabla 3.4 se presentan los valores de makespan reportados en la literatura (Behnke & Geiger, 2012) frente a los valores de makespan obtenidos mediante el modelo propuesto, lo que permite analizar el desempeño del modelo desarrollado, identificando su capacidad para aproximarse o alcanzar los resultados documentados en la literatura.

Tabla 3.4. Comparación entre makespan reportados en la literatura con el modelo propuesto.

Instancia	**Makespan (Behnke & Geiger, 2012)**	**Makespan propuesto (C_{max})**
SFJS1	66	66
SFJS2	107	107
SFJS3	221	221
SFJS4	355	355
SFJS5	119	119
SFJS6	320	320
SFJS7	397	397
SFJS8	253	253

SFJS9	210	210
SFJS10	516	516
MFJS1	468	468
MFJS2	446	446
MFJS3	466	466
MFJS4	554	554
MFJS5	514	514
MFJS6	634	634
MFJS7	931	879
MFJS8	884	887
MFJS9	1070	1147
MFJS10	1208	1288

Fuente: Elaboración propia.

De los resultados obtenidos, es posible observar que el modelo MILP propuesto presenta un buen rendimiento para instancias de menor tamaño (SFJS1 a SFJS10), logrando obtener un makespan idéntico a los reportados en la literatura en fracciones de segundo. Sin embargo, en instancias de mayor complejidad (MFJS8 a MFJS10), el modelo comienza a presentar algunas limitaciones. Si bien logra encontrar soluciones viables, estas no alcanzan el valor óptimo reportado, llegando incluso al límite máximo de tiempo establecido para la ejecución, lo cual es esperable debido a la complejidad del FJSSP.

Pese a las limitaciones encontradas, la calidad de las soluciones obtenidas y la capacidad del modelo para resolver con precisión instancias de distinta escala permiten validar la eficacia del modelo como punto de partida para generar datos confiables. En particular, su desempeño justifica su uso como base para el entrenamiento supervisado de la red neuronal desarrollada, permitiendo que esta aprenda patrones de asignación a partir de soluciones óptimas o cercanas al óptimo previamente calculadas.

3.5.2. Resultados modelo híbrido

Las instancias seleccionadas para evaluar el desempeño del modelo híbrido corresponden a las 10 instancias MFJS (MFJS1 a MFJS10) previamente abordadas en la resolución del modelo MILP. El objetivo de esta etapa es analizar la capacidad del nuevo modelo para generar soluciones factibles en

un menor tiempo de cómputo, especialmente donde la resolución exacta resulta computacionalmente ineficiente.

Los resultados obtenidos por el modelo híbrido se presentan en la Tabla 3.5, la cual muestra para cada instancia MFJS el valor del makespan encontrado y el tiempo de cómputo asociado. Adicionalmente, con la finalidad de proporcionar una visualización estructurada de las secuencias de operaciones, se incorpora en el Anexo 4 el correspondiente diagrama de Gantt para cada instancia, los cuales permiten observar la distribución temporal de las operaciones por máquina y el uso de recursos en cada caso.

Tabla 3.5. Makespan y tiempo de cómputo de cada instancia FJSSP resuelta mediante el Modelo Híbrido.

Instancia	Makespan	Tiempo [s]
MFJS1	495	1.51
MFJS2	515	0.93
MFJS3	559	0.97
MFJS4	625	0.91
MFJS5	514	1.59
MFJS6	634	1.93
MFJS7	989	3.46
MFJS8	1016	4.20
MFJS9	1481	2.48
MFJS10	1496	3.54

Fuente: Elaboración propia.

Para facilitar la comparación directa entre el modelo MILP y el modelo híbrido propuesto, se presenta la Tabla 3.6, donde se contrastan los makespan y tiempos de cómputo obtenidos por ambos métodos.

Tabla 3.6. Comparación entre makespan y tiempos de cómputo proporcionados por el modelo MILP e híbrido.

Instancia	Makespan (MILP)	Tiempo [s]	Makespan (Modelo Híbrido)	Tiempo [s]
MFJS1	468	1.34	495	1.51
MFJS2	446	1.68	515	0.93
MFJS3	466	4.97	559	0.97
MFJS4	554	26.64	625	0.91
MFJS5	514	12.06	514	1.59

MFJS6	634	91.18	634	1.93
MFJS7	879	887.74	989	3.46
MFJS8	887	3600	1016	4.20
MFJS9	1147	3600	1481	2.48
MFJS10	1288	3600	1496	3.54

Fuente: Elaboración propia.

Los resultados obtenidos permiten observar que, los valores de makespan obtenidos mediante el modelo híbrido no siempre alcanzan los valores óptimos hallados por el modelo MILP. Sin embargo, es capaz de generar soluciones factibles en tiempos significativamente menores, especialmente en instancias donde el modelo MILP no converge antes del tiempo límite asignado de 3600 segundos. En particular, para las instancias MFJS8, MFJS9 y MFJS10, el modelo híbrido es capaz de entregar soluciones en menos de 5 segundos, mientras que el MILP alcanza el límite máximo de ejecución sin mejoras adicionales.

Gracias a la combinación de asignaciones generadas por la red neuronal y la capacidad exploratoria del AG es posible construir soluciones factibles y competitivas en tiempos operacionales muy reducidos. Sin embargo, se debe destacar que la calidad de las soluciones entregadas por el modelo híbrido depende fuertemente del desempeño predictivo de la red neuronal. Una red entrenada de mejor manera ya sea con más entradas de datos o una arquitectura más compleja podría mejorar la calidad de las asignaciones iniciales, permitiendo al AG explorar más regiones del espacio de soluciones.

A pesar de que el modelo demostró robustez y escalabilidad, siendo capaz de entregar soluciones viables para las instancias MFJS en tiempos inferiores a los requeridos por el MILP, uno de los desafíos que persiste en este tipo de modelos es su baja interpretabilidad. Las decisiones tomadas por la red neuronal no son fácilmente comprensibles ni auditables, lo cual limita su adopción en entornos donde el seguimiento de la decisión es fundamental. Dicho de otro modo, el modelo actúa como una caja negra, cuya lógica interna es opaca para los operadores humanos, lo cual pone en evidencia la necesidad de incorporar herramientas que permitan explicar y justificar el comportamiento del modelo, un aspecto fundamental de la Industria 5.0, donde la colaboración humano-máquina exige confianza y comprensibilidad en los sistemas inteligentes.

Atendiendo a estas consideraciones, la siguiente sección aborda la aplicación de técnicas XAI, enfocándose en la reinterpretación de valores SHAP como medio para analizar e interpretar el

razonamiento de la red neuronal en el proceso de asignación de máquinas, lo que contribuye a cerrar la brecha entre precisión predictiva y explicabilidad del modelo.

3.5.3. Resultados SHAP

Para evaluar la interpretabilidad del modelo de red neuronal, se seleccionaron las 100 primeras muestras del conjunto de entrenamiento como background y 10 instancias del conjunto de prueba para su análisis individual, es decir, datos no utilizados durante el entrenamiento del modelo, lo que garantiza que el análisis se realice sobre casos nuevos para la red, evaluando así la coherencia y generalización de su comportamiento. Para cada una de estas instancias se calcularon los valores SHAP correspondiente a las 35 variables de entrada del modelo, determinando su impacto sobre la probabilidad de asignación a cada una de las 8 máquinas posibles. De esta forma, es posible analizar no solo la predicción final del modelo, sino también comprender que tan influyentes resultaron ser los distintos atributos del entorno de producción para inclinar la decisión hacia una máquina específica.

A modo ilustrativo, en esta sección se presenta el análisis detallado de la primera instancia seleccionada, cuya estructura, al igual que la de las instancias restantes se encuentra en el Anexo 5 de este documento. En primer lugar, la Figura 3.4 muestra la distribución de probabilidades asignada por el modelo a cada una de las 8 máquinas, lo que permite visualizar la confianza en la predicción. En particular, para esta primera instancia, el modelo asigna una probabilidad del 87,91% a la máquina 5, lo cual indica una predicción altamente confiada.

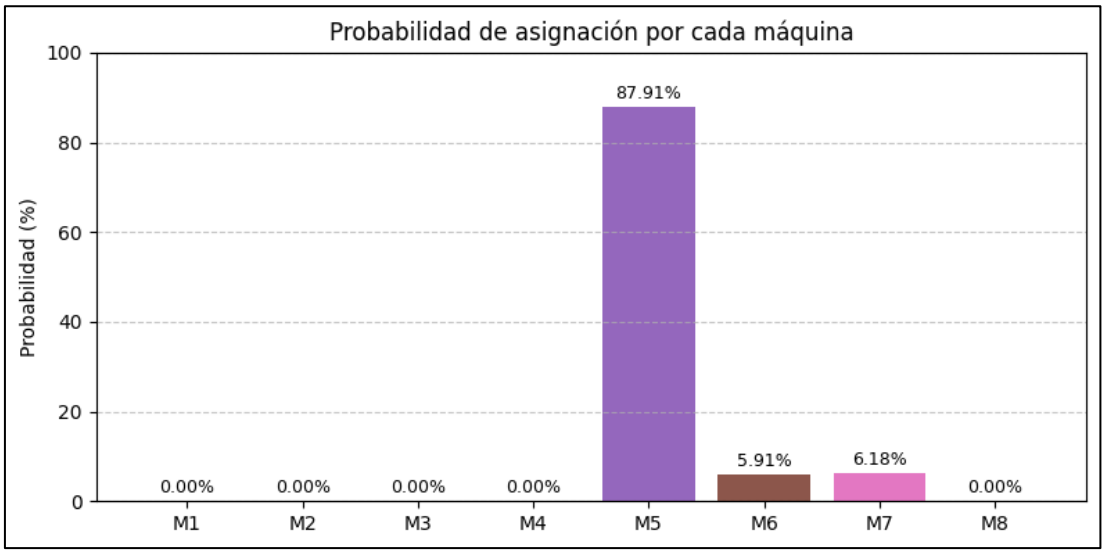

Figura 3.4. Probabilidad de asignación de máquinas para la instancia 1.

Fuente: Elaboración propia.

A continuación, se presentan los valores SHAP para cada una de las 8 máquinas en las Figuras 3.5 a 3.12, lo que permite visualizar de forma detallada cómo influyó cada variable de entrada en la probabilidad de asignación de cada máquina. De esta forma, también es posible comprender de manera clara y cuantitativa que factores aumentan o disminuyen la probabilidad de asignación, revelando la lógica que utiliza la red neuronal para tomar una decisión.

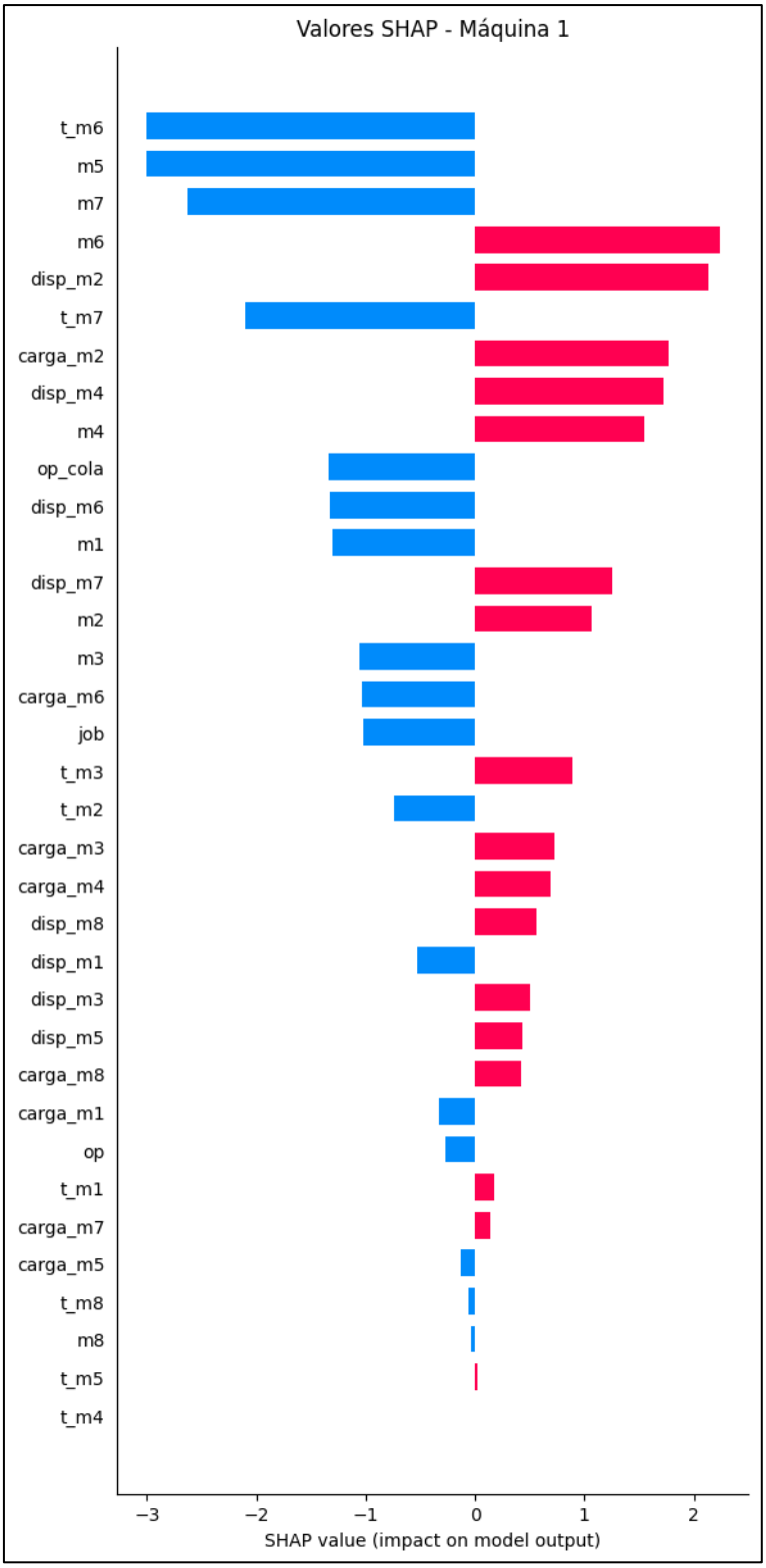

Figura 3.5. Valores SHAP asociados a la máquina 1 para la instancia 1.

Fuente: Elaboración propia.

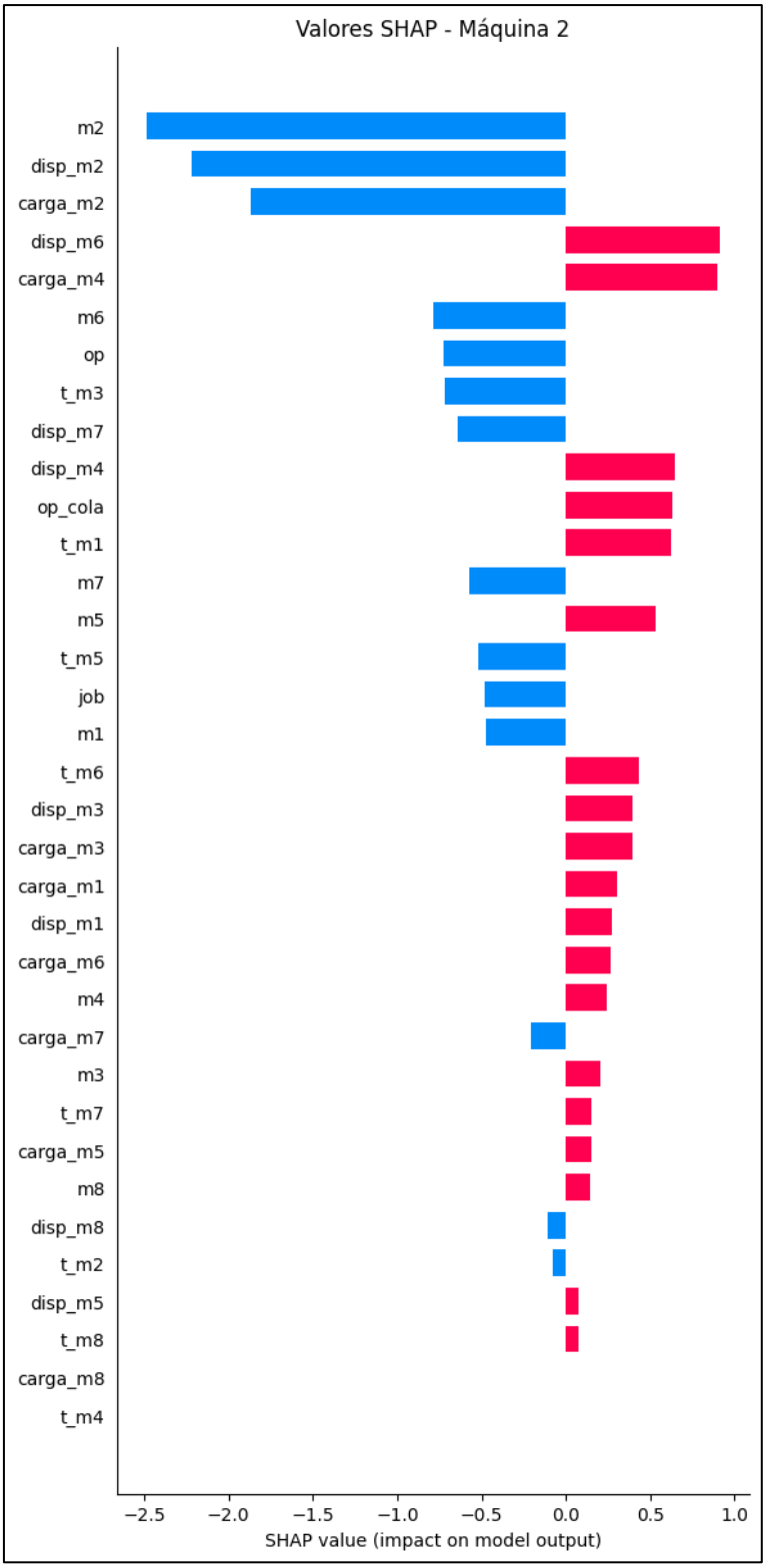

Figura 3.6. Valores SHAP asociados a la máquina 2 para la instancia 1.

Fuente: Elaboración propia.

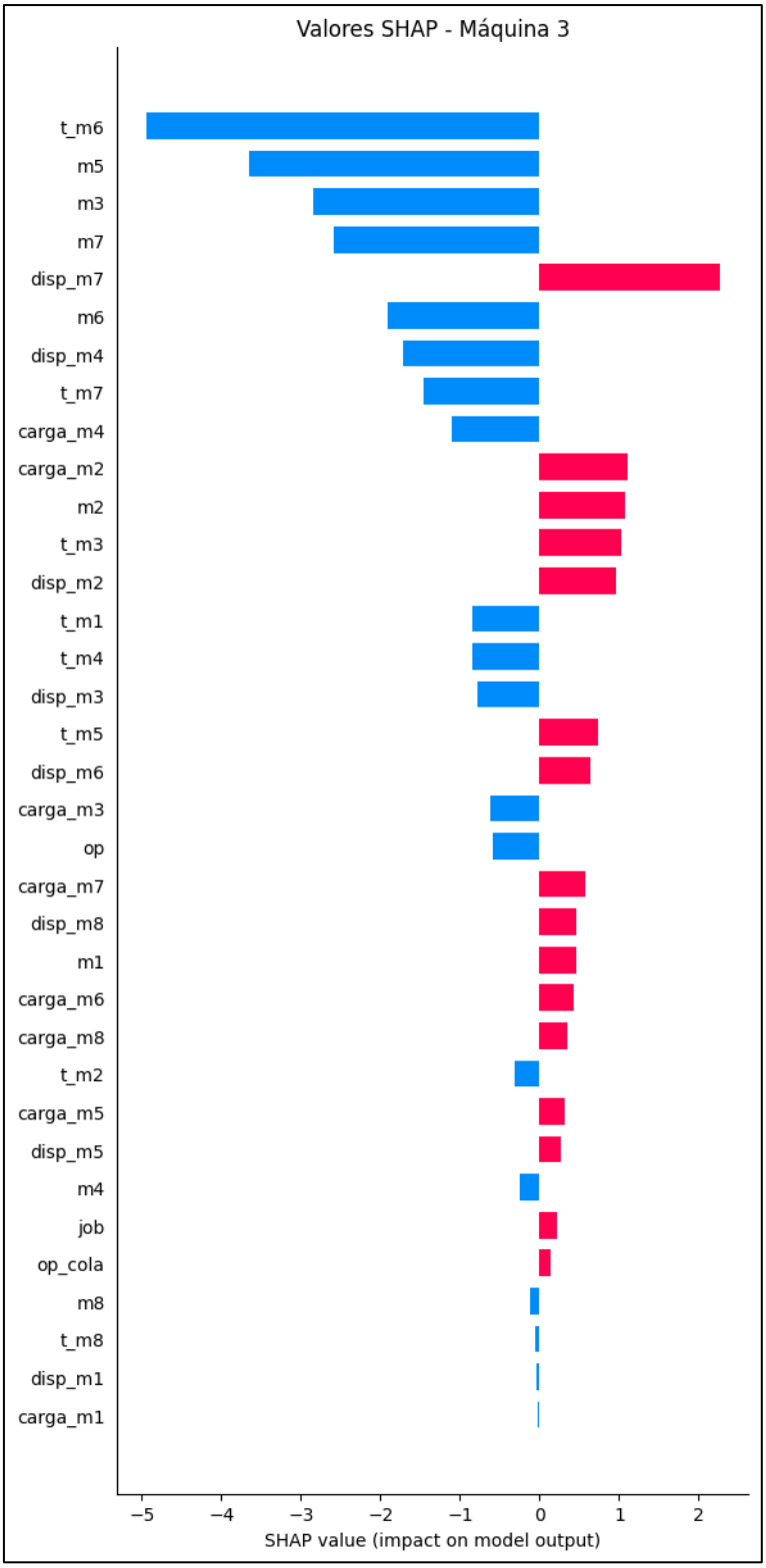

Figura 3.7. Valores SHAP asociados a la máquina 3 para la instancia 1.

Fuente: Elaboración propia.

