Automatización en la Asignación de Tareas en un ERP para Talleres Industriales utilizando Algoritmos Genéticos

Pontificia Universidad
JAVERIANA
Cali

[VIGILADA MINEDUCACIÓN. Res. 12220 de 2016]

Maria José Pava Echeverry

Jose Daniel Ramirez Delgado

Directora:
PhD. Gloria Ines Alvares Vargas

Co-director:
PhD. Diego Luis Linares Ospina

FACULTAD DE INGENIERÍA Y CIENCIAS
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Pontificia Universidad Javeriana Cali

Santiago de Cali, 2025

Abstract

This project develops and validates a genetic algorithm (GA) based model for automating task assignment in industrial workshop ERP systems. Manual task allocation in automotive technology workshops often results in worker overload, idle times, and delivery delays. The proposed GA-based approach integrates into the ERP work order management module, considering worker capabilities, spare parts availability, and workload distribution constraints.

The core problem is modeled as a variant of Job Shop Scheduling with practical constraints including task precedences, skill requirements, parts availability, and daily time horizons. The GA employs a dual-representation chromosome encoding both operator assignments and task sequencing priorities, evaluated through a fitness function that respects domain-specific constraints. The implementation uses Python 3.14.0, NumPy 1.26.4, and PyGAD 3.2.0, with calibrated parameters: 300 generations, population of 100, 15 % mutation rate, and ranking-based selection.

Experimental validation across 30 test instances from Rectificadora Recti-Ram workshop demonstrates significant advantages over the Shortest Processing Time (SPT) heuristic. The GA achieved 15.6 % reduction in makespan (157.37 vs 186.53 time units) and 14.3 % improvement in load balance (standard deviation 30.10 vs 35.11), while maintaining equivalent task completion rates. Both methods executed 29.53 tasks daily, confirming quality gains stem from superior sequencing rather than increased capacity. The GA requires 3.13 seconds per instance compared to milliseconds for SPT, yielding operationally viable computational costs.

Industrially, the 15.6 % efficiency gain represents approximately 29.16 additional productive minutes per day, equivalent to 117.6 annual hours. Improved load balance reduces bottlenecks, worker fatigue, and operational errors while enabling greater order volume capacity. This research contributes formalized methodology and reproducible solution using open-source tools,

reducing adoption barriers for small and medium enterprises. Future extensions include hybrid metaheuristics, flexible job shop scheduling, multi-objective optimization, and Industry 4.0 integration.

Genetic algorithms demonstrate viability as practical tools for industrial task automation, outperforming classical heuristics while maintaining adaptability and reproducibility in real operational environments.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

El presente proyecto de grado se centró en abordar los desafíos que enfrentan los talleres industriales en la gestión eficiente de sus operaciones, específicamente en la asignación de tareas dentro de un sistema de Planificación de Recursos Empresariales (ERP). La asignación tradicional de tareas, frecuentemente realizada de forma manual, puede conducir a ineficiencias significativas, tales como la sobrecarga de operarios, tiempos muertos en la producción y retrasos en las entregas, afectando directamente la productividad y la rentabilidad del taller.

En este contexto, la automatización de la asignación de tareas emerge como una solución crucial para optimizar el uso de los recursos y mejorar el flujo de trabajo. Este proyecto propuso el desarrollo de un modelo de automatización basado en algoritmos genéticos, una técnica de inteligencia artificial que ha demostrado su eficacia en la resolución de problemas complejos de optimización. Al implementar algoritmos genéticos en el módulo de gestión de órdenes de trabajo de un ERP, se buscó automatizar la distribución de tareas considerando variables clave como las capacidades técnicas de los operarios, la disponibilidad de insumos y la carga de trabajo actual.

Para el desarrollo del proyecto, se llevó a cabo un proceso metodológico que incluyó el análisis de los procesos actuales de asignación de tareas en Rectificadora Recti-Ram, la recopilación y estructuración de datos operacionales, el diseño e implementación de un algoritmo genético adaptado a las necesidades específicas del taller, y la evaluación comparativa del modelo propuesto frente a la heurística clásica *Shortest Processing Time* (SPT), ampliamente utilizada en el campo de la ingeniería industrial. Los resultados obtenidos demostraron que el algoritmo genético desarrollado logró una re-

ducción del 15.6 % en el makespan promedio (de 186.53 a 157.37 unidades de tiempo) y una disminución del 14.3 % en la desviación estándar de carga entre operarios (de 35.11 a 30.10), logrando cronogramas más compactos, mejor balance de trabajo y mayor estabilidad operativa. Estas mejoras validaron la eficacia y aplicabilidad del enfoque propuesto en un entorno industrial real, demostrando que el algoritmo genético no solo es competitivo frente a métodos tradicionales, sino que los supera en los aspectos más relevantes para la operación diaria del taller.

La investigación se llevó a cabo en colaboración con la empresa Rectificadora Recti-Ram, lo que proporcionó un contexto real y datos relevantes para el desarrollo y la validación del modelo.

A través de este proyecto, se contribuyó al avance en la automatización de los talleres industriales, proporcionando una herramienta que permite mejorar la eficiencia operativa, reducir los costos y aumentar la satisfacción del cliente mediante la optimización de los procesos de asignación de tareas.

Capítulo 2

Definición y Planteamiento del Problema

2.1. Planteamiento del Problema

El entorno productivo de los talleres industriales enfrenta diversas problemáticas que afectan su eficiencia y rendimiento. Entre los principales desafíos se encuentran la capacidad de producción limitada, los largos tiempos de fabricación y los retrasos en las fechas de entrega [1]. Además, la falta de sistemas estructurados para la evaluación de calidad y la baja integración tecnológica pueden impactar negativamente la productividad del proceso [1].

La asignación de tareas en los talleres industriales es un factor clave para optimizar el uso de recursos humanos y técnicos. Una planificación ineficiente puede provocar retrasos, una distribución desequilibrada de la carga de trabajo y una reducción en la productividad [2]. La complejidad de estos entornos productivos, caracterizados por la variabilidad en la demanda y la necesidad de coordinar múltiples operaciones, hace necesario el uso de enfoques avanzados que permitan gestionar eficazmente la distribución de actividades y mejorar el cumplimiento de los plazos de entrega [2].

Para abordar este problema, se propone la implementación de algoritmos genéticos (AG) en un módulo ERP de gestión de órdenes de trabajo. Esta solución permitirá automatizar la distribución de tareas considerando factores clave como:

- Capacidades técnicas de cada operario.

- Disponibilidad de insumos requeridos para cada tarea.
- Carga de trabajo actual y disponibilidad horaria de los operarios.

Adicionalmente, este estudio evaluará el desempeño del algoritmo genético en comparación con otro método de optimización utilizado en ingeniería industrial, con el objetivo de determinar su viabilidad y eficacia en este contexto.

2.1.1. Formulación

¿Cómo mejorar la asignación de tareas en un ERP de talleres industriales utilizando algoritmos genéticos, y cómo se compara su rendimiento con otras técnicas de automatización utilizadas en ingeniería industrial?

2.1.2. Sistematización

Para abordar de manera integral el problema de investigación, se plantean las siguientes preguntas:

1. ¿Cuáles son los principales factores que afectan la asignación de tareas en talleres industriales?
2. ¿Cómo pueden los algoritmos genéticos mejorar la automatización de tareas dentro del ERP?
3. ¿Qué otras técnicas de automatización existen en ingeniería industrial para resolver problemas similares?
4. ¿Cómo se comparan los resultados obtenidos con algoritmos genéticos respecto a otra técnica de automatización en términos de eficiencia y calidad de la solución?

2.2. Objetivos

2.2.1. Objetivo General

Desarrollar un modelo de automatización basado en algoritmos genéticos para apoyar en el módulo de asignación de tareas de un sistema ERP de talleres industriales, comparándolo con otro método de automatización utilizado en ingeniería industrial.

2.2.2. Objetivos Específicos

Para alcanzar el objetivo general, se plantean los siguientes objetivos específicos:

1. Identificar las caracaterísticas claves que influyen en la asignación de tareas en talleres industriales.

2. Diseñar e implementar un modelo utilizando un algoritmo genético para automatizar la asignación de tareas tales como desmontaje del motor, limpieza y diagnóstico, rectificación de cilindros, rectificación de cigüeñales, ajuste de válvulas, ensamblaje y prueba del motor.

3. Evaluar la funcionalidad del modelo y el desempeño del algoritmo genético en comparación con otra técnicas de automatización utilizadas en ingeniería industrial.

4. Analizar los beneficios y limitaciones de los algoritmos genéticos en este contexto.

2.3. Justificación

La gestión eficiente de órdenes de trabajo en talleres industriales impacta directamente la productividad y rentabilidad. El uso de algoritmos genéticos puede ofrecer una solución flexible y adaptable que automatiza los tiempos de ejecución, minimiza costos y mejora la distribución equitativa de la carga de trabajo.

La empresa Rectificadora Recti-Ram cuenta con datos históricos sobre tiempos de reparación, asignación de operarios y disponibilidad de recursos, lo que facilita el entrenamiento y validación del modelo propuesto. Asimismo, la disponibilidad de la empresa para proporcionar información clave sobre su flujo de trabajo y sus restricciones operativas permitirá diseñar un modelo que se ajuste a las necesidades reales.

Además, comparar esta solución con otro método de automatización permitirá validar su aplicabilidad en escenarios industriales reales, asegurando que la mejor estrategia sea implementada dentro del ERP del taller, garantizando así una planificación eficiente y adaptable a las condiciones dinámicas del entorno productivo.

2.4. Delimitaciones y Alcances

- Se enfocará en la asignación de tareas dentro del módulo de ordenes de trabajo en un ERP para talleres industriales.
- Se implementará un modelo basado en algoritmos genéticos.
- Se comparará con otro método de automatización (a definir en consulta con expertos en ingeniería industrial).

Capítulo 3

Marco de Referencia

3.1. Marco Teórico

Para comprender el desarrollo de un módulo de ERP destinado a la automatización de la asignación de tareas en un taller industrial, es esencial establecer una base teórica sólida sobre los conceptos clave involucrados. Este proyecto se enfoca en la aplicación de algoritmos genéticos como una técnica de inteligencia artificial que permite la automatización de procesos de toma de decisiones en entornos industriales. Los algoritmos genéticos permiten analizar múltiples combinaciones posibles para la asignación de recursos y actividades, facilitando la gestión eficiente de órdenes de trabajo en un ERP.

Dentro del campo de la ingeniería industrial, se han desarrollado diversas metodologías para mejorar la productividad y la gestión de recursos, muchas de ellas basadas en principios de automatización y sistemas inteligentes. En este proyecto, se desarrollará un módulo de ERP que empleará algoritmos genéticos para la automatización de la asignación de tareas, permitiendo reducir la intervención manual y mejorar la eficiencia operativa en talleres industriales. A lo largo del estudio, se comparará este enfoque con otras metodologías existentes en la industria, evaluando su desempeño en términos de tiempos de procesamiento, flexibilidad y adaptabilidad a escenarios reales.

Los sistemas ERP (Enterprise Resource Planning) son herramientas integradas que permiten la automatización y gestión de los procesos empresariales esenciales, tales como finanzas, producción, logística y ventas. Un ERP centraliza la información empresarial, optimizando la toma de decisiones y reduciendo la fragmentación de datos, lo que mejora la eficiencia operativa. Estos sistemas

han evolucionado desde los sistemas MRP (Material Requirements Planning) y actualmente constituyen una pieza clave en la digitalización de la industria [3].

3.1.1. Automatización con ERP en talleres industriales

La implementación de ERP en talleres industriales responde a la necesidad de automatizar y estructurar procesos que, de otro modo, se gestionarían de forma manual, lo que puede generar ineficiencias. En los talleres de tecnomecánica, donde se requiere un control riguroso de los repuestos y tiempos de trabajo, un ERP automatizado permite la asignación eficiente de tareas según la disponibilidad de operarios y recursos, reduciendo tiempos de espera y mejorando la productividad.

En un estudio realizado en 2019 [4], se identificó que el 70.4 % de las quejas de los clientes en una compañía analizada estaban relacionadas con retrasos en la entrega de órdenes de trabajo, mientras que el 29.6 % restante correspondía a problemas de órdenes no confirmadas y reprocesos. Estos problemas, comunes en talleres industriales, pueden ser mitigados mediante un ERP que automatice la distribución de tareas y optimice la administración de recursos, minimizando errores humanos y mejorando la trazabilidad de las órdenes de trabajo.

3.1.2. Algoritmos Genéticos (GA) para la Automatización

La automatización de procesos en entornos industriales ha avanzado con el uso de algoritmos evolutivos, entre los cuales los algoritmos genéticos (GA) han demostrado ser eficaces para resolver problemas de asignación de tareas y planificación de recursos. Estos algoritmos, inspirados en la selección natural y la evolución biológica, trabajan con poblaciones de soluciones potenciales, aplicando operadores como la selección, el cruce (crossover) y la mutación para encontrar soluciones óptimas de manera iterativa [5].

A diferencia de los métodos tradicionales de asignación de tareas, que pueden depender de reglas predefinidas y cálculos deterministas, los algoritmos genéticos ofrecen una solución flexible que se adapta dinámicamente a cambios en el entorno de producción. Gracias a su capacidad de explorar múltiples combinaciones de asignación de tareas y su enfoque probabilístico, los GA

pueden reducir tiempos improductivos y mejorar la distribución del trabajo en talleres industriales.

3.2. Técnicas de Automatización en Ingeniería Industrial

La automatización en ingeniería industrial comprende un conjunto de metodologías, técnicas y herramientas orientadas a optimizar el uso de recursos, reducir desperdicios y mejorar el rendimiento operativo en entornos productivos. Diversos estudios han señalado que la mejora de la eficiencia en talleres y plantas manufactureras depende directamente de la capacidad para programar tareas, asignar recursos y minimizar tiempos improductivos [6, 7]. En los talleres industriales modernos, donde existe alta variabilidad de tareas y heterogeneidad en los tiempos de ejecución, la automatización se convierte en un elemento clave para garantizar un flujo de trabajo continuo y una gestión eficiente de las órdenes de trabajo [8].

Estas técnicas abarcan desde enfoques de mejora continua, como Lean Manufacturing y Six Sigma, hasta métodos formales de secuenciación basados en reglas de prioridad —tales como SPT, FCFS y EDD— que constituyen la base histórica de la programación industrial [9, 10]. Asimismo, la literatura reciente destaca la importancia de combinar estas metodologías con algoritmos avanzados de optimización, dada la complejidad creciente de los sistemas productivos y la naturaleza NP-hard de muchos problemas de scheduling [6, 8].

A continuación se presentan las principales técnicas de automatización utilizadas en la industria para la asignación y programación de tareas, todas ellas relevantes como puntos de comparación frente a métodos evolutivos como los algoritmos genéticos.

3.2.1. Lean Manufacturing y mejora continua

Lean Manufacturing es una filosofía de gestión orientada a la eliminación sistemática de desperdicios (muda) y a la optimización del flujo de trabajo mediante la mejora continua. Sus principios, derivados del Sistema de Producción Toyota, se aplican en talleres industriales para reducir tiempos improductivos, minimizar inventarios y mejorar la estabilidad del proceso productivo. La literatura en secuenciación de operaciones y en sistemas de

manufactura destaca que los talleres que adoptan prácticas Lean presentan niveles más estables de procesamiento y menor variabilidad temporal en sus operaciones [7, 6].

En entornos de tecnomecánica y mantenimiento industrial, Lean permite estructurar procesos que tradicionalmente se ejecutan de forma manual o desorganizada, facilitando la estandarización y priorización de tareas. Herramientas como 5S, Value Stream Mapping y Kanban han demostrado ser efectivas para mejorar la trazabilidad de órdenes de trabajo, reducir cuellos de botella y aumentar la utilización de operarios y máquinas [11]. Estas prácticas aportan la base conceptual necesaria para implementar métodos de automatización más avanzados dentro de un ERP, permitiendo que reglas de despacho y algoritmos de optimización operen sobre flujos de trabajo más consistentes.

