Modelo de secuenciamiento para un sistema N/2 (N trabajos – 2 máquinas) a partir de la regla de Johnson y algoritmo genético.

Lina Marcela Dallos Sanabria

Trabajo de grado para optar por el título de ingeniero industrial.

Director
Vlakxmir Robles Marín
Magister en Ingeniería

Universidad Industrial de Santander
Facultad de Ingenierías Fisicomecánicas
Escuela de Estudios Industriales y Empresariales
Bucaramanga
2024

Dedicatoria

Este logro es un tributo lleno de amor y gratitud hacia mi familia y Cristian Aguilar. Su apoyo incondicional y colaboración constante han sido el pilar que me sostuvo en los momentos difíciles y me impulsó hacia el éxito en mi camino académico. Mis padres y hermano han sido mi motivación, iluminando cada etapa de mi vida y moldeando a la persona llena de determinación y gratitud que soy hoy en día. Sin ellos, este logro no sería posible. ¡Gracias por siempre creer en mí y por ser mi mayor fuente de inspiración y amor!

Agradecimientos

En este momento trascendental, doy gracias a Dios por haberme guiado hasta aquí, donde concluye esta etapa tan significativa de mi vida.

A mi familia y a Cristian Aguilar les agradezco desde lo más profundo de mi ser por su apoyo incondicional y dedicación en mi formación. Cada sacrificio y gesto de amor ha sido el sostén que me impulsó a seguir adelante.

A mis amigos, les debo una parte de mi corazón por acompañarme en este viaje lleno de aprendizaje y vivencias que llevaré grabadas por siempre en mi ser.

Quiero dirigir un agradecimiento sincero a mi director de proyecto, Vlakxmir Robles Marín, por su confianza, apoyo constante y paciencia infinita. Su orientación y respaldo fueron esenciales para alcanzar este logro.

Por último, mi alma mater, la Universidad Industrial de Santander, merece mi más profundo agradecimiento por brindarme la oportunidad de crecer y desarrollarme como ingeniera. Su compromiso y dedicación han dejado una marca imborrable en mi camino académico y personal.

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Lista de Apéndices

Los apéndices están adjuntos y puede visualizarlos en el Drive compartido

https://drive.google.com/drive/folders/1zMngwjNIEDV-eEgRvOd83GfQIbjneQKH

Apéndice A. Articulo publicable

Apéndice B. Programación regla de Johnson y algoritmo genético

Apéndice C. Programación mantenimiento correctivo

Apéndice D. Diagrama de Gantt

Apéndice E. Comparación de los modelos

Glosario

Con el fin de darle solución al caso de estudio, es importante conocer algunos conceptos que están vinculados con su desarrollo, los cuales se encuentran relacionados con la programación, a continuación, se relacionan las definiciones correspondientes.

Heurísticas: Conjunto de instrucciones utilizados para realizar tareas y darle solución al problema, algunos heurísticos producen soluciones sin dar un óptimo, pero sí mejoran la solución inicial.

Problemas de optimización: Es un modelo matemático que ayuda a evaluar las opciones más factibles por medio de alternativas. Consiste en optimizar la función objetivo definiendo el mejor resultado posible, mediante un objetivo, una función que cumpla con dicho objetivo, variables y restricciones.

Makespan: Es el lapso que abarca desde que se comienza el primer trabajo en la primera máquina hasta que se concluye el último trabajo en la última máquina. Es decir, es el tiempo total que transcurre desde el inicio de la producción hasta su terminación.

Centro de trabajo: Espacio que destina la empresa para la organización y gestión de los recursos productivos, donde se llevan a cabo diversas labores. Estos centros pueden ser un grupo de máquinas, una máquina o un lugar donde se ejecuten cierta clase de trabajos.

Resumen

Título: Modelo de secuenciamiento para un sistema N/2 (N trabajos – 2 máquinas) a partir de la regla de Johnson y algoritmo genético*

Autor: Lina Marcela Dallos Sanabria**

Director: Vlakxmir Robles Marín

Palabras Clave: Secuenciamiento, procesamiento de tareas, algoritmo, regla de Johnson y algoritmo genético, makespan.

Descripción:

Los problemas referentes a la programación de trabajos en entornos de producción han sido objeto de un exhaustivo análisis, ya que el objetivo primordial de estos estudios es determinar la secuenciación óptima aplicadas a las líneas de producción. Se incluyen variables como son el número de máquinas disponibles, los tiempos de preparación de las máquinas y la capacidad operativa de ellas. Por lo tanto, el propósito de este análisis es alcanzar el uso eficiente de los recursos en el proceso de producción para ello se consideraron parámetros como son los retrasos durante el proceso y la maximización de la utilización.

Con respecto a lo anterior, se abordó el problema secuenciación de un sistema N/2 (N trabajos en 2 máquinas en serie), donde se propuso un modelo de programación de trabajos considerando tiempo de preparación y de tal manera minimizando el makespan, mediante algoritmos que permitieron lograr una reducción del tiempo. Adicionalmente, se quiere analizar el mantenimiento correctivo de las máquinas al momento de estar operando ya que es una actividad que pasa muy frecuente en las empresas.

Abstract

Title: Sequencing model for an N/2 system (N jobs - 2 machines) based on Johnson's rule and genetic algorithm*

Author(s): Lina Marcela Dallos Sanabria**

Director: Vlakxmir Robles Marín

Key Words: Sequencing, task processing, algorithm, Johnson rule and genetic algorithm, flow time or makespan, processing time.

Description:

The problems related to job scheduling in production environments have been the subject of an exhaustive analysis, since the primary objective of these studies is to determine the optimal sequencing applied to production lines. Variables such as the number of machines available, machine setup times and machine operating capacity are included. Therefore, the purpose of this analysis is to achieve the efficient use of resources in the production process, for which parameters such as delays during the process and maximization of utilization were considered.

With respect to the above, the sequencing problem of an N/2 system (N jobs in 2 machines in series) was approached, where a job scheduling model was proposed considering setup time and thus minimizing the makespan, through algorithms that allowed to achieve a reduction of time. Additionally, we want to analyze the corrective maintenance of the machines when they are operating since it is an activity that happens very frequently in the companies.

Introducción

La secuenciación de tareas, conocida como Scheduling, se refiere a la asignación de recursos a tareas en intervalos de tiempo específicos, con el propósito de optimizar una medida determinada de comportamiento (Pinedo, 2016). Scheduling tuvo su origen en la época de la revolución industrial, donde se buscaba la forma más eficiente de gestionar el flujo de producción, a pesar de haberse ocasionado antes de la revolución. Este enfoque marco un punto de partida importante para la evolución de la secuenciación.

"La secuenciación y la programación son procesos fundamentales en la toma de decisiones dentro de las industrias manufactureras y de servicios" (Pinedo, 2016). En la actualidad, la eficiente programación y organización de la producción es un desafío que se encuentra frecuentemente en diversos contextos, no solamente en el ámbito industrial, sino también en la vida cotidiana.

Henry Gantt fue uno de los pioneros que contribuyó con un diagrama que facilitó la visualización de los problemas de secuenciación, en la actualidad, este diagrama es conocido como el "Diagrama de Gantt" en su honor.

A finales de 1970 Michael Garey y David Johnson crearon una rama de la teoría de la computación conocida como la teoría de la complejidad computacional. Esta teoría se encarga de analizar el tiempo y los recursos necesarios para resolver un problema, clasificándolos en grados de complejidad como son P y NP. Años más tarde, publicaron su libro sobre los problemas NP-Hard dentro de los cuales se encuentra el problema de secuenciación de tareas (M. R. Garey, 1976).