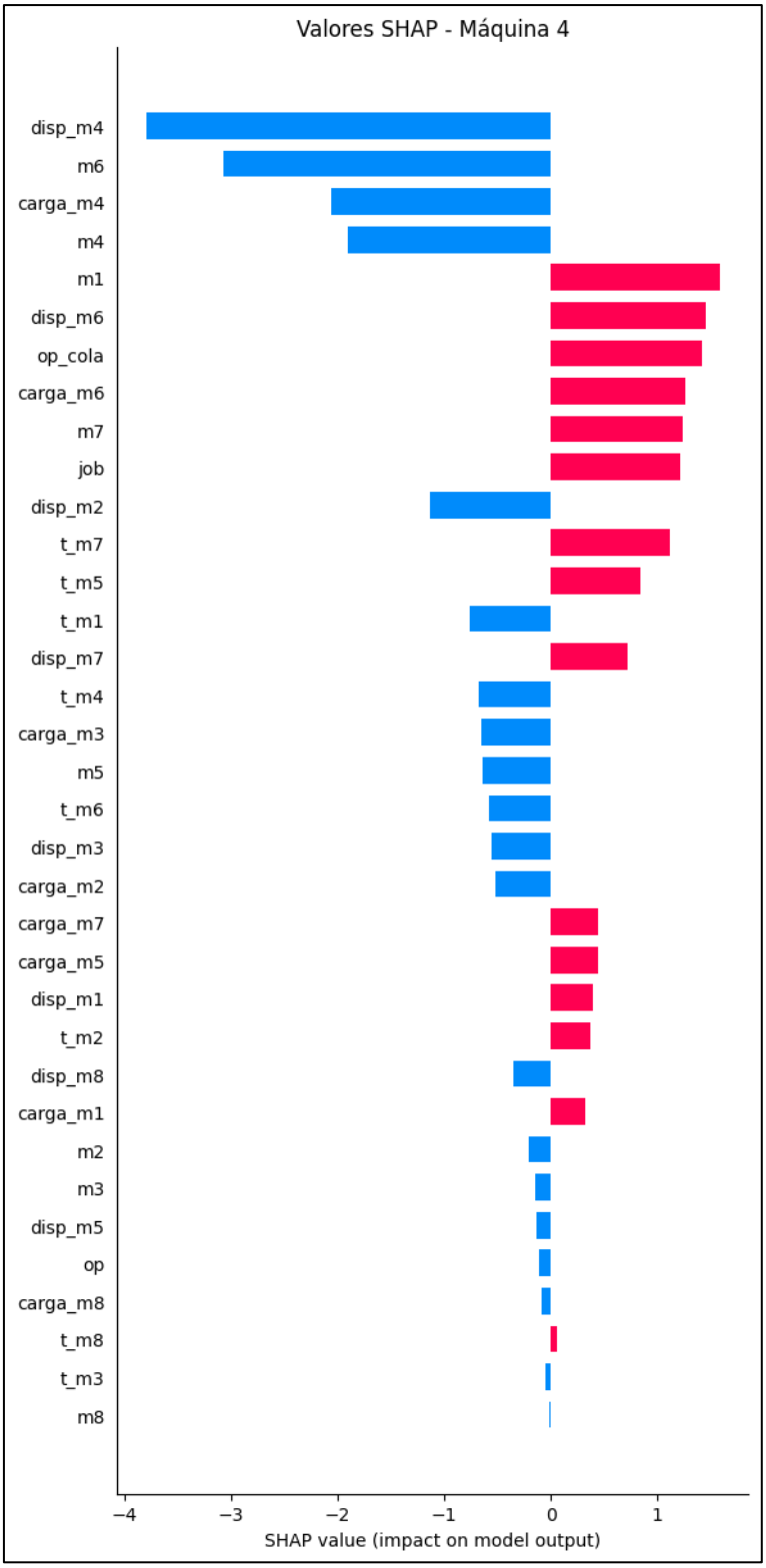

Figura 3.8. Valores SHAP asociados a la máquina 4 para la instancia 1.

Fuente: Elaboración propia.

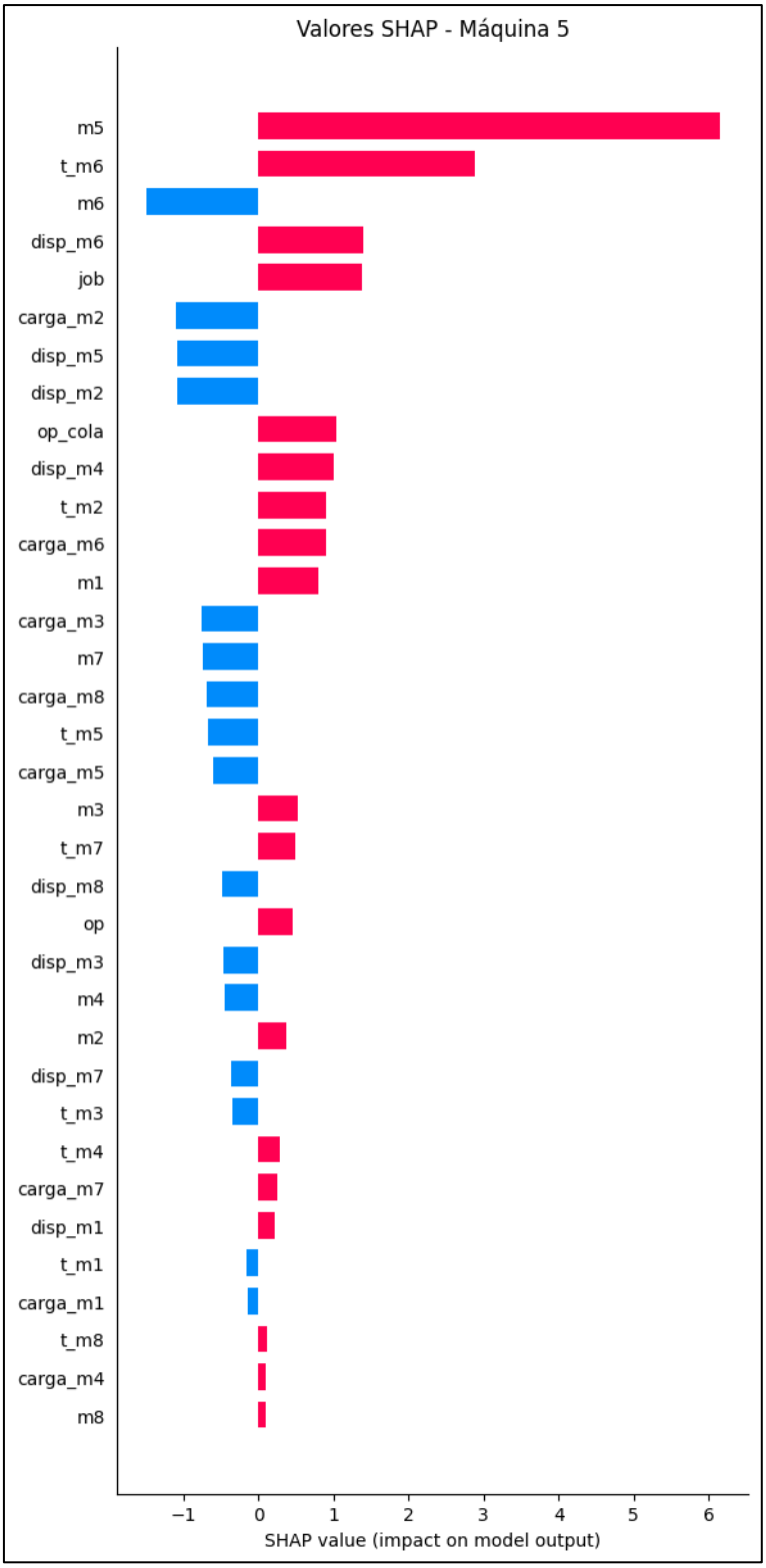

Figura 3.9. Valores SHAP asociados a la máquina 5 para la instancia 1.

Fuente: Elaboración propia.

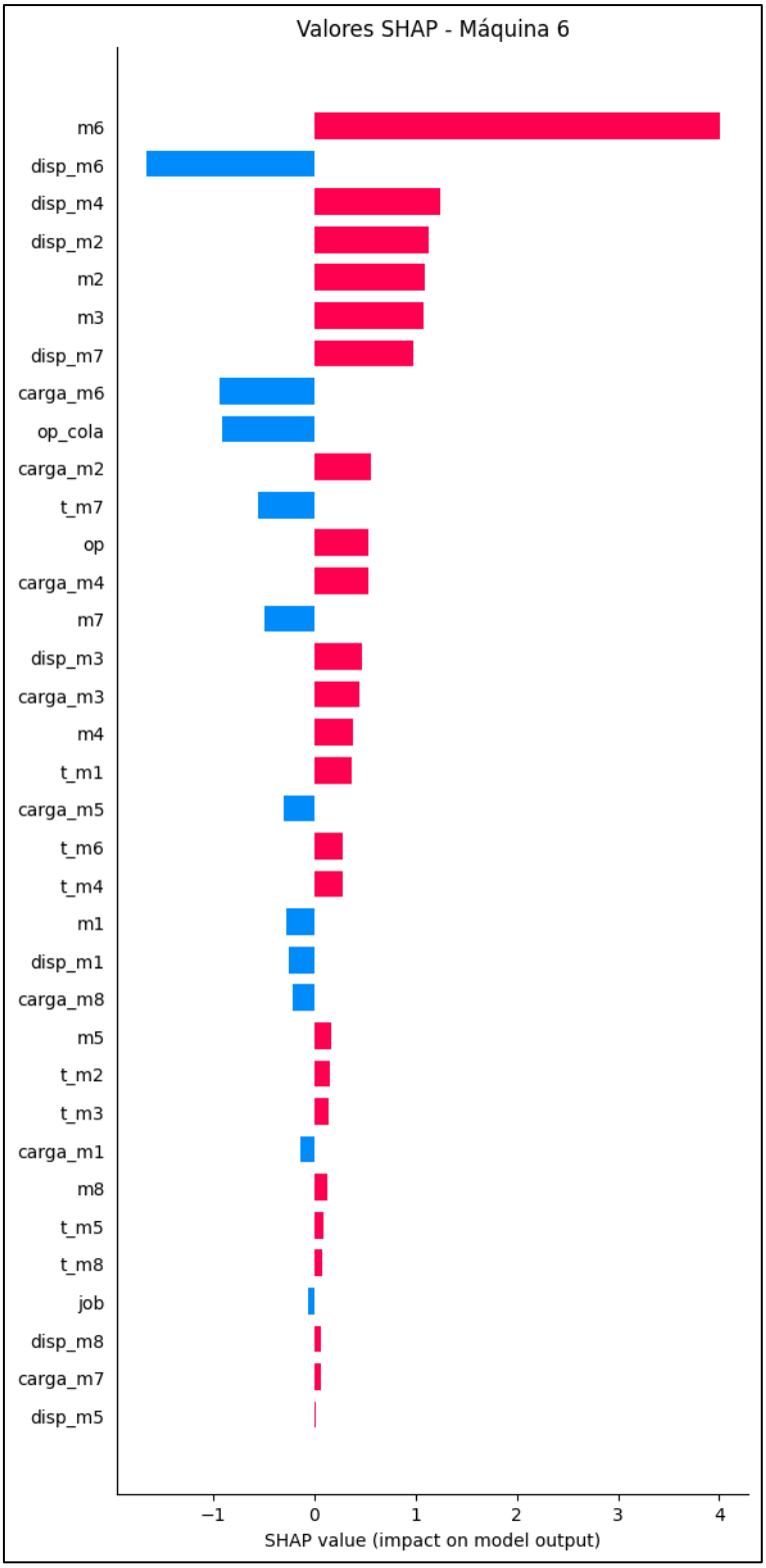

Figura 3.10. Valores SHAP asociados a la máquina 6 para la instancia 1.

Fuente: Elaboración propia.

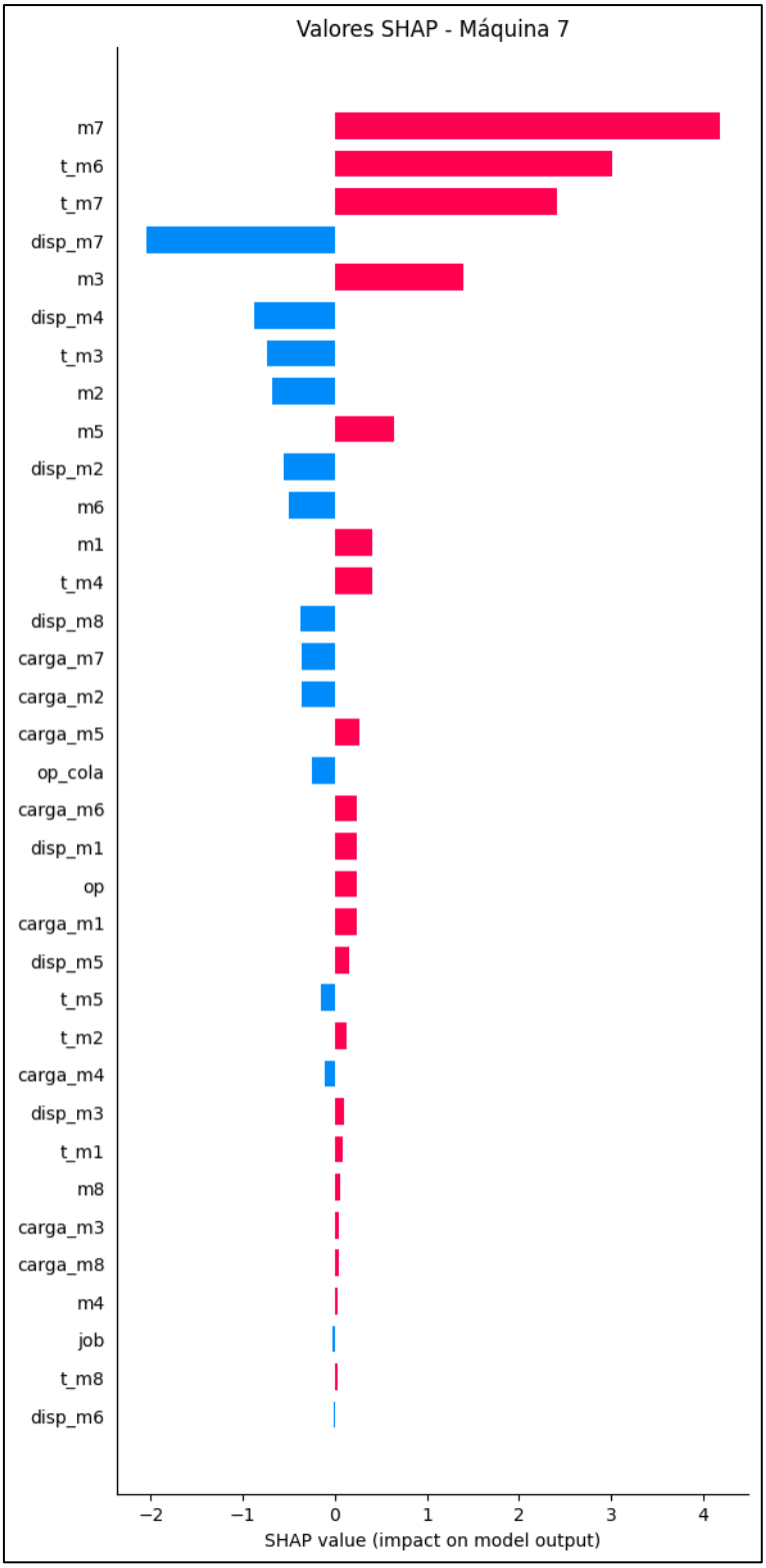

Figura 3.11. Valores SHAP asociados a la máquina 7 para la instancia 1.

Fuente: Elaboración propia.

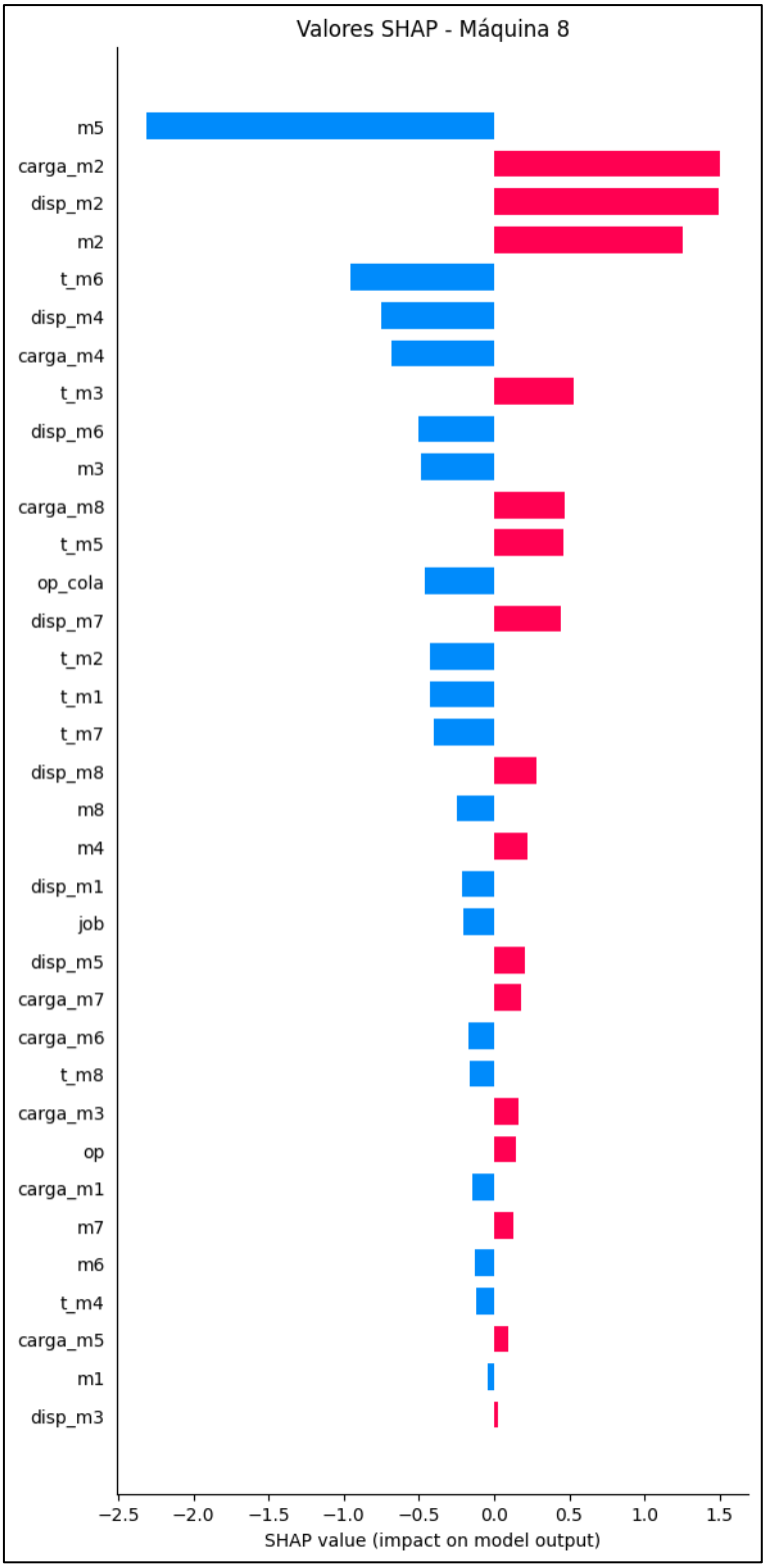

Figura 3.12. Valores SHAP asociados a la máquina 8 para la instancia 1.

Fuente: Elaboración propia.

Los resultados correspondientes a las 9 instancias restantes se incluyen en el Anexo 6 de este documento. Para cada una de ellas, se presenta el gráfico de probabilidad de asignación a cada máquina y un gráfico con los valores SHAP detallados exclusivamente para la máquina con mayor probabilidad, es decir, aquella que fue predicha por el modelo. Esta selección permite concentrar el análisis interpretativo en los casos más relevantes, facilitando la comparación entre las distintas predicciones del modelo y la coherencia lógica existente en contextos de asignación de máquinas.

A partir del análisis de los valores SHAP obtenidos para las distintas instancias, es posible identificar ciertos patrones en el comportamiento del modelo de red neuronal al momento de asignar máquinas a operaciones. En primer lugar, se observó que el modelo solo considera aquellas máquinas que están habilitadas para llevar a cabo una operación, es decir, aquellas que forman parte del conjunto de máquinas disponibles (m_i). Esto se refleja en la distribución de probabilidades de asignación, donde solo las máquinas habilitadas presentan una mayor probabilidad de ser escogidas, lo que evidencia que el modelo ha aprendido a descartar de manera efectiva opciones inviables.

En segundo lugar, el modelo tiende a favorecer máquinas con una menor carga de trabajo. Las variables relacionadas a la carga actual de cada máquina ($carga_{m_i}$) muestran valores SHAP positivos cuando estas presentan una carga relativamente baja en comparación con otras máquinas, lo que aumenta considerablemente la probabilidad de ser seleccionada. Este comportamiento resulta coherente con la lógica de optimización del makespan, ya que asignar operaciones a máquinas menos saturadas puede tender a mejorar la eficiencia global del sistema.

En tercer lugar, variables como la disponibilidad de la máquina ($disp_{m_i}$) también muestran una influencia relevante sobre la predicción del modelo. En contextos donde existe una alta congestión, la disponibilidad operativa parece ser un criterio importante en la asignación de las máquinas.

Por último, variables como el número de operaciones en cola (op_{cola}) o la posición de la operación dentro del trabajo (op) también aportan cierta influencia, aunque en menor medida. En general, este tipo de variables tienden a reforzar decisiones que buscan balancear la carga del sistema y evitar cuellos de botella.

De este modo, los resultados obtenidos sugieren que el modelo de redes neuronales ha sido capaz de aprender una lógica alineada a ciertos criterios de eficiencia operacional, como evitar asignaciones a máquinas no habilitadas, priorizar aquellas con menor carga o disponibilidad, y tener en cuenta el

contexto del flujo de trabajo. Esto refuerza la validez del modelo como una herramienta de apoyo en la toma de decisiones y demuestra que, a pesar de poseer un carácter de caja negra, es posible extraer la lógica con la opera mediante técnicas de explicabilidad como SHAP.

4. Discusión y conclusiones

El presente estudio abordó el problema de programación flexible de talleres (FJSSP) integrando técnicas de optimización exacta, aprendizaje automático e inteligencia artificial explicable. Para tal efecto, se desarrolló un modelo híbrido que combina una red neuronal artificial con un algoritmo genético para predecir asignaciones de máquinas y optimizar la secuenciación de operaciones. También, se aplicó la metodología SHAP como herramienta de explicabilidad post hoc, con la finalidad de evaluar la lógica interna del modelo de red neuronal y aportar transparencia al proceso de toma de decisiones, lo que responde a los lineamientos emergentes de la Industria 5.0, donde la eficiencia operativa debe ir acompañada de confianza, auditabilidad, comprensión y colaboración humano-máquina.

En virtud de los resultados obtenidos, es posible afirmar que el modelo híbrido propuesto es capaz de generar soluciones factibles, de buena calidad y en tiempos de cómputo significativamente inferiores a los requeridos por el modelo MILP. Si bien los makespan obtenidos no siempre igualaron los resultados óptimos, se demostró que, en situaciones de mayor complejidad, donde el solver exacto no entrega mejoras dentro del tiempo asignado, el modelo híbrido es capaz de presentar una alternativa viable, entregando soluciones en pocos segundos, lo que valida su utilidad como herramienta de apoyo en contextos donde la rapidez en la toma de decisiones es crítica.