3.2.2. Six Sigma y reducción de variabilidad

Six Sigma es una metodología orientada a reducir la variabilidad de los procesos mediante técnicas estadísticas avanzadas y un enfoque basado en datos. Su ciclo DMAIC permite analizar de manera estructurada las causas de ineficiencias y establecer mejoras que impactan directamente la estabilidad del sistema productivo. La investigación en sistemas de manufactura reporta que Six Sigma mejora la precisión en los tiempos de proceso y disminuye la ocurrencia de operaciones re-trabajadas o reprocesos [6].

En talleres de mantenimiento automotriz o metalmecánica, Six Sigma facilita la estandarización de tiempos de reparación, la identificación de cuellos de botella y la reducción de variaciones entre operarios, contribuyendo a un entorno más predecible para la programación de tareas. La reducción de variabilidad es fundamental para que los métodos de scheduling —tanto heurísticos como metaheurísticos— puedan operar bajo condiciones de mayor estabilidad y precisión [8, 9].

3.2.3. Investigación de Operaciones aplicada al Scheduling

La Investigación de Operaciones (IO) ha desarrollado modelos matemáticos para resolver problemas de asignación, secuenciación y optimización de recursos. En el ámbito de la producción industrial, estos modelos incluyen:

- Programación Lineal Entera Mixta (MILP)
- Branch and Bound
- Programación Dinámica
- Modelos deterministas de Flow Shop, Job Shop y Flexible Job Shop

Estos problemas son ampliamente conocidos por su complejidad computacional, siendo Job Shop y Flexible Job Shop NP-hard [6, 7]. Si bien los modelos exactos permiten obtener soluciones óptimas, en la práctica su uso se limita a problemas de pequeña escala debido al crecimiento exponencial del espacio de búsqueda. Investigaciones recientes han demostrado que, incluso con técnicas avanzadas como Constraint Programming, resolver problemas de scheduling con múltiples restricciones sigue siendo un reto significativo para sistemas reales [12].

Por esta razón, en escenarios industriales complejos se emplean heurísticas, reglas de despacho y metaheurísticas que permiten obtener soluciones de buena calidad en tiempos computacionales reducidos [9, 10].

3.2.4. Reglas de prioridad (Dispatching Rules)

Las reglas de prioridad constituyen uno de los métodos clásicos más utilizados para la programación de tareas en entornos industriales debido a su simplicidad, bajo coste computacional y fácil implementación. Estas reglas se basan en atributos específicos de las tareas, como tiempo de procesamiento, fecha de entrega o urgencia.

Las reglas más representativas incluyen:

- **SPT (Shortest Processing Time)**

  Prioriza las tareas más cortas, reduciendo el tiempo promedio de flujo y el tiempo total en el sistema. Es una técnica particularmente eficiente en sistemas sin restricciones complejas [6]. Esta regla se utiliza como base comparativa en esta investigación.

- **FCFS (First Come, First Served)** Procesa tareas en el orden en que llegan. Aunque sencilla, puede generar tiempos improductivos o cargas desbalanceadas [7].

- **EDD (Earliest Due Date)** Minimiza el retraso (tardiness) priorizando tareas con fechas de entrega más próximas. Es una regla ampliamente usada en la industria gráfica y manufactura ligera [9].
- **LPT (Longest Processing Time)** Favorece las tareas más largas y es útil en escenarios donde se busca balancear carga entre máquinas paralelas.
- **Reglas híbridas** Combinan múltiples criterios como tiempo, urgencia y vencimiento. Son más flexibles y representan la transición entre heurísticas clásicas y metaheurísticas [10].

Estas reglas son consideradas baseline industrial y sirven como punto de referencia para evaluar técnicas más avanzadas como algoritmos genéticos o algoritmos híbridos [9, 10].

3.2.5. Métodos heurísticos clásicos

Las heurísticas permiten obtener soluciones aproximadas para problemas de scheduling de manera rápida. Aunque no garantizan optimalidad, son valiosas en entornos industriales por su eficiencia computacional.

Entre las más destacadas se encuentran:

- Heurísticas de cuello de botella
- Algoritmos constructivos como NEH (Flow Shop) o G&T (Job Shop)
- List Scheduling
- Heurísticas parametrizadas por prioridad

Estudios en Flow Shop y Flexible Job Shop señalan que estas heurísticas ofrecen buen desempeño cuando el entorno productivo es estable y presenta pocas restricciones [6, 7]. Sin embargo, su desempeño disminuye considerablemente cuando existen múltiples dependencias, precedencias o recursos limitados —como es el caso en talleres industriales reales—, por lo que requieren ser complementadas con metaheurísticas [9].

3.2.6. Metaheurísticas aplicadas al scheduling

Debido a la complejidad de los problemas industriales de programación de tareas, especialmente aquellos con múltiples máquinas, precedencias y restricciones, las metaheurísticas se han convertido en herramientas fundamentales para obtener soluciones de alta calidad.

Las técnicas más utilizadas incluyen:

- Recocido Simulado (SA)
- Búsqueda Tabú (TS)
- Algoritmos Genéticos (GA)
- PSO (Particle Swarm Optimization)
- Algoritmos híbridos (GA + SA, GA + TS)

La literatura presenta numerosos casos donde los GA y los métodos híbridos superan significativamente a las heurísticas tradicionales y reglas de prioridad [9, 6]. Del mismo modo, revisiones recientes sobre programación en Industria 4.0 y 5.0 destacan el papel de las metaheurísticas en sistemas productivos dinámicos y altamente automatizados [8].

3.2.7. Métodos deterministas y exactos

Aunque su aplicabilidad es limitada en escenarios complejos, los métodos exactos cumplen un rol fundamental como referencia teórica para evaluar heurísticas y metaheurísticas.

Estos métodos incluyen:

- Branch & Bound
- Branch & Cut
- Constraint Programming (CP)
- Programación Lineal y Entera

Investigaciones recientes en CP demuestran que si bien estas técnicas logran acotar soluciones y obtener resultados cercanos al óptimo en algunos casos, su escalabilidad aún es limitada en problemas como FJSSP y JSSP [12]. Por ello, suelen emplearse solo para casos pequeños o como benchmark teórico.

3.3. Antecedentes

La programación de tareas en entornos industriales ha sido un campo de estudio ampliamente abordado en la literatura debido a la complejidad inherente de problemas como Flow Shop, Job Shop y Flexible Job Shop, todos ellos catalogados como NP-hard [6, 7]. Esta complejidad ha motivado el desarrollo de metodologías basadas en heurísticas, reglas de despacho y metaheurísticas avanzadas para obtener soluciones eficientes dentro de límites computacionales razonables. En este contexto, los algoritmos genéticos (AG) han demostrado un rendimiento destacado en la optimización de tareas, especialmente en escenarios donde múltiples restricciones interactúan entre sí.

Uno de los trabajos más relevantes es el de Delgado et al. [13], quienes propusieron un algoritmo genético para la programación de tareas en una celda de manufactura y compararon su desempeño con métodos heurísticos tradicionales. Sus resultados mostraron que los AG pueden reducir tiempos improductivos y mejorar la utilización de recursos, destacando su capacidad para explorar soluciones que las reglas de prioridad no alcanzan debido a su naturaleza determinista.

De manera complementaria, Meisel y Prado [9] desarrollaron un algoritmo genético híbrido que incorpora enfriamiento simulado, demostrando que las metaheurísticas híbridas pueden ofrecer soluciones más estables y de mayor calidad en problemas de tipo Job Shop. Este estudio respalda la idea de que los AG son especialmente adecuados para problemas con múltiples restricciones, como precedencias, máquinas en paralelo y tiempos variables de procesamiento.

En el ámbito de la automatización industrial desde la perspectiva de Industria 4.0, Zhang et al. [8] presentan un análisis del estado del arte de algoritmos evolutivos aplicados a la programación en sistemas inteligentes. Su investigación concluye que los AG, junto con técnicas híbridas de optimización, son herramientas clave para la toma de decisiones en entornos altamente dinámicos y automatizados. Esta tendencia refuerza la pertinencia del uso de AG como enfoque moderno para la gestión de recursos en talleres industriales.

En cuanto a la asignación de tareas en problemas de tipo Task Assignment Problem (TAP), Savić et al. [14] demostraron que los AG pueden encontrar soluciones óptimas o cercanas al óptimo en espacios de búsqueda complejos, superando métodos deterministas y heurísticos en términos de tiempo de ejecución y calidad de resultados. De forma similar, Wang [15] aplicó algoritmos genéticos al diseño automático de maquinaria, mostrando mejo-

ras significativas frente a métodos tradicionales en precisión y tiempos de procesamiento.

Por otra parte, investigaciones sobre heurísticas industriales, como las reglas de despacho SPT, EDD y FCFS, han demostrado ser métodos simples y eficientes para problemas básicos de secuenciación, pero con limitaciones marcadas en problemas con múltiples restricciones o estructuras complejas de precedencia [10, 6]. Esta brecha entre simplicidad y capacidad de adaptación es precisamente lo que motiva la comparación entre SPT y algoritmos evolutivos en el presente estudio.

En conjunto, estos antecedentes evidencian que los algoritmos genéticos han sido utilizados con éxito en diversos dominios de la optimización industrial, mientras que las heurísticas tradicionales siguen siendo herramientas fundamentales como métodos comparativos o de referencia. En el caso particular de un taller industrial basado en un sistema ERP, la literatura existente es limitada. Por ello, el presente proyecto aporta un enfoque novedoso al integrar un algoritmo genético dentro de un entorno simulado que replica el flujo real de un taller automotriz, comparándolo directamente con una heurística industrialmente aceptada como SPT. Esta comparación permite evaluar el potencial de los AG para mejorar la asignación de tareas y la eficiencia operativa en un contexto aplicado.

3.4. Contexto Operativo del Taller Recti-Ram

Se utiliza un módulo de configuración que centraliza toda la información estructural del taller requerida para la simulación y los algoritmos de asignación. Este artefacto actúa como la representación formal del sistema productivo real y contiene los catálogos de tareas, las relaciones de precedencia, la clasificación de operarios, sus habilidades, la duración base de cada operación y las reglas operativas fundamentales. Fue construido a partir del levantamiento de información realizado con la empresa Recti-Ram y sirve como la "fuente de verdad" sobre la cual el generador de instancias, el simulador, el algoritmo genético y el método SPT basan todas sus decisiones.

Para implementar y evaluar los algoritmos de asignación desarrollados en este trabajo fue necesario construir una representación formal del funcionamiento real del taller. Esta modelación constituye un puente entre el dominio operativo del taller y los métodos computacionales descritos en capítulos posteriores. Con este fin, se diseñó un archivo de configuración denominado

`taller_config`, el cual consolida la estructura de tareas, las relaciones de precedencia, las habilidades de los operarios, la disponibilidad de repuestos, la duración base de las actividades y el horizonte de planificación diario.

En términos de implementación, este archivo no es un texto descriptivo sino un módulo Python con estructuras explícitas que actúan como parámetros del modelo. El diccionario `TAREAS` asigna un identificador entero a cada operación real del taller (35 en total), y `TAREAS_INV` permite recuperar el nombre a partir del identificador para fines de trazabilidad y reporte. `PRECEDENCIAS` define, para cada tarea, la lista de prerequisitos técnicos que deben completarse antes de su ejecución. `DURACIONES_BASE` fija los tiempos nominales de proceso por tipo de tarea, usados como referencia para generar duraciones diarias con variación controlada. `BLOQUES` y `CULATAS` agrupan los IDs de operarios por frente de trabajo, mientras que `OPERARIOS_APTOS` establece la matriz de habilidades: para cada operario, el subconjunto exacto de tareas que puede ejecutar. Finalmente, `HORIZONTE` define el límite temporal diario (480 minutos) que restringe la programación factible.

Esta separación entre configuración estructural fija (catálogos, precedencias, habilidades y horizonte) y parámetros variables por instancia (órdenes del día, tareas activas, repuestos disponibles y duraciones ajustadas) permite mantener consistencia operativa en todos los experimentos y, al mismo tiempo, introducir variabilidad controlada para comparar los algoritmos en escenarios diversos.

3.4.1. Flujo operativo y restricciones reales

El taller de reparación de motores opera bajo un flujo secuencial condicionado por restricciones de precedencia, disponibilidad de repuestos, habilidades del personal e interacción entre tareas dentro de una misma orden de trabajo. Cada día se recibe un conjunto de órdenes de trabajo que agrupan tareas mecánicas con distintos requerimientos técnicos, y el objetivo operativo es completar la mayor cantidad posible de dichas tareas dentro de la jornada laboral. El proceso presenta dificultades estructurales:

- Los operarios poseen habilidades heterogéneas.
- Las órdenes contienen dependencias internas que condicionan el flujo.
- Los repuestos no siempre están disponibles.

- El tiempo límite de jornada restringe la cantidad de tareas ejecutables.
- Tareas pertenecientes a una misma orden no pueden solaparse.

Operativamente, el taller funciona bajo este flujo conceptual:

1. Llegan varias órdenes de trabajo, cada una con un conjunto de tareas.
2. Cada tarea requiere un operario apto y, en algunos casos, un repuesto.
3. Algunas tareas dependen de otras dentro de la misma orden.
4. Los operarios ejecutan las tareas secuencialmente en una jornada limitada.
5. El objetivo es completar la mayor cantidad posible de tareas.

Este flujo sirve como referencia para validar la simulación planteada en la formalización del problema.

3.4.2. Familias y distribución empírica de tareas

El proceso completo de reparación de motores en un taller de rectificación se compone de múltiples operaciones que pueden agruparse en cuatro grandes familias, según la naturaleza mecánica y el componente del motor intervenido:

- **Tareas de Bloque**: incluyen inspección, pruebas hidrostáticas, rectificación de cilindros, encamisado, glaseado y reconstrucción de bancadas.
- **Tareas de Culata**: comprenden instalación y fabricación de guías, rectificación de válvulas, cepillado de culata, adaptación y reconstrucción de alojamientos.
- **Tareas de Cigüeñal**: incluyen operaciones como metalizado, pulido, rectificación de muñones y balanceo del cigüeñal.
- **Tareas de Bielas**: abarcan asentado, instalación de pistón y armado del conjunto biela–pistón.

El conjunto completo está compuesto por 35 tipos de tareas identificadas como las más asignadas y validadas directamente con el taller, cada una asignada a una de estas cuatro familias. Durante la fase de levantamiento

de información se revisaron las órdenes de trabajo típicas manejadas por Recti-Ram. Con base en esta revisión y en entrevistas con el personal técnico se estimó la proporción mensual en que aparecen cada una de las familias de tareas. El resultado permitió establecer la siguiente distribución operativa, que se utiliza posteriormente para la generación de instancias de prueba:

- **60 % Culatas**: la rectificación de culatas constituye la mayor demanda mensual.
- **20 % Bloque**: tareas costosas en maquinaria pero menos frecuentes.
- **10 % Cigüeñal**.
- **10 % Bielas**.

3.4.3. Operarios y habilidades técnicas

La empresa cuenta con dos grupos principales de operarios:

- **Operarios especializados en Bloque**: un grupo de 6 operarios entrenados para tareas de rectificación y mecanizado pesado.
- **Operarios especializados en Culata**: un grupo de 12 operarios capacitados en procesos de precisión asociados a la culata.

Ambos grupos poseen, además, habilidades parciales o complementarias para tareas de Cigüeñal y Bielas, lo que se refleja directamente en la matriz de aptitudes utilizada en el modelo (`OPERARIOS_APTOS`). Cada operario cuenta con un subconjunto específico de tareas que puede ejecutar, lo cual introduce restricciones reales en la asignación y limita el espacio de búsqueda del algoritmo genético, obligándolo a considerar soluciones factibles desde el punto de vista operativo.

Capítulo 4

Formalización del problema

El presente capítulo describe la estructura formal del problema de programación diaria del taller industrial, incluyendo la caracterización del flujo operativo, la definición matemática de los elementos que componen el sistema y la modelación del comportamiento del taller bajo sus restricciones reales. Los elementos descriptivos del funcionamiento real del taller se documentan en el capítulo de Marco de Referencia; aquí se retoman como insumo para la formulación matemática. Esta formalización constituye la base para el desarrollo posterior del algoritmo genético y el método SPT, presentados en los capítulos siguientes.