El problema de secuenciamiento se reconoce en la literatura como un problema NP-Hard para tamaños grandes por la cantidad de soluciones posibles que pueden darse, dependiendo el tamaño de las órdenes.

La finalidad de este trabajo es ampliar información sobre el problema de secuenciamiento para un sistema de N trabajos en 2 máquinas, utilizando una de las técnicas más comunes para resolver dicho sistema es la regla Johnson utilizada para resolver situaciones de secuenciación, minimizando el makespan total del grupo de trabajos, y el algoritmo genético siendo esta una técnica donde se resuelven los problemas mediante la generación de poblaciones sucesivas a las que se les aplican operadores de mutación y cruce, cada individuo en estas poblaciones representan una posible solución al problema, y el objetivo es encontrar el individuo que mejor represente la solución óptima (Naupari Quiroz & Rosales Geronimo , 2010). Con lo anterior, se quiere evaluar qué método es más eficaz, dado que las máquinas están en serie y los trabajos en cada método pueden variar dado que el makespan varía dependiendo de dicho ordenamiento escogiendo el más factible para ser procesados los pedidos; por esto se propone validar un algoritmo que contribuya a la reducción del makespan en la fabricación de los pedidos, donde no haya desperdicio de tiempos, que repercute directamente en el producto final y los tiempos de entrega, adicionalmente se quiere analizar el tiempo de reparación de una máquina cuando presenta fallas, ya que es un problema que se encuentra presente hoy en día. Con el fin de incrementar los índices de desempeño y productividad de una línea o un sistema de producción.

El desarrollo de esta investigación se enfoca en los procesos empresariales y de manufactura en general que puedan ser descritos mediante un sistema N/2, ya que es la actividad que más representa costos y esfuerzos, es por esto por lo que los procesos dependen en gran medida

entregar los pedidos a tiempo, sin retardos y en óptimas condiciones. Con el fin de aumentar la productividad en las empresas o compañías.

El presente trabajo está organizado en secciones comprendidas por: la sección 3 expone la revisión de literatura, la sección 4 y 5 muestra el planteamiento del problema y los objetivos de la investigación. La sección 6 y 7 presentan los resultados esperados y el marco de referencia, la sección 8 aborda la metodología de investigación, constituida por 6 etapas, las cuales dan cumplimiento a los objetivos específicos. Por último, las secciones 9, 10 y 11 exponen la estructura del proyecto, el cronograma y el presupuesto estimado para llevar a cabo el desarrollo del proyecto de investigación.

Tabla 1.

Tabla de cumplimiento de objetivos.

Objetivos	**Capítulos y subcapítulos**
Establecer una revisión bibliográfica sobre el sistema N/2 y sus diferentes métodos de solución. • Construir la ecuación de búsqueda • Realizar el análisis bibliométrico de los resultados obtenidos a partir de la ecuación de búsqueda	Capítulo 2 Subcapítulo 2.1
Definir específicamente las características del problema de secuenciamiento a procesar.	Capítulo 5

Formular un modelo matemático para la asignación de tiempos de las máquinas y los trabajos.	Capítulo 5 Subcapítulo 5.2
Desarrollar y formular los algoritmos a usar para dar solución al modelo formulado.	Capítulo 6
Validar el algoritmo para el problema abordado	Capítulo 7
Elaborar un artículo abordando los resultados de la investigación	Apéndice A

1. Generalidades de la Investigación

1.1. Objetivos

1.1.1. Objetivo general

Comparar soluciones factibles de secuenciamiento[**] en serie para un sistema N/2 mediante la regla Johnson vs la metaheurística algoritmo genético.

1.1.2. Objetivos específicos

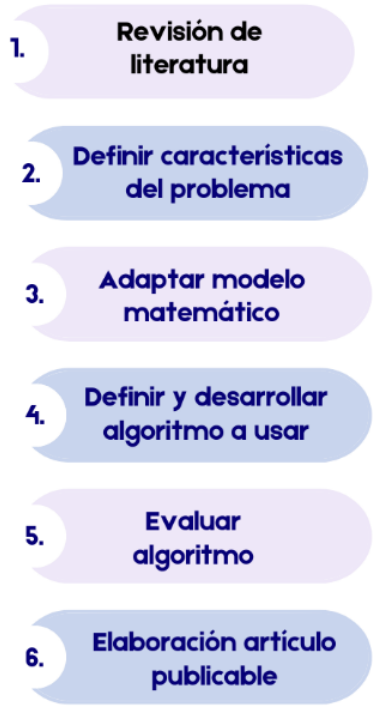
- Establecer una revisión bibliográfica sobre el sistema N/2 y sus diferentes métodos de solución.
- Definir específicamente las características del problema de secuenciamiento a procesar.
- Enunciar modelo matemático para la asignación de tiempos de las máquinas y los trabajos.
- Desarrollar y formular los algoritmos a usar para dar solución al modelo formulado.
- Validar el algoritmo para el problema abordado.
- Elaborar un artículo abordando los resultados de la investigación.

[**] Secuenciamiento es un término usado en contextos de gestión de operaciones y programación de tareas, sin embargo, secuenciación es utilizada en contextos como la genética

1.2. Metodología

Figura 1.

Metodología



Fuente: Realizada de manera propia

1.2.1. Fase I. Realizar una revisión bibliográfica sobre el problema a tratar

- Revisión de literatura
- Elaboración de la ecuación de búsqueda en la base de datos de SCOPUS y Web of Science, para realizar el análisis bibliométrico de lo obtenido en las bases de datos.

1.2.2. Fase II. Definir características del problema

- Seleccionar e identificar el diseño que se va a trabajar

- Decidir la cantidad de trabajos que se van a realizar en las máquinas (en este caso, son 2 máquinas)

1.2.3. Fase III. Adaptar el modelo matemático

- Adaptar un modelo que represente el problema a solucionar

1.2.4. Fase IV. Definir y desarrollar algoritmos que se utilizan

- Metaheurística algoritmo genético y regla de Johnson aplicado al secuenciamiento para un sistema N/2, identificados en la revisión de literatura.

1.2.5. Fase V. Evaluación de algoritmos

- Verificar que tanto la estructura como el funcionamiento de los modelos sean adecuados y estén correctamente, corrigiendo cualquier error que pueda existir en ellos.

1.2.6. Fase VI. Conclusión y recomendaciones

- Concluir sobre los hallazgos significativos que se obtuvieron a lo largo del estudio y los resultados derivados de la investigación.
- Formular recomendaciones para estudios futuros.

2. Revisión de literatura

Se presenta la revisión de literatura teniendo como objetivo analizar y recopilar la información obtenida de diferentes artículos y documentos, relacionados con el secuenciamiento de máquinas para un sistema N/2; identificando fuentes de información, autores, años y países con mayor crecimiento investigativo en lo concerniente al tema, a fin de encontrar un punto de referencia relevante para la investigación actual.

A continuación, se mostrará la ecuación de búsqueda utilizada y se detallará el análisis realizado en dos bases de datos como Scopus y Web of Science.

2.1. Análisis Bibliométrico

Con la utilización de la herramienta VOSviewer para análisis bibliométrico de datos y las bases de datos Web of Science y Scopus, se realizó un análisis de los estudios y artículos encontrados en estas sobre secuenciación para un sistema N/2. A continuación, se presentan los hallazgos más relevantes obtenidos de este análisis.

2.1.1. Web of Science

Se empleó la siguiente ecuación para realizar la búsqueda en Web of Science.

Figura 2.