En cuanto a la interpretabilidad del modelo, la integración de SHAP permitió analizar en detalle la influencia de cada variable de entrada sobre la decisión del modelo. Los resultados obtenidos mostraron que la red neuronal aprendió a identificar patrones coherentes priorizando las máquinas habilitadas para la operación y seleccionando aquellas con una menor carga de trabajo y buena disponibilidad. Estos resultados refuerzan la confianza del modelo y dejan en evidencia que, a pesar de la naturaleza de caja negra del modelo, es posible extraer información relevante sobre su funcionamiento interno mediante técnicas XAI.

Cabe destacar que este estudio también presenta ciertas limitaciones. En primer lugar, el tamaño relativamente reducido del dataset utilizado puede reducir la capacidad predictiva del modelo, generalizando su comportamiento ante instancias con distintas estructuras en cuanto al número de trabajos, operaciones o niveles de flexibilidad. Además, el entrenamiento de la red neuronal se realizó sobre instancias específicas, lo que impide asegurar que el modelo mantenga su precisión en problemas nuevos no incluidos en la etapa de entrenamiento.

En cuanto a las proyecciones futuras, se identifican múltiples líneas de desarrollo. Una primera mejora consiste en ampliar el dataset con una mayor cantidad de instancias y decisiones de asignación, permitiendo entrenar modelos más robustos y generalizables. Asimismo, se podría explorar el uso d arquitecturas más avanzadas como redes neuronales convolucionales (CNN) que podrían capturar relaciones más profundas entre variables. Finalmente, sería de gran valor validar este modelo en entornos reales o simulaciones industriales más complejas, integrando restricciones adicionales como tiempos de preparación, mantenimiento predictivo o una doble flexibilidad mediante grupos de trabajadores.

En síntesis, el estudio realizado presenta un aporte metodológico y conceptual al área de la planificación de la producción en la nueva era de la Industria 5.0. La metodología propuesta para abordar el FJSSP no solo busca eficiencia en los resultados, sino también transparencia en las decisiones tomadas por el modelo de machine learning. La incorporación de técnicas XAI como SHAP permite avanzar hacia sistemas inteligentes más confiables, comprensibles y alineados con el rol activo que deben mantener los operadores humanos. Bajo este contexto, se abren nuevas oportunidades para desarrollar nuevos modelos que no solo sean capaces de resolver problemas complejos, sino que puedan construir puentes entre la inteligencia artificial y la toma de decisiones humanas.

5. Glosario

1. Adam: *Adaptive Moment Estimation*, optimizador utilizado en el entrenamiento de redes neuronales que combina las ventajas de los métodos AdaGrad y RMSProp.

2. AG: *Algoritmo Genético*, técnica de búsqueda heurística inspirada en la evolución natural, utilizada para encontrar soluciones aproximadas a problemas de optimización.

3. ANN: *Artificial Neural Network*, modelo computacional inspirado en el funcionamiento del cerebro humano, utilizado para resolver tareas de clasificación, regresión y predicción.

4. CNN: *Convolutional Neural Network*, tipo especializado de red neuronal artificial diseñada principalmente para el procesamiento de datos con estructura de grilla, como imágenes.

5. CPS: *Cyber Physical Systems*, sistemas que integran componentes físicos y digitales, interactuando mediante sensores, actuadores y software para monitorear y controlar procesos.

6. D-HCA: *Dynamic Hybrid Control Architecture*, arquitectura de control híbrida y dinámica utilizada para la gestión adaptable de sistemas complejos en entornos industriales.

7. DFJSSP: *Double Flexible Job Shop Scheduling Problem*. Variante del FJSSP en la que existe doble flexibilidad: en la asignación de máquinas y en la secuencia de rutas posibles para cada operación, lo que incrementa significativamente la complejidad del problema.

8. DJSSP: *Dynamic Flexible Job Shop Scheduling Problem*, variante del FJSSP que incorpora cambios dinámicos en el sistema, como la llegada inesperada de trabajos o la falla de máquinas.

9. DRL: *Deep Reinforcement Learning*, técnica de aprendizaje automático que combina aprendizaje profundo con aprendizaje por refuerzo para tomar decisiones secuenciales.

10. FJSSP: *Flexible Job Shop Scheduling Problem*, problema de planificación de producción donde cada operación puede ser ejecutada en más de una máquina, permitiendo rutas alternativas.

11. HGNN: *Heterogeneous Graph Neural Network*, tipo de red neuronal que opera sobre grafos con múltiples tipos de nodos y relaciones, útil para modelar estructuras complejas.

12. IA: *Inteligencia Artificial*, campo de estudio enfocado en desarrollar sistemas capaces de realizar tareas que requieren inteligencia humana, como el aprendizaje, razonamiento y percepción.

13. IoT: *Internet of Things*, red de objetos físicos interconectados a través de Internet que recopilan y comparten datos en tiempo real.

14. JSSP: *Job Shop Scheduling Problem*, problema clásico de programación en el cual un conjunto de trabajos debe ser procesado en un orden específico por diferentes máquinas.

15. MILP: *Mixed Integer Linear Programming*, técnica de optimización matemática que combina variables enteras y continuas sujetas a restricciones lineales.

16. MIP: *Mixed Integer Programming*, técnica de optimización matemática en la que algunas variables son enteras y otras continuas, utilizada para resolver problemas complejos de planificación y asignación.

17. MLP: *Multilayer Perceptron*, arquitectura clásica de red neuronal con una o más capas ocultas totalmente conectadas, utilizada para resolver problemas supervisados.

18. NHGA: *Novel Hybrid Genetic Algorithm*, variante avanzada de algoritmos genéticos que combina operadores tradicionales con estrategias adicionales para mejorar la calidad y eficiencia de búsqueda de soluciones.

19. ReLU: *Rectified Linear Unit*, función de activación ampliamente utilizada en redes neuronales que introduce no linealidad al permitir que solo los valores positivos pasen.

20. SHAP: *SHapley Additive exPlanations*, técnica de interpretabilidad de modelos de machine learning basada en teoría de juegos, que atribuye a cada característica su contribución individual a la predicción.

21. XAI: *Explainable Artificial Intelligence*, conjunto de técnicas y métodos que buscan hacer los modelos de inteligencia artificial más comprensibles para los humanos.

6. Referencias

Aribi, D., Driss, O. B., & Haouzi, H. B. E. (2023). A real-time double flexible job shop scheduling problem under Industry 5.0. *2023 IEEE Afro-Mediterranean Conference on Artificial Intelligence (AMCAI)*, Hammamet, Tunisia, 1-7. https://doi.org/10.1109/AMCAI59331.2023.10431523

Behnke, D., & Geiger, M. J. (2012). Test instances for the flexible job shop scheduling problem with work centers (Research Report RR-12-01-01). Helmut-Schmidt-Universität, Universität der Bundeswehr Hamburg, Lehrstuhl für Betriebswirtschaftslehre, insbesondere Logistik-Management. 10.24405/436

Destouet, C., Tlahig, H., Bettayeb, B., & Mazari, B. (2023). Flexible Job Shop Scheduling Problem Under Industry 5.0: A Survey on Human Reintegration, Environmental Consideration and Resilience Improvement. *Journal of Manufacturing Systems*, 67, 155-173. https://doi.org/10.1016/j.jmsy.2023.01.004

Destouet, C., Tlahig, H., Bettayeb, B., & Mazari, B. (2024). Multi-objective sustainable flexible job shop scheduling problem: Balancing economic, ecological, and social criteria. *Computers & Industrial Engineering, 195*, 110419. https://doi.org/10.1016/j.cie.2024.110419

Erlenbusch, F., & Stricker, N. (2025). Explainable reinforcement learning in job-shop scheduling: A systematic literature review. *Procedia CIRP, 132*, 25–30. https://doi.org/10.1016/j.procir.2025.01.005

European Commission: Directorate-General for Research and Innovation, Renda, A., Schwaag Serger, S., Tataj, D., Morlet, A., Isaksson, D., Martins, F., Mir Roca, M., Hidalgo, C., Huang, A., Dixson-Declève, S., Balland, P.-A., Bria, F., Charveriat, C., Dunlop, K., & Giovannini, E. (2021). Industry 5.0, a transformative vision for Europe: Governing systemic transformations towards a sustainable industry. *Publications Office of the European Union.* https://doi.org/10.2777/17322

Fattahi, P., Saidi Mehrabad, M. & Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing, 18, 331-342*. https://doi.org/10.1007/s10845-007-0026-8

Fonseca, D. J., & Navaresse, D. (2002). Artificial neural networks for job shop simulation. *Advanced Engineering Informatics, 16*(4), 241–246. https://doi.org/10.1016/S1474-0346(03)00005-3

Gong, G., Deng, Q., Gong, X., Liu, W., & Ren, Q. (2018). A new double flexible job-shop scheduling problem integrating processing time, green production, and human factor indicators. *Journal of Cleaner Production, 174,* 560-576. https://doi.org/10.1016/j.jclepro.2017.10.188

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. https://www.deeplearningbook.org/

Herrera, F. (2025). Reflections and attentiveness on eXplainable Artificial Intelligence (XAI). The journey ahead from criticisms to human–AI collaboration. *Information Fusion, 121*, 103133. https://doi.org/10.1016/j.inffus.2025.103133

Jimenez, J.-F., Bekrar, A., Zambrano-Rey, G., Trentesaux, D., & Leitão, P. (2017). Pollux: A dynamic hybrid control architecture for flexible job shop systems. *International Journal of Production Research, 55(15*), 4229-4247. https://doi.org/10.1080/00207543.2016.1218087

Leng, J., Sha, W., Wang, B., Zheng, P., Zhuang, C., Liu, Q., Wuest, T., Mourtzis, D., & Wang, L. (2022). Industry 5.0: Prospect and Retrospect. *Journal of Manufacturing Systems*, 65, 279-295. https://doi.org/10.1016/j.jmsy.2022.09.017

Liu, C.-L., Weng, P.-H., & Tseng, C.-J. (2025). Harnessing heterogeneous graph neural networks for Dynamic Job-Shop Scheduling Problem solutions. *Computers & Industrial Engineering, 203*, 111060. https://doi.org/10.1016/j.cie.2025.111060

Ma, L., Zhong, R. Y., Yuan, M., Ding, K., Thürer, M., Pan, Y., Qu, T., & Huang, G. Q. (2025). A human-centric order release method based on workload control in high-variety make-to-order shops towards Industry 5.0. *Robotics and Computer-Integrated Manufacturing, 94*, 102946. https://doi.org/10.1016/j.rcim.2024.102946

Morinaga, E., Tang, X., Iwamura, K., & Hirabayashi, N. (2023). An improved method of job shop scheduling using machine learning and mathematical optimization. *Procedia Computer Science, 217,* 1479-1486. https://doi.org/10.1016/j.procs.2022.12.347

Morinaga, E., Tang, X., Iwamura, K., & Hirabayashi, N. (2024). Improvement of job shop scheduling method based on mathematical optimization and machine learning. *Procedia Computer Science, 232,* 871-879. https://doi.org/10.1016/j.procs.2024.01.087

Murtaza, A. A., Saher, A., Zafar, M. H., Moosavi, S. K. R., Aftab, M. F., & Sanfilippo, F. (2024). Paradigm shift for predictive maintenance and condition monitoring from Industry 4.0 to Industry 5.0: A systematic review, challenges and case study. *Results in Engineering, 24*, 102935. https://doi.org/10.1016/j.rineng.2024.102935

Nikiforidis, K., Kyrtsoglou, A., Vafeiadis, T., Kotsiopoulos, T., Nizamis, A., Ioannidis, D., Votis, K., Tzovaras, D., & Sarigiannidis, P. (2025). Enhancing transparency and trust in AI-powered manufacturing: A survey of explainable AI (XAI) applications in smart manufacturing in the era of Industry 4.0/5.0. *ICT Express, 11*(1), 135–148. https://doi.org/10.1016/j.icte.2024.12.001

Rieger, T., Schindler, H., Onnasch, L., & Roesler, E. (2025). Explaining AI weaknesses improves human–AI performance in a dynamic control task. *International Journal of Human-Computer Studies, 199*, 103505. https://doi.org/10.1016/j.ijhcs.2025.103505

Scikit-learn Developers. (n.d.). *sklearn.preprocessing.StandardScaler*. Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

Sharma, R., Nibedita, B., & Gupta, H. (2025). Innovating tomorrow: Industry 5.0's role in shaping the workforce and socio-economic development in the sustainable energy transition era. *Journal of Environmental Management, 381*, 125170. https://doi.org/10.1016/j.jenvman.2025.125170

Smit, I. G., Zhou, J., Reijnen, R., Wu, Y., Chen, J., Zhang, C., Bukhsh, Z., Zhang, Y., & Nuijten, W. (2025). Graph neural networks for job shop scheduling problems: A survey. *Computers & Operations Research, 176*, 106914. https://doi.org/10.1016/j.cor.2024.106914

Ukwaththa, J., Herath, S., & Meddage, D. P. P. (2024). A review of machine learning (ML) and explainable artificial intelligence (XAI) methods in additive manufacturing (3D Printing). *Materials Today Communications, 41*, 110294. https://doi.org/10.1016/j.mtcomm.2024.110294

Yahata, E., dos Santos, P. D., de Souza Pires, M. M., Suyama, R., & Simoes, P. W. (2025). Neural Networks and Explainable Artificial Intelligence for Breast Cancer Prediction and Classification. *Procedia Computer Science, 256*, 1159–1166. https://doi.org/10.1016/j.procs.2025.02.224

7. Anexos

Anexo 1. Instancias de FJSSP por Fattahi et al.

La siguiente tabla resume las características principales de las 20 instancias del FJSSP propuestas originalmente por Fattahi et al. (2007) y reestructuradas por Behnke & Geiger (2012).

Tabla 7.1. Instancias de FJSSP por Fattahi et al.

Instancia	n	m	Flexibilidad	LB	UB	CP
SFJS1	2	2	Total	66	66	66
SFJS2	2	2	Parcial	107	107	107
SFJS3	3	2	Parcial	212	221	221
SFJS4	3	2	Parcial	331	355	355
SFJS5	3	2	Total	107	119	119
SFJS6	3	2	Parcial	310	320	320
SFJS7	3	5	Total	397	397	397
SFJS8	3	4	Total	216	253	253
SFJS9	3	3	Total	210	210	210
SFJS10	4	5	Parcial	427	516	516
MFJS1	5	6	Parcial	403	468	468
MFJS2	5	7	Parcial	396	446	446
MFJS3	6	7	Parcial	396	466	466
MFJS4	7	7	Parcial	496	554	554
MFJS5	7	7	Parcial	414	514	514
MFJS6	8	7	Parcial	614	635	634
MFJS7	8	7	Parcial	764	879	931
MFJS8	9	8	Parcial	764	884	884
MFJS9	11	8	Parcial	807	1088	1070
MFJS10	12	8	Parcial	944	1267	1208

Fuente: Behnke & Geiger (2012).

Cada fila de la tabla representa una instancia individual y contiene la siguiente información:

- **Instancia:** Identificador único de la instancia, donde el prefijo “S” indica instancias pequeñas (Small FJSP) y “M” indica instancias medianas (Medium FJSP).

- $\boldsymbol{n}$**:** Número total de trabajos.
- $\boldsymbol{m}$**:** Número total de máquinas.
- **Flexibilidad:** Nivel de flexibilidad definido para las operaciones. Una flexibilidad parcial indica que solo algunas operaciones tienen múltiples alternativas, mientras que una flexibilidad total indica que todas las operaciones pueden ser procesadas por más de una máquina.
- **LB (Lower Bound):** Representa la cota inferior del makespan. Es el tiempo mínimo teóricamente posible para completar todos los trabajos.
- **UB (Upper Bound):** Representa la cota superior del makespan y es obtenido generalmente de métodos exactos o heurísticos.
- **CP (Computed Performance):** Valor del makespan obtenido por el algoritmo propuesto.

Anexo 2. Estructura de cada instancia propuesta por Fattahi et al.

Las instancias utilizadas en este estudio están estructuradas por Behnke & Geiger (2012) en base a un estándar comúnmente adoptado en la literatura para representar el FJSSP. La estructura de los datos para cada instancia es la siguiente:

La primera fila contiene al menos dos números enteros, donde el primero indica el número total de trabajos y el segundo el número total de máquinas. Se puede presentar un posible tercer valor, el cual corresponde al número promedio de máquinas por operación, sin embargo, no es esencial para la interpretación del formato.

Para las filas siguientes, cada una de ellas representa un trabajo individual, donde el primer número indica la cantidad de operaciones que componen el trabajo. Luego, para cada operación se incluye primero la cantidad de máquinas que pueden procesar dicha operación ($k \geq 1$), seguido de k pares de valores, donde cada par contiene el identificador de la máquina y el tiempo de procesamiento correspondiente a la operación en dicha máquina.

A continuación, se presentan las 20 instancias utilizadas en el formato previamente descrito.

Instancia SFJS1:

2 2 2

2 2 1 25 2 37 2 1 32 2 24

2 2 1 45 2 65 2 1 21 2 65

Instancia SFJS2:

2 2 1.5

2 1 1 43 2 1 64 2 71

2 2 1 21 2 35 1 2 43

Instancia SFJS3:

3 2 1.7

2 1 1 43 2 1 87 2 95

2 2 1 63 2 53 1 2 73

2 2 1 125 2 135 2 1 43 2 61

Instancia SFJS4:

3 2 1.7

2 2 1 54 2 63 2 1 87 2 95

2 1 2 120 1 2 152

2 2 1 125 2 135 2 1 143 2 124

Instancia SFJS5:

3 2 2

2 2 1 43 2 36 2 1 64 2 71

2 2 1 34 2 53 2 1 36 2 21

2 2 1 21 2 35 2 1 43 2 37

Instancia SFJS6:

3 3 1.6

3 1 1 17 2 1 40 2 130 2 2 50 3 60

3 1 1 30 2 1 150 2 160 1 3 70

3 2 1 50 2 60 2 2 170 3 180 2 2 90 3 100

Instancia SFJS7:

3 5 2

3 2 1 117 2 125 2 4 140 2 130 2 4 150 5 160

3 2 1 214 3 150 2 3 55 2 66 2 5 78 3 65

3 2 1 87 2 62 2 4 70 3 80 2 4 190 5 100

Instancia SFJS8:

3 4 2

3 2 1 17 2 25 2 4 40 2 30 2 4 150 3 160

3 2 1 30 3 50 2 4 55 2 66 2 4 78 3 65

3 2 1 56 2 62 2 2 70 3 80 2 4 90 3 100

Instancia SFJS9:

3 3 2

3 2 1 17 2 25 2 1 40 2 30 2 2 50 3 60

3 2 1 30 3 50 2 1 50 2 60 2 2 70 3 60

3 2 1 50 2 60 2 2 70 3 80 2 2 90 3 100

Instancia SFJS10:

4 5 1.7

3 1 1 147 2 4 140 2 130 2 4 150 5 160

3 2 1 214 3 150 2 3 87 2 66 1 5 178

3 2 1 87 2 62 1 3 180 2 4 190 5 100

3 2 1 87 2 65 1 5 173 2 4 145 3 136

Instancia MFJS1:

5 6 2.2

3 3 1 147 2 123 3 145 2 4 140 2 130 2 4 150 5 160

3 2 1 214 3 150 2 3 87 2 66 2 5 178 6 95

3 2 1 87 2 62 2 3 180 4 105 3 4 190 5 100 6 153

3 2 1 87 2 65 1 5 173 2 4 145 6 136

3 3 2 123 3 145 1 128 3 3 86 4 65 5 47 2 5 110 6 85

Instancia MFJS2:

5 7 2.7

3 3 1 147 2 123 3 145 3 4 140 2 130 1 123 3 4 150 5 160 7 200

3 2 1 214 3 150 3 3 87 2 66 4 99 3 5 178 6 95 7 150

3 2 1 87 2 62 3 7 145 3 180 4 105 3 4 190 5 100 6 153

3 2 1 87 2 65 2 3 250 5 173 2 4 145 6 136

3 3 2 123 3 145 1 128 3 3 86 4 65 5 47 2 5 110 6 85

Instancia MFJS3:

6 7 2.7

3 3 1 147 2 123 3 145 3 4 140 2 130 1 123 3 4 150 5 160 7 200

3 2 1 214 3 150 3 3 87 2 66 4 99 3 5 178 6 95 7 150

3 2 1 87 2 62 3 7 145 3 180 4 105 3 4 190 5 100 6 153

3 2 1 87 2 65 2 3 250 5 173 2 4 145 6 136

3 3 2 123 3 145 1 128 3 3 86 4 65 5 47 2 5 110 6 85

3 3 2 145 3 320 4 154 3 4 150 3 123 5 192 3 5 120 6 240 7 180

Instancia MFJS4:

7 7 2.7

3 2 1 247 2 223 3 4 140 2 130 7 123 3 4 150 5 160 7 200

3 2 1 214 3 150 3 3 87 2 66 4 99 3 5 178 6 95 7 150

3 2 1 87 2 62 3 7 145 3 180 4 105 3 4 190 5 100 6 153

3 2 1 87 2 65 2 3 250 5 173 2 4 145 6 136

3 3 2 123 3 145 1 128 3 7 86 4 65 5 47 2 5 110 6 85

3 3 2 145 3 320 4 154 3 4 150 3 123 5 192 3 5 120 6 240 7 180

3 3 3 145 2 210 1 157 3 3 124 4 168 5 154 3 5 145 6 165 7 178

Instancia MFJS5:

7 7 2.6

3 3 1 247 2 223 3 100 3 4 140 2 130 7 123 3 4 150 5 160 7 200

3 2 1 214 3 150 2 3 87 2 66 3 5 178 6 95 7 150

3 2 1 87 2 62 3 7 145 3 180 4 105 3 4 190 5 100 6 153

3 2 1 87 2 65 2 3 250 5 173 2 4 145 6 136

3 3 2 123 3 145 1 128 3 7 86 4 65 5 47 3 5 110 6 85 2 100

3 3 2 145 3 320 4 154 3 4 150 3 123 5 192 3 5 120 6 240 7 180

3 2 3 145 2 157 2 3 124 5 168 3 5 145 6 165 7 178

Instancia MFJS6:

8 7 2.6

3 3 1 247 2 223 3 100 3 4 140 2 130 7 123 3 4 150 5 160 7 200

3 2 1 214 3 150 2 3 87 2 66 3 5 178 6 95 7 150

3 2 1 87 2 62 3 7 145 3 180 4 105 3 4 190 5 100 6 153

3 2 1 87 2 65 2 3 250 5 173 2 4 145 6 136

3 3 2 123 3 145 1 128 3 7 86 4 65 5 47 3 5 110 6 85 2 100

3 3 2 145 3 320 4 154 3 4 150 3 123 5 192 3 5 120 6 240 7 180

3 2 3 145 2 157 2 3 124 5 168 3 5 145 6 165 7 178

3 2 4 245 3 257 2 6 224 5 268 3 5 145 6 165 7 178

Instancia MFJS7:

8 7 2.4

4 3 1 247 2 223 3 100 3 4 140 2 130 7 123 3 4 150 5 160 7 200 2 7 210 6 145

4 2 1 214 3 150 2 3 87 2 66 3 5 178 6 95 7 150 2 5 120 7 150

4 2 1 87 2 62 3 7 145 3 180 4 105 3 4 190 5 100 6 153 2 6 170 7 165

4 2 1 87 2 65 2 3 250 5 173 2 4 145 6 136 2 5 250 6 170

4 3 2 123 3 145 1 128 3 7 86 4 65 5 47 3 5 110 6 85 2 100 2 4 165 6 180

4 3 2 145 3 320 4 154 3 4 150 3 123 5 192 3 5 120 6 240 7 180 2 4 120 7 50

4 2 3 145 2 157 2 3 124 5 168 3 5 145 6 165 7 178 2 7 230 5 140

4 2 4 245 3 257 2 6 224 5 268 3 5 145 6 165 7 178 2 7 150 6 150

Instancia MFJS8:

9 8 2.4

4 3 1 247 2 123 3 100 3 4 140 2 130 7 123 3 4 150 5 160 7 200 2 7 210 8 145

4 2 1 214 3 150 2 3 87 2 66 3 5 178 6 95 7 150 2 5 120 8 150

4 2 1 87 2 62 3 7 145 3 180 4 105 3 4 190 5 100 6 153 2 6 170 8 165

4 2 1 87 2 65 2 3 250 5 173 2 4 145 6 136 2 5 250 6 170

4 3 2 123 3 145 1 128 3 7 86 4 65 5 47 3 5 110 6 85 2 100 2 8 165 6 180

4 3 2 145 3 320 4 154 3 4 150 3 123 5 192 3 5 120 6 240 7 180 2 4 120 7 50

4 2 3 145 2 157 2 3 124 5 168 3 5 145 6 165 7 178 2 8 230 5 140

4 2 4 245 3 257 2 6 224 5 268 3 5 145 6 165 7 178 2 7 150 8 150

4 2 2 150 3 150 2 3 180 4 220 2 5 40 7 50 2 6 150 8 170

Instancia MFJS9:

11 8 2.3

4 3 1 247 2 123 3 100 3 4 140 2 130 7 123 3 4 150 5 160 7 200 2 7 210 8 145

4 2 1 214 3 150 2 3 87 2 66 3 5 178 6 95 7 150 2 5 120 8 150

4 2 1 87 2 62 3 7 145 3 180 4 105 3 4 190 5 100 6 153 2 6 170 8 165

4 2 1 87 2 65 2 3 250 5 173 2 4 145 6 136 2 5 250 6 170

4 3 2 123 3 145 1 128 3 7 86 4 65 5 47 3 5 110 6 85 2 100 2 8 165 6 180

4 3 2 145 3 320 4 154 3 4 150 3 123 5 192 3 5 120 6 240 7 180 2 4 120 7 50

4 2 3 145 2 157 2 3 124 5 168 3 5 145 6 165 7 178 2 8 230 5 140

4 2 4 245 3 257 2 6 224 5 268 3 5 145 6 165 7 178 2 7 150 8 150

4 2 2 150 3 150 2 3 180 4 220 2 5 40 7 50 2 6 150 8 170

4 2 4 245 3 257 2 6 224 5 268 3 5 145 6 165 7 178 2 7 150 8 150

4 2 2 150 3 150 2 3 180 4 220 2 5 40 7 50 2 6 150 8 170

Instancia MFJS10:

12 8 2.3

4 3 1 247 2 223 3 100 3 4 140 2 130 7 123 3 4 150 5 160 7 200 2 7 210 8 145

4 2 1 214 3 150 2 3 87 2 66 3 5 178 6 95 7 150 2 5 120 8 150

4 2 1 87 2 62 3 7 145 3 180 4 105 3 4 190 5 100 6 153 2 6 170 8 165

4 2 1 87 2 65 2 3 250 5 173 2 4 145 6 136 2 5 250 6 170

4 3 2 123 3 145 1 128 3 7 86 4 65 5 47 3 5 110 6 85 2 100 2 8 165 6 180

4 3 2 145 3 320 4 154 3 4 150 3 123 5 192 3 5 120 6 240 7 180 2 4 120 7 50

4 2 3 145 2 157 2 3 124 5 168 3 5 145 6 165 7 178 2 8 230 5 140

4 2 4 245 3 257 2 6 224 5 268 3 5 145 6 165 7 178 2 7 150 8 150

4 2 2 150 3 150 2 3 180 4 220 2 5 40 7 50 2 6 150 8 170

4 2 4 245 3 257 2 6 224 5 268 3 5 145 6 165 7 178 2 7 150 8 150

4 2 2 150 3 150 2 3 180 4 220 2 5 40 7 50 2 6 150 8 170

4 2 3 345 2 357 2 3 224 5 268 3 5 145 6 165 7 178 2 8 230 5 340

Anexo 3. Diagrama de Gantt obtenido mediante MILP

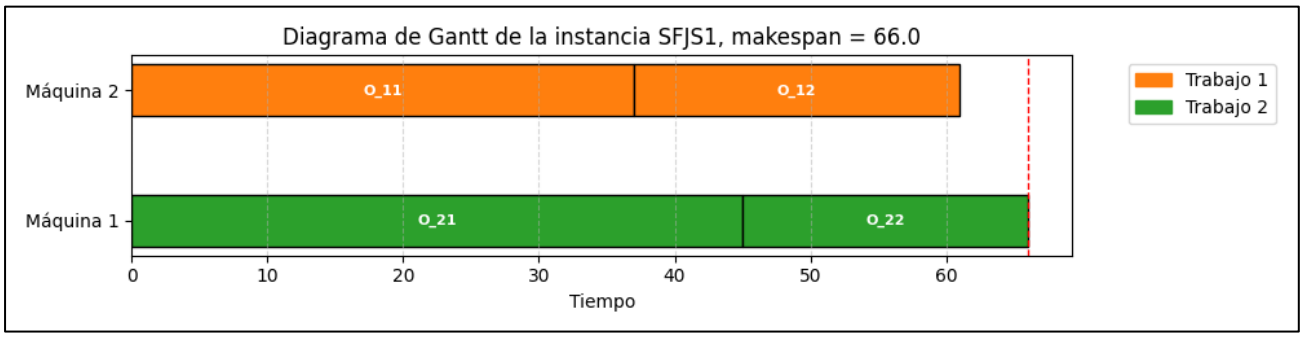

Figura 7.1. Diagrama de la instancia SFJS1 mediante MILP.

Fuente: Elaboración propia.

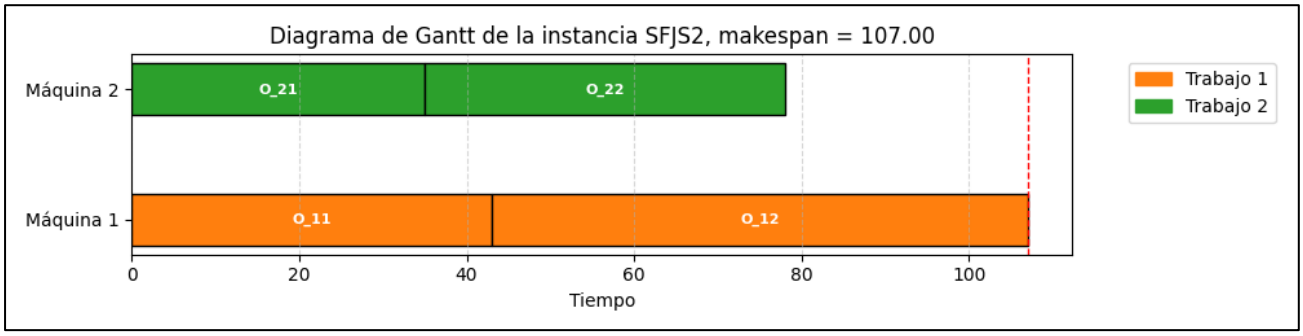

Figura 7.2. Diagrama de la instancia SFJS2 mediante MILP.

Fuente: Elaboración propia.

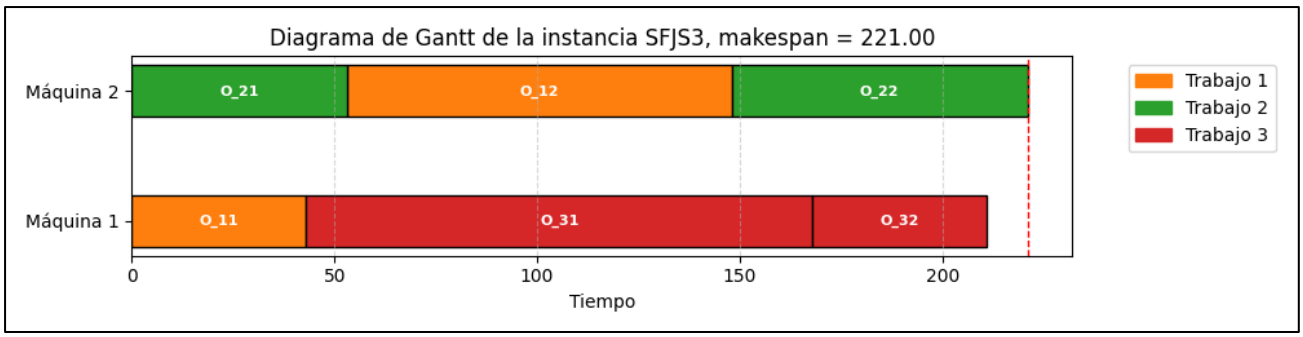

Figura 7.3. Diagrama de la instancia SFJS3 mediante MILP.

Fuente: Elaboración propia.

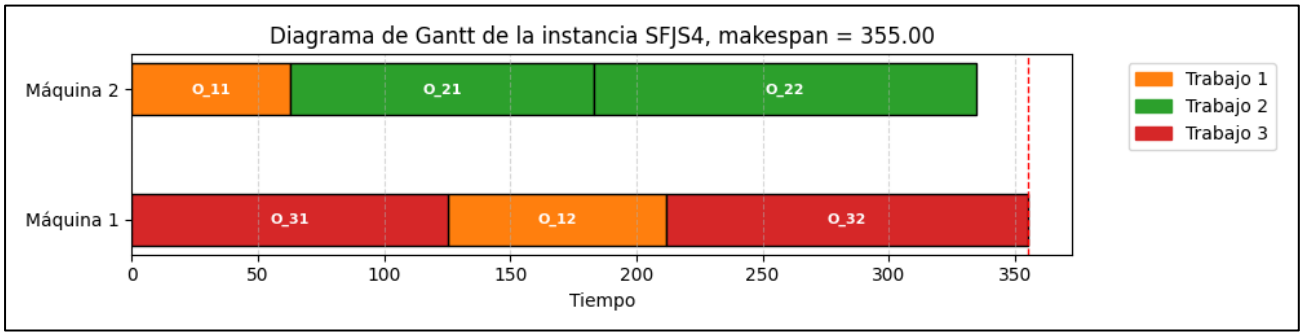

Figura 7.4. Diagrama de la instancia SFJS4 mediante MILP.

Fuente: Elaboración propia.

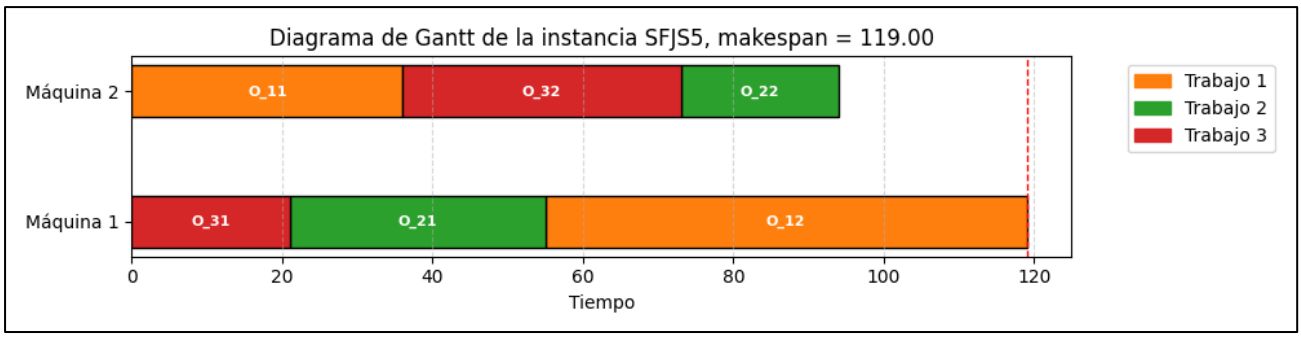

Figura 7.5. Diagrama de la instancia SFJS5 mediante MILP.

Fuente: Elaboración propia.

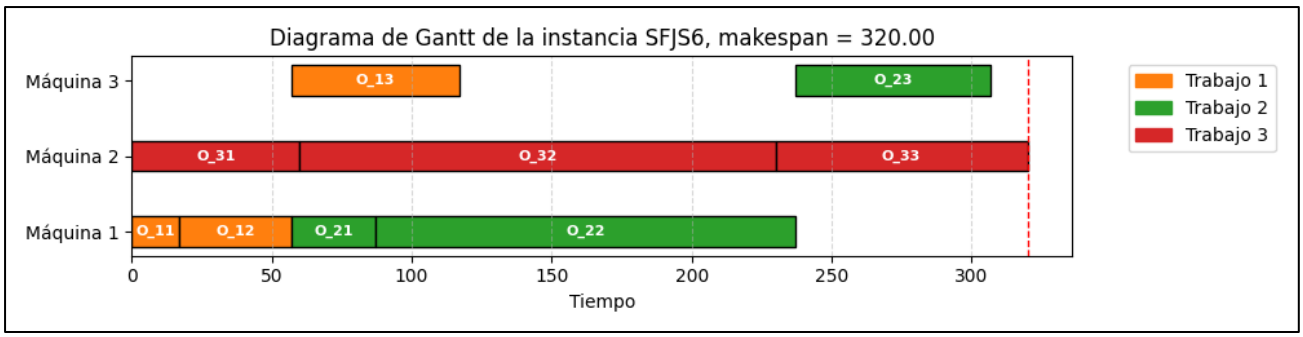

Figura 7.6. Diagrama de la instancia SFJS6 mediante MILP.

Fuente: Elaboración propia.

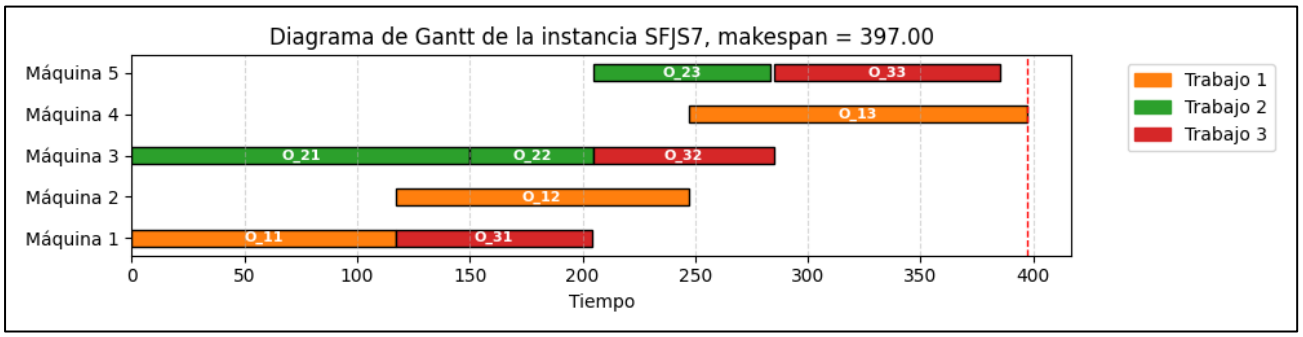

Figura 7.7. Diagrama de la instancia SFJS7 mediante MILP.

Fuente: Elaboración propia.

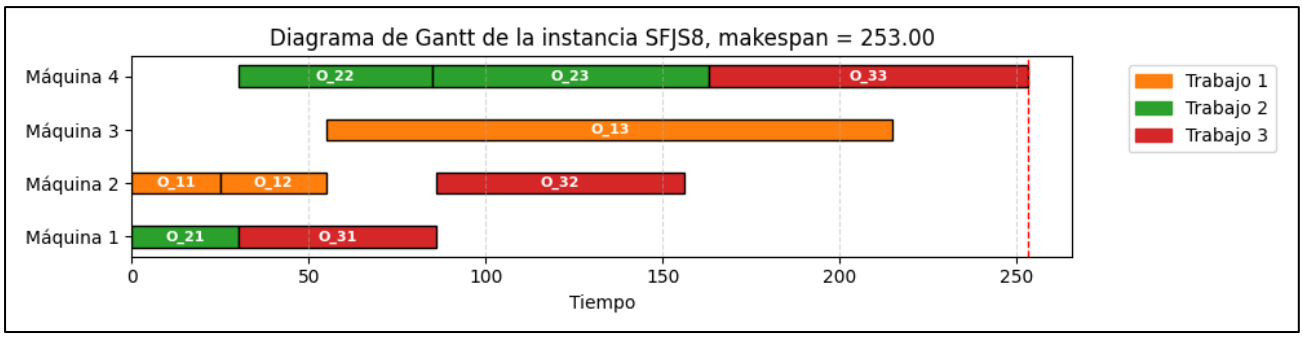

Figura 7.8. Diagrama de la instancia SFJS8 mediante MILP.

Fuente: Elaboración propia.

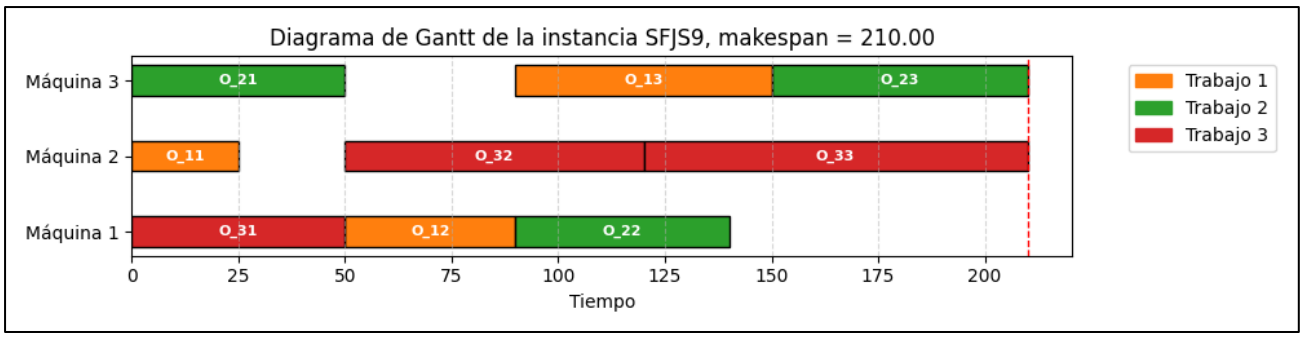

Figura 7.9. Diagrama de la instancia SFJS9 mediante MILP.

Fuente: Elaboración propia.

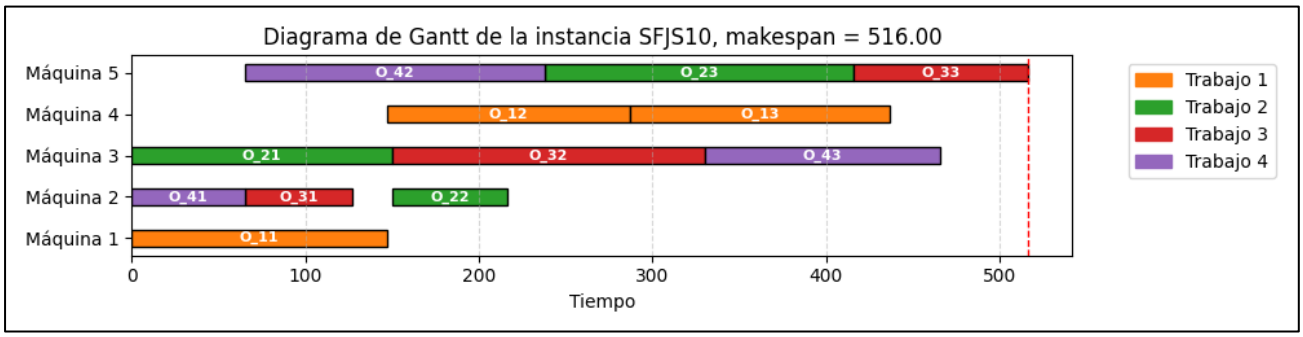

Figura 7.10. Diagrama de la instancia SFJS10 mediante MILP.

Fuente: Elaboración propia.

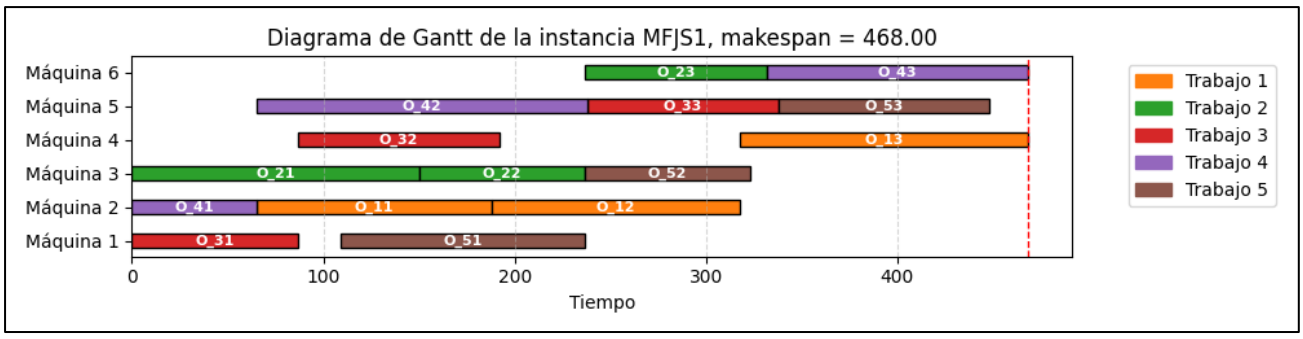

Figura 7.11. Diagrama de la instancia MFJS1 mediante MILP.

Fuente: Elaboración propia.

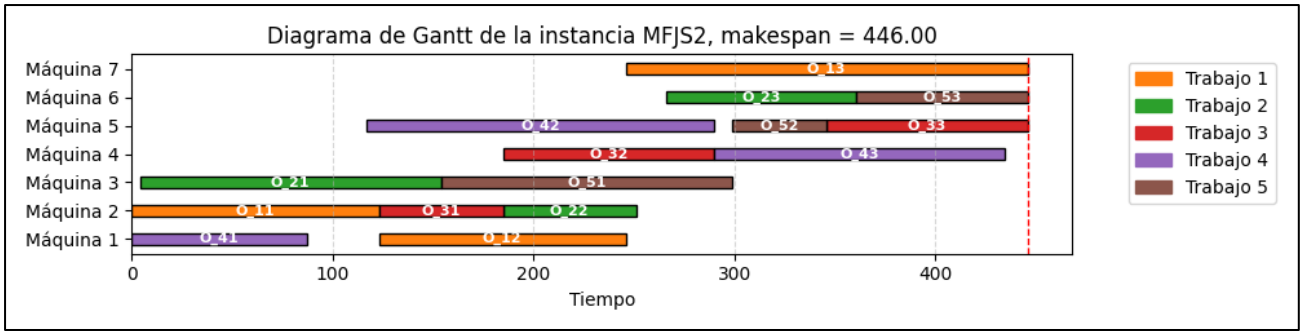

Figura 7.12. Diagrama de la instancia MFJS2 mediante MILP.