4.1. Estudio del flujo de trabajo del taller industrial

El taller de reparación de motores opera bajo un flujo secuencial condicionado por restricciones de precedencia, disponibilidad de repuestos, habilidades del personal e interacción entre tareas dentro de una misma orden de trabajo. Cada día se recibe un conjunto de órdenes de trabajo que agrupan tareas mecánicas con distintos requerimientos técnicos, y el objetivo operativo es completar la mayor cantidad posible de dichas tareas dentro de la jornada laboral.

Operativamente, la jornada sigue el siguiente comportamiento conceptual:

1. Llegan varias órdenes de trabajo, cada una con un conjunto de operaciones asociadas.

2. Se identifican los recursos requeridos para cada operación: operarios aptos y repuestos específicos.

3. Algunas tareas deben esperar a que otras concluyan dentro de una misma orden.

4. Los operarios ejecutan las tareas secuencialmente dentro del horizonte diario disponible.

5. Se prioriza maximizar el número de tareas completadas sin violar las restricciones operativas.

Este flujo describe el contexto real que posteriormente se formaliza para alimentar la simulación y los algoritmos de optimización.

4.1.1. Ejemplo de Estructura de Precedencias en una Orden Real

La Figura 4.1 muestra la estructura de dependencias de una orden de trabajo real del taller. Cada nodo representa una tarea con su identificador y nombre, mientras que las flechas direccionadas indican la relación de precedencia: una tarea solo puede iniciarse una vez que sus tareas predecesoras hayan sido completadas. Este tipo de restricciones son típicas en procesos de reparación de motores, donde ciertas operaciones (como pruebas de funcionamiento) deben realizarse al final, y otras (como rimar guías) pueden ejecutarse en paralelo siempre que no sea por el mismo operario. El diagrama ilustra cómo las restricciones de precedencia generan un grafo dirigido acíclico (DAG) que debe ser respetado durante la programación de las tareas.

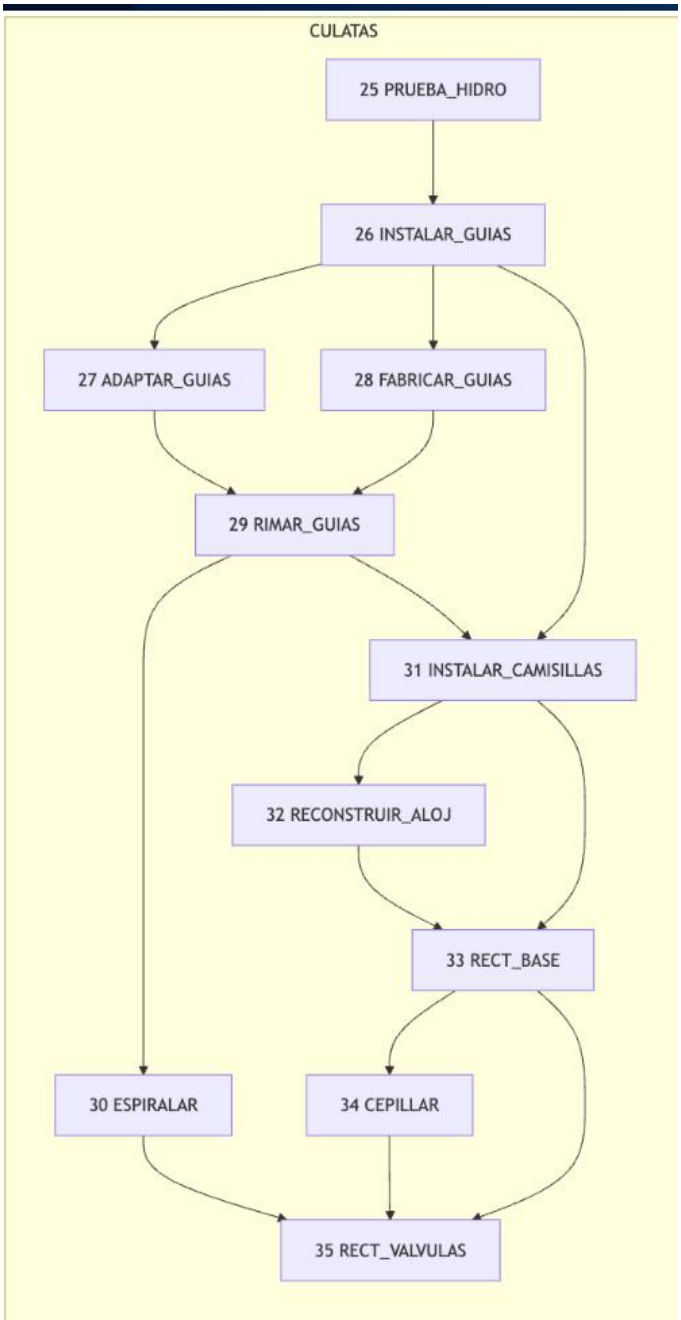


Figura 4.1: Estructura de precedencias de una orden de trabajo real del taller. Los nodos representan tareas identificadas por número y nombre, y las flechas indican relaciones de dependencia donde una tarea no puede iniciarse hasta que todas sus predecesoras hayan sido completadas. Este grafo dirigido acíclico es típico en procesos de reparación de motores.

4.2. Supuestos operativos y alcance

Los supuestos del modelo parten de la caracterización levantada con el taller Recti-Ram. Se considera que:

- La duración base de cada tipo de tarea es conocida y corresponde a tiempos promedio validados con expertos.
- Las habilidades de cada operario son estáticas durante la jornada y definen qué tareas puede ejecutar.
- Las precedencias sólo aplican dentro de una misma orden de trabajo.
- La disponibilidad de repuestos se conoce al inicio del día y se representa mediante parámetros binarios.
- Cada operario cuenta con un horizonte máximo de 480 minutos, equivalente a una jornada de 8 horas.

Estos supuestos delimitan el alcance del problema y sirven como base para la formalización matemática descrita a continuación.

4.3. Formalización Matemática del Problema

4.3.1. Conjuntos y parámetros fundamentales

La estructura formal se define sobre los conjuntos:

$$\begin{aligned} OT &= \{1, \ldots, m\} && \text{Órdenes de trabajo,} \\ T_d &= \{1, \ldots, n\} && \text{Tareas concretas del día,} \\ O &= \{1, \ldots, r\} && \text{Operarios disponibles,} \\ T &= \{1, \ldots, q\} && \text{Tipos de tarea identificados.} \end{aligned}$$

Los parámetros que caracterizan el sistema son:

$$\begin{aligned} p(t) : T \to \mathbb{N} &\quad \text{Duración base del tipo } t, \\ E(t) \subseteq O &\quad \text{Operarios aptos para el tipo } t, \\ P(t) \subseteq T &\quad \text{Tipos que deben preceder a } t, \\ R_{ot} \in \{0,1\}^{|T|} &\quad \text{Disponibilidad de repuestos por orden,} \\ H = 480 &\quad \text{Horizonte diario en minutos.} \end{aligned}$$

Para cada tarea $i \in T_d$ se conocen los parámetros $ot(i) \in OT$ y $tipo(i) \in T$.

4.3.2. Variables de decisión y auxiliares

La formalización utiliza el siguiente conjunto de variables:

- $x_i \in \{0, 1\}$: indica si la tarea i se ejecuta dentro del horizonte.
- $a_i \in O$: operario asignado a la tarea i.
- $s_i \geq 0$ y $f_i = s_i + p(tipo(i))$: inicio y fin de la tarea i.
- $tpo_o \geq 0$: tiempo acumulado utilizado por el operario o.
- $comp_{ot}$: conjunto de tipos ya completados en la orden ot.
- occ_{ot}: conjunto de intervalos ocupados por tareas aceptadas dentro de la orden ot.

Las tres últimas variables operan como estados auxiliares dentro de la simulación y permiten verificar factibilidad al evaluar cada cronograma candidato.

4.3.3. Estructura temporal

El tiempo se modela como un intervalo continuo $[0, H]$. Cada tarea i programada por el operario a_i ocupa el intervalo $[s_i, f_i)$, donde $f_i = s_i + p(tipo(i))$. El tiempo acumulado por operario se actualiza mediante $tpo_{a_i} \leftarrow f_i$ cuando se acepta la tarea, garantizando que la jornada individual no exceda H.

4.3.4. Restricciones operativas formales

La factibilidad de una solución requiere cumplir simultáneamente las siguientes restricciones:

$$a_i \in E(tipo(i)) \qquad \forall i \in T_d \tag{4.1}$$

$$R_{ot(i)}(tipo(i)) \geq x_i \qquad \forall i \in T_d \tag{4.2}$$

$$P(tipo(i)) \subseteq comp_{ot(i)} \qquad \forall i \in T_d \text{ con } x_i = 1 \tag{4.3}$$

$$f_i \leq H \qquad \forall i \in T_d \tag{4.4}$$

$$[s_i, f_i) \cap occ_{ot(i)} = \emptyset \qquad \forall i \in T_d \text{ con } x_i = 1 \tag{4.5}$$

La restricción (4.1) garantiza que cada tarea se asigne a un operario competente, mientras que (4.2) asegura la existencia de repuestos antes de iniciar la operación. La relación de precedencia (4.3) obliga a que los tipos requeridos se encuentren en el conjunto de tareas completadas de la misma orden. La cota temporal (4.4) impide exceder la jornada y (4.5) evita traslapos entre tareas pertenecientes a la misma orden de trabajo.

Adicionalmente, la coherencia entre la variable binaria x_i y el cronograma se mantiene mediante las implicaciones:

$$\begin{aligned} &x_i = 0 \Rightarrow a_i \text{ y } s_i \text{ quedan sin asignar,} \\ &x_i = 1 \Rightarrow s_i = tpo_{a_i}, \quad f_i = s_i + p(tipo(i)). \end{aligned}$$

4.3.5. Definición formal del problema de optimización

El objetivo operativo consiste en maximizar el número de tareas completadas durante la jornada, lo cual se expresa como:

$$\begin{aligned} \max_{x,a,s} \quad & \sum_{i \in T_d} x_i \\ \text{s.a.} \quad & \text{Restricciones (4.1)–(4.5),} \\ & x_i \in \{0,1\},\ a_i \in O,\ s_i \geq 0, \quad \forall i \in T_d. \end{aligned}$$

La solución del problema proporciona un cronograma factible que respeta las reglas industriales y utiliza de forma eficiente la capacidad diaria disponible.

4.4. Representación computacional del sistema

Para ejecutar la simulación y alimentar los algoritmos, la formalización anterior se implementa mediante una configuración digital que integra todos los elementos estructurales del taller:

- **Catálogo de tareas**: lista de los tipos contenidos en T, cada uno con su duración base $p(t)$ y la familia operativa correspondiente.
- **Matriz de habilidades**: representación explícita de $E(t)$, indicando qué operarios están calificados para cada tipo de tarea.

- **Relaciones de precedencia**: estructuras de datos que codifican $P(t)$ y permiten validar dependencias dentro de una misma orden.
- **Parámetros diarios**: conjunto de tareas T_d, asociación con órdenes $ot(i)$ y disponibilidad de repuestos R_{ot} derivada del levantamiento de información real.
- **Horizonte y estados auxiliares**: valores iniciales de tpo_o, $comp_{ot}$ y occ_{ot} que se actualizan durante la evaluación de cada solución.

Esta representación computacional garantiza consistencia entre la formalización matemática y el comportamiento del simulador que se utiliza en los capítulos posteriores para evaluar SPT y el algoritmo genético.

Capítulo 5

Desarrollo de la Solución

La asignación de tareas en talleres industriales suele realizarse mediante procesos manuales que dependen de la experiencia del operario y de la disponibilidad momentánea de recursos. Este enfoque puede generar sobrecarga de trabajo, tiempos inactivos, secuencias operativas subóptimas y retrasos en la entrega de órdenes, especialmente cuando cada tarea requiere habilidades técnicas específicas, disponibilidad de repuestos y una coordinación adecuada con otras actividades del proceso de reparación. El problema resultante puede entenderse como una variante del Job Shop Scheduling con restricciones adicionales, como precedencias internas por orden de trabajo, diferencias en la aptitud de los operarios y un horizonte diario limitado. Esta combinación de factores da lugar a un espacio de búsqueda complejo y no lineal, típico de problemas NP-hard, lo que dificulta resolverlo mediante métodos exactos en tiempos razonables.

5.1. Arquitectura General del Sistema

Antes de detallar los componentes de optimización y simulación, se presenta una vista estructural del flujo completo implementado en el proyecto. Esta arquitectura muestra cómo se conectan los módulos desde la configuración base del taller hasta la consolidación final de métricas.

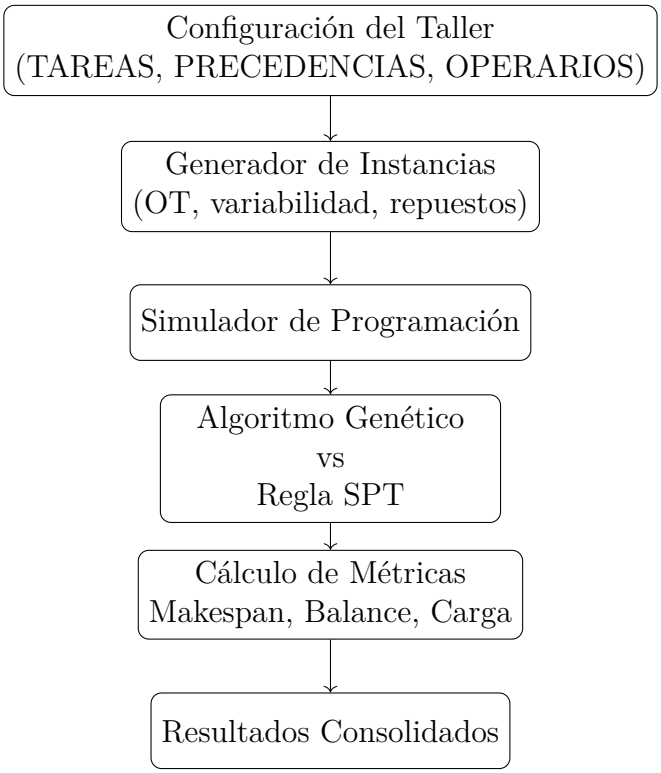


Figura 5.1: Arquitectura general del sistema desarrollado. El módulo de configuración alimenta el generador de instancias, el cual produce escenarios evaluados por el simulador y los algoritmos de asignación.

5.2. Entorno de Desarrollo

El sistema se implementó en el intérprete Python disponible en el entorno local (`python3 --version` reporta 3.14.0), aprovechando su compatibilidad con heurísticas evolutivas y las utilidades de la biblioteca estándar. La portabilidad se garantizó mediante un entorno virtual administrado con `venv`, herramienta incluida en la propia distribución de Python.

La ejecución requiere dos dependencias externas: **NumPy 1.26.4** para el tratamiento eficiente de arreglos y cálculos numéricos, y **PyGAD 3.2.0** como base de la implementación del algoritmo genético. Las rutinas de aleatoriedad se apoyan en el módulo `random`, parte del núcleo de Python, por lo que no demanda instalación adicional. Con esta configuración mínima se reproducen

todas las etapas del proyecto sin depender de librerías adicionales.

5.3. Metodología de Gestión del Desarrollo

La implementación del presente proyecto se realizó bajo la metodología **Kanban**, un enfoque de gestión visual basado en el flujo de trabajo continuo originario de la manufactura esbelta japonesa. Kanban se estructura alrededor de tres principios fundamentales: (i) visualización del trabajo mediante tableros que muestran tareas en diferentes estados (por hacer, en progreso, completado); (ii) limitación del trabajo en proceso (WIP, *Work In Progress*) para evitar sobrecarga y mejorar la concentración en tareas actuales; y (iii) mejora continua mediante la retroalimentación y ajuste dinámico del flujo.

Para este proyecto, Kanban resultó especialmente útil por las siguientes razones:

1. **Transparencia del progreso:** El tablero Kanban permitió visualizar en tiempo real el estado de cada componente del proyecto (adquisición de requisitos, desarrollo del simulador, implementación del AG, validación experimental, redacción de documentación). Esta transparencia facilitó la identificación rápida de cuellos de botella y permitió priorizar esfuerzos de forma ágil.