Ecuación de búsqueda Web of Science.

$$sequencing\ of\ machines\ OR\ machine\ programming\ OR\ johnson's\ rule\ OR\ genetic\ algorithm$$
$$AND\ carrer\ OR\ industrial\ engineering\ NOT\ medicine\ OR\ medical$$

Basándonos en la ecuación previamente mencionada, se han obtenido 200 documentos hasta el 26 de junio de 2023, seleccionando únicamente aquellos documentos relacionados con el enfoque del proyecto.

2.1.1.1. Indicadores básicos

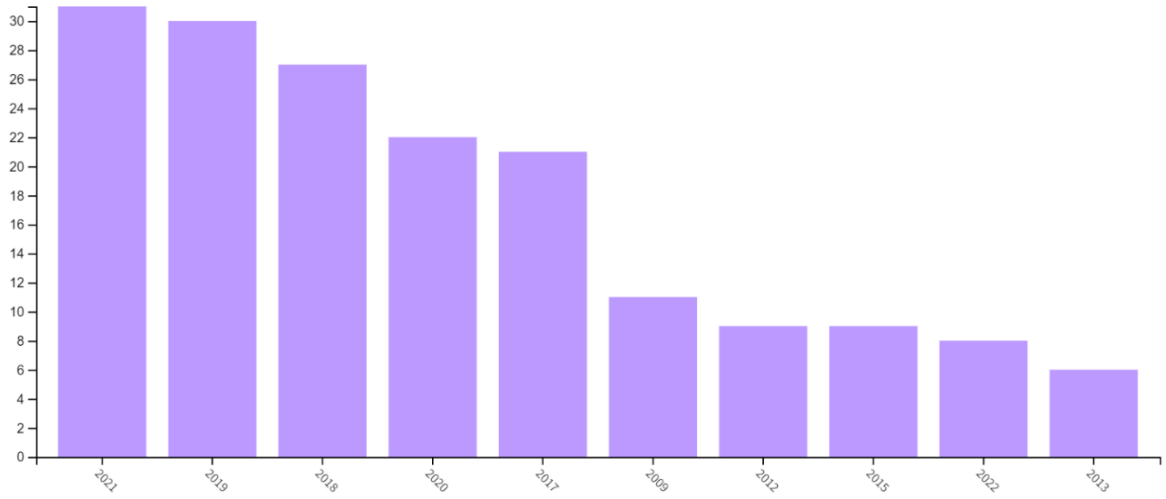
Durante el análisis de los datos obtenidos en la búsqueda, se procedió a la generación de los gráficos estadísticos mostrados a continuación, presentando de manera clara y concisa la información relevante en la investigación.

2.1.1.1.1. Documentos por año

En la figura 3 se aprecia un incremento gradual en el interés por el tema de investigación a lo largo de los años, por ejemplo, desde el año 2017 hasta el año 2021 se observa un crecimiento significativo, mientras que en el año 2021 hay 31 registros de documentos publicados acerca de este tema.

Figura 3.

Documentos por años, tomado de Web of Science (2023)



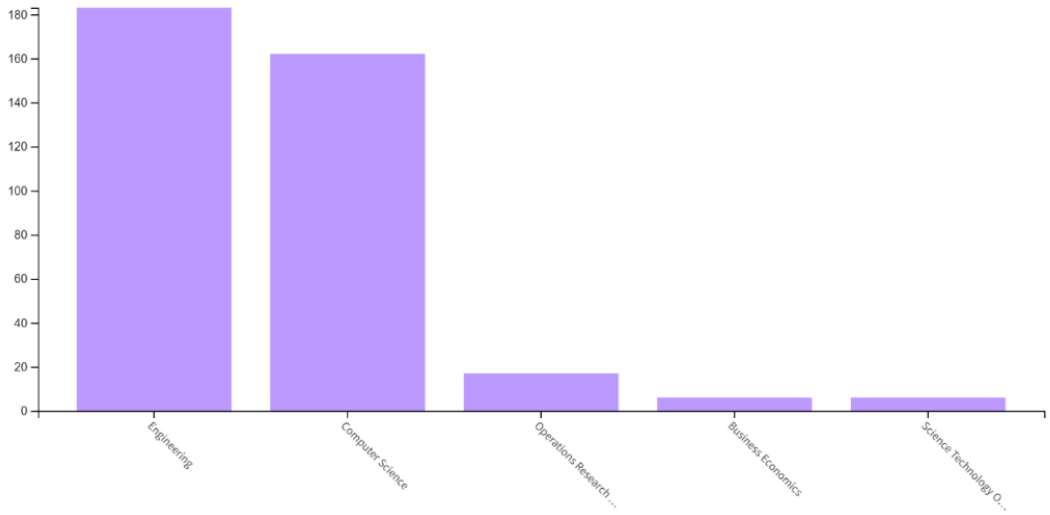
2.1.1.1.2. Área temática

En la figura 4 se presentan las áreas de investigación más destacadas en este tema. Los campos de mayor importancia, considerando los documentos publicados son: Engineering con 183

registros de publicaciones, Computer science con 162 registros y operations research con 17 registros.

Figura 4.

Área temática, tomado de Web of Science (2023)

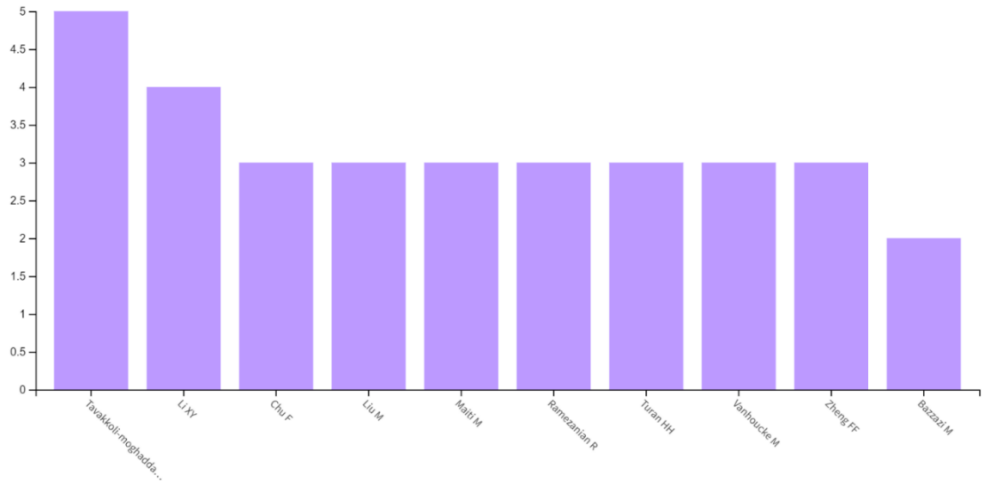


2.1.1.1.3. Autores

La figura 5 exhibe a los 10 autores más destacados en la investigación, teniendo en cuenta la cantidad de documentos que han publicado. Es relevante mencionar que el autor con mayor cantidad de publicaciones registradas en Web Of Science es Tavakkoli-Moghaddam R. con 5 artículos, seguido por Li XY. con 4 registros, mientras otros autores como Chu F., Liu m., Maití M., entre otros, cuentan con 3 documentos publicados en la base de datos.

Figura 5.