Fuente: Elaboración propia.

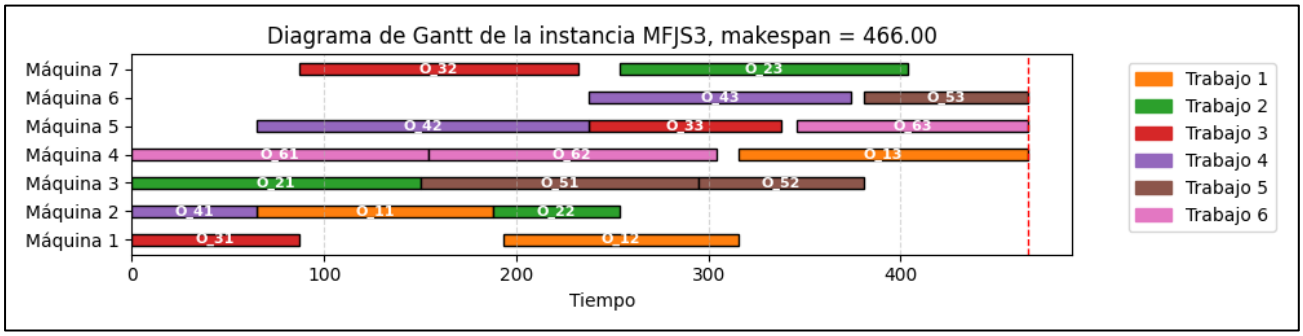

Figura 7.13. Diagrama de la instancia MFJS3 mediante MILP.

Fuente: Elaboración propia.

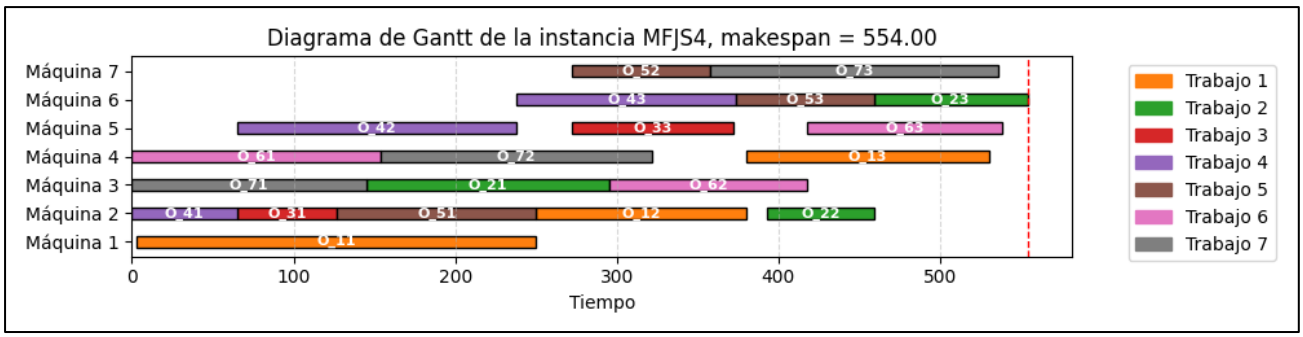

Figura 7.14. Diagrama de la instancia MFJS4 mediante MILP.

Fuente: Elaboración propia.

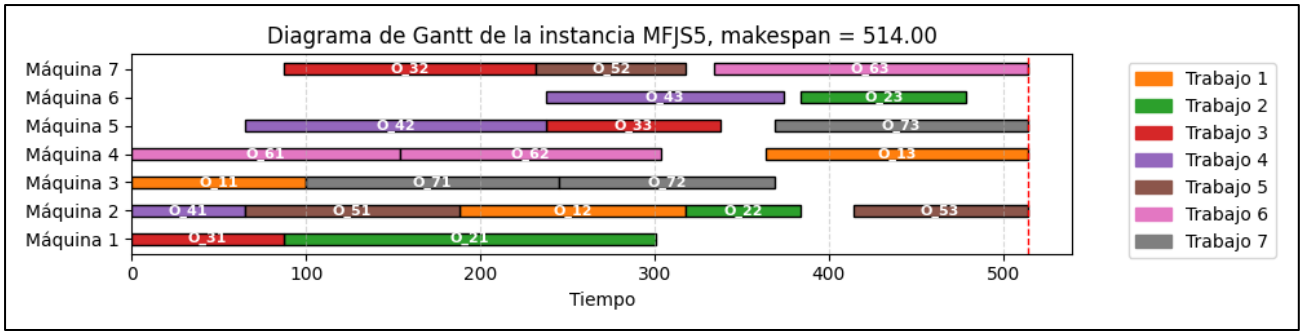

Figura 7.15. Diagrama de la instancia MFJS5 mediante MILP.

Fuente: Elaboración propia.

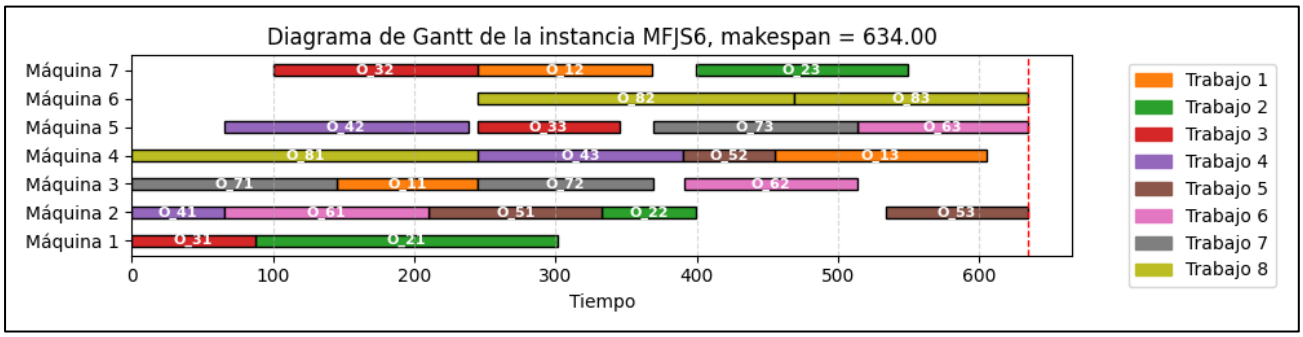

Figura 7.16. Diagrama de la instancia MFJS6 mediante MILP.

Fuente: Elaboración propia.

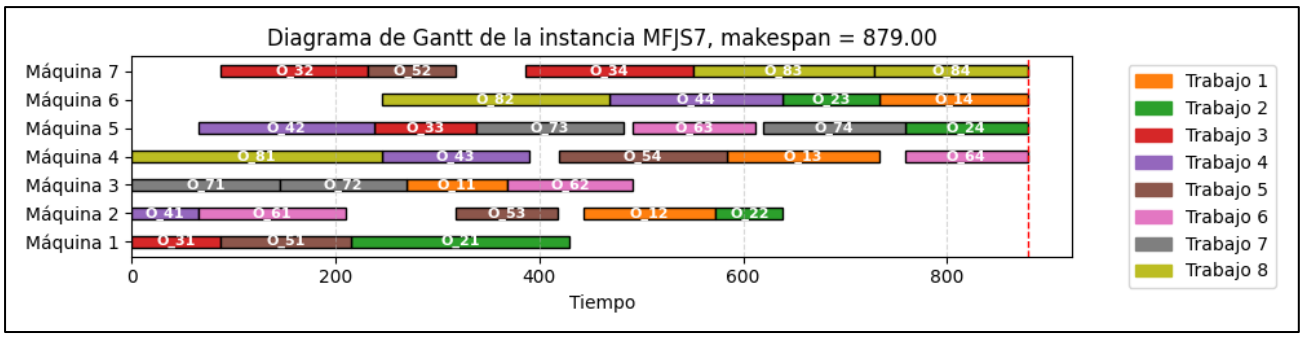

Figura 7.17. Diagrama de la instancia MFJS7 mediante MILP.

Fuente: Elaboración propia.

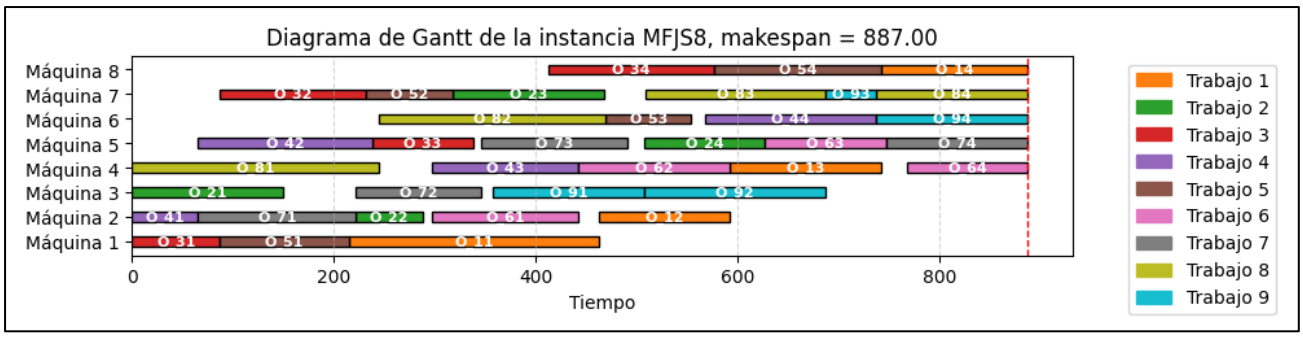

Figura 7.18. Diagrama de la instancia MFJS8 mediante MILP.

Fuente: Elaboración propia.

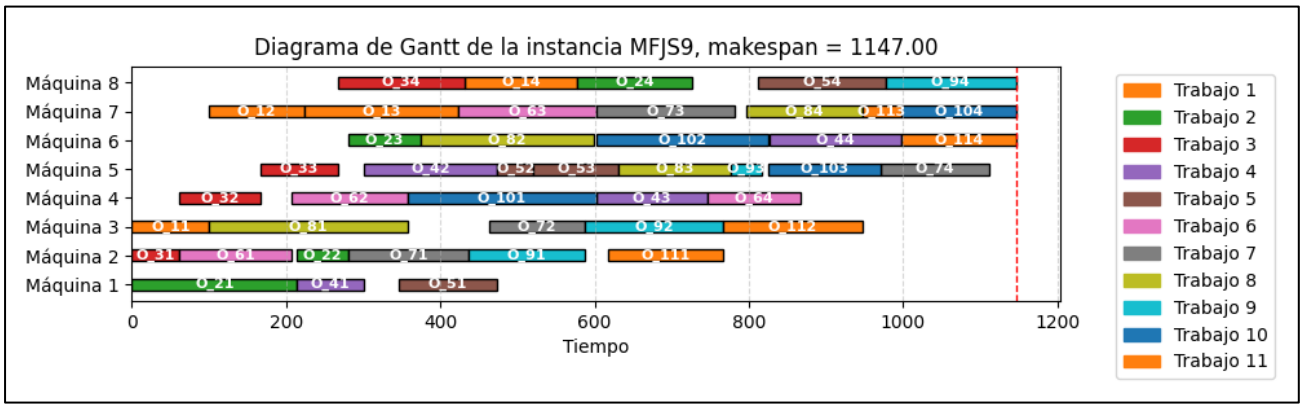

Figura 7.19. Diagrama de la instancia MFJS9 mediante MILP.

Fuente: Elaboración propia.

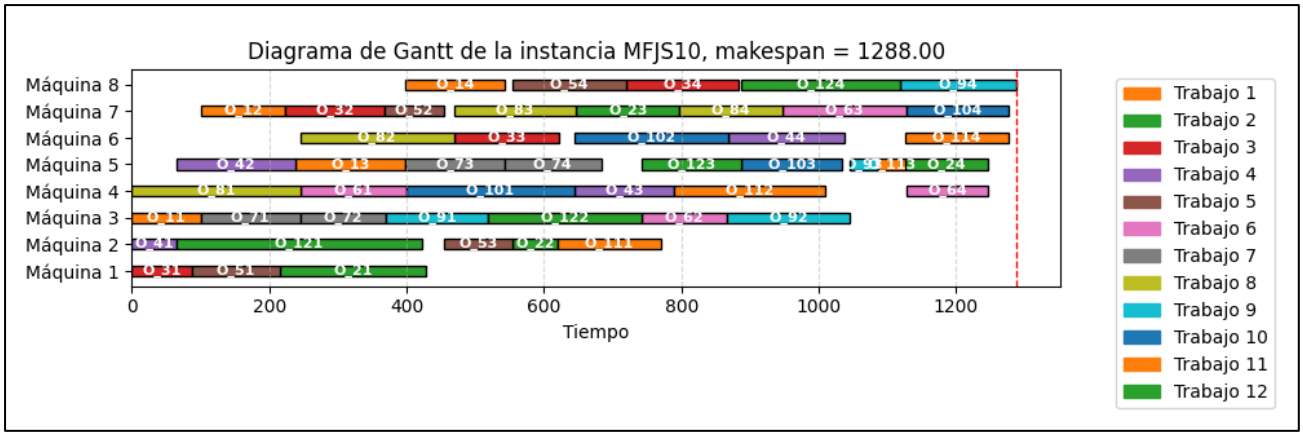

Figura 7.20. Diagrama de la instancia MFJS10 mediante MILP.

Fuente: Elaboración propia.

Anexo 4. Diagrama de Gantt obtenido mediante el Modelo Híbrido

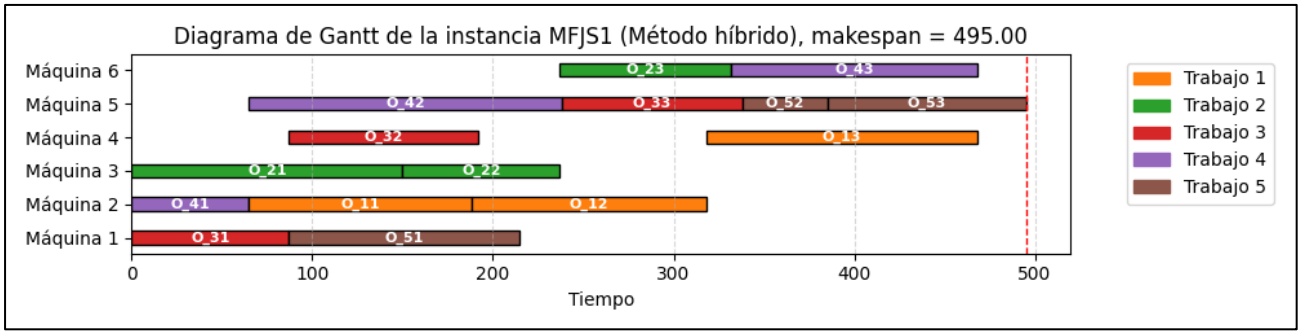

Figura 7.21. Diagrama de la instancia MFJS1 mediante Modelo Híbrido.

Fuente: Elaboración propia.

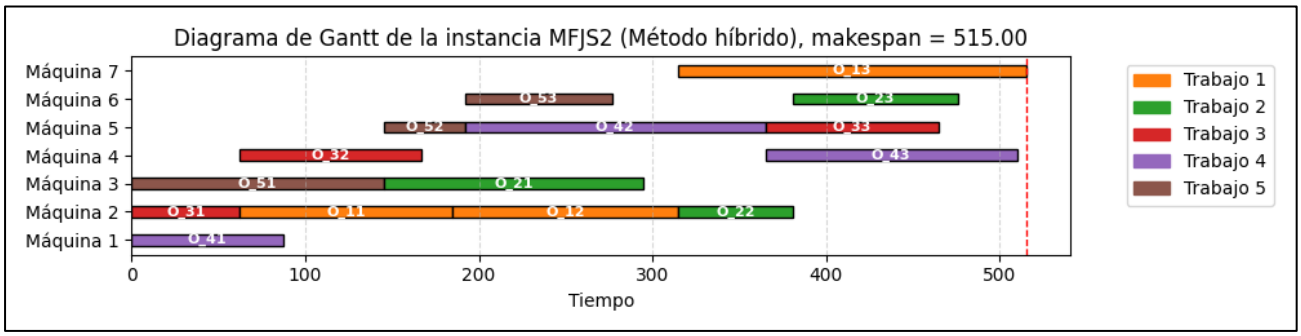

Figura 7.22. Diagrama de la instancia MFJS2 mediante Modelo Híbrido.

Fuente: Elaboración propia.

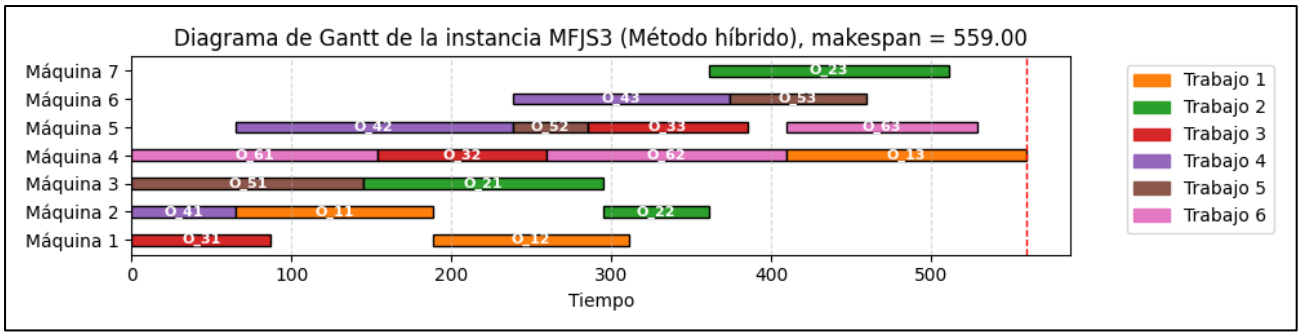

Figura 7.23. Diagrama de la instancia MFJS3 mediante Modelo Híbrido.

Fuente: Elaboración propia.

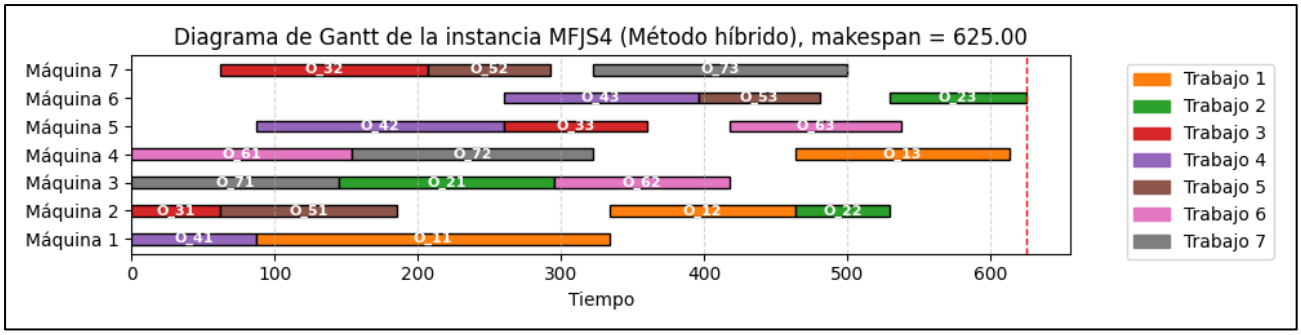

Figura 7.24. Diagrama de la instancia MFJS4 mediante Modelo Híbrido.

Fuente: Elaboración propia.

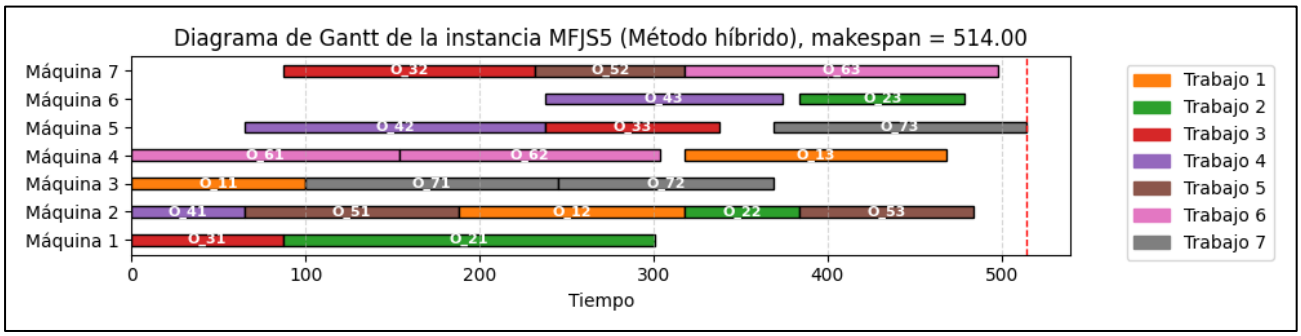

Figura 7.25. Diagrama de la instancia MFJS5 mediante Modelo Híbrido.

Fuente: Elaboración propia.

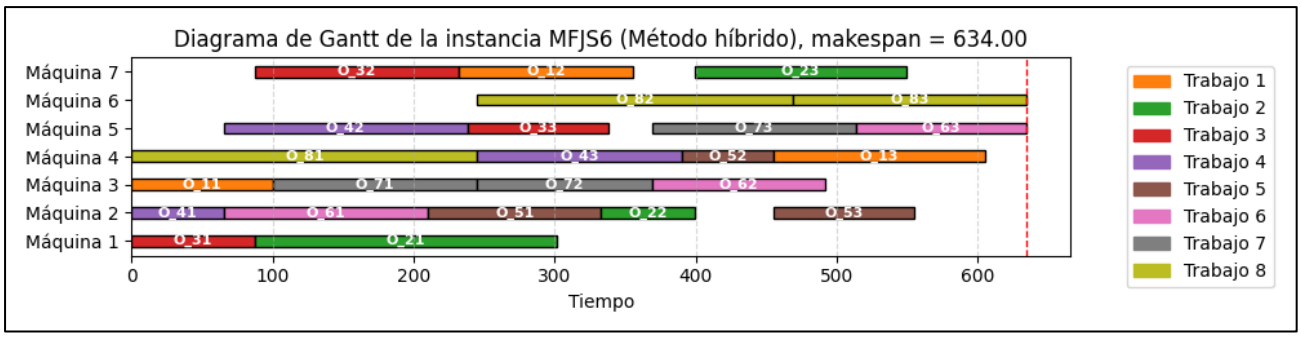

Figura 7.26. Diagrama de la instancia MFJS6 mediante Modelo Híbrido.