2. **Flexibilidad ante cambios:** Durante el desarrollo emergieron necesidades imprevistas, tales como ajustes en la representación del cromosoma, recalibración de parámetros genéticos y expansión del conjunto de instancias de prueba. Kanban permite incorporar estos cambios sin replanificación exhaustiva, adaptándose orgánicamente al flujo existente.

3. **Mejora iterativa:** Kanban incentiva la reflexión periódica sobre el proceso. Retrospectivas frecuentes facilitaron la identificación de ineficiencias, como la necesidad de automatizar regeneración de instancias o mejorar la documentación de parámetros genéticos, conduciendo a mejoras concretas en la productividad.

Esta aproximación resultó particularmente alineada con la naturaleza iterativa de la optimización mediante algoritmos genéticos, permitiendo paralelizar validaciones experimentales mientras se avanzaba en refinamientos del código.

5.4. Enfoque de Solución: Algoritmos Genéticos (AG)

Frente a la complejidad operativa descrita, se implementa un enfoque de optimización basado en Algoritmos Genéticos (AG) cuya configuración se adapta específicamente al flujo de trabajo del taller. Este enfoque permite representar simultáneamente la asignación de operarios y la secuenciación de las tareas de cada orden de trabajo, incorporando las restricciones reales del sistema —habilidades requeridas, disponibilidad de insumos, dependencias entre actividades y límite diario de jornada— dentro del proceso evolutivo. De esta manera, el AG actúa como un mecanismo que genera y mejora iterativamente configuraciones de programación factibles y alineadas con las condiciones operativas del entorno industrial.

5.4.1. Representación del Individuo

La solución que manipula el algoritmo genético se estructura mediante un cromosoma que integra explícitamente las dos decisiones fundamentales del problema: la asignación de operarios y el orden de intento de programación de cada tarea. Para ello, cada individuo se modela como un vector de dimensión $2n$, donde n corresponde al número de tareas del día.

$$\mathbf{x} \;=\; \big[\, y_1,\, y_2,\, \ldots,\, y_n \;\big|\; \pi_1,\, \pi_2,\, \ldots,\, \pi_n \big]$$

donde:

$$y_i \in E(\text{tipo}(i)), \qquad \pi_i \in [0,1], \qquad i = 1, \ldots, n.$$

La primera mitad del cromosoma contiene la asignación propuesta para cada tarea. El gen y_i especifica el operario responsable de ejecutar la tarea i, y su valor está restringido al conjunto de operarios aptos $E(\text{tipo}(i))$, tal como se definió en la formalización del problema. Con esto, la representación inicial garantiza que cada asignación respete las habilidades mínimas requeridas por el taller.

La segunda mitad del cromosoma codifica la prioridad asociada a cada tarea mediante un valor continuo $\pi_i \in [0,1]$. Estas prioridades determinan el orden en que las tareas serán consideradas durante el proceso de evaluación, orientando la construcción del flujo operativo. Este mecanismo permite

explorar distintas secuencias de programación sin necesidad de trabajar directamente con permutaciones, lo cual simplifica el espacio de búsqueda y facilita la integración con las restricciones industriales del sistema.

La combinación de ambas partes —asignación y prioridad— permite que cada individuo represente un cronograma candidato completo, el cual será posteriormente evaluado mediante la función de aptitud. Esta estructura dual ofrece flexibilidad para adaptarse a las condiciones reales del taller, manteniendo al mismo tiempo una codificación compacta y eficiente para la evolución genética.

5.4.2. Evaluación de la Solución

Una vez representado el individuo mediante la asignación de operarios y las prioridades de programación, el siguiente paso consiste en evaluar su desempeño como cronograma operativo. Este proceso corresponde a la simulación interna que ejecuta el algoritmo genético para determinar qué tan viable y eficiente es la solución candidata.

La evaluación parte de separar las tareas asignadas a cada operario a partir del vector $\mathbf{y}$. Posteriormente, cada conjunto T_o se ordena según las prioridades π_i, definiendo así el orden en que las tareas serán consideradas. Para cada tarea, el proceso verifica secuencialmente las restricciones operativas del taller: aptitud del operario, disponibilidad de repuestos, cumplimiento de precedencias internas dentro de la orden de trabajo y ausencia de solapamientos en los intervalos ya programados. Si la tarea satisface dichas condiciones y su tiempo de finalización no excede el horizonte diario del operario, se programa y se actualizan las estructuras internas de la evaluación. En caso contrario, se registra la penalización correspondiente.

Al finalizar el recorrido, la evaluación produce un conjunto de métricas que describen el comportamiento del cronograma: número de tareas ejecutadas, violaciones de precedencia, faltas de repuestos, asignaciones no aptas, solapamientos internos, excedentes del horizonte, balance de carga y tiempo máximo de finalización. Estas métricas alimentan posteriormente la función de aptitud.

Representación Formal del Proceso de Evaluación. La evaluación del individuo sigue un procedimiento estructurado en seis pasos principales, como se describe a continuación:

1. Separación por operario. Dado un individuo $\mathbf{x} = (\mathbf{y}, \boldsymbol{\pi})$, se construye el conjunto de tareas asignadas a cada operario:

$$T_o = \{i \in T_d : y_i = o\} \quad \forall o \in O$$

Cada conjunto T_o contiene las tareas asignadas al operario o según el vector de asignación $\mathbf{y}$.

2. Ordenamiento por prioridad. Para cada operario, se ordena el conjunto T_o según los valores de prioridad π_i contenidos en la segunda mitad del cromosoma:

$$\text{Orden}_o = \text{sort}(T_o, \text{clave} = \pi_i)$$

donde el operador sort organiza las tareas en orden ascendente (o descendente, según la implementación) de prioridad. Esto define el orden en que cada tarea será considerada para su programación.

3. Intento de programación. Sea s_i el inicio tentativo y f_i el fin tentativo de la tarea i. Se asigna:

$$s_i = tpo[o], \quad f_i = s_i + p(i)$$

donde $tpo[o]$ es el tiempo acumulado (tiempo posterior ocupado) del operario o, y $p(i)$ es la duración de la tarea i.

4. Validación de restricciones. Antes de confirmar la programación, se verifica que se cumplan:

1. **Aptitud:** $y_i \in E(\text{tipo}(i))$
2. **Disponibilidad de repuesto:** $\text{rep}(OT(i)) = \text{disponible}$
3. **Precedencias:** $\forall t' \in P(i) : t' \in \text{comp}[OT(i)]$
4. **Posible ubicación en agenda:** $s_i = tpo[o], f_i = s_i + p(i)$
5. **No solapamiento dentro de la misma orden:** $\forall [s', f') \in \text{occ}[OT(i)] : \neg(s_i < f' \wedge f_i > s')$
6. **Horizonte diario:** $f_i \leq H$

Si todas las condiciones se satisfacen, la tarea se programa y se registra su intervalo $[s_i, f_i)$ en $\text{occ}[OT(i)]$. En caso contrario, se registra la penalización correspondiente.

Actualización de estado. Si la programación es exitosa:

$$\text{programar}(i, o), \quad tpo[o] \leftarrow f_i, \quad \text{comp}[OT(i)] \leftarrow \text{comp}[OT(i)] \cup \{i\}$$

Se actualiza el tiempo acumulado del operario y el conjunto de tareas completadas de la orden.

Registro de penalización. En caso contrario, se registra:

$$\text{registrar penalización correspondiente}$$

Este proceso se repite para cada tarea en Orden_o de cada operario $o \in O$.

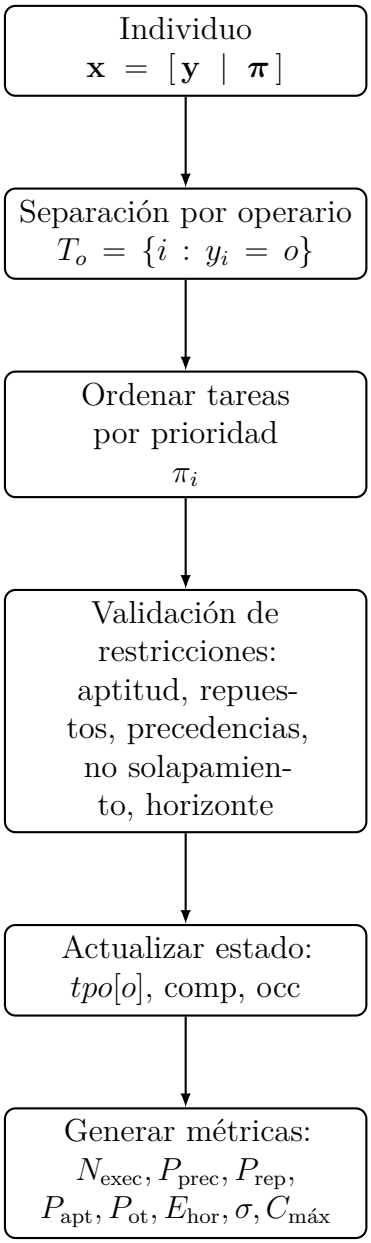


Figura 5.2: Flujo del proceso de evaluación de un individuo.

5.4.3. Función de Aptitud

La función de aptitud asigna un valor cuantitativo al desempeño de cada individuo a partir de las métricas generadas durante la evaluación. Su diseño busca orientar el proceso evolutivo hacia soluciones que maximicen la cantidad de tareas programadas y minimicen las violaciones operativas. Para ello se emplean ponderaciones que reflejan la importancia relativa de cada componente dentro del contexto del taller.

El término principal corresponde al número de tareas ejecutadas, ponderado con un valor significativamente mayor, dado que la productividad diaria es el objetivo fundamental del proceso de programación. Las penalizaciones restantes representan violaciones a restricciones que afectan la factibilidad o la calidad de la programación. Estas penalizaciones se escalan según su impacto operativo: las restricciones relacionadas con aptitud, precedencias y repuestos reciben pesos más altos, ya que su incumplimiento impide la ejecución de la tarea. Los solapamientos dentro de una orden se penalizan con magnitud similar, pues afectan la coherencia del proceso de reparación. Por su parte, los excedentes del horizonte, el balance de carga y el makespan tienen ponderaciones menores, dado que no invalidan la tarea pero sí influyen en la eficiencia global del cronograma.

La función de aptitud se define formalmente como:

$$\begin{aligned}\text{fitness}(\mathbf{x}) = 100 \cdot N_{\text{exec}} \\ - 10 \cdot P_{\text{prec}} \\ - 10 \cdot P_{\text{rep}} \\ - 15 \cdot P_{\text{apt}} \\ - 12 \cdot P_{\text{ot}} \\ - 0{,}1 \cdot E_{\text{hor}} \\ - 15 \cdot \sigma \\ - 10 \cdot C_{\text{máx}}\end{aligned}$$

donde:

- N_{exec}: número de tareas ejecutadas satisfactoriamente.
- P_{prec}: penalización acumulada por violaciones de precedencias.
- P_{rep}: penalización acumulada por falta de repuestos disponibles.

- P_{apt}: penalización acumulada por asignaciones a operarios no aptos.
- P_{ot}: penalización acumulada por solapamientos dentro de órdenes de trabajo.
- E_{hor}: excedente total del horizonte diario acumulado.
- σ: desviación estándar de la carga de trabajo entre operarios (desbalance).
- $C_{\text{máx}}$: makespan o tiempo máximo de finalización del cronograma.

Los coeficientes responden a los siguientes criterios operativos:

- **100 en** $N_{\mathbf{exec}}$: maximizar la ejecución de tareas es el objetivo operativo principal. Cada tarea adicional contribuye significativamente a la aptitud.
- **15 en** $P_{\mathbf{apt}}$: una asignación a un operario no apto hace imposible la ejecución de la tarea, por lo que recibe la penalización más alta entre restricciones.
- **12 en** $P_{\mathbf{ot}}$: los solapamientos dentro de una orden de trabajo comprometen la coherencia del proceso de reparación.
- **10 en** $P_{\mathbf{prec}}$ **y** $P_{\mathbf{rep}}$: la falta de precedencias o repuestos impide ejecutar la tarea, recibiendo penalizaciones significativas.
- **0.1 en** $E_{\mathbf{hor}}$: el excedente del horizonte indica ineficiencia temporal pero no invalida la tarea, recibiendo baja penalización.
- **15 en** σ: el desbalance de carga se penaliza fuertemente para promover una distribución equitativa de trabajo entre operarios.
- **10 en** $C_{\text{máx}}$: favorece cronogramas más compactos, reduciendo el tiempo total de ejecución.

En conjunto, estas ponderaciones guían la búsqueda del algoritmo genético hacia soluciones factibles, eficientes y alineadas con las necesidades reales del entorno industrial.

5.4.4. Operadores Genéticos y Parámetros del AG

El proceso evolutivo implementado en este proyecto se rige por los operadores clásicos de los algoritmos genéticos. Cada uno cumple una función específica dentro del ciclo evolutivo y puede ajustarse mediante parámetros que equilibran exploración y explotación del espacio de búsqueda.

Selección. La selección por ranking ordena a los individuos según su aptitud y asigna probabilidades en función de su posición relativa dentro de la población, lo cual evita que diferencias extremas en la aptitud dominen prematuramente la evolución y favorece una presión selectiva estable. Este tipo de mecanismos ha sido ampliamente documentado en la literatura de algoritmos genéticos, donde se considera una alternativa robusta frente a métodos proporcionales al fitness [5, 16].

Cruce. El operador de cruce de un punto (single-point crossover) divide el cromosoma en dos segmentos y recombina material genético de ambos progenitores, promoviendo variabilidad y manteniendo estructuras internas útiles. Este enfoque se ha utilizado tradicionalmente en problemas combinatorios y de asignación debido a su simplicidad y capacidad para preservar bloques funcionales de información [5, 14].

Mutación. La mutación introduce variación en la población mediante modificaciones aleatorias en los genes, lo que ayuda a prevenir la convergencia prematura hacia óptimos locales. La literatura coincide en que la mutación es esencial para mantener la diversidad y garantizar una exploración adecuada del espacio de soluciones [5, 16]. En problemas de planificación y asignación, mutaciones sobre componentes discretos del cromosoma permiten generar configuraciones no presentes en los progenitores, ampliando el alcance de la búsqueda.

Elitismo. El elitismo asegura que los mejores individuos de una generación se preserven sin alteraciones en la siguiente, lo que contribuye a estabilizar la evolución y acelerar la convergencia. Su pertinencia ha sido demostrada en problemas de scheduling y asignación, donde retener soluciones de alta calidad evita retrocesos en el desempeño global del algoritmo [13].

Parámetros generales. Además de los operadores evolutivos, el algoritmo genético depende de un conjunto de parámetros que controlan la dinámica del proceso de búsqueda. Entre ellos se encuentran el tamaño de la población, que determina cuántos individuos se consideran en cada generación; la cantidad de generaciones ejecutadas, que define la profundidad evolutiva del proceso; y la probabilidad de mutación, que regula el nivel de variación introducido en cada ciclo. Otros parámetros relevantes incluyen la cantidad de padres seleccionados para la reproducción, los criterios de reemplazo y la semilla aleatoria utilizada para garantizar reproducibilidad. Estos elementos no constituyen operadores en sí mismos, pero modulan la intensidad de la exploración y la velocidad de convergencia del algoritmo, y por tanto se ajustan experimentalmente en las secciones de configuración base y calibración.

5.4.5. Configuración Base del Algoritmo Genético

Con los operadores y parámetros generales establecidos, el siguiente paso es construir una configuración base sobre la cual se realizarán los ajustes experimentales descritos en la sección de calibración. Para ello se fijaron los elementos no sujetos a ensayo en cada experimento. Este arreglo inicial emplea una población de 100 individuos durante 300 generaciones, con diez padres seleccionados, cinco preservados mediante elitismo, selección por *ranking*, cruce de un punto y mutación *random* aplicada al 5 % de los genes. Toda la configuración se ejecuta con semilla aleatoria 42 para garantizar reproducibilidad. Cuando se analiza un parámetro específico (por ejemplo, tipo de cruce o probabilidad de mutación), únicamente se modifica ese componente, manteniendo constantes los restantes valores base.