Autores de documento, tomado de Web of Science (2023)

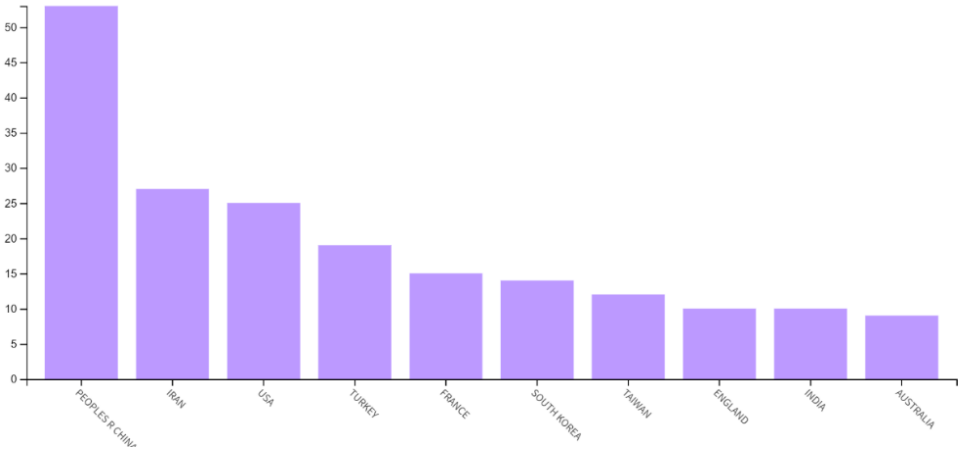


2.1.1.1.4. Países

En la figura 6 se muestran los países con el mayor número de documentos publicados, donde destaca Peoples R China con un total de cincuenta y tres (53) registros, seguido por Irán con veintisiete (27) registros y Estados Unidos con veinticinco (25) registros.

Figura 6.

Países, tomado de Web of Science (2023)



La búsqueda en la base de datos Web of Science muestra resultados favorables para el tema de investigación gracias al aumento constante de artículos publicados en este campo cada año. Esto resalta la importancia de seguir explorando y profundizando en esta área específica de estudio, otorgando así un valor significativo a la investigación.

2.1.1.2. Indicadores de relación

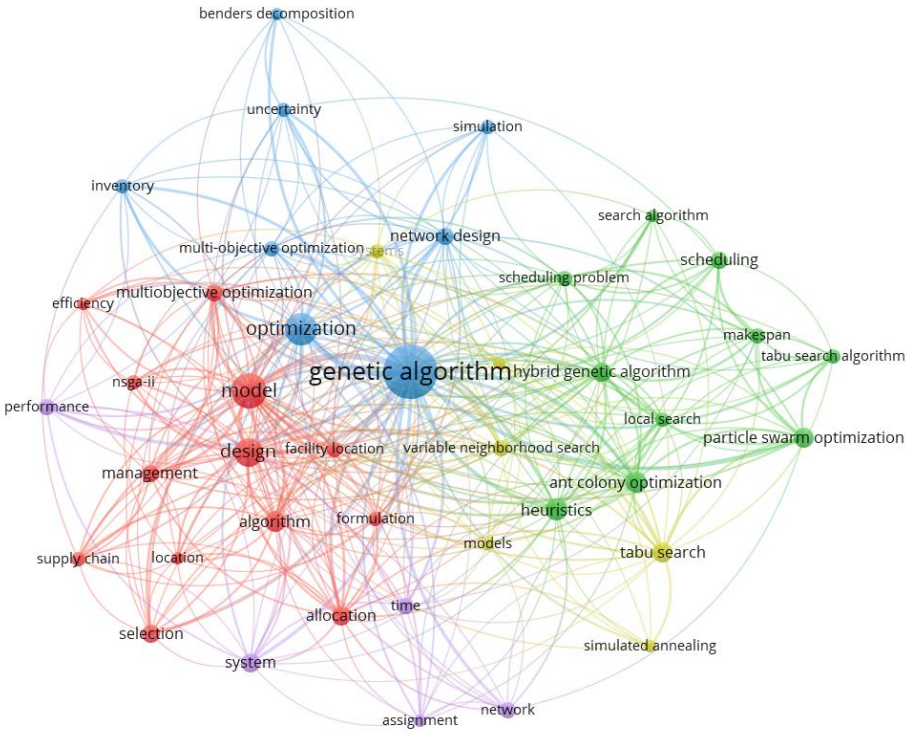
Con el objetivo de analizar los indicadores de relación, se recurrió al software VOSviewer, el cual facilita la creación y visualización de redes bibliométricas. Esta herramienta posee extracción de texto que resulta útil para identificar los términos relevantes extraídos de la literatura científica. Para llevar a cabo este proceso, se utilizó la información de Web of Science en formato CSV que posteriormente fue importada a VOSviewer.

2.1.1.2.1. Palabras claves

En la figura 7 se muestra cómo se agrupan palabras clave en 5 cluters diferentes, cada uno representado por un color distinto, para categorizar algoritmos de agrupamiento de acuerdo con temas relevantes. Se destacan los colores azul, rojo y verde, siendo "genetic algorithm" la palabra más frecuente, seguida de "optimization", "model" y "heuristics".

Figura 7.

Palabras clave, tomado de Web of Science.



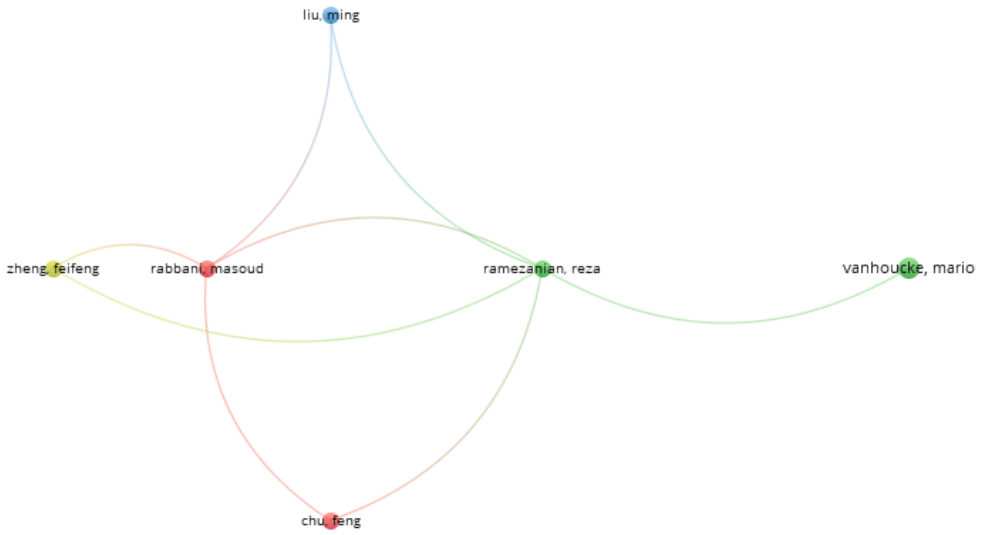
Nota: Esta figura fue generada por VOSviewer y elaborada teniendo en cuenta las palabras claves. (Visualizing scientific landscapes, 2023)

2.1.1.2.2. Citación de autores

En la figura 8 se presentan 4 clústeres que muestran grupos de autores más relevantes según las citaciones a lo largo del tiempo. Se destaca que el autor Ramezaniam, Reza posee el círculo de mayor tamaño, lo que evidencia que ha sido citado con mayor frecuencia a lo largo de los años.

Figura 8.

Citación de autores, tomado de Web of Science.



Nota: Esta figura es extraída de VOSviewer, con elaboración propia teniendo en cuenta las palabras clave ya mencionadas. (Visualizing scientific landscapes, 2023)

2.1.2. Scopus

Se empleó la siguiente ecuación para realizar la búsqueda en Scopus.

Figura 9.

Ecuación búsqueda Scopus

$$sequencing\ of\ machines\ OR\ machine\ programming\ OR\ johnson's\ rule\ OR\ genetic\ algorithm$$
$$AND\ carrer\ OR\ industrial\ engineering\ AND\ NOT\ medicine\ OR\ medical$$

Según la ecuación previamente mencionada, se han encontrado un total de 487 documentos hasta la fecha del 27 de julio de 2023. Luego de aplicar un proceso de depuración manual siguiendo los lineamientos del proyecto en cuestión, se han seleccionado un total de 96 documentos.