Fuente: Elaboración propia.

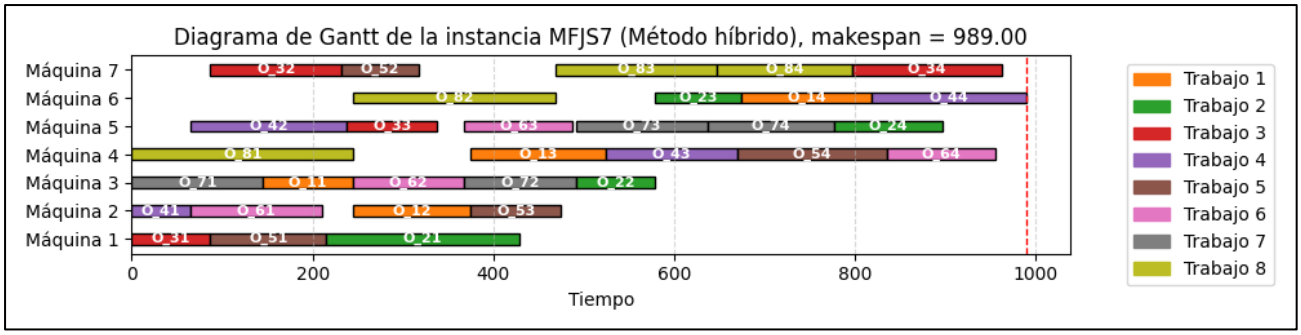

Figura 7.27. Diagrama de la instancia MFJS7 mediante Modelo Híbrido.

Fuente: Elaboración propia.

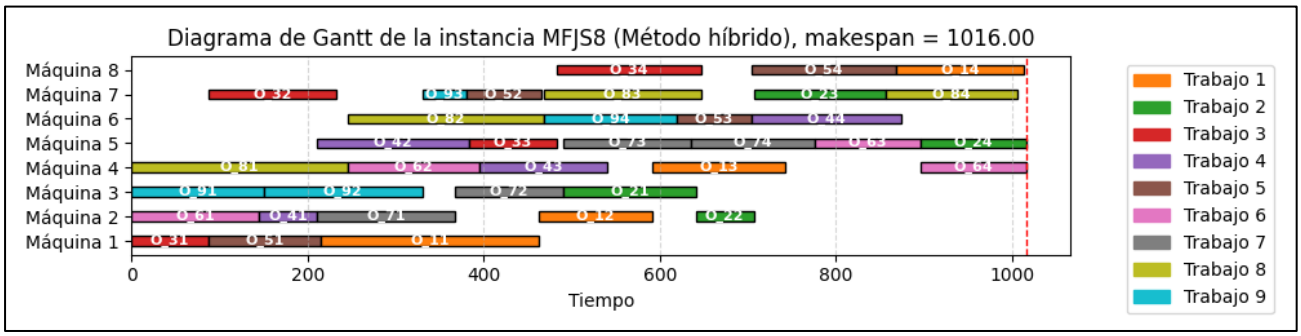

Figura 7.28. Diagrama de la instancia MFJS8 mediante Modelo Híbrido.

Fuente: Elaboración propia.

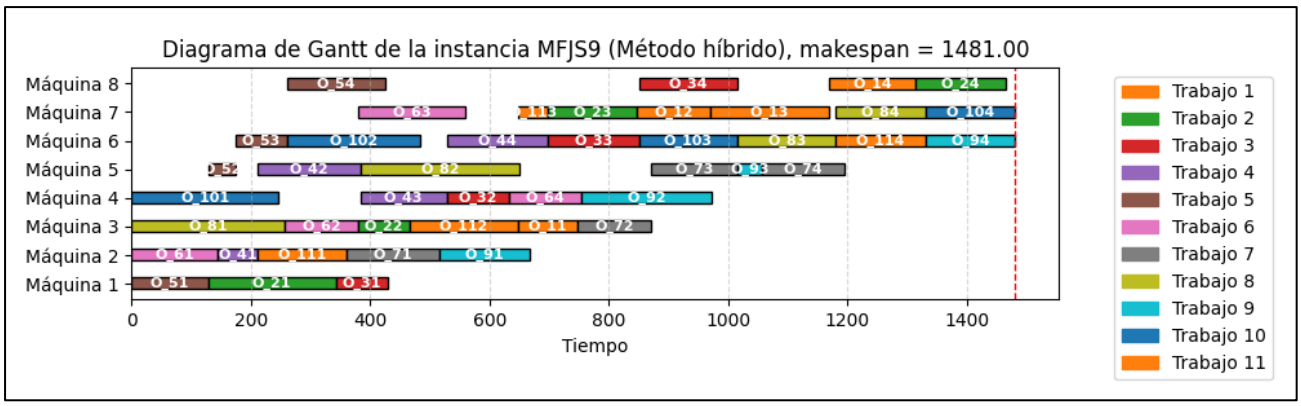

Figura 7.29. Diagrama de la instancia MFJS9 mediante Modelo Híbrido.

Fuente: Elaboración propia.

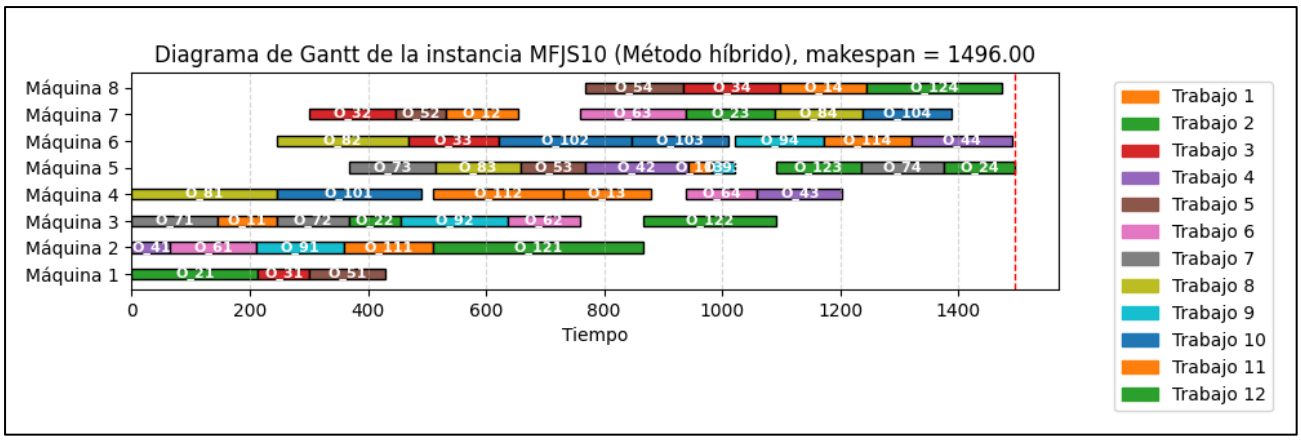

Figura 7.30. Diagrama de la instancia MFJS10 mediante Modelo Híbrido.

Fuente: Elaboración propia.

Anexo 5. Estructura de las 10 instancias analizadas mediante SHAP

Instancia 1:

- ***job*:** (6)
- ***op*:** (3)
- $\boldsymbol{m_i}$**:** (0,0,0,0,1,1,1,0)
- $\boldsymbol{t_{m_i}}$**:** (0,0,0,0,120,240,180,0)
- $\boldsymbol{disp_{m_i}}$**:** (250.0,459.0,418.0,530.0,372.0,459.0,536.0,0.0)
- $\boldsymbol{op_{cola}}$**:** (1)
- $\boldsymbol{carga_{m_i}}$**:** (247,446,418,472,273,221,264,0)

Instancia 2:

- ***job*:** (1)
- ***op*:** (2)
- $\boldsymbol{m_i}$**:** (1,1,0,1,0,0,0,0)
- $\boldsymbol{t_{m_i}}$**:** (123,130,0,140,0,0,0,0)
- $\boldsymbol{disp_{m_i}}$**:** (87.0,123.0,154.0,0.0,290.0,0.0,0.0,0.0)
- $\boldsymbol{op_{cola}}$**:** (2)
- $\boldsymbol{carga_{m_i}}$**:** (87,123,150,0,173,0,0,0)

Instancia 3:

- $\boldsymbol{job}$**:** (8)
- $\boldsymbol{op}$**:** (3)
- $\boldsymbol{m_i}$**:** (0,0,0,0,1,1,1,0)
- $\boldsymbol{t_{m_i}}$**:** (0,0,0,0,145,165,178,0)
- $\boldsymbol{disp_{m_i}}$**:** (342.0,729.0,691.0,799.0,625.0,636.0,774.0,630.0)
- $\boldsymbol{op_{cola}}$**:** (2)
- $\boldsymbol{carga_{m_i}}$**:** (342,729,691,715,488,462,653,310)

Instancia 4:

- $\boldsymbol{job}$**:** (2)
- $\boldsymbol{op}$**:** (1)
- $\boldsymbol{m_i}$**:** (1,1,0,0,0,0,0,0)
- $\boldsymbol{t_{m_i}}$**:** (34,53,0,0,0,0,0,0)
- $\boldsymbol{disp_{m_i}}$**:** (21.0,36.0,0.0,0.0,0.0,0.0,0.0,0.0)
- $\boldsymbol{op_{cola}}$**:** (2)
- $\boldsymbol{carga_{m_i}}$**:** (21,36,0,0,0,0,0,0)

Instancia 5:

- $\boldsymbol{job}$**:** (3)
- $\boldsymbol{op}$**:** (2)
- $\boldsymbol{m_i}$**:** (0,1,1,0,0,0,0,0)
- $\boldsymbol{t_{m_i}}$**:** (0,70,80,0,0,0,0,0)
- $\boldsymbol{disp_{m_i}}$**:** (90.0,25.0,50.0,0.0,0.0,0.0,0.0,0.0)
- $\boldsymbol{op_{cola}}$**:** (2)
- $\boldsymbol{carga_{m_i}}$**:** (90,25,50,0,0,0,0,0)

Instancia 6:

- $\boldsymbol{job}$**:** (1)
- $\boldsymbol{op}$**:** (2)

- $\boldsymbol{m_i}$**:** (0,1,0,1,0,0,1,0)
- $\boldsymbol{t_{m_i}}$**:** (0,130,0,140,0,0,123,0)
- $\boldsymbol{disp_{m_i}}$**:** (301.0,188.0,245.0,304.0,238.0,0.0,232.0,0.0)
- $\boldsymbol{op_{cola}}$**:** (2)
- $\boldsymbol{carga_{m_i}}$**:** (301,188,245,304,173,0,145,0)

Instancia 7:

- $\boldsymbol{job}$**:** (6)
- $\boldsymbol{op}$**:** (2)
- $\boldsymbol{m_i}$**:** (0,0,1,1,1,0,0,0)
- $\boldsymbol{t_{m_i}}$**:** (0,0,123,150,192,0,0,0)
- $\boldsymbol{disp_{m_i}}$**:** (342.0,579.0,444.0,557.0,625.0,636.0,594.0,485.0)
- $\boldsymbol{op_{cola}}$**:** (3)
- $\boldsymbol{carga_{m_i}}$**:** (342,579,444,495,488,462,473,165)

Instancia 8:

- $\boldsymbol{job}$**:** (7)
- $\boldsymbol{op}$**:** (3)
- $\boldsymbol{m_i}$**:** (0,0,0,0,1,1,1,0)
- $\boldsymbol{t_{m_i}}$**:** (0,0,0,0,145,165,178,0)
- $\boldsymbol{disp_{m_i}}$**:** (301.0,399.0,369.0,390.0,345.0,469.0,368.0,0.0)
- $\boldsymbol{op_{cola}}$**:** (1)
- $\boldsymbol{carga_{m_i}}$**:** (301,399,369,390,273,224,268,0)

Instancia 9:

- $\boldsymbol{job}$**:** (4)
- $\boldsymbol{op}$**:** (2)
- $\boldsymbol{m_i}$**:** (0,0,1,0,1,0,0,0)
- $\boldsymbol{t_{m_i}}$**:** (0,0,250,0,173,0,0,0)
- $\boldsymbol{disp_{m_i}}$**:** (342.0,272.0,357.0,412.0,175.0,320.0,223.0,0.0)

- $\boldsymbol{op_{cola}}$: (3)
- $\boldsymbol{carga_{m_i}}$: (342,272,357,350,47,153,123,0)

Instancia 10:

- $\boldsymbol{job}$: (9)
- $\boldsymbol{op}$: (1)
- $\boldsymbol{m_i}$: (0,1,1,0,0,0,0,0)
- $\boldsymbol{t_{m_i}}$: (0,150,150,0,0,0,0,0)
- $\boldsymbol{disp_{m_i}}$: (429.0,567.0,740.0,640.0,704.0,622.0,647.0,705.0)
- $\boldsymbol{op_{cola}}$: (4)
- $\boldsymbol{carga_{m_i}}$: (429,567,680,640,568,377,532,310)

Anexo 6. Visualización de predicciones y explicaciones SHAP por instancia seleccionada

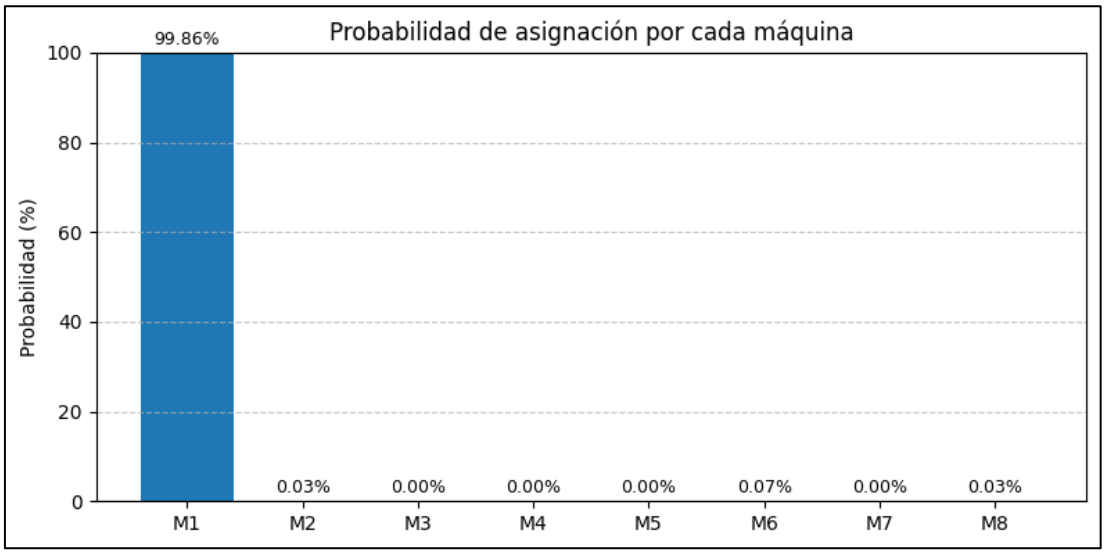

Figura 7.31. Probabilidad de asignación de máquinas para la instancia 2.

Fuente: Elaboración propia.

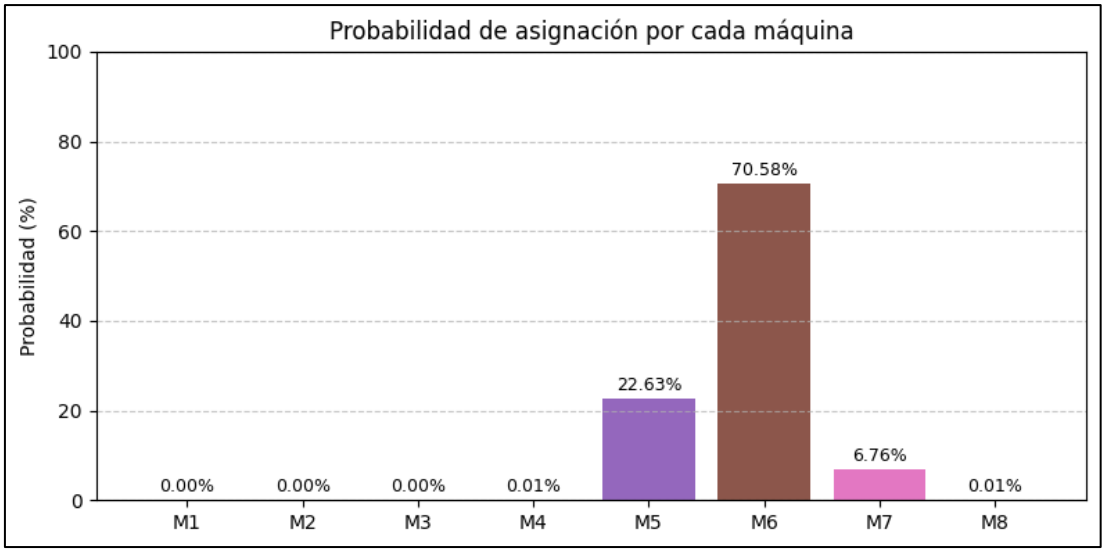

Figura 7.32. Probabilidad de asignación de máquinas para la instancia 3.

Fuente: Elaboración propia.

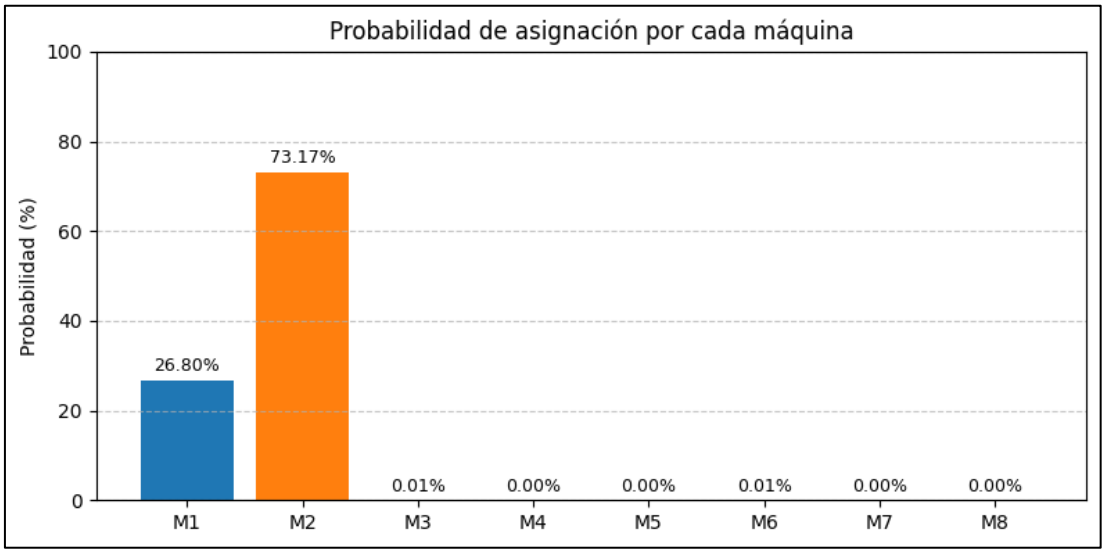

Figura 7.33. Probabilidad de asignación de máquinas para la instancia 4.

Fuente: Elaboración propia.

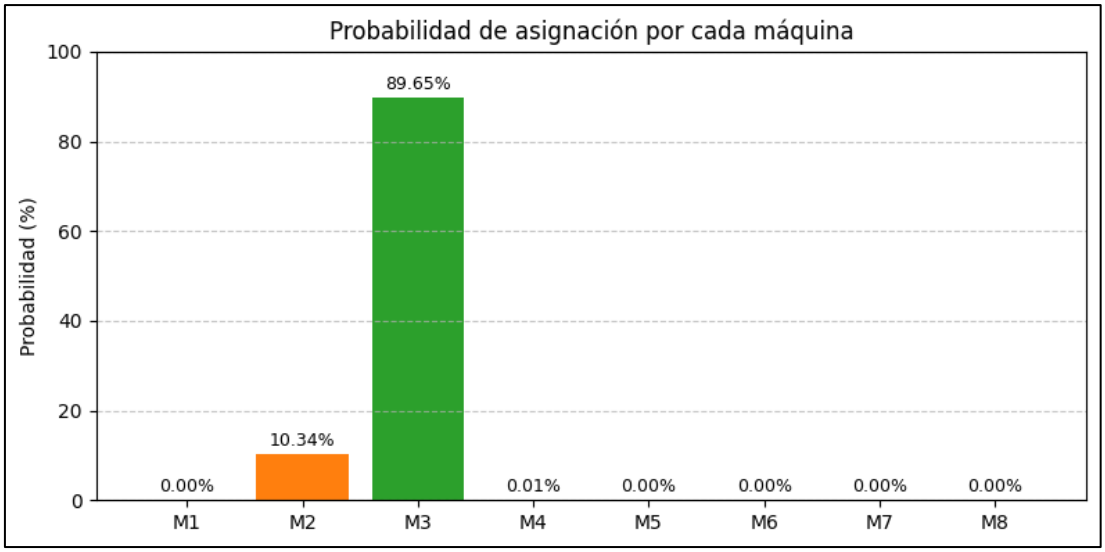

Figura 7.34. Probabilidad de asignación de máquinas para la instancia 5.

Fuente: Elaboración propia.

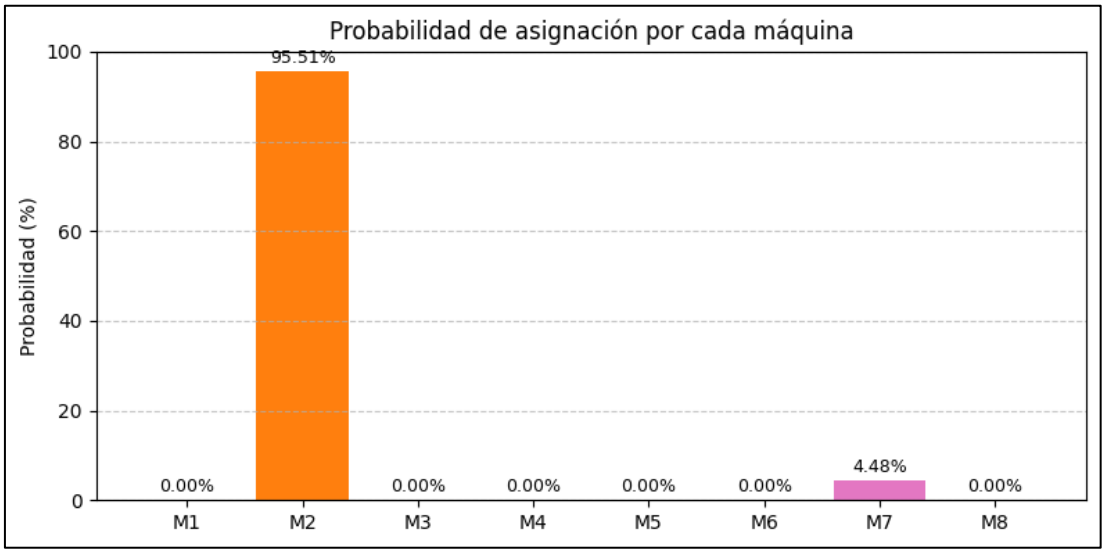

Figura 7.35. Probabilidad de asignación de máquinas para la instancia 6.