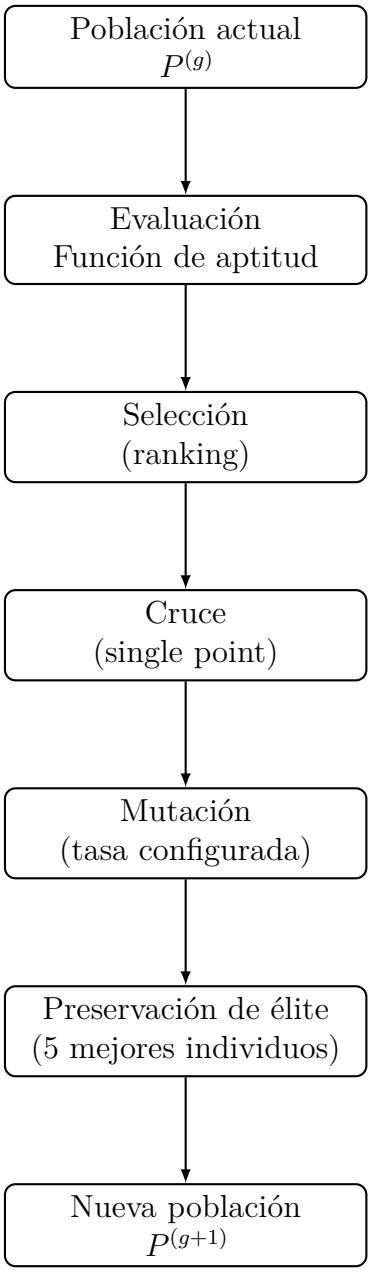


Figura 5.3: Flujo del ciclo evolutivo del algoritmo genético en PyGAD.

5.5. Calibración del Algoritmo Genético

La calibración de parámetros en un algoritmo genético es un paso fundamental para garantizar su rendimiento, dado que aspectos como el operador de cruce, el tamaño de la población o la probabilidad de mutación afectan directamente la convergencia y la estabilidad del proceso evolutivo. La literatura especializada señala que los parámetros de un GA no son independientes entre sí y que su ajuste requiere un proceso sistemático que explore configuraciones alternativas manteniendo constantes los demás factores [5, 16].

Adicionalmente, estudios sobre control de parámetros en metaheurísticas evidencian que valores estáticos pueden ser subóptimos y que la elección adecuada depende del tipo de problema y de la dinámica de la población a lo largo de las generaciones [17, 18]. Por esta razón, siguiendo prácticas reconocidas en la experimentación con metaheurísticas, los ensayos de calibración se realizaron variando un parámetro a la vez, evaluando su efecto sobre el desempeño del algoritmo mientras se mantuvieron constantes los demás componentes.

Este enfoque coincide con recomendaciones de investigaciones previas sobre planificación de tareas con algoritmos genéticos, donde la calibración permite equilibrar calidad de solución, tiempo computacional y estabilidad evolutiva [13, 6].

5.5.1. Evaluación del Operador de Cruce

Se compararon los operadores *single_point* y *two_points* bajo exactamente las mismas condiciones: misma población, mismas instancias, misma semilla aleatoria y la configuración base descrita en la Sub-sección 5.4.5. El experimento se ejecutó sobre veinte instancias independientes, todas simuladas bajo un horizonte operativo de 480 minutos. Los resultados promedio se presentan en la Tabla 5.1.

Cuadro 5.1: Resultados promedio del AG para cada operador de cruce (horizonte de 480 minutos).

Operador de cruce	**Tareas ejecutadas**	**Tareas rechazadas**	**Makespan**	**Carga total**
single_point	26.20	17.55	212.60	613.85
two_points	24.55	14.85	206.10	568.80

Aun cuando *two_points* obtiene un makespan ligeramente menor, *single_point* supera sistemáticamente a su contraparte en las métricas más relevantes para el taller (tareas ejecutadas y carga total). Por ello se seleccionó *single_point* para los experimentos de calibración restantes.

5.5.2. Evaluación del Número de Generaciones

Se evaluaron tres niveles de profundidad evolutiva (300, 800 y 1300 generaciones) manteniendo la misma configuración base. Cada configuración se aplicó sobre veinte instancias independientes con horizonte de 480 minutos. Los resultados promedio se presentan en la Tabla 5.2.

Cuadro 5.2: Resultados promedio del AG para distintos números de generaciones (480 minutos).

Generaciones	Tareas ejecutadas	Tareas rechazadas	Makespan	Carga total
300	**31.40**	6.75	178.58	**758.80**
800	29.90	8.55	174.33	722.60
1300	28.40	8.65	171.40	696.65

El mejor rendimiento se obtiene con 300 generaciones, valor que maximiza tareas ejecutadas y carga total sin incrementar el costo computacional. Las configuraciones más profundas no aportan mejoras y tienden a reducir la diversidad de la población.

5.5.3. Evaluación de la Probabilidad de Mutación

Finalmente se estudiaron probabilidades de mutación de 5 %, 15 % y 25 % empleando veinte instancias independientes. Los resultados promedio se muestran en la Tabla 5.3.

Cuadro 5.3: Resultados promedio del AG para diferentes probabilidades de mutación (480 minutos).

Probabilidad de mutación	Tareas ejecutadas	Tareas rechazadas	Makespan	Carga total
5 %	**34.90**	10.35	178.65	**832.00**
15 %	30.20	**6.65**	**168.75**	712.30
25 %	32.00	8.00	181.45	786.30

Una probabilidad de mutación del 15 % ofrece el mejor equilibrio entre exploración y estabilidad: reduce tareas rechazadas y logra el makespan más bajo sin sacrificar significativamente la productividad, por lo que se adoptó como valor final.

5.5.4. Configuración Final del AG

La calibración descrita anteriormente permitió definir una configuración que equilibra calidad de soluciones y costo computacional. Se utiliza el operador de cruce *single_point* sobre un horizonte de 480 minutos, con 300 generaciones, población de 100 individuos, selección por *ranking*, diez padres por generación, cinco preservados mediante elitismo, mutación *random* aplicada al 15 % de los genes y semilla fija 42. Esta configuración proporciona un desempeño robusto en términos de tareas ejecutadas, carga total y estabilidad evolutiva, además de tiempos de ejecución compatibles con la operación diaria del taller.

Cuadro 5.4: Configuración final del algoritmo genético empleada en las simulaciones.

Parámetro	**Valor seleccionado**
Operador de cruce	*single_point*
Número de generaciones	300
Tamaño de población	100
Método de selección	*rank*
Padres por generación	10
Padres preservados	5
Probabilidad de mutación	15 %
Semilla aleatoria	42

5.6. Enfoque de Solución: Método SPT

Como alternativa al enfoque evolutivo basado en algoritmos genéticos, se consideró una estrategia heurística clásica de secuenciación: la regla SPT

(*Shortest Processing Time*). Este método es ampliamente utilizado en entornos industriales debido a su simplicidad, su bajo costo computacional y su buen desempeño en escenarios donde las restricciones operativas son mínimas. Por estas razones, SPT constituye un punto de referencia habitual en estudios comparativos de métodos de programación.

En su forma tradicional, SPT prioriza las tareas con menor tiempo de procesamiento. No obstante, dado que el problema del taller incorpora múltiples restricciones —precedencias internas, disponibilidad de repuestos, aptitud del operario y un horizonte diario limitado—, la regla fue adaptada para integrarse al entorno de evaluación del sistema.

De este modo, tanto SPT como el algoritmo genético abordan el mismo problema de programación y son plenamente capaces de generar un cronograma operativo completo. La diferencia radica en la lógica que guía la construcción de la solución: mientras el enfoque evolutivo explora múltiples configuraciones mediante un proceso de búsqueda global, SPT define un orden determinista basado en la duración de las tareas.

5.6.1. Modelado del SPT para el Taller

La aplicación de la regla SPT en el taller requiere establecer con claridad la información operativa que interviene en la selección y ordenamiento de las tareas. En este enfoque, SPT actúa directamente sobre los datos de la instancia, priorizando las tareas en función de su tiempo de procesamiento, pero integrado a las restricciones reales del sistema para asegurar un orden viable.

El método utiliza como entrada las tareas pendientes por orden de trabajo, los tiempos de procesamiento por tipo de tarea, las relaciones de precedencia interna, la disponibilidad de repuestos, el conjunto de operarios disponibles y las tareas para las cuales son aptos, así como el horizonte diario máximo permitido.

Con esta información, el algoritmo identifica en cada iteración las tareas listas para ser consideradas. Una tarea se considera lista cuando: (i) todas las tareas precedentes dentro de su orden han sido completadas, (ii) el repuesto correspondiente se encuentra disponible, (iii) su duración está definida y (iv) permanece pendiente en la instancia.

Entre todas las tareas que cumplen estas condiciones, la regla SPT selecciona aquellas con menor tiempo de procesamiento, conformando un conjunto ordenado dinámicamente. Este ordenamiento se actualiza a medida que nuevas

tareas se completan, repuestos se consumen y las relaciones de precedencia modifican el estado del sistema.

El resultado de este proceso es una política determinista que, basada en la duración de las tareas y en el cumplimiento de las restricciones operativas, establece un orden de ejecución coherente con las características del taller. A partir de dicho orden, el entorno de programación determina los horarios de inicio y finalización, respetando los tiempos disponibles, la aptitud del operario asignado y las ventanas temporales de la orden de trabajo.

5.6.2. Representación Formal del Orden SPT

El ordenamiento SPT aplicado al taller puede describirse formalmente como un proceso iterativo que opera sobre el conjunto de tareas pendientes. En cada paso, el método determina el subconjunto de tareas que cumplen las condiciones necesarias para ser consideradas y luego aplica un criterio de priorización basado en el tiempo de procesamiento.

Sea $\mathcal{T}$ el conjunto de tareas pendientes, $p(i)$ la duración de la tarea i, $\mathcal{P}(ot, i)$ el conjunto de prerequisitos internos para la tarea i dentro de la orden ot, $R(ot, i)$ un indicador de disponibilidad del repuesto correspondiente, y $C(ot)$ el conjunto de tareas completadas en dicha orden. El conjunto de tareas listas en la iteración k se define como:

$$\mathcal{L}^{(k)} = \Big\{(ot, i) \in \mathcal{T}^{(k)} \ : \ \mathcal{P}(ot, i) \subseteq C^{(k)}(ot),\ R(ot, i) = 1,\ p(i) \text{ definido}\Big\}.$$

Sobre este conjunto, el criterio SPT establece un ordenamiento ascendente por duración:

$$\text{SPT}(\mathcal{L}^{(k)}) = \text{sort}\Big(\mathcal{L}^{(k)},\ p(i)\Big).$$

Dado que los elementos dentro de una misma orden pueden compartir la misma duración, se emplean criterios secundarios de desempate consistentes con la implementación: primero por identificador de orden y luego por identificador de tarea. Esto asegura un orden total determinista:

$$(ot_a, i_a) \prec (ot_b, i_b) \quad \Longleftrightarrow \quad \begin{cases} p(i_a) < p(i_b), \\ \text{o bien } p(i_a) = p(i_b) \text{ y } ot_a < ot_b, \\ \text{o bien } p(i_a) = p(i_b),\ ot_a = ot_b \text{ y } i_a < i_b. \end{cases}$$

El resultado final es una secuencia ordenada de tareas listas, actualizada dinámicamente en cada iteración a medida que nuevas tareas son completadas y otras entran en condición de ser consideradas.

5.6.3. Aplicación Operativa del SPT en el Taller

Una vez definido el orden SPT de las tareas listas, el sistema procede a construir el cronograma operativo respetando la disponibilidad de los operarios, las restricciones de cada orden de trabajo y el horizonte diario. Este proceso se desarrolla de manera iterativa sobre el conjunto de tareas pendientes, actualizando el estado del sistema a medida que se programan nuevas actividades.

En cada iteración, el método toma la lista ordenada de tareas generada por la regla SPT y selecciona, para cada una de ellas, un operario apto para su ejecución. La asignación se realiza siguiendo un criterio de carga acumulada: entre los operarios capaces de ejecutar la tarea, se elige aquel que tenga el menor tiempo total programado hasta el momento, buscando equilibrar progresivamente la utilización de la capacidad diaria.

Una vez identificado el operario, el sistema determina el primer intervalo disponible donde la tarea puede ser ubicada. Para ello se utiliza una rutina de búsqueda del hueco más temprano, que evalúa: (i) los intervalos ya ocupados por actividades de la misma orden de trabajo, (ii) el tiempo más próximo en el que el operario puede iniciar nuevas tareas, (iii) la duración de la actividad y (iv) el límite superior del horizonte diario.

Si existe un intervalo $[s, f)$ dentro de estos límites donde la tarea puede ejecutarse sin solaparse con otras actividades de la misma orden y sin exceder la jornada asignada, la tarea es programada en dicho espacio. Con esto se actualizan simultáneamente la ocupación temporal de la orden, el tiempo acumulado del operario y el conjunto de tareas completadas.

En caso de que ninguna combinación válida de operario e intervalo esté disponible, la tarea permanece pendiente y será reconsiderada más adelante, siempre que las condiciones del sistema cambien a su favor. El ciclo continúa hasta que no sea posible programar nuevas tareas o todas hayan sido ubicadas dentro del horizonte.

Al finalizar, el método genera el cronograma de tareas con tiempos de inicio y fin, la lista de tareas que no pudieron ser programadas, la ocupación temporal por orden de trabajo, el tiempo total utilizado por cada operario y métricas como el número de tareas ejecutadas y el makespan.

De esta manera, el enfoque SPT produce un cronograma completo y coherente con las restricciones del taller, combinando un orden de secuenciación basado en los tiempos de procesamiento con un proceso operativo de programación ajustado al estado real del sistema.

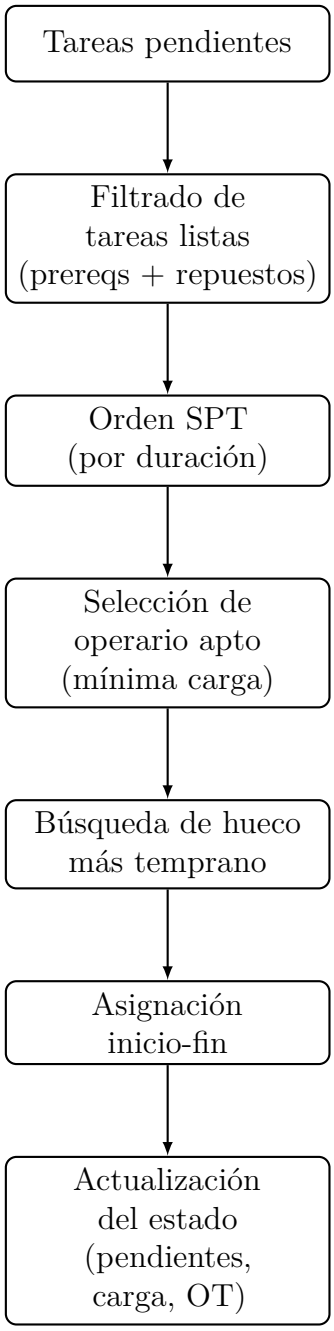


Figura 5.4: Flujo operativo del método SPT en el taller.

5.7. Ejecución de Simulaciones

El módulo de simulación coordina el experimento completo. Para cada instancia generada se ejecutan consecutivamente el método SPT y el algoritmo genético, registrando métricas comparables que luego se consolidan para el análisis.

5.7.1. Procedimiento de Generación de Instancias

Las instancias se generan mediante un componente dedicado a la creación de escenarios, que consume la configuración formal del taller y construye casos completos para la evaluación de ambos métodos. Este generador se implementó para articular los datos base alojados en la configuración del sistema. Se ejecuta sobre la versión de Python disponible (3.14.0 en esta instalación, compatible con el requerimiento original de Python 3.11) y utiliza únicamente librerías estándar: `random` provee la aleatoriedad controlada y `typing` las anotaciones estáticas.