2.1.2.1. Indicadores básicos

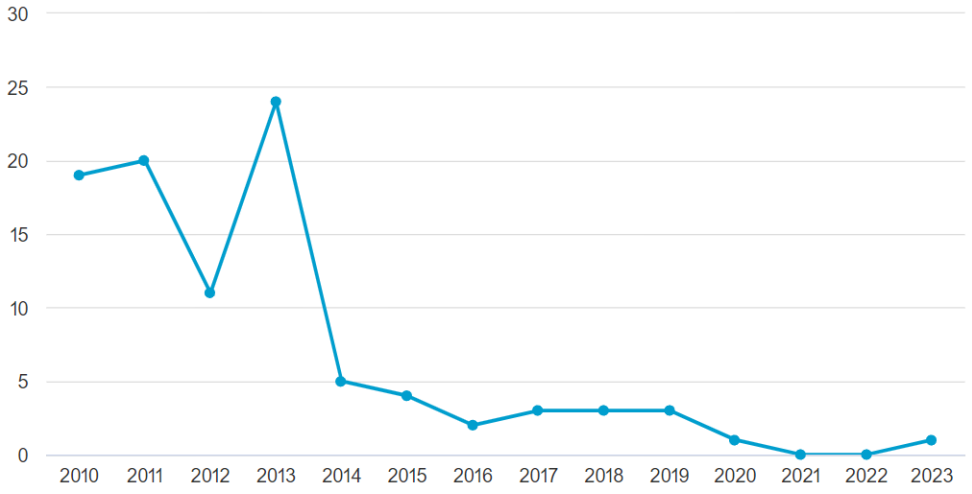
Luego de realizar el análisis de los datos obtenidos durante la búsqueda, se procedió a la generación de una serie de gráficos estadísticos; esto con el objetivo de visualizar de manera más clara y precisa los patrones y distribuciones observadas en los resultados.

2.1.2.1.1. Documentos por año

En la figura 10 se observa un incremento progresivo en el interés sobre el tema en el año 2013, seguido de un descenso en los años siguientes. No obstante, en el transcurso del año 2023, se identifica la publicación de un documento relacionado con este tema. De manera congruente con lo reportado en la base de datos de Web of Science, se puede constatar un interés limitado por parte de la comunidad académica en realizar investigaciones vinculadas con este tema.

Figura 10.

Documentos por año, tomado de Scopus (2023)

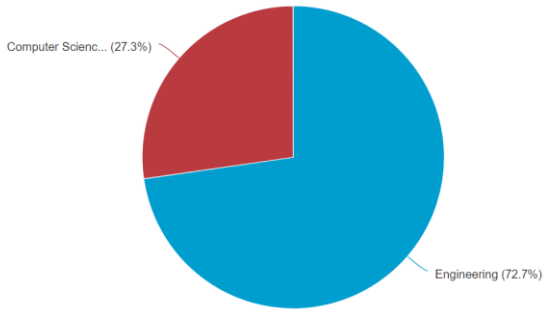


2.1.2.1.2. Área temática

En la figura 11 se resaltan las áreas de investigación de mayor relevancia, basándose en los documentos más importantes publicados. Se observa que el 27.3% de las publicaciones pertenecen al campo de Ciencias de la Computación, mientras que el 72.7% de las publicaciones están relacionadas con la Ingeniería. Esto demuestra una alta presencia de investigación en los campos de Ingeniería y Ciencias de la Computación, similar a lo que se observa en la base de datos de Scopus.

Figura 11.

Área temática, tomado de Scopus (2023).

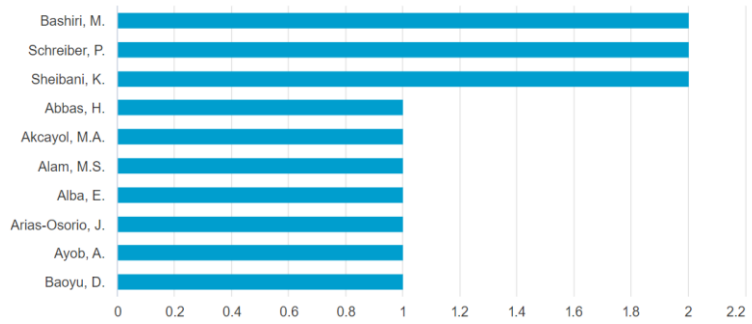


2.1.2.1.3. Autores

La figura 12 muestra a los 10 autores más relevantes en la investigación, basándose en el número de publicaciones realizadas. Se destaca que los autores Bashiri, M., Schreiber, P. y Sheibani, K., tienen 2 publicaciones en la base de datos Scopus, otros autores como Abbas, H. y Akcayol, M.A. tienen al menos un documento publicado en esta base de datos. Esta diversidad de autores refleja la variedad de material relevante disponible como referencia para la investigación. Es importante tener en cuenta que los autores presentes en Scopus difieren de los de Web of Science.

Figura 12.

Autores, tomado de Scopus (2023).



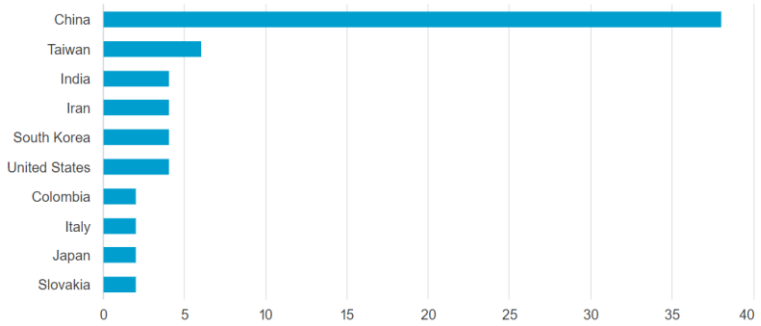
Nota: Esta figura fue tomada de Scopus y elaborada teniendo en cuenta a los autores que tienen mayor relevancia en la investigación.

2.1.2.1.4. Países

La Figura 13 presenta el ranking de países con mayor número de publicaciones en la base de datos, destacando a China como líder con un treinta y ocho (38), le siguen Taiwán con seis (6) registros, India, Irán, Corea del Sur y Estados Unidos tienen cuatro (4) publicaciones cada uno.

Figura *13*.

Países, tomado de Scopus (2023)



Nota: Esta figura fue tomada de Scopus y elaborada teniendo en cuenta los que tienen mayor relevancia en la investigación.