Fuente: Elaboración propia.

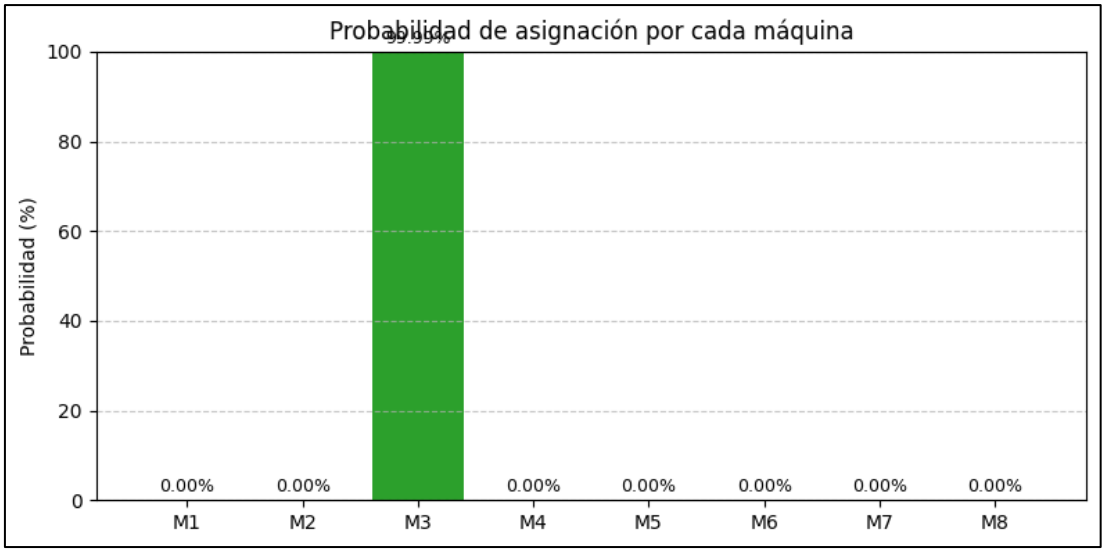

Figura 7.36. Probabilidad de asignación de máquinas para la instancia 7.

Fuente: Elaboración propia.

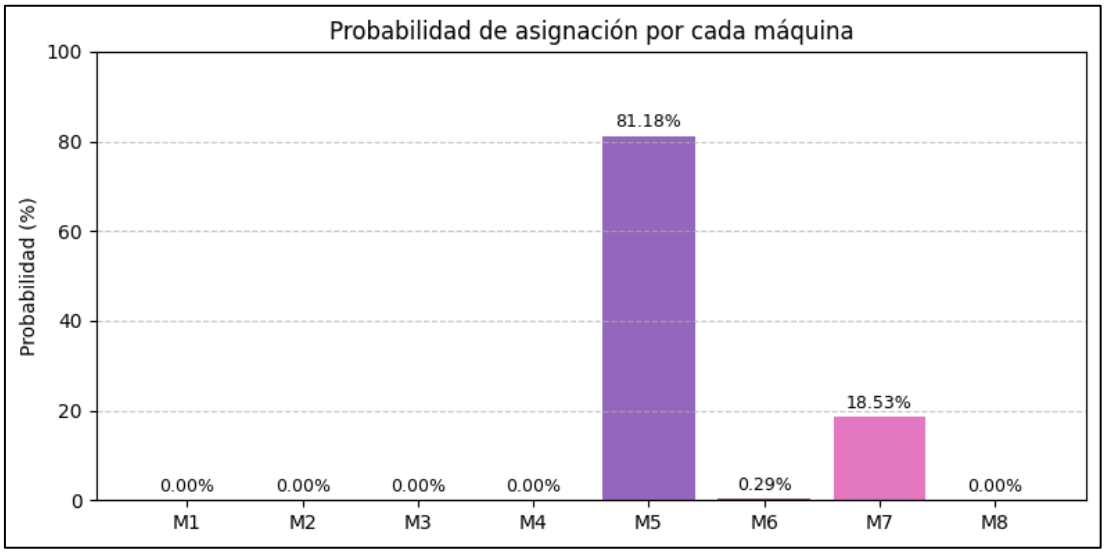

Figura 7.37. Probabilidad de asignación de máquinas para la instancia 8.

Fuente: Elaboración propia.

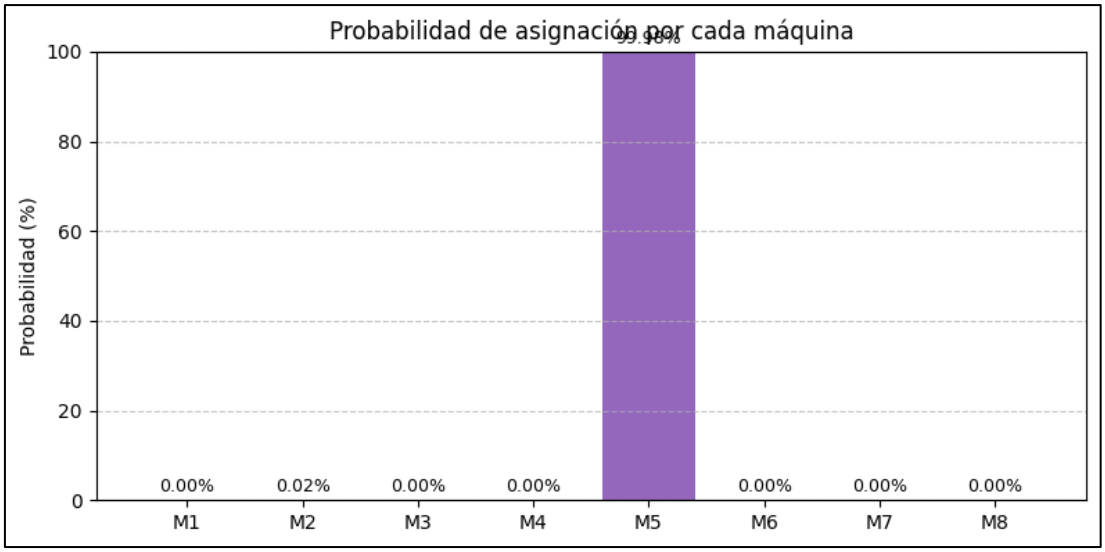

Figura 7.38. Probabilidad de asignación de máquinas para la instancia 9.

Fuente: Elaboración propia.

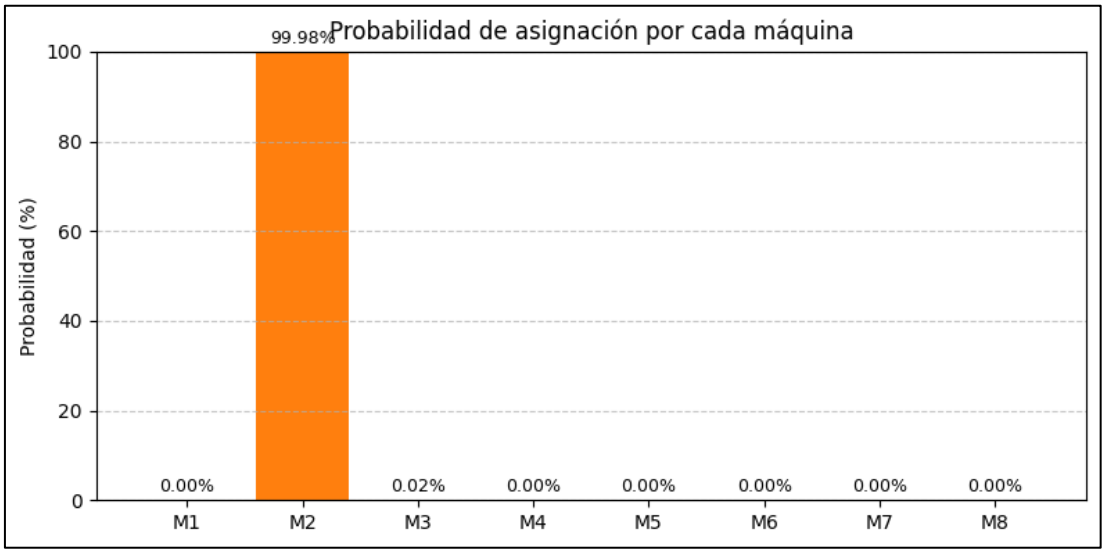

Figura 7.39. Probabilidad de asignación de máquinas para la instancia 10.

Fuente: Elaboración propia.

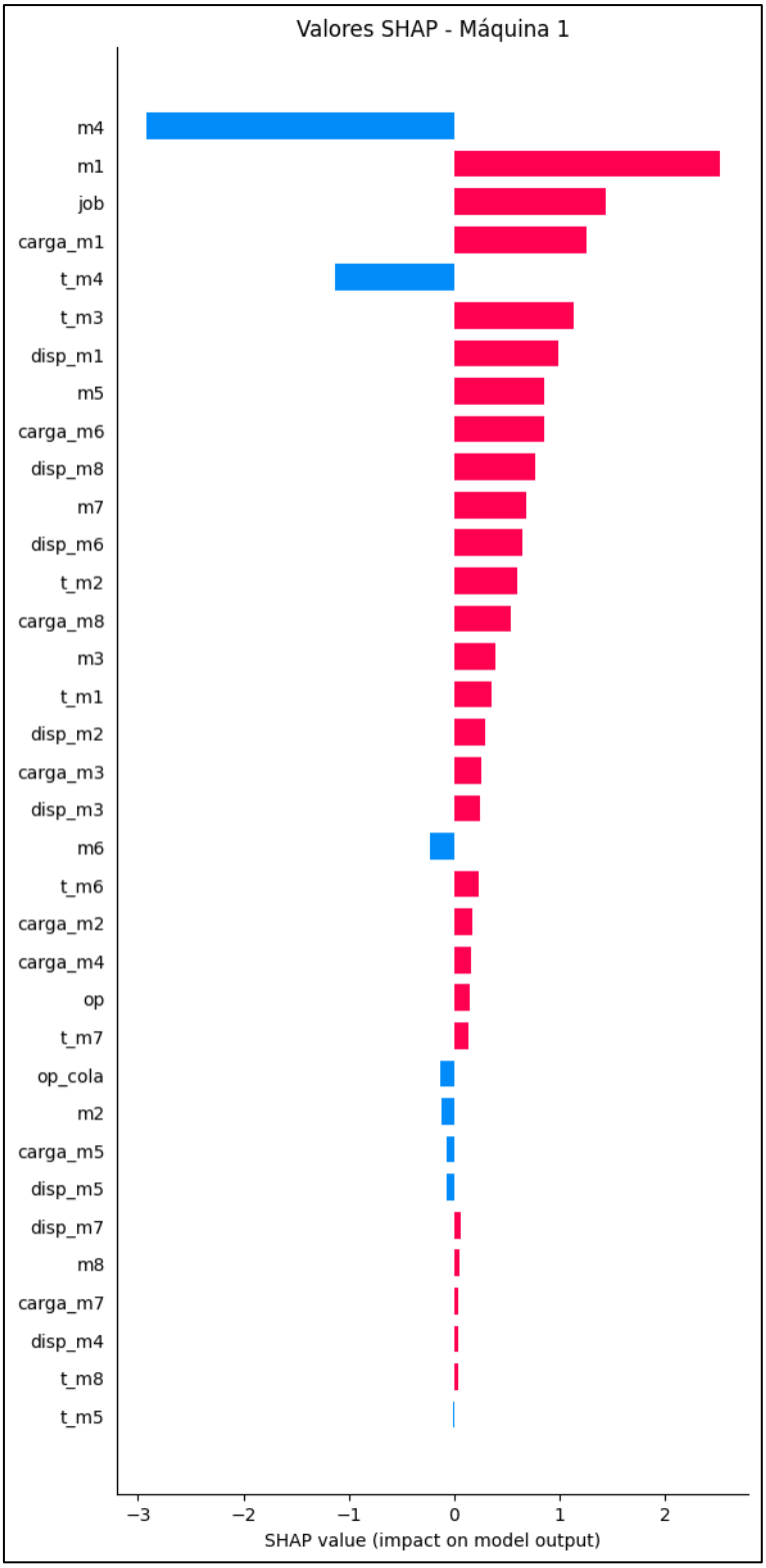

Figura 7.40. Valores SHAP asociados a la máquina 1 para la instancia 2.

Fuente: Elaboración propia.

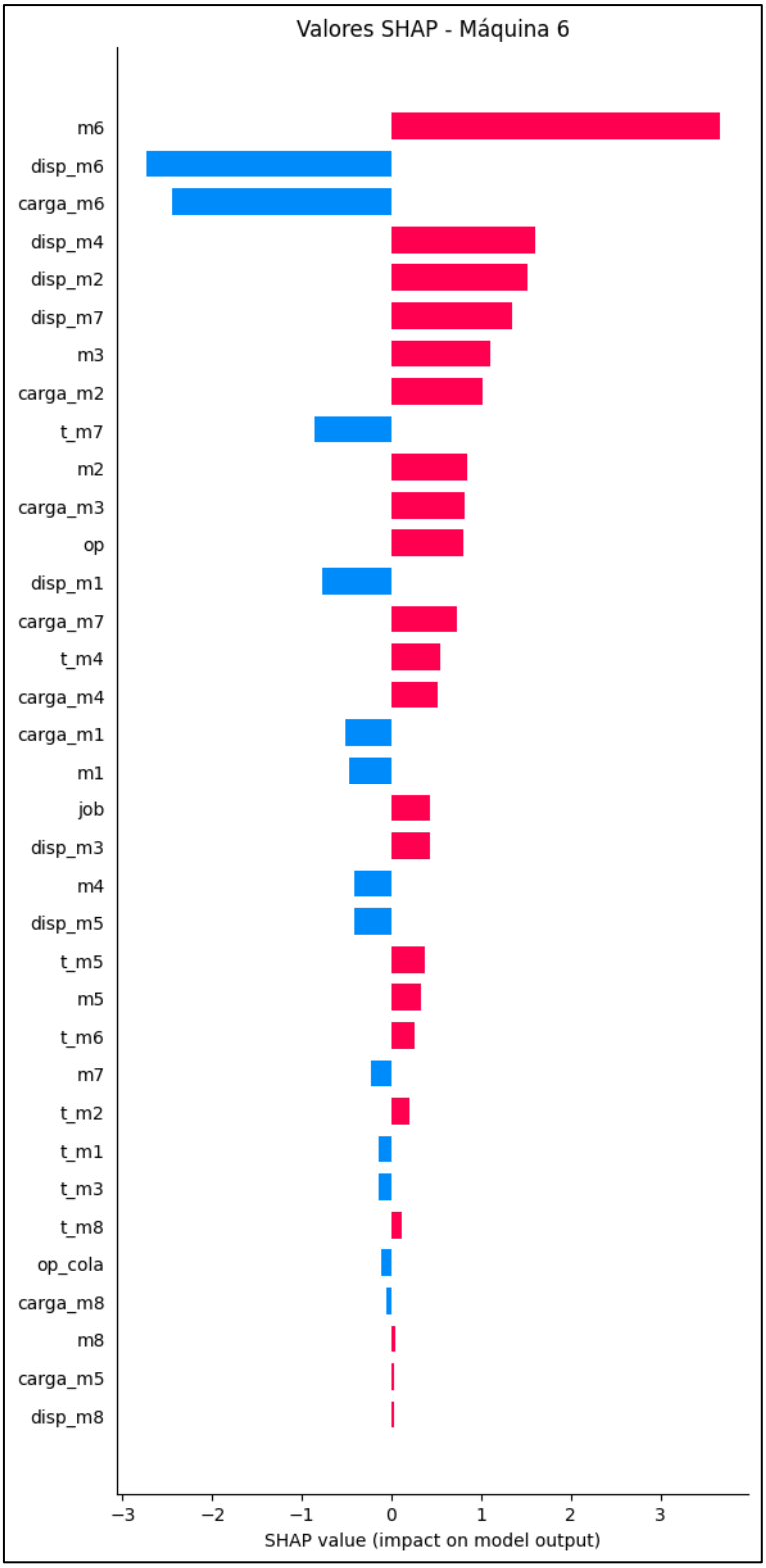

Figura 7.41. Valores SHAP asociados a la máquina 6 para la instancia 3.

Fuente: Elaboración propia.

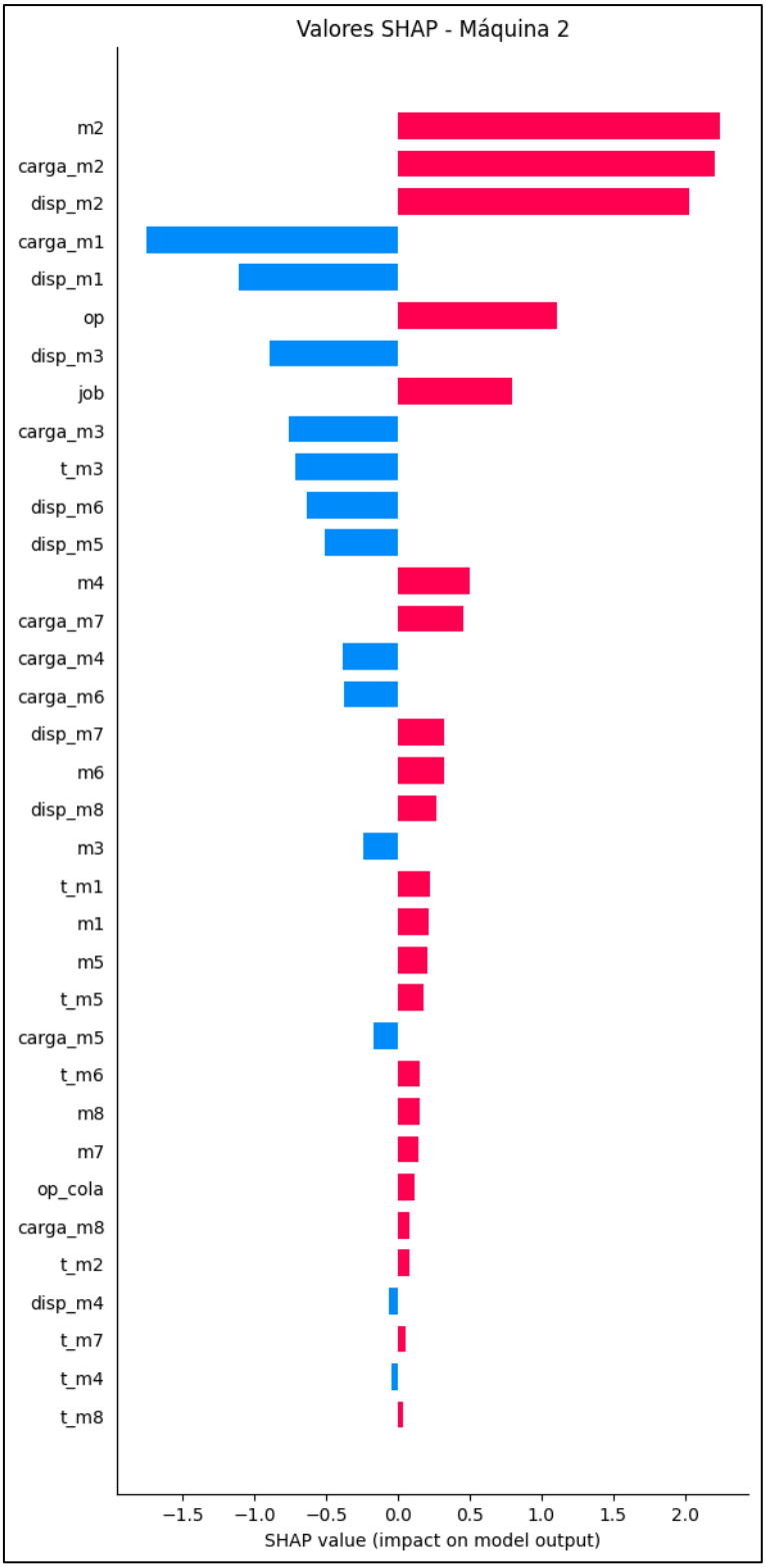

Figura 7.42. Valores SHAP asociados a la máquina 2 para la instancia 4.

Fuente: Elaboración propia.

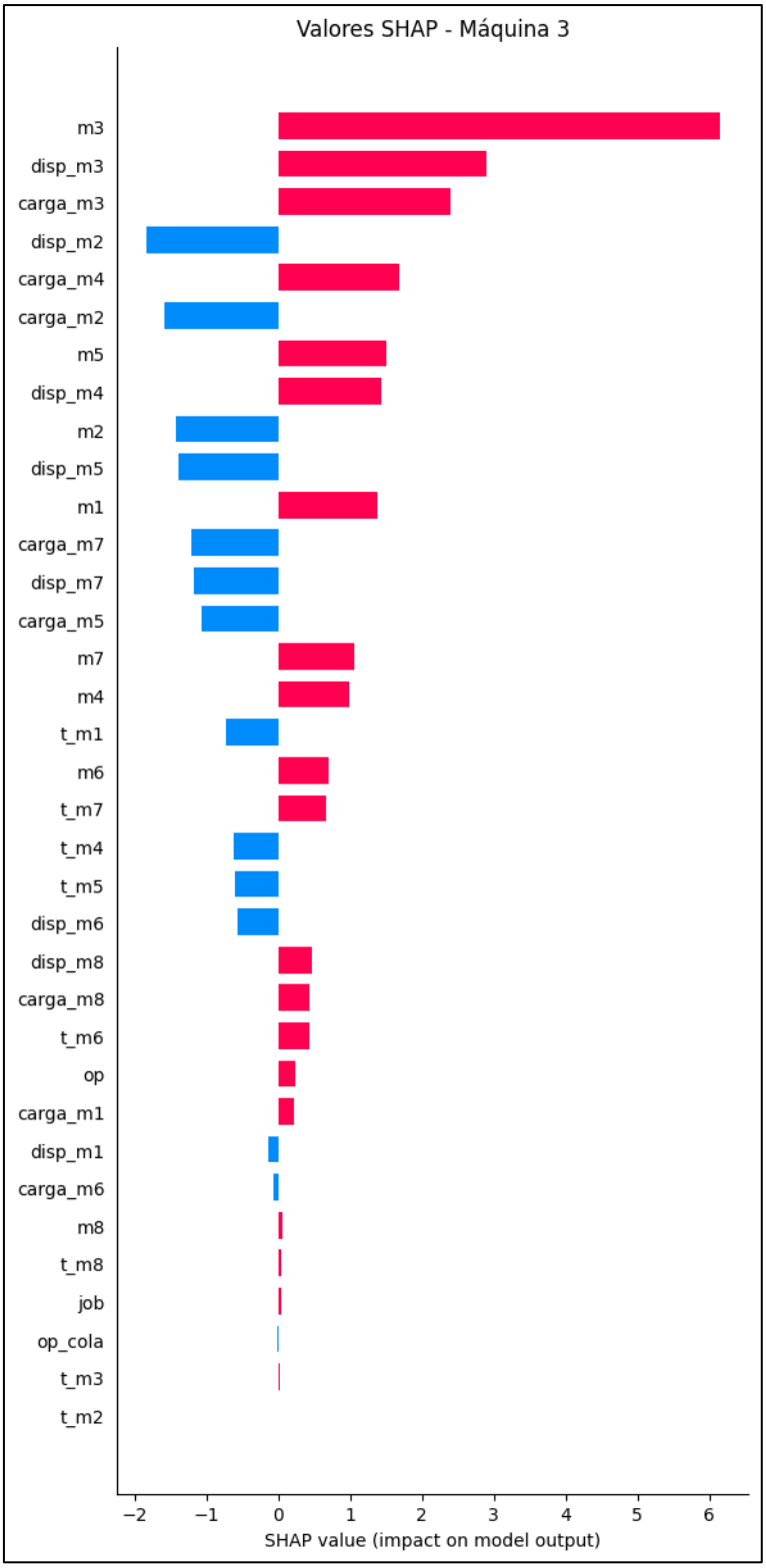

Figura 7.43. Valores SHAP asociados a la máquina 3 para la instancia 5.