Conceptualmente, la herramienta construye un conjunto de órdenes de trabajo y, para cada una, selecciona un número aleatorio de tareas dentro de rangos configurables. Esta selección respeta las precedencias reales definidas para el taller y puede relajar algunas relaciones de forma probabilística, lo que introduce ramificaciones opcionales sin violar las reglas esenciales del proceso. Las duraciones operativas se derivan de tiempos base catalogados en el módulo de configuración y se modifican aplicando factores de variación acotados para obtener duraciones diferenciadas pero coherentes. En paralelo, se asigna de manera estocástica la disponibilidad de repuestos por tarea y por orden, replicando los cuellos de suministro que inciden en la programación diaria. La Figura 5.5 ilustra una orden de trabajo ejemplo construida con esta lógica de precedencias.

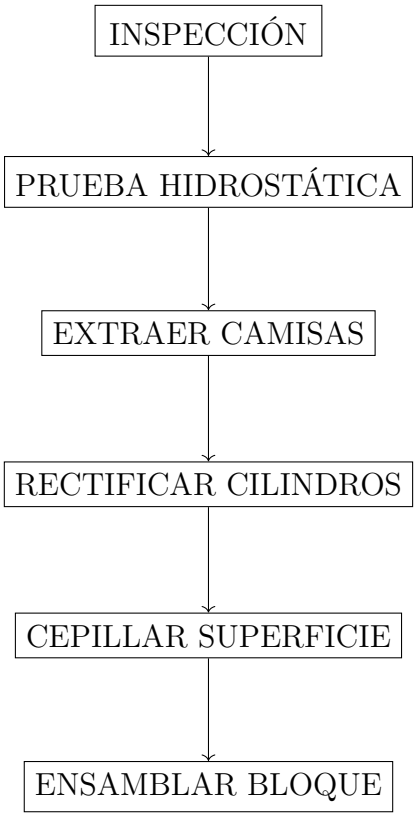


Figura 5.5: Ejemplo ilustrativo de una Orden de Trabajo generada automáticamente por el módulo de instancias. Las tareas seleccionadas respetan las precedencias definidas en el sistema.

La disponibilidad de repuestos se modela con la estructura `repuestos_por_ot: Dict[int, List[int]]`. Para cada orden de trabajo *ot*, la lista binaria asociada se alinea por índice con `mapeo_ot[ot]`: cada posición representa una tarea concreta de esa orden, con valor 1 si el repuesto está disponible y 0 si no lo está. La generación es aleatoria por tarea mediante una prueba de Bernoulli: `random.random() <prob_falta_repuesto`. Si la condición se cumple, se registra 0; en caso contrario, se registra 1. Con el valor por defecto `prob_falta_repuesto = 0.15`, en promedio se obtiene aproximadamente un 15 % de tareas sin repuesto y un 85 % con repuesto disponible.

Cada instancia resultante incorpora todos los elementos necesarios para la simulación: lista de órdenes consideradas, tareas elegidas por orden, mapeo entre órdenes y tareas, duraciones ajustadas, prerequisitos vigentes, estado de

repuestos, plantillas de operarios disponibles y aptos y el horizonte temporal sobre el cual se evalúan los algoritmos. Gracias a este ensamblaje, el generador produce instancias realistas, reproducibles y con suficiente variabilidad para comparar enfoques como el algoritmo genético y la regla SPT. Cuando se requiere analizar múltiples escenarios, el mismo módulo crea lotes completos utilizando una semilla aleatoria, lo que garantiza reproducibilidad o variación controlada.

Parámetros considerados. Cada instancia especifica los elementos estructurales necesarios para recrear un día de operación:

- conjunto de órdenes a atender;
- listado de tareas concretas por orden;
- habilidades y disponibilidad del personal;
- duraciones base de cada tipo de tarea;
- precedencias aplicables;
- indicadores de repuestos disponibles.

Aleatoriedad y distribución de tiempos. El generador recibe como entrada:

- número mínimo y máximo de órdenes,
- número de tareas por orden,
- distribución temporal inducida por $p(t)$,
- disponibilidad probabilística de repuestos,
- habilidades factibles para cada tipo de tarea.

Procedimiento de creación de escenarios

A partir de estos parámetros se ejecuta el siguiente procedimiento:

1. Seleccionar aleatoriamente las órdenes del día y las tareas asociadas.
2. Asignar tipos, duraciones y familias respetando la estructura de precedencias.
3. Generar la disponibilidad de repuestos para cada orden.
4. Determinar los operarios aptos para cada tipo de tarea.
5. Consolidar la instancia final (OT, T_d, O) para su evaluación en la simulación.

Validación del modelo simulado con expertos

Los parámetros del generador se calibraron con el personal de Recti-Ram, confirmando que las duraciones reflejan tiempos reales de procesamiento, las precedencias respetan la secuencia técnica del taller, las matrices de habilidades representan la estructura del personal y el horizonte de 480 minutos coincide con la jornada laboral. Este procedimiento asegura que los escenarios utilizados sean representativos de las condiciones de producción del taller.

Construcción de escenarios de prueba. Para la fase experimental se emplearon combinaciones que exploran distintos números de órdenes, tamaños del conjunto de tareas, niveles de complejidad en las precedencias y disponibilidades de recursos y repuestos. Las instancias incorporan variabilidad en carga operativa, composición de órdenes y mezcla de repuestos, lo que permite comparar el desempeño de SPT y del algoritmo genético en contextos heterogéneos.

5.7.2. Algoritmo de Evaluación Operativa

El componente de simulación determina si una asignación propuesta es factible y cuantifica su desempeño bajo las restricciones del taller; en la práctica, corresponde al algoritmo que implementa la función de aptitud del sistema. El pseudocódigo resumido se presenta a continuación:

Algorithm 1: Simulación operativa de programación de tareas

```
Inicializar estructuras: tpo[o] ← 0, comp[ot] ← ∅, occ[ot] ← ∅;
foreach operario o ∈ O do
    Obtener conjunto de tareas asignadas T_o;
    Ordenar T_o según prioridad descendente;
    foreach tarea i ∈ T_o do
        Verificar aptitud: o ∈ E(tipo(i));
        Verificar repuesto disponible;
        Verificar precedencias cumplidas en comp[ot(i)];
        Calcular intervalo [s_i, f_i) = [tpo[o], tpo[o] + p(tipo(i))];
        if f_i > H then
            rechazar tarea; continuar
        end
        if [s_i, f_i) solapa con algún intervalo en occ[ot(i)] then
            rechazar tarea; continuar
        end
        Aceptar tarea: actualizar tpo[o], comp, occ;
    end
end
```

Este simulador es el núcleo de la evaluación: recibe las soluciones propuestas por cada algoritmo, aplica las restricciones industriales y produce las métricas utilizadas en el análisis comparativo.

5.7.3. Conjunto Final de Escenarios Utilizados en la Comparación

Número y estructura de las instancias

Para la fase experimental se consideraron múltiples instancias generadas bajo diferentes configuraciones de número de órdenes, número de tareas, complejidad de precedencias y disponibilidad de recursos.

Variabilidad incorporada

Las instancias presentan variabilidad en tamaño del conjunto de tareas, carga operativa, composición de órdenes y mezcla de repuestos disponibles.

Justificación de los casos de estudio

Estas instancias permiten evaluar el desempeño de SPT y del algoritmo genético en condiciones diversas, representando escenarios realistas del taller.

5.7.4. Consolidación de Resultados

Para cada instancia se ejecuta primero la heurística SPT y se registra su desempeño; luego se ejecuta el AG para obtener el mejor individuo y ambos resultados se evalúan con el simulador operativo, generando métricas comparables.

Los datos generados por cada ejecución se almacenan en archivos CSV que posteriormente se utilizan en el Capítulo 7 para el análisis cuantitativo.

5.8. Código fuente disponible

La implementación completa del algoritmo genético, el método SPT y el simulador operativo están disponibles en el repositorio de GitHub:

`https://github.com/mariajosepa/algoritmo-tesis-jave.git`

5.9. Limitaciones del Modelo

Aunque el algoritmo genético desarrollado demuestra superior desempeño respecto a heurísticas clásicas, es importante reconocer sus limitaciones en relación al contexto industrial real. Estas limitaciones no invalidan los resultados, pero sí establecen un alcance explícito del modelo y orientan futuras extensiones.

5.9.1. Disponibilidad de Máquinas no Contemplada

La limitación más significativa del modelo actual radica en que **el algoritmo genético no incorpora la disponibilidad de máquinas (equipos o recursos específicos) como restricción de programación**. El modelo únicamente representa restricciones respecto a la disponibilidad de operarios (asignación de habilidades) y operarios específicos para cada tipo de tarea.

En consecuencia, es posible que el algoritmo asigne simultáneamente dos o más tareas a operarios distintos en intervalos temporales que se solapan, cuando ambas tareas requieren el uso de la misma máquina o equipo compartido. En el contexto real del taller, esta situación es inviable: una máquina no puede ser utilizada por múltiples operarios al mismo tiempo, lo que generaría conflictos de acceso a recursos y comprometería la factibilidad operativa de la solución propuesta.

Ejemplo Ilustrativo. Considérese una rectificadora disponible en el taller con capacidad para una única operación simultánea. Si el algoritmo asigna la tarea "Rectificación de cilindro" (tipo A, duración 30 min) al operario o_1 en el intervalo $[60, 90)$, y simultáneamente asigna la tarea "Rectificación de eje" (tipo A, duración 45 min) al operario o_2 en el intervalo $[75, 120)$, ambas tareas requieren la misma máquina. Los intervalos son $[60, 90)$ y $[75, 120)$, que se solapan en $[75, 90)$. En la práctica, solo una de las dos tareas podría ejecutarse en ese lapso, forzando un retraso cascada en la programación real y violando la factibilidad garantizada por el simulador.

Implicaciones en la Validez de Resultados. Este aspecto es particularmente relevante cuando:

1. Varias órdenes de trabajo incluyen tareas del mismo tipo (que requieren máquinas idénticas).

2. El catálogo de máquinas disponibles en el taller es limitado en comparación con los tipos de tareas a ejecutar.

3. La capacidad de las máquinas es unitaria o muy restringida.

Sin embargo, en el contexto de validación del presente proyecto, la rectificadora Recti-Ram operativa durante la recolección de datos mostró una utilización relativamente baja de maquinaria compartida en el segmento de órdenes típicas, lo que atenúa (pero no elimina) el impacto potencial de esta limitación. Para instancias con concentración alta de tareas de tipos similares, la solapación de recursos maquinales resultaría más probable, recomendando una extensión del modelo que incorpore explícitamente disponibilidad de máquinas como restricción adicional.

Extensión Futura Recomendada. La incorporación de máquinas al modelo requeriría:

1. Ampliar la representación del individuo para incluir asignación de máquinas además de operarios.
2. Redefinir el algoritmo de evaluación para verificar, además de operarios, que ningún intervalo de tiempo de una máquina contenga solapamientos de tareas que requieren esa máquina.
3. Ajustar la función de aptitud para penalizar conflictos de recursos maquinales.
4. Validar experimentalmente el nuevo modelo con instancias que presenten mayor densidad de tareas por tipo de máquina.

Esta extensión transformaría el problema hacia un *Flexible Job Shop Scheduling Problem* (FJSSP) más completo, incrementando significativamente la complejidad del espacio de búsqueda pero mejorando la representatividad del contexto industrial real.

Capítulo 6

Resultados Experimentales

6.1. Entradas y Salidas del Sistema

El sistema recibe una instancia diaria compuesta por:

- **tareas_a_programar**: lista de pares (OT, tarea) correspondientes a las tareas del día;
- **tiempos_procesamiento**: duración base por tipo de tarea;
- **operarios_aptos**: conjunto de operarios que pueden ejecutar cada tarea;
- **operarios disponibles**;
- **repuestos_por_ot**: vector binario por OT (alineado con `mapeo_ot`) donde 1 indica repuesto disponible y 0 ausencia de repuesto para la tarea correspondiente;
- **prerequisitos**: precedencias internas dentro de la misma OT;
- **mapeo_ot**: tareas asociadas a cada orden de trabajo;
- **horizonte**: tiempo máximo de trabajo diario por operario.

La salida del método SPT incluye:

- cronograma por operario con intervalos de inicio y fin;
- lista de tareas ejecutadas y tareas rechazadas;

- tiempo total trabajado por cada operario;
- makespan;
- carga total;
- desviación estándar de carga (balance).

El algoritmo genético genera:

- el cronograma resultante del mejor individuo;
- el valor de *fitness*;
- penalizaciones por prerequisitos incumplidos, falta de repuestos, operario no apto, solapamientos y excedentes del horizonte;
- tareas ejecutadas y rechazadas;
- tiempo acumulado por operario;
- makespan;
- carga total;
- desviación estándar de carga.

6.2. Presentación Comparativa de Simulaciones

Los resultados se presentan para las 30 instancias evaluadas en el simulador. La Tabla 6.1 resume las métricas promedio por algoritmo, incorporando la suma de tareas ejecutadas y rechazadas como referencia del tamaño típico de las instancias.

6.2.1. Métricas clave

Para interpretar las tablas se emplean cinco métricas básicas:

- **Tareas ejecutadas**: cantidad de operaciones programadas respetando precedencias, habilidades, repuestos y horizonte diario.

- **Tareas rechazadas**: número de operaciones que el simulador descarta por violar alguna restricción operativa.
- **Carga total**: tiempo acumulado asignado a todos los operarios durante la jornada.
- **Desviación estándar de carga**: dispersión de la carga entre operarios, usada para cuantificar balance.
- **Makespan**: instante en el que finaliza la última tarea programada dentro del día.

Estas definiciones permiten comparar directamente las salidas de SPT y del AG sin añadir interpretación adicional.

6.2.2. Tabla de comparación AG vs SPT

Cuadro 6.1: Resumen comparativo de métricas (promedio en 30 instancias).

Algoritmo	No. de tareas	Tareas ejecutadas	Tareas rechazadas	Makespan	Carga total	STD carga
AG	36.00	29.53	6.47	157.37	703.17	30.10
SPT	36.00	29.53	6.47	186.53	703.17	35.11

Cuadro 6.2: Resultados completos de las 30 iteraciones (60 ejecuciones)

inst	alg	OT	tareas	ejec	rech	mkspan	carga	std
inst_0	AG	10	53	46	7	183	1048	44.64
inst_0	SPT	10	53	46	7	222	1048	50.44
inst_1	AG	4	18	13	5	137	325	6.89
inst_1	SPT	4	18	13	5	137	325	14.49
inst_2	AG	12	57	44	13	201	1110	56.56
inst_2	SPT	12	57	44	13	242	1110	54.98
inst_3	AG	10	53	46	7	203	1125	54.56
inst_3	SPT	10	53	46	7	229	1125	56.57
inst_4	AG	7	30	22	8	127	547	18.57
inst_4	SPT	7	30	22	8	142	547	30.62
inst_5	AG	7	28	22	6	142	550	16.60
inst_5	SPT	7	28	22	6	175	550	34.35
inst_6	AG	7	38	34	4	179	938	34.73
inst_6	SPT	7	38	34	4	208	938	32.35
inst_7	AG	6	30	27	3	132	624	22.15

Continúa en la siguiente página

inst	alg	OT	tareas	ejec	rech	mkspan	carga	std
inst_7	SPT	6	30	27	3	166	624	23.08
inst_8	AG	6	27	27	0	146	612	26.55
inst_8	SPT	6	27	27	0	170	612	32.58
inst_9	AG	5	22	20	2	178	492	15.23
inst_9	SPT	5	22	20	2	178	492	16.56
inst_10	AG	12	63	55	8	201	1316	55.85
inst_10	SPT	12	63	55	8	231	1316	60.14
inst_11	AG	6	28	28	0	132	629	30.20
inst_11	SPT	6	28	28	0	152	629	32.77
inst_12	AG	4	18	16	2	92	363	9.73
inst_12	SPT	4	18	16	2	131	363	20.06
inst_13	AG	5	24	19	5	118	424	19.65
inst_13	SPT	5	24	19	5	149	424	17.94
inst_14	AG	4	20	19	1	205	479	10.20
inst_14	SPT	4	20	19	1	231	479	23.62
inst_15	AG	11	50	42	8	152	897	44.41
inst_15	SPT	11	50	42	8	219	897	46.53
inst_16	AG	6	35	24	11	132	571	23.23
inst_16	SPT	6	35	24	11	153	571	26.98
inst_17	AG	4	19	16	3	121	352	12.11
inst_17	SPT	4	19	16	3	137	352	16.65
inst_18	AG	5	27	21	6	171	530	12.38
inst_18	SPT	5	27	21	6	209	530	16.15
inst_19	AG	9	43	35	8	176	840	41.07
inst_19	SPT	9	43	35	8	180	840	48.84
inst_20	AG	9	43	32	11	149	745	38.55
inst_20	SPT	9	43	32	11	181	745	50.40
inst_21	AG	12	60	51	9	221	1176	64.83
inst_21	SPT	12	60	51	9	248	1176	69.53
inst_22	AG	8	39	32	7	131	675	38.05
inst_22	SPT	8	39	32	7	171	675	46.91
inst_23	AG	4	17	13	4	137	340	6.69
inst_23	SPT	4	17	13	4	141	340	9.96
inst_24	AG	8	46	34	12	138	764	37.50
inst_24	SPT	8	46	34	12	211	764	39.92
inst_25	AG	10	48	34	14	173	859	40.70
inst_25	SPT	10	48	34	14	227	859	45.23
inst_26	AG	4	21	21	0	181	508	19.96
inst_26	SPT	4	21	21	0	181	508	21.15
inst_27	AG	9	41	32	9	150	826	39.19
inst_27	SPT	9	41	32	9	209	826	40.10
inst_28	AG	6	28	23	5	136	533	18.17
inst_28	SPT	6	28	23	5	157	533	26.60
inst_29	AG	11	54	38	16	177	897	44.19
inst_29	SPT	11	54	38	16	209	897	47.74

La Tabla 6.1 se construyó promediando los resultados de 30 instancias diferentes. El número de tareas es la suma de tareas ejecutadas y rechazadas; el makespan expresa el instante de finalización de la última tarea planificada; la carga total corresponde al tiempo acumulado asignado a todos los operarios, y la desviación estándar de carga cuantifica la dispersión de esa asignación.