2.1.2.2. Indicadores de relación

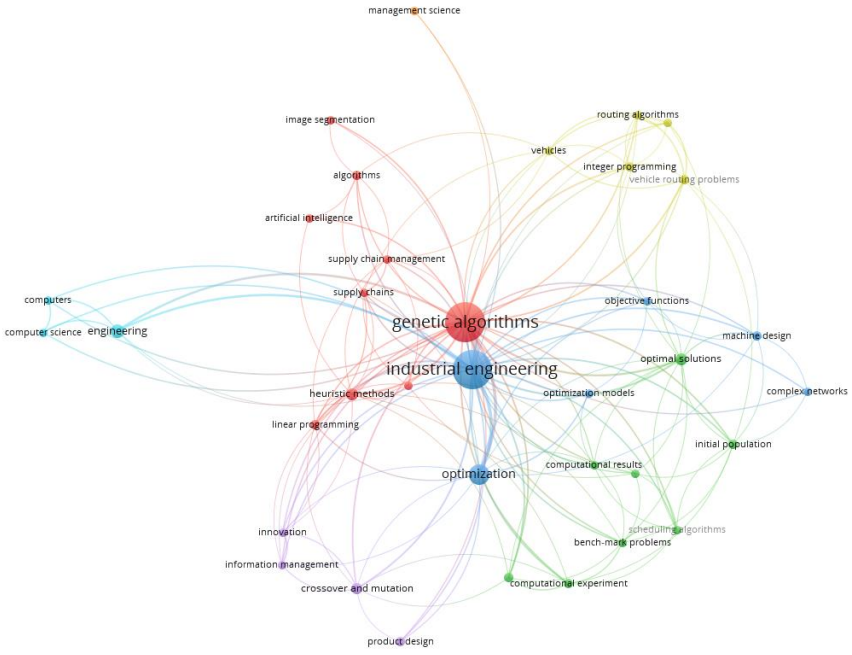
En base a lo realizado con Web of Science, se utilizó el software de VOSviewer para calcular los indicadores de relación.

2.1.2.2.1. Palabras claves

En la figura 14 se evidencian siete (7) clúster, las palabras con mayor repetición son genetic algorithms e industrial engineering. Es importante destacar que las esferas de mayor tamaño representan la frecuencia con la que aparecen dichas palabras en los artículos.

Figura 14.

Palabras clave en Scopus.



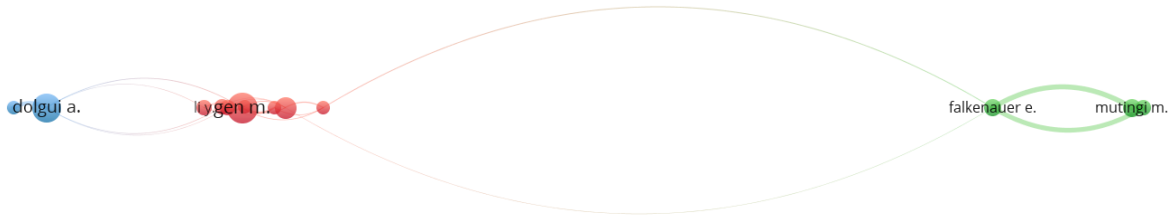
Nota: Esta figura fue tomada de VOSviewer y elaborada teniendo en cuenta las palabras clave con mayor relevancia en la investigación.

2.1.2.2.2. Citación de autores

En la figura 15 se muestra la citación de autores plasmados por colores en 3 clusters que representan la citación de los autores más destacados a lo largo del tiempo. En este contexto, se destaca que las esferas más grandes corresponden a Dolgui A., Gen M. y Falkenauer E., lo que indica que son los autores más relevantes a lo largo del tiempo en términos de citaciones.

Figura 15.

Citación de autores en Scopus.



Nota: Esta figura fue tomada de VOSviewer y elaborada teniendo en cuenta la citación de autores con mayor relevancia en la investigación.

2.1.3. Análisis preliminar de la literatura

En la actualidad, es esencial que las empresas dispongan de herramientas que contribuyan a lograr las exigencias del mercado buscando satisfacer las necesidades de los consumidores. Las empresas están sujetas al cambio, al mejoramiento continuo, por tal motivo, de manera adecuada realizan la programación de producción ya que día a día deben optimizar los procesos con el fin de maximizar la utilización de sus recursos obteniendo mayor eficiencia en las plantas de producción, mejorar los tiempos de entrega y reducir costos, logrando así tener un progreso sobre la competencia.

Los siguientes aportes de investigación tienen como objetivo indagar sobre secuenciamiento para un sistema de N trabajos en 2 máquinas, en cualquier parte del mundo.

En el año 2002, Jatinder N.D. Gupta, Karin Krüger, Volker Lauff, Frank Werner y Yuri N. Sotskov, realizaron un proyecto referente a: Heurística para talleres de flujo híbrido con tiempos de procesamiento controlables y fechas de vencimiento asignables (Gupta, Krüger, Lauff, Werner, & Sotskov, 2002). Este proyecto considera un problema de la programación de talleres de flujo como son las medidas de desempeño, tiempo de procesamiento y fechas de vencimiento, donde n trabajos se procesan en m máquinas utilizando algoritmos iterativos, búsqueda local, y algoritmos de inserción, obteniendo un buen rendimiento en dichos problemas.

Posteriormente, se aborda un trabajo titulado “Aplicación de la heurística de palmer en la secuenciación de n tareas en m máquinas: un caso de estudio” (Correa, 2010). Se planteó un problema de 10 tareas en 5 máquinas cuyo objetivo es minimizar el tiempo de terminación de cada una de las tareas, a través de la ecuación de índice propuesto por Palmer (French, 1982) se ordenaron las tareas de orden decreciente para obtener el valor de C_{max} (makespan).

En la búsqueda se aborda un trabajo referente a la aplicación de una heurística constructiva para la asignación de varios trabajos a varias máquinas en paralelo (Silva, Rivero, & Bolivar, 2013). Se tiene un grupo de 5 tareas en un taller de 4 máquinas, se inicia con la operación que menor tiempo de iniciación presente, se aplica el algoritmo hasta terminar la programación de todas las operaciones, el programa se ejecutó en Matlab donde permitió encontrar la solución del algoritmo obteniendo un buen resultado. Cabe resaltar, las empresas teniendo una adecuada administración de los sistemas de producción, logran un control significativo y que sobresalgan competitivamente.

Por otro lado, Eduardo Salazar Hornig y Juan Carlo Medina S, realizaron un artículo sobre la minimización del makespan en máquinas paralelas idénticas con tiempo de preparación dependientes de la secuencia utilizando un algoritmo genético (Salazar Hornig & Medina S, 2013). Consiste en minimizar el intervalo de tiempo en el que se procesan en su totalidad todos los trabajos, es decir, el proceso del primer trabajo y el tiempo de procesamiento de último trabajo, mediante el algoritmo genético (estándar con operador de cruzamiento PMX y operador de mutación swap), algoritmo genético incorporando la heurística del mejor vecino como algoritmo de mejora, la heurística LPT y la simulación Montecarlo al que se comparan. En el problema se consideran 2 máquinas paralelas idénticas y 6 trabajos, donde se presentan los tiempos de proceso de los trabajos y la matriz de tiempo de setup a través del diagrama de Gantt se ordenan los tiempos de procesamiento de mayor a menor. Obteniendo como resultado que el algoritmo LPT es el más bajo en rendimiento respecto a los otros, el algoritmo genético y simulación Montecarlo tienen un comportamiento similar, el procedimiento de algoritmo genético y mejor vecino tiene un buen rendimiento mejorando significativamente el desempeño respecto a el AG.

En el año 2014, Enrique Pérez Pérez, Ilse Castillo Pérez y Manuel E. Jiménez Bahri, realizaron un estudio sobre: Algoritmo genético para secuenciación de pedidos en taller de mecanizado con máquinas en paralelo, recirculación y tiempos de preparación (Pérez Pérez , Pérez Castillo , & Jiménez Bahri , 2014). Dicho proyecto consiste en establecer la secuencia donde debe ordenarse un conjunto de 20 pedidos de piezas mecánicas, correspondiente a una semana con jornadas de 8 horas diarias, minimizando el tiempo total de fabricación, recirculación, las tareas pueden pasar cero o más veces por la misma máquina, y tiempos de preparación. Generalmente, estos talleres trabajan bajo pedido, es decir, las piezas se elaboran a medida que los clientes las requieran siendo estas realizadas en el menor tiempo posible, sin alterar los tiempos de entrega y minimizando costos de producción. La fabricación de una pieza requiere una secuenciación de operaciones, donde se requiere una máquina para cada operación, en algunos casos, se repite eventualmente para diversas operaciones la misma máquina.