Fuente: Elaboración propia.

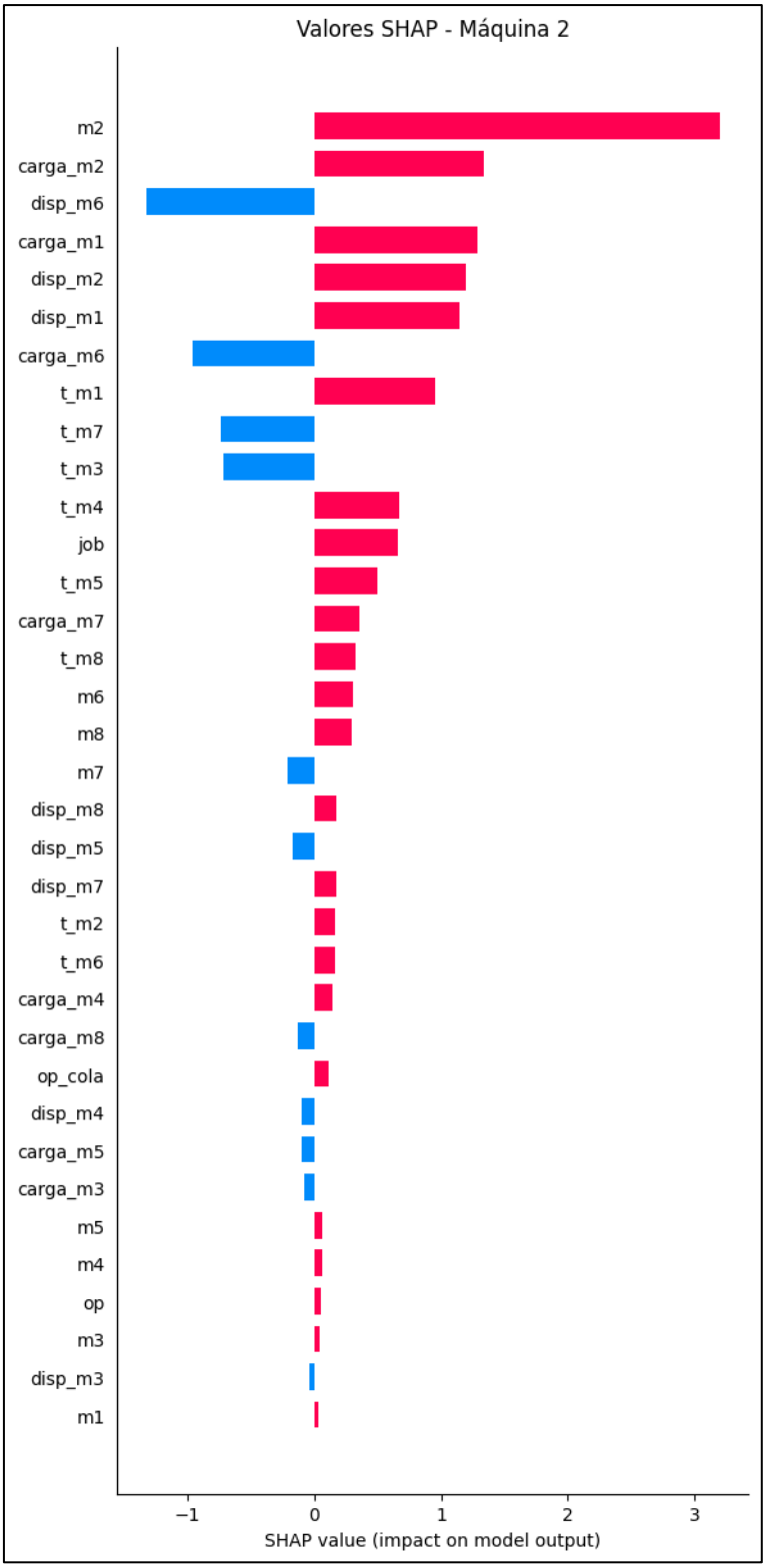

Figura 7.44. Valores SHAP asociados a la máquina 2 para la instancia 6.

Fuente: Elaboración propia.

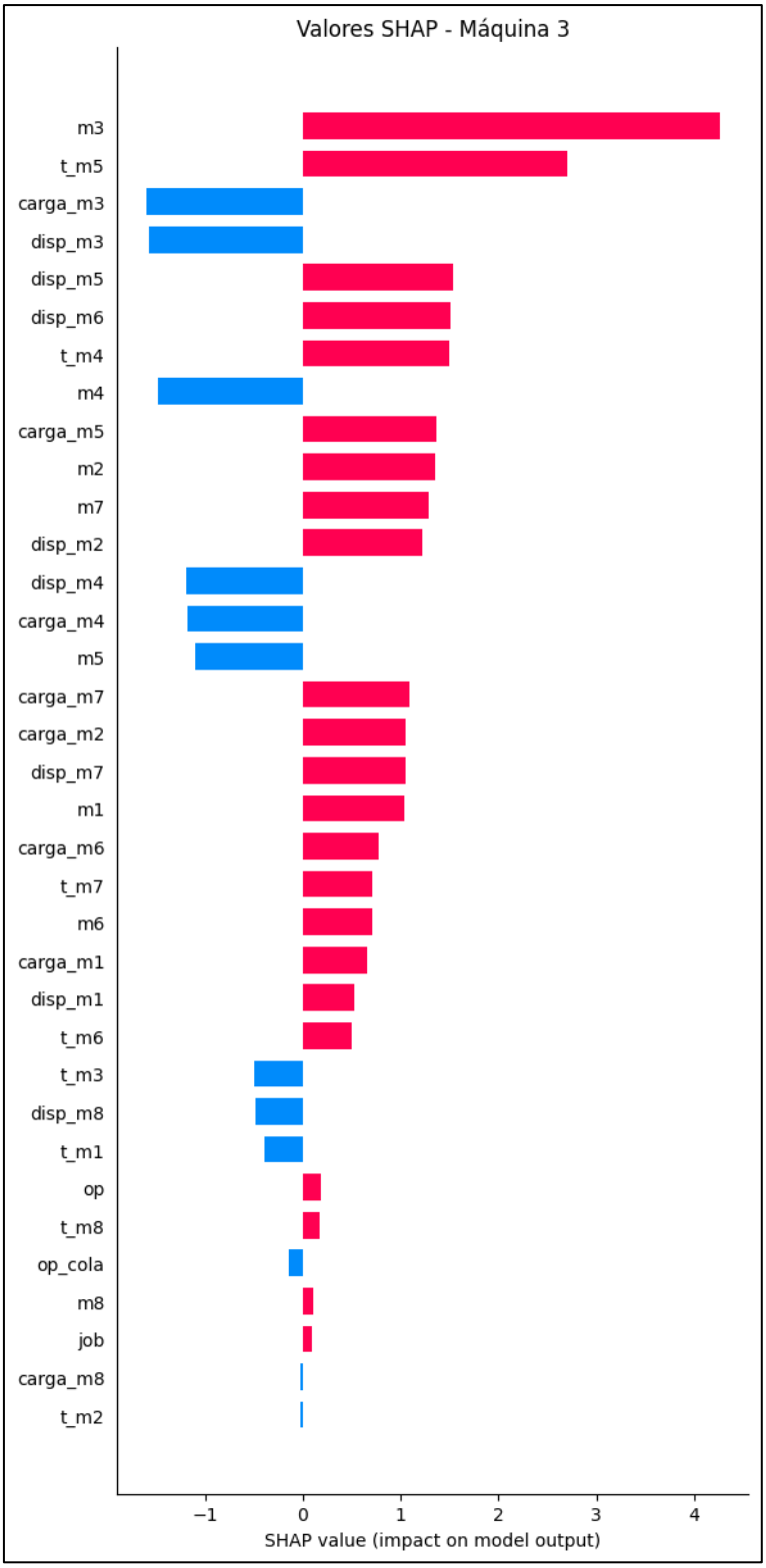

Figura 7.45. Valores SHAP asociados a la máquina 3 para la instancia 7.

Fuente: Elaboración propia.

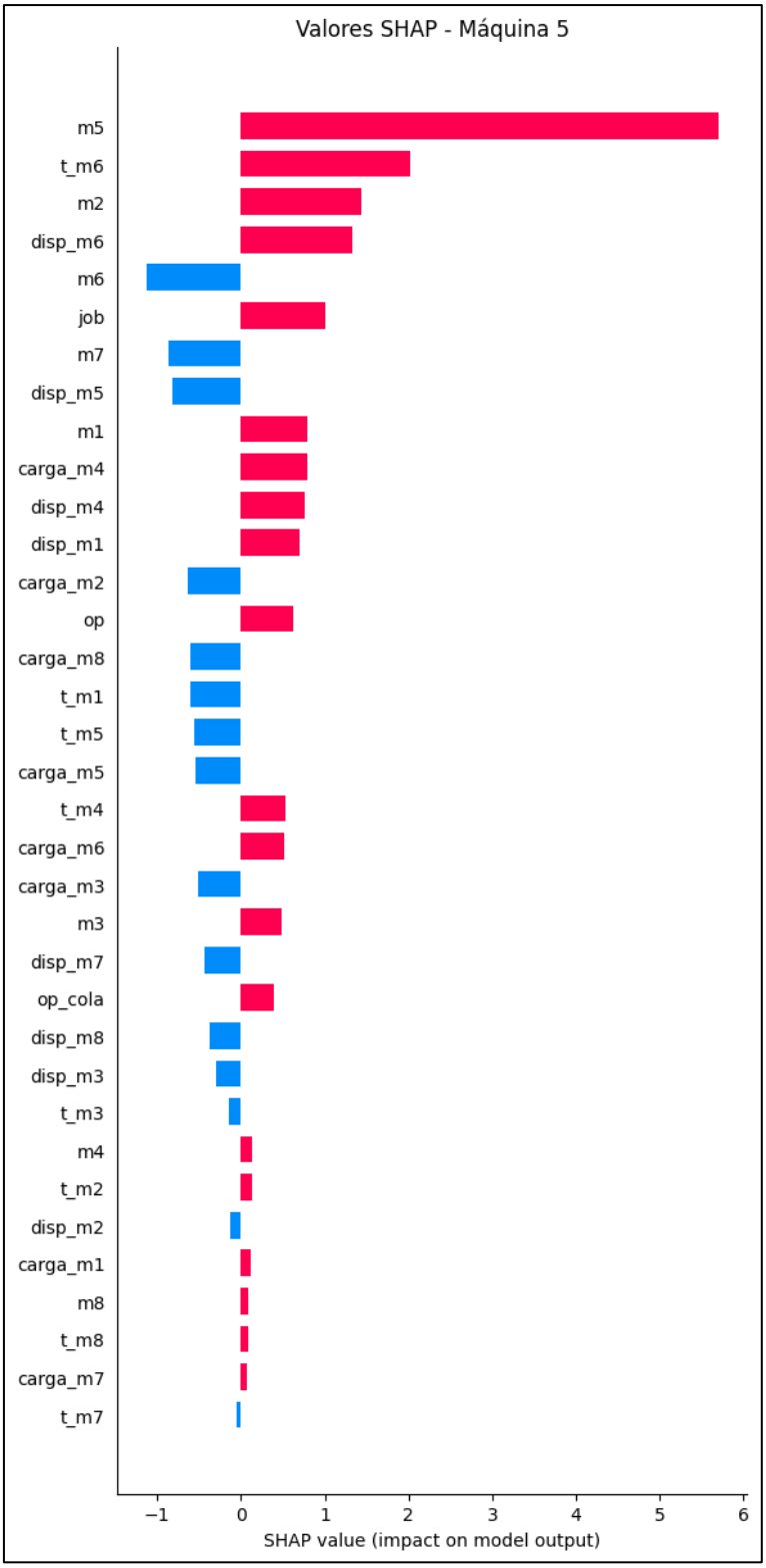

Figura 7.46. Valores SHAP asociados a la máquina 5 para la instancia 8.

Fuente: Elaboración propia.

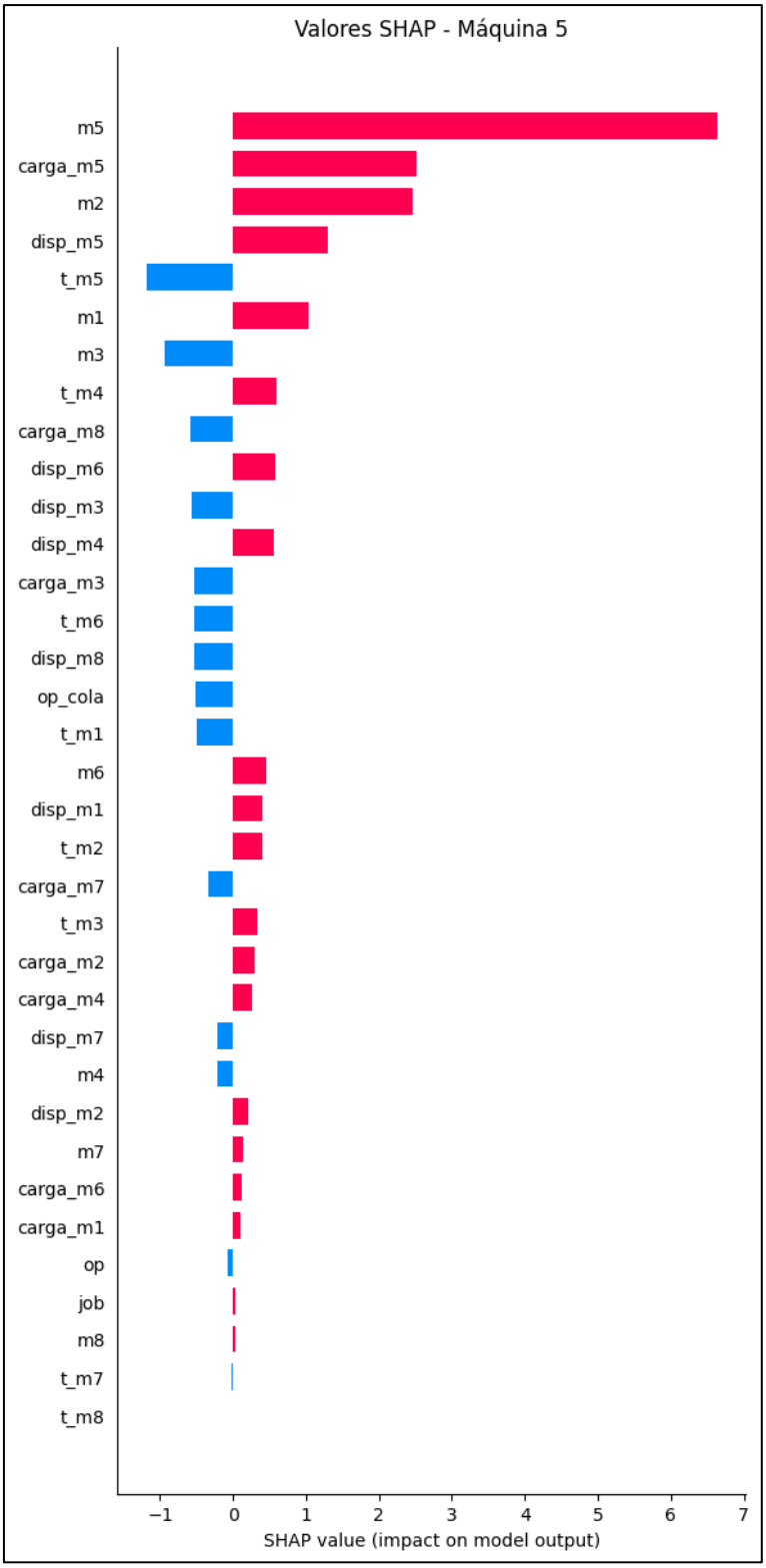

Figura 7.47. Valores SHAP asociados a la máquina 5 para la instancia 9.

Fuente: Elaboración propia.

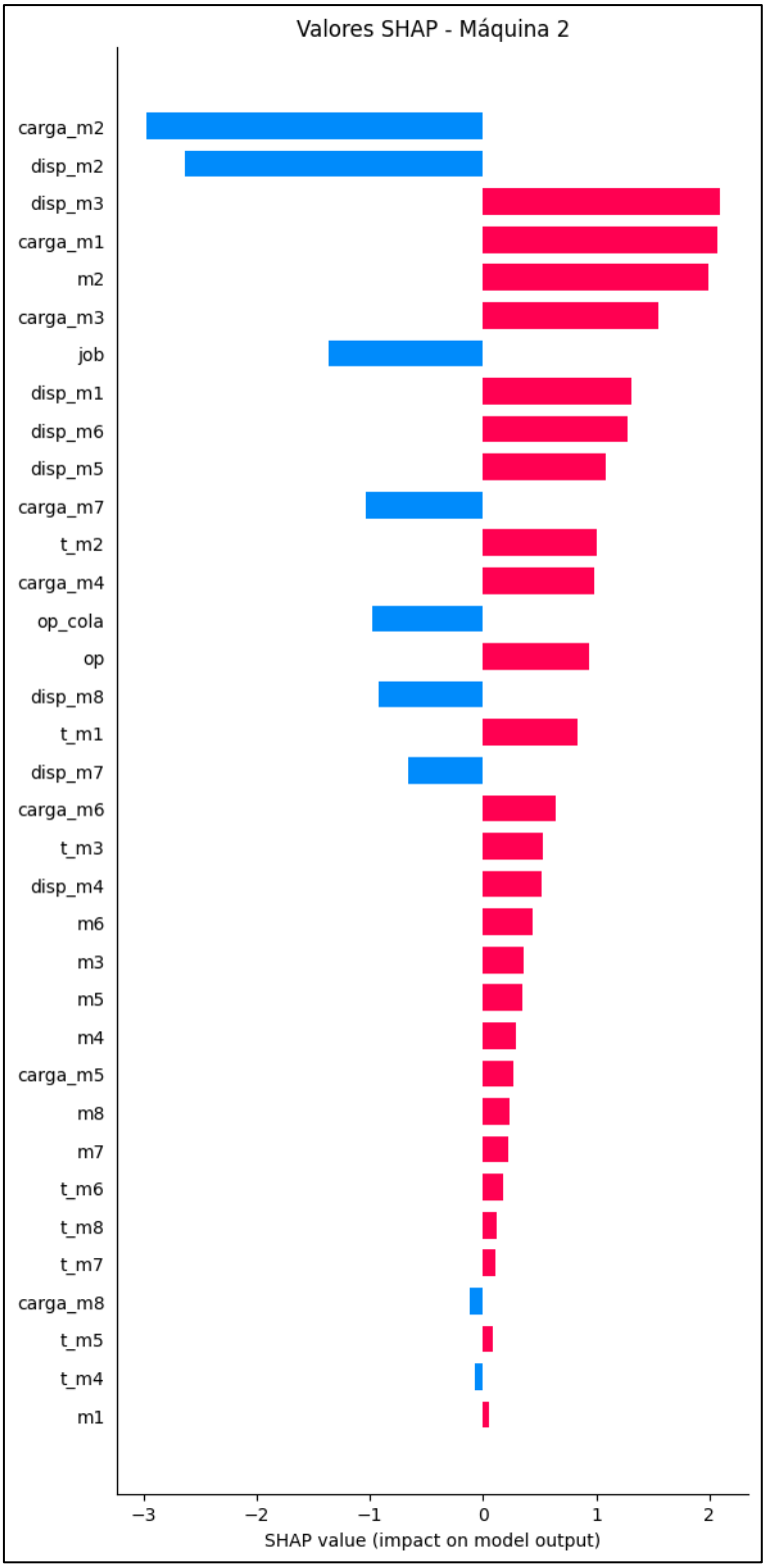

Figura 7.48. Valores SHAP asociados a la máquina 2 para la instancia 10.

Fuente: Elaboración propia.

Resumen FI

UNIVERSIDAD DE CONCEPCIÓN – FACULTAD DE INGENIERÍA

RESUMEN DE MEMORIA DE TÍTULO

Departamento	: Departamento de Ingeniería Industrial
Carrera	: Ingeniería Civil Industrial
Nombre del memorista	: Miguel Ignacio Oliva Moraga
Título de la memoria	: Interpretabilidad en la Industria 5.0: Análisis de un problema Job Shop Flexible.
Fecha de la presentación oral	:
Profesor(es) Guía	: Patricio Antonio Sáez Bustos
Profesor(es) Revisor(es)	: Eduardo Javier Salazar Hornig
Concepto	:
Calificación	:

Resumen (máximo 200 palabras)

El presente estudio aborda el problema de programación flexible de talleres de trabajo (FJSSP) en línea con los principios de la Industria 5.0, la cual promueve el desarrollo de sistemas inteligentes más comprensibles, confiables y colaborativos, capaces de dar apoyo en las decisiones críticas en entornos industriales altamente exigentes. Se propone un modelo híbrido que combina un modelo de red neuronal artificial con un algoritmo genético para predecir asignaciones de máquinas y optimizar la secuenciación de operaciones. Para ello, la red neuronal es entrenada a partir de datos generados por un modelo de programación entera mixta (MILP), mientras que el algoritmo genético busca soluciones eficientes reduciendo el makespan del problema. Adicionalmente, se incorpora la metodología SHAP como herramienta de inteligencia artificial explicable (XAI), permitiendo analizar la influencia de cada variable de entrada sobre las decisiones del modelo. Los resultados muestran que el modelo híbrido entrega soluciones de buena calidad en tiempos de cómputo significativamente menores que los métodos exactos, y que el modelo es capaz de aprender patrones coherentes con criterios operacionales, revelando así parte de la lógica interna del modelo y contribuyendo a superar su naturaleza de caja negra mediante explicaciones claras y cuantificables.