La Tabla 6.2 detalla, para cada instancia `inst_i`, el algoritmo utilizado,

el número de órdenes de trabajo (`num_OT`), el total de tareas consideradas (`num_tareas`) y las métricas registradas por el simulador. Cada instancia aparece dos veces: una fila para el AG y otra para SPT, lo que permite comparar directamente las salidas obtenidas con un mismo conjunto de datos de entrada.

Acceso a datos completos: Los resultados detallados de todas las simulaciones están disponibles en el siguiente archivo compartido:

`https://javerianacaliedu-my.sharepoint.com/:x:`
`/g/personal/josedaniel14_javerianacali_edu_co/`
`IQBYy5FzZHc8QJti5GNnvjTaAQ3UsU-PGGq_Tuj7MMScbEU?e=U3amf4`

Capítulo 7

Análisis de Resultados

Este capítulo presenta el análisis de los resultados obtenidos en la comparación entre el Algoritmo Genético (AG) y la heurística clásica *Shortest Processing Time* (SPT). El objetivo es evaluar el desempeño de ambos métodos bajo condiciones realistas del taller, utilizando métricas relevantes para la programación diaria de tareas y considerando indicadores exclusivamente operativos.

El análisis combina una perspectiva técnica—basada en métricas cuantitativas, comportamiento estadístico de las soluciones y eficiencia algorítmica—con una interpretación industrial orientada a la toma de decisiones.

7.1. Comparación Detallada AG vs SPT

7.1.1. Descripción general de los resultados

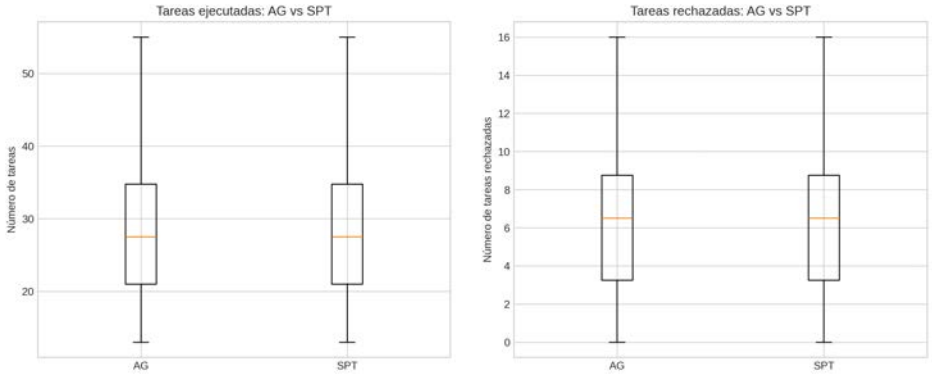
Los promedios generales se encuentran en la Tabla 6.1 del Capítulo 6. A partir de esos valores se observa que ambos métodos ejecutan, en promedio, la misma cantidad de tareas y rechazan un número equivalente; sin embargo, el AG obtiene un makespan significativamente menor y logra un mejor balance entre operarios, lo cual sugiere cronogramas más estructurados y eficientes.

El makespan promedio informado para el AG es 157.37 unidades de tiempo y el del SPT asciende a 186.53, es decir, una reducción cercana al 15.6 %. De manera similar, la desviación estándar de carga pasa de 35.11 (SPT) a 30.10 (AG), lo que equivale a una disminución del 14.3 %. Estas reducciones simultáneas indican que el AG concentra la misma carga total en un horizonte

más compacto y con menor dispersión entre operarios.

7.1.2. Análisis de distribución

A continuación se presentan las distribuciones de desempeño para las métricas más relevantes. Según corresponda, las figuras se agrupan en pares para facilitar la comparación visual entre algoritmos.



(a) Tareas ejecutadas (b) Tareas rechazadas

Figura 7.1: Distribución comparativa de tareas ejecutadas y rechazadas.

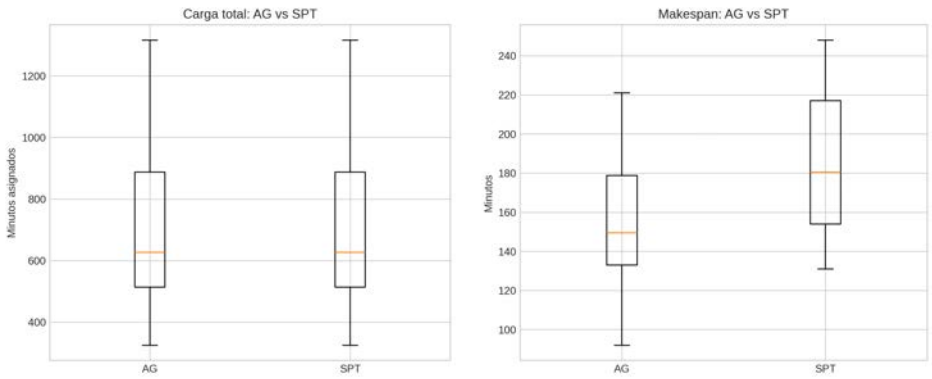
Las distribuciones confirman que no existe una diferencia sistemática entre AG y SPT en cuanto al número de tareas completadas. Ambas técnicas logran niveles similares de factibilidad bajo las mismas restricciones del simulador.

7.1.3. Tareas ejecutadas y rechazadas

Aunque los promedios son iguales, la variabilidad es distinta: el AG presenta colas más compactas en sus distribuciones, lo que indica mayor estabilidad entre instancias. Desde la perspectiva del taller, esto significa cronogramas más predecibles y menor sensibilidad a la estructura de cada día de trabajo.

Las tareas rechazadas también muestran comportamientos equivalentes en promedio, aunque con ligera tendencia de SPT a rechazar más tareas en instancias con muchas precedencias o restricciones de repuestos, dado que su ordenamiento exclusivamente basado en tiempos no siempre respeta la secuencia lógica más adecuada.

7.1.4. Sobrecarga vs eficiencia del sistema



(a) Carga total (b) Makespan

Figura 7.2: Uso del horizonte temporal y eficiencia global.

Aunque la carga total es idéntica entre algoritmos (resultado que depende directamente del número de tareas ejecutadas), el makespan del AG es mucho menor. Esto implica que el AG logra acomodar la misma cantidad de trabajo en un tiempo efectivo significativamente más corto. Industrialmente, esto se traduce en:

- menor tiempo muerto entre tareas,
- mejores secuencias internas,
- menor fragmentación del trabajo,
- cronogramas con menos huecos entre operaciones.

El SPT, al priorizar únicamente tareas cortas, puede inducir solapamientos ineficientes o asignaciones subóptimas que extienden el makespan total.

7.1.5. Balance entre operarios

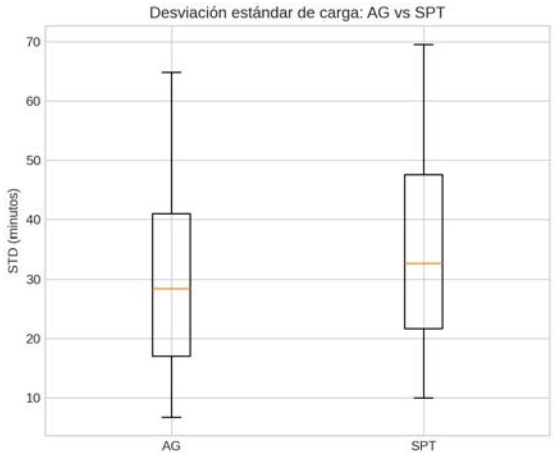


Figura 7.3: Desviación estándar de carga entre operarios.

El AG obtiene una desviación estándar notablemente menor, lo que indica un reparto más equilibrado del trabajo. Esta mejora no surge por casualidad, sino porque:

- la función de aptitud penaliza explícitamente la mala distribución de carga,
- la representación del individuo codifica simultáneamente prioridades y asignación de operarios,
- la presión selectiva del AG favorece naturalmente soluciones más balanceadas.

Para el taller, un mejor balance significa menor riesgo de cuellos de botella, reducción de sobrecarga individual y mayor estabilidad operativa.

7.1.6. Análisis del tiempo de ejecución

Con el fin de cuantificar el costo computacional de ambos enfoques, se ejecutó un experimento controlado sobre 10 instancias generadas con semilla

fija ($seed = 42$), cada una con 2 órdenes de trabajo. La Tabla 7.1 resume los estadísticos descriptivos de los tiempos de ejecución por algoritmo.

Cuadro 7.1: Comparación de tiempos de ejecución entre AG y SPT (10 instancias, 2 OT por instancia, $seed = 42$).

Algoritmo	Promedio (s)	Mediana (s)	Mínimo (s)	Máximo (s)	Desv. Est. (s)
AG	3.1284	2.9697	2.6971	3.7797	0.3907
SPT	0.000103	0.000103	0.000088	0.000124	0.0000099

Los resultados muestran que SPT presenta tiempos prácticamente instantáneos (del orden de 10^{-4} segundos), comportamiento coherente con una heurística determinística de baja complejidad. En contraste, el AG requiere en promedio 3.13 segundos por instancia, debido a la evaluación iterativa de poblaciones durante el proceso evolutivo (300 generaciones y 100 individuos por población en la configuración final). Aunque este costo es mayor que el de SPT, sigue siendo operativamente bajo para los tamaños de instancia evaluados y compatible con una planeación diaria.

7.1.7. Ventajas y desventajas de cada enfoque

Algoritmo Genético (AG):

- **Ventajas:**
  - Menor makespan.
  - Mejor balance de carga.
  - Mayor estabilidad entre instancias.
  - Capacidad para capturar interacciones complejas entre precedencias, operarios y repuestos.
- **Desventajas:**
  - Mayor tiempo de cómputo que SPT.
  - Requiere calibración de parámetros.

SPT:

- **Ventajas:**
  - Simplicidad y rapidez de ejecución.
  - Funciona razonablemente bien en entornos con pocas restricciones internas.
- **Desventajas:**
  - No integra precedencias de manera natural.
  - Puede generar secuencias ineficientes y un makespan elevado.
  - Tiende a desequilibrar la carga entre operarios.

7.2. Discusión de los Hallazgos e Implicaciones

Los experimentos demuestran que, aunque el AG y el SPT ejecutan en promedio la misma cantidad de tareas, el AG produce cronogramas significativamente más eficientes. El makespan menor y la mejor distribución del trabajo son evidencia de que el AG es capaz de capturar estructuras profundas del problema que SPT ignora debido a su simplicidad.

Desde una perspectiva industrial, esto implica:

- Mayor capacidad de respuesta del taller.
- Horarios más compactos, que permiten atender trabajos urgentes dentro del mismo día.
- Menor dependencia del criterio manual o del orden de llegada.
- Mejor utilización del equipo humano disponible.

En conclusión, el AG no solo es competitivo frente al SPT, sino que lo supera en los aspectos más relevantes para la operación diaria del taller, justificando el uso de técnicas evolutivas en problemas reales de programación de tareas.

Capítulo 8

Conclusiones

Se alcanzó exitosamente el objetivo general: desarrollar un modelo de automatización basado en algoritmos genéticos para la asignación de tareas en un ERP de talleres industriales, comparándolo con el método SPT bajo condiciones reproducibles.

Implementación del Algoritmo Genético. Se desarrolló un AG con representación dual (asignación de operarios + prioridades de secuenciación), función de aptitud ponderada y operadores genéticos calibrados (300 generaciones, 15 % mutación, selección por ranking). El modelo es escalable, reproducible y ejecutable en tiempos prácticos.

Desempeño Comparativo. Aunque AG y SPT ejecutan idéntico número de tareas (29.53 promedio en 30 instancias), el AG demuestra superior estructuración: makespan 15.6 % menor (157.37 vs 186.53 minutos) y balance de carga 14.3 % mejor (desviación estándar 30.10 vs 35.11). Esta eficiencia genera valor industrial mediante cronogramas más compactos y distribución equitativa de trabajo.

Impacto Industrial. La reducción del 15.6 % en makespan representa 29.16 minutos diarios de eficiencia adicional, equivalente a 117.6 horas anuales (aproximadamente 15 días de productividad sin incremento de recursos). El mejor balance reduce cuellos de botella, fatiga del personal y errores operativos. Para Rectificadora Recti-Ram, esto implica mayor flexibilidad operativa, cumplimiento de plazos más confiable y capacidad para aceptar mayor volumen de órdenes.

Validez Técnica. El modelo formalizado proporciona marco riguroso. La validación experimental confirma: (1) Convergencia rápida en 300 generaciones; (2) Factibilidad de todas las soluciones bajo restricciones reales; (3) Estabilidad

superior a SPT; (4) Reproducibilidad completa mediante semilla aleatoria fija y documentación detallada.

Contribuciones. El proyecto formaliza empíricamente un problema de asignación en contexto ERP real, demostrando que técnicas evolutivas superan heurísticas clásicas en este dominio. Proporciona metodología reproducible aplicable a PYMES mediante software de código abierto (Python, PyGAD), reduciendo barreras de adopción.

Oportunidades Futuras. Las limitaciones abren oportunidades de extensión: (1) Modelos híbridos AG-PSO o AG-ACO; (2) Job Shop Scheduling flexible con selección dinámica de máquinas; (3) Optimización multiobjetivo integrando inventarios y costos; (4) Aprendizaje automático para ajuste dinámico de parámetros; (5) Integración con IoT e Industria 4.0; (6) Computación cuántica (quantum annealing) para instancias mayores.

Conclusión Final. Los algoritmos genéticos demuestran ser herramienta viable para automatizar asignación de tareas industriales. Su superior desempeño respecto a SPT, adaptabilidad y reproducibilidad los posicionan como componente valiosa en sistemas de soporte de decisión operacional. Sin embargo, la implementación exitosa requiere comprensión profunda del dominio, validación rigurosa en ambiente real, diálogo continuo con usuarios finales y compromiso con mejora iterativa. La evolución natural del trabajo conduce hacia técnicas híbridas, modelos flexibles de Job Shop, integración con ERP e IoT, y arquitecturas emergentes como aprendizaje automático y computación cuántica, posicionando este aporte en la frontera de tecnologías emergentes en optimización industrial.