En el año 2022, Dilan Jhoanny Mogollón Carreño y Sebastián Elias Paéz Becerra llevaron a cabo un estudio titulado “Comparación del desempeño de dos algoritmos genéticos híbridos para el problema de formación de celdas de manufactura considerando el movimiento de los trabajadores” (Paéz Becerra & Mogollón Carreño, 2022). El objetivo principal del proyecto es reducir el tiempo máximo de finalización de las tareas (makespan) en un entorno productivo que involucra máquinas y trabajadores, con el fin de comparar cuál de las dos es la mejor se definieron parámetros fijos para cada algoritmo. Posteriormente, obtenidos los valores (el mínimo tiempo en la matriz inicial, el makespan y solución proporcionada por el algoritmo genético) al correr el modelo matemático para cada metaheurística por medio de Matlab, se obtuvo el mejor rendimiento en el algoritmo genético híbrido.

3. Planteamiento del problema

La reducción de costos hoy en día es uno de los desafíos que enfrentan las empresas, con el fin de hacer más eficaz y óptima la utilización de recursos, obteniendo una competitividad en el mercado y sobresalir ante sus competidores.

Los problemas de secuenciación se presentan constantemente en los entornos industriales y manufactureros, siendo este un problema de optimización combinatoria caracterizado por tener un número finito de soluciones (a nivel práctico son soluciones muy grandes, es decir, son cantidades grandes de soluciones que para calcularlas se vuelven extensos los procesos). La teoría de la complejidad los denomina NP-hard (o NP-complejo, o NP-difícil), dando lugar a utilizar metaheurísticas como el algoritmo genético (Gen & Cheng, 2000) siendo esta una buena opción ya que pueden adaptarse a dar una solución en un rango de tiempo establecido.

Una de las problemáticas fundamentales que abordamos en este proyecto se relaciona con los métodos exactos, dichos métodos nos ayudan a obtener mejores soluciones en tiempo real o práctico, a diferencia de las heurísticas, estas nos arrojan soluciones buenas en tiempo real a problemas complejos.

Para darle solución a la programación de n trabajos en 2 máquinas a través de la regla de Johnson, cuyo principal objetivo de este método es minimizar el tiempo de procesamiento desde el comienzo del primer trabajo hasta el final del último (Chase, Jacobs, & Aquilano, 2009, pág. 631). Dicho método consiste en una serie de pasos que se describirán a continuación:

- Paso 1, se nota el tiempo de operación de cada trabajo en ambas máquinas.
- Paso 2, se selecciona el trabajo con el tiempo más corto.

- Paso 3, si el trabajo con el tiempo más corto debe realizarse en la primera máquina, se programa como el primer trabajo, si debe realizarse en la segunda máquina, se programa con el último trabajo. En caso de empate, se asigna el trabajo a la primera máquina.
- Paso 4, se repiten los pasos 2 y 3 con los trabajos restantes hasta completar la programación.

Este trabajo se centra en la creación de un algoritmo genético y la regla de Johnson utilizados para el secuenciamiento de un sistema de N trabajos en 2 máquinas mediante Python. La metaheurística, algoritmo genético, empleada se utiliza para obtener soluciones que mejoren el rendimiento o contribuyan en el desempeño en los problemas de optimización combinatoria. La heurística, regla de Johnson, se emplea con el propósito de hacer más sencilla la solución de problemas complejos o difíciles.

Con esta investigación se busca comparar la metaheurística algoritmo genético y la heurística regla de Johnson a través de Python, con el fin de encontrar soluciones factibles teniendo como objetivo minimizar los tiempos de procesamiento de las máquinas (makespan). Cabe resaltar que en la base de datos los artículos encontrados no se detectan proyectos donde hagan la comparación entre estos dos.

3.1. Pregunta de investigación

Basándonos en el análisis del problema, surge la pregunta que esta investigación busca resolver. ¿Es más eficaz la metaheurística algoritmo genético en comparación a la regla de Johnson para el secuenciamiento de un sistema N/2 (N trabajos en 2 máquina)?

4. Marco de antecedentes

En primer lugar, se presenta un proyecto de grado que se realizó en la Universidad Libre Seccional de Bogotá, como tesis pregrado de Ingeniería Industrial, Karen Yulied Alfonso Albarracín y Sandra Aponte Limas (2013) desarrollaron un proyecto titulado "Desarrollo de una aplicación computacional bajo algoritmos genéticos para la secuenciación de trabajos/ordenes de la celda de manufactura HAS-200".

Este proyecto tiene como objetivo desarrollar una aplicación computacional que utiliza algoritmos genéticos para mejorar la secuenciación de la celda de manufactura HAS-200, utiliza la aplicación Edmes la cual presenta deficiencias en el cálculo del tiempo total de procesamiento (makespan). Para ello se utiliza la aplicación Sekvens con el fin de obtener el mínimo makespan a través de algoritmos genéticos. Se lleva a cabo una comparación detallada entre ambas aplicaciones, tanto cuantitativa como cuantitativamente, evaluando las interfaces de cada una y los tiempos totales de procesamiento de las órdenes a lanzar, evidenciando las mejoras que se proponen para la nueva aplicación obteniendo beneficios en el usuario, con el fin de facilitar el aprendizaje sobre la secuenciación utilizando algoritmos genéticos, se llevará a cabo un tutorial a través de la plataforma Sekvens.

En segundo lugar, se presenta un proyecto de grado que se realizó en la Universidad Libre, como tesis pregrado de Ingeniería Industrial, Malka Garcia Pereira (2017) desarrollaron un proyecto titulado "Diseño de una heurística eficiente basada en teoría de restricciones para la programación de sistemas Job Shop".

El propósito de este proyecto es crear una heurística eficaz basada para la programación de sistemas productivos Job Shop, los datos que serán utilizados son generados de manera aleatoria

utilizando un diseño experimental y analizados a través de una hoja de cálculo de Excel. Por medio de visual Basic de Excel se ejecutó la heurística cuello de botella, este problema se probó para un total de 100 instancias donde cada una presenta 20 trabajos; se obtuvo un buen rendimiento de la heurística.

Por último, se presenta un proyecto de grado que se realizó en la Universidad Tecnológica de Pereira, como tesis pregrado de Ingeniería Industrial, Valentina Salazar Álvarez (2019) desarrollaron un proyecto titulado “Estado del arte del problema de secuenciación de tareas implementando reglas de despacho”.

El propósito de este proyecto es profundizar en el estudio de los problemas de secuenciación de tareas utilizando reglas de despacho mediante métodos heurísticos y algoritmos, logrando la optimización del tiempo y el costo. Se identifican las reglas de despacho y heurísticas existentes en la literatura que han surgido a lo largo de las décadas, siendo estas clasificadas como: técnicas exactas y técnicas aproximadas. Posteriormente, se lleva a cabo la verificación de los resultados obtenidos mediante software.

4.1. Marco teórico

4.1.1. Secuenciación de trabajos

El proceso de determinar qué tarea se debe realizar primero en un centro de trabajo se denomina secuenciamiento de prioridades. Las reglas de prioridad son criterios que se utilizan para organizar las tareas en función de su fecha de vencimiento, orden de llegada o la regla de Johnson. Estas reglas se aplican para determinar el orden en el que se realizan las actividades. Para llevar a cabo este proceso de manera eficiente, es necesario contar con un programa de computación. (Carranza, 2023)

Las medidas de desempeño que se emplean para evaluar las normas de prioridad son las siguientes (Chase, Jacobs, & Aquilano, 2009, pág. 627):

- Cumplir con los plazos establecidos por los clientes o por las operaciones que siguen a fin de garantizar un buen servicio.
- Reducir el tiempo de tránsito siendo este el tiempo que requiere un trabajo en el proceso.
- Reducir la cantidad de proyectos inconclusos almacenados.
- Reducir el tiempo de inactividad de las máquinas o las tareas es fundamental para optimizar la productividad y eficiencia en cualquier entorno laboral.