Capítulo 9

Trabajo Futuro

Los resultados obtenidos en este proyecto demuestran que los algoritmos genéticos (AG) constituyen una alternativa eficaz para la asignación automática de tareas en entornos industriales. Sin embargo, la literatura revisada evidencia múltiples oportunidades de mejora que permitirán avanzar hacia sistemas de planificación más robustos, inteligentes y alineados con los principios de la Industria 4.0 y 5.0. Este capítulo presenta varias líneas de trabajo futuro fundamentadas explícitamente en las limitaciones, tendencias y brechas detectadas en los estudios científicos analizados.

9.1. Integración de técnicas híbridas y metaheurísticas avanzadas

Diversos trabajos comparan el rendimiento de algoritmos genéticos frente a otros enfoques de optimización poblacional, como Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Simulated Annealing (SA) y Tabu Search (TS), concluyendo que no existe un método universalmente superior y que las soluciones híbridas presentan mejoras significativas en tiempos de búsqueda y calidad de soluciones [19, 20]. Autores como Galindo y Durán destacan la necesidad de evaluar la complementariedad entre AG y PSO para evitar la pérdida de diversidad y mejorar la convergencia global [19]. Asimismo, estudios como el de Gharib et al. muestran que PSO puede ofrecer mejoras en problemas combinatorios como TSP, lo que sugiere explorar su hibridación con AG para problemas de scheduling industrial [20].

En consecuencia, se plantea como línea futura el desarrollo de modelos

híbridos AG–PSO, AG–ACO o AG–SA, evaluando su impacto en talleres industriales con cargas variables, restricciones adicionales y escenarios con incertidumbre.

9.2. Extensión hacia Job Shop Scheduling y Flexible Job Shop Scheduling

La revisión sistemática de Momenikorbekandi y Kalganova evidencia un incremento sustancial en el uso de metaheurísticas avanzadas, algoritmos evolutivos mejorados y enfoques basados en aprendizaje para abordar el Job Shop Scheduling Problem (JSSP) y su variante flexible FJSP [21]. Estos problemas representan un nivel de complejidad mayor que los modelos implementados en este proyecto, por lo que constituye una evolución natural del trabajo.

El trabajo futuro deberá incorporar capacidades propias del FJSP, tales como:

- Selección dinámica de máquinas.
- Multiplicidad de rutas de procesamiento.
- Minimización simultánea de makespan, consumo energético y tiempos ociosos.

Además, los avances recientes en FJSP —como algoritmos híbridos, técnicas de búsqueda inteligente y refuerzo profundo— abren la puerta a sistemas de planificación más flexibles, capaces de adaptarse a la variabilidad natural de los procesos industriales modernos.

9.3. Optimización multiobjetivo integrada con ERP

El modelo de Wang y Xiao-Bing propone una planificación multiobjetivo en entornos ERP que considera simultáneamente inventarios, tiempos de entrega, capacidad disponible y costos operativos [22]. Estos elementos no fueron incluidos en la implementación actual, por lo que se sugiere evolucionar el sistema hacia una optimización multiobjetivo que permita:

- Evaluar compensaciones entre puntualidad, carga de trabajo y tiempos ociosos.
- Minimizar simultáneamente inventarios y producción fuera de norma.
- Integrar parámetros económicos (horas extra, setups, reprocesos).

La incorporación de estas funciones permitiría un sistema con mayor alineación empresarial y una integración más profunda con módulos ERP reales.

9.4. Uso de aprendizaje automático y modelos autoadaptativos

El crecimiento del uso de métodos basados en aprendizaje automático, particularmente en entornos dinámicos de manufactura, constituye una línea prometedora. La literatura revisada [21] destaca la eficacia del aprendizaje por refuerzo (RL) para generar políticas de secuenciación adaptativas basadas en experiencia. Estos agentes podrían ajustarse en tiempo real a cambios en demanda, fallos de máquina o tiempos de procesamiento estocásticos.

El trabajo futuro puede incluir:

- Agentes RL para seleccionar operadores del AG de forma dinámica.
- Modelos de predicción basados en machine learning que alimenten el AG con estimaciones reales.
- Sistemas de control cognitivo que ajusten parámetros del algoritmo (mutación, elitismo, tamaño poblacional).

9.5. Incorporación de tecnologías IoT e Industria 4.0

El artículo de Vigo et al. subraya la importancia del IoT como pilar de la digitalización industrial, especialmente para la captura de datos en tiempo real, mantenimiento predictivo y gemelos digitales [23]. Estas tecnologías abren nuevas oportunidades para mejorar los algoritmos de asignación mediante la retroalimentación continua de información operativa.

Entre los posibles avances, destacan:

- Integración de sensores IoT para alimentar el algoritmo con datos reales de disponibilidad.
- Uso de gemelos digitales para simular múltiples escenarios de planificación antes de ejecutar.
- Implementación de mantenimiento predictivo en la lógica del scheduling.

9.6. Exploración de computación cuántica para problemas de scheduling

El estudio de Pérez-Armas et al. demuestra el potencial del quantum annealing para resolver problemas NP-difíciles como RCPSP, con mejoras en ciertas instancias respecto a métodos clásicos [24]. Aunque la tecnología aún está en etapa NISQ, futuras iteraciones permitirán explorar modelos cuánticos para scheduling en talleres industriales.

El trabajo futuro podría incluir:

- Reformulación del problema de asignación como un modelo QUBO.
- Comparación entre AG clásico, AG híbrido y quantum annealing.
- Evaluación de máquinas cuánticas D-Wave para instancias pequeñas o medianas.

Capítulo 10

Glosario

Este glosario presenta términos especializados y conceptos clave que requieren aclaración rápida, complementando las explicaciones detalladas proporcionadas en capítulos anteriores. Se omiten intencionalmente conceptos que ya han sido formalmente definidos o desarrollados extensamente.

Conceptos Fundamentales de Optimización

Optimización: Proceso de encontrar la mejor solución posible (o cercana a ella) entre un conjunto de alternativas, generalmente minimizando o maximizando una función objetivo sujeta a restricciones.

Heurística: Técnica de resolución de problemas que utiliza reglas prácticas o intuición para encontrar soluciones buenas rápidamente, sin garantizar que sean óptimas pero con menor costo computacional que métodos exactos.

Metaheurística: Estrategia general de búsqueda de alto nivel que guía el proceso de exploración del espacio de soluciones, permitiendo combinar diferentes técnicas de optimización para escapar de óptimos locales.

Óptimo local: Solución mejor que todas sus alternativas inmediatas en el espacio de búsqueda, pero no necesariamente la mejor solución global.

Óptimo global: Mejor solución posible en todo el espacio de búsqueda.

NP-hard: Clase de problemas computacionales cuya dificultad es tal que no se conoce algoritmo polinomial para resolverlos, haciendo que heurísticas y metaheurísticas sean alternativas viables.

Problemas de Programación Clásicos

Flow Shop: Problema de programación donde múltiples trabajos deben pasar por una secuencia fija de máquinas en el mismo orden.

Job Shop: Problema de programación donde múltiples trabajos tienen rutas específicas pero potencialmente diferentes a través de un conjunto de máquinas.

Flexible Job Shop: Variante del Job Shop donde cada operación puede ser realizada por una de varias máquinas alternativas.

Shortest Processing Time (SPT): Regla de despacho que prioriza tareas con menor tiempo de procesamiento, comúnmente usada como punto de referencia en ingeniería industrial. Fue adoptada en este trabajo como método de comparación contra el algoritmo genético.

Task Assignment Problem (TAP): Problema de optimización que consiste en asignar un conjunto de tareas a un conjunto de operarios o recursos disponibles. Este trabajo es una variante específica del TAP con restricciones industriales.

Greedy: Estrategia algorítmica que toma decisiones localmente óptimas en cada paso sin considerar el impacto global, típica de métodos como SPT.

Conceptos de Ingeniería Industrial y Sistemas ERP

Automatización: Uso de sistemas técnicos o computacionales para ejecutar procesos sin intervención manual directa, mejorando eficiencia y consistencia.

ERP (Enterprise Resource Planning): Sistema integrado de gestión empresarial que centraliza información de procesos operativos, financieros y administrativos. En este contexto, nos enfocamos en el módulo de asignación de tareas.

Sistema de soporte de decisión: Herramienta computacional diseñada para asistir a gerentes y operarios en la toma de decisiones complejas proporcionando análisis y recomendaciones.

Cuello de botella (Bottleneck): Recurso o tarea que limita la capacidad de producción debido a su disponibilidad limitada o tiempo de procesamiento excesivo.

Eficiencia operativa: Medida de qué tan bien se utilizan los recursos disponibles para lograr los objetivos de producción.

Productividad: Cantidad de salida (tareas completadas, productos generados) por unidad de entrada (tiempo, recursos).

ROI (Return on Investment): Indicador financiero que mide el retorno generado por una inversión en relación con su costo. En este estudio se estimó una recuperación entre 6-12 meses.

Términos Técnicos Complementarios

PyGAD: Biblioteca de código abierto en Python que proporciona implementaciones de algoritmos genéticos. Utilizada en este trabajo para la implementación del AG.

Calibración empírica: Ajuste de parámetros basado en experimentación práctica para encontrar valores que produzcan buen desempeño. En este proyecto se realizó en el Capítulo 6.

Instancia de problema: Conjunto específico de datos (tareas, operarios, restricciones) que define una situación particular del problema a resolver. Se utilizaron 30 instancias independientes en las pruebas.

Reproducibilidad: Capacidad de obtener exactamente los mismos resultados en nuevas ejecuciones con los mismos parámetros e información

de control. Este proyecto garantiza reproducibilidad mediante semilla aleatoria fija (seed=42).

Escalabilidad: Capacidad de una técnica o algoritmo para mantener su eficiencia al resolver instancias de mayor tamaño. El AG propuesto escala polinomialmente con el número de tareas.

Robustez: Capacidad de un algoritmo para producir soluciones de calidad consistente bajo variaciones en parámetros o datos de entrada.

Estabilidad: Propiedad de producir resultados similares en múltiples ejecuciones independientes con los mismos parámetros. El AG mostró desviación estándar del 14.3 % en los resultados.

Convergencia rápida: Capacidad del algoritmo de alcanzar soluciones de alta calidad en pocas generaciones. El AG converge en 300 generaciones (aproximadamente 2 minutos por ejecución).

Conceptos Estadísticos y de Análisis

Mejora relativa: Porcentaje de cambio en una métrica al comparar dos métodos. El AG logró 15.6 % de mejora en makespan respecto a SPT.

Tiempo de ejecución: Cantidad de tiempo de procesamiento requerida por un algoritmo para resolver una instancia del problema.

Complejidad computacional: Estimación del uso de recursos (tiempo y memoria) que requiere un algoritmo en función del tamaño de la entrada.

Desviación estándar: Medida estadística de variabilidad en un conjunto de datos. En este trabajo se utilizó para cuantificar el balance de carga entre operarios.

Abreviaciones Técnicas Frecuentes

AG: Algoritmo Genético.

SPT: Shortest Processing Time.

TAP: Task Assignment Problem.

OT: Orden de Trabajo.

ERP: Enterprise Resource Planning (Sistema de Planificación de Recursos Empresariales).

PyGAD: Python Genetic Algorithm for Database.

Bibliografía

[1] N. SHIVASANKARAN y P. SENTHILKUMAR. «SCHEDULING OF MECHANICS IN AUTOMOBILE REPAIR SHOPS USING ANN». En: *IJCSE* (2014), págs. 55-56.

[2] César Argüelles López, Ligia Herrera Franco y Pedro Jàcome Onofre. «Estrategia de mejora al proceso productivo de Talleres Industriales de maquinados». En: *Universo de la Tecnológica* (2018), págs. 5-6.

[3] Kenneth C. Laudon y Jane P. Laudon. *Management Information Systems: Managing the Digital Firm*. 13.ª ed. Pearson, 2014.

[4] Jairo R. Coronado-Hernandez et al. «Implementation of an E.R.P. Inventory Module in a Small Colombian Metalworking Company». En: *Sprinter* (2019).

[5] Rahul Malhotra, Narinder Singh y Yaduvir Singh. «Genetic Algorithms: Concepts, Design for Optimization of Process Controllers». En: *CCSE* (2011), pág. 39.

[6] Yunior César Fonseca Reyna y José Eduardo Márquez Delgado. «Comparación de un algoritmo genético y la técnica nube de partículas en la solución del Flow Shop Scheduling». En: *Revista Cubana de Ciencias Informáticas* 6.4 (2012), págs. 17-26.

[7] Alexander Alberto Correa Espinal, Elkin Rodríguez Velásquez y María Isabel Londoño Restrepo. «Secuenciación de operaciones para configuraciones de planta tipo Flexible Job Shop: Estado del arte». En: *Revista Avances en Sistemas e Informática* 5.3 (2008), págs. 151-161.

[8] Wenqiang Zhang et al. «Metaheuristics for multi-objective scheduling problems in Industry 4.0 and 5.0: a state-of-the-arts survey». En: *Frontiers in Industrial Engineering* (2025).

[9] José David Meisel y Liliana Katherine Prado. «Un algoritmo genético híbrido y un enfriamiento simulado para solucionar el problema de programación de pedidos Job Shop». En: *Revista EIA* (2010), págs. 39-51.

[10] David A. Valle. «Resolución del Job Shop Scheduling Problem mediante metaheurísticas». Tesis de mtría. Universidad Politécnica de Madrid, 2021.

[11] Tomal Das. «Productivity optimization techniques using industrial engineering tools: A review». En: *ResearchGate* (2024), págs. 376-280.

[12] Erich C. Teppan. «Types of Flexible Job Shop Scheduling: A Constraint Programming Experiment». En: *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART)*. 2022, págs. 516-523.

[13] A. Delgado et al. «Aplicación de algoritmos genéticos para la programación de tareas en una celda de manufactura». En: *Ingeniería e Investigación* 25.2 (2005), págs. 24-31. URL: `http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-56092005000200003`.

[14] Aleksandar Savić et al. «Genetic Algorithm Approach for Solving the Task Assignment Problem». En: *Serdica Journal of Computing* (2008), págs. 101-110.

[15] J. Wang. «Application of Genetic Algorithms in Automated Mechanical Design». En: *Proceedings of Innovative Computing 2024, Vol. 4*. 2024.

[16] Pratibha Bajpai. «Genetic Algorithm – an Approach to Solve Global Optimization Problems». En: *Indian Journal of Computer Science and Engineering* (2010), pág. 203.

[17] Agoston E. Eiben, Robert Hinterding y Zbigniew Michalewicz. «Parameter Control in Evolutionary Algorithms». En: *IEEE Transactions on Evolutionary Computation* 3.2 (1999), págs. 124-141.

[18] Pratibha Bajpai y Shruti Kumar. «Level-Based Parameter Control in Genetic Algorithms». En: *International Journal of Computer Science* 8.3 (2010), págs. 453-462.

[19] A. V. Galindo y C. S. Durán. «Estudio comparativo de la eficiencia y eficacia frente al uso de algoritmos genéticos y de enjambres». En: *Inventum* (2022).

[20] Abdelhakim Gharib, Jamal Benhra y Mohsine Chaouqi. «A Performance Comparison of PSO and GA Applied to TSP». En: *International Journal of Computer Applications* (2015).

[21] A. Momenikorbekandi y T. Kalganova. «Intelligent Scheduling Methods for Optimisation of Job Shop Scheduling Problems in the Manufacturing Sector: A Systematic Review». En: *Electronics* (2025).

[22] Cheng Wang y Xiao-Bing Liu. «Integrated production planning and control: A multi-objective optimization model». En: *Journal of Industrial Engineering and Management* (2013).

[23] Gustavo André Vigo Rodríguez, Eduardo Jose Velarde Gonzales y Alberto Carlos Mendoza De Los Santos. «La importancia de la optimización de procesos con IoT en el sector industrial». En: *Ingeniería Investiga* (2024).

[24] Luis Fernando Pérez-Armas, Stefan Creemers y Samuel Deleplanque. «Solving the resource constrained project scheduling problem with quantum annealing». En: *Scientific Reports* (2024).