Los métodos de secuenciación se rigen por reglas de prioridad que permiten establecer un orden adecuado en la realización de los trabajos. Entre las reglas de prioridad más utilizadas se encuentran:

- Primero en entrar primero en salir (PEPS), los trabajos se procesan en el orden en que llegan al centro de trabajo.

- Tiempo de procesamiento más corto (TPC), se priorizan los trabajos de menor duración, lo que permite completarlos más rápidamente.
- Fecha de entrega más próxima (FEP), los trabajos con fecha de entrega más cercana se programan primero para asegurar su cumplimiento.
- Tiempo de procesamiento más largo (TPL), los trabajos más largos y significativos se seleccionan primero, ya que su duración puede influir en el proceso.
- Razón crítica (RC), se calcula dividiendo el tiempo restante hasta la fecha de entrega entre el tiempo disponible para completar la tarea, lo que ayuda a priorizar los trabajos críticos.
- Regla de Johnson, este algoritmo heurístico se utiliza para abordar la secuenciación de N trabajos que deben pasan a través de 2 máquinas o centros de trabajo en procesos de producción (López B. S., 2019).

4.1.2. Optimización combinatoria

Un problema de optimización combinatoria es aquel en el que el conjunto de posibles soluciones es discreto, lo que implica una alta complejidad para encontrar la solución óptima debido a la gran cantidad de soluciones factibles que se tienen (Garcia, 2006). La optimización combinatoria desde la perspectiva computacional busca el desarrollo de soluciones algorítmicas a problemas donde se requiere optimizar (maximizar o minimizar) una función objetivo (Aguilar, 2017).

Los algoritmos de optimización combinatoria se relacionan con problemas Np Hard. Estos son los modelos más complejos que se encuentran en la investigación de operaciones, además no existe un método que garantice soluciones óptimas, es decir, solo se logran soluciones factibles (Garcia, 2006).

4.1.2.1. Teoría de la complejidad computacional.

La creación de herramientas y mecanismos que puedan abordar y analizar la complejidad de un problema es esencial (Complejidad computacional, 2023). A continuación, se abordarán las clases de complejidad computacional con el fin de identificar los problemas de acuerdo con su "dificultad" (Vidal Esmorís, 2013).

- **Clase P:** Está conformada por problemas que son fáciles o tratables, los cuales se pueden resolver eficazmente mediante la ejecución de un programa con un tiempo de ejecución considerablemente rápido.
- **Clase NP:** Esta clase trata diversos problemas de optimización que abarcan una amplia variedad de enfoques y soluciones, cuyo objetivo principal es encontrar buenas soluciones o mejores a las conocidas, dichos problemas tienen una gran complejidad ya que incluyen aquellos que pertenecen a los problemas de la clase P.
- **Clase NP-Complete:** Se refiere a los problemas que requieren una cantidad significativa de tiempo para su solución y, en ocasiones, resultan ser prácticamente intratables. Esta clase representa los problemas más arduos dentro de la categoría NP.
- **Clase NP-Hard:** Estos problemas presentan una complejidad desconocida, a diferencia de la clase NP, donde los tiempos de solución y verificación aumentan significativamente a medida que la magnitud del problema aumenta se vuelve más complicado de abordar.

4.1.3. Algoritmo de solución

4.1.3.1. Algoritmos exactos o búsqueda exhaustiva.

Un algoritmo es una secuencia finita de operaciones que se ejecutan de manera inequívoca y que proporciona una solución a un problema. Los algoritmos exactos son aquellos que siempre ofrecen una solución óptima, estos algoritmos implican evaluar todas las posibles soluciones en el conjunto de opciones hasta encontrar la mejor solución (Vidal Esmorís, 2013).

4.1.3.2. Heurística.

Una heurística es un procedimiento diseñado específicamente para encontrar una solución a un problema, un algoritmo se considera heurístico cuando busca una solución a través de ensayos, pruebas y ensayos, suele hacer referencia a un procedimiento que busca una solución, no garantiza necesariamente encontrar la mejor solución (Garcia, 2006). Es decir, encuentran soluciones de buena calidad, pero renuncian a encontrar una solución global del problema.

4.1.3.2.1. Regla de Johnson.

Es un algoritmo heurístico empleado para resolver problemas de secuenciación de procesos que involucran dos o más operaciones que transitan por dos máquinas o centros de trabajo. Su principal objetivo radica en la minimización del tiempo total de procesamiento del grupo de trabajos (López B. S., 2019), dicho algoritmo va teniendo una secuencia de tareas que van a ser procesadas, teniendo el orden de las tareas de menor tiempo. La regla de Johnson consta de los siguientes pasos (Chase, Jacobs, & Aquilano, 2009, pág. 631):

1. Se registra el tiempo de operación de cada trabajo en ambas máquinas.
2. Se selecciona el trabajo con el tiempo más corto.

3. Si el tiempo más corto corresponde a la primera máquina, se programa como el primer trabajo; si corresponde a la segunda máquina, se programa como el último trabajo. En caso de empate, se programa en la primera máquina.
4. Repita los pasos 2 y 3 con los trabajos restantes hasta completar la programación.

4.1.3.3. Metaheurística

Se especializa en la resolución de problemas de optimización que son definidos como difíciles de resolver mediante la exploración del espacio de soluciones (Jimenez Morales, 2012), el término metaheurística se deriva de la combinación de heurística con el sufijo meta, que significa "más allá" o "a un nivel superior". Las metaheurísticas son estrategias inteligentes diseñadas para mejorar procedimientos heurísticos muy generales, ofreciendo un alto rendimiento (Melián, Moreno Perez, & Moreno Vega, 2003). En este proyecto, se va a utilizar la metaheurística de Algoritmo Genético (GA), estas metaheurísticas se basan en conjuntos de procedimientos evolutivos a lo largo del espacio de soluciones para alcanzar la solución deseada.

4.1.3.3.1. Algoritmo genético.

Estos algoritmos utilizan la reproducción de seres vivos como modelo para resolver problemas de optimización, imitando el proceso evolutivo (López J. C., 2010), se utilizan particularmente en problemas que no permiten una solución eficiente, ya que exploran el espacio de soluciones con el propósito de evitar quedar atrapados en óptimos locales y en lugar de eso buscar óptimos globales (Garduño Juárez, 2018).

4.1.4. Mantenimiento correctivo.

La efectividad del sistema productivo de una empresa se ve directamente influenciada por el mantenimiento de su maquinaria, un fallo en el equipo puede ocasionar una interrupción significativa en la producción, prolongando así los tiempos de inactividad. El mantenimiento correctivo es una acción que no está planificada ya que se presenta en cualquier momento cuando la máquina está trabajando (Juan-Muns, 2016).

4.1.4.1. Tiempo de reparación

El tiempo de inactividad de una máquina va desde la detección de fallos hasta la finalización de su reparación, momento en el cual la máquina vuelve a estar operativa. Esta duración puede variar según la complejidad del daño, la disponibilidad de repuestos y el rendimiento del equipo de mantenimiento, entre otros factores.

4.1.4.1.1. Tiempo medio de reparación

Representa el promedio del tiempo requerido para restaurar un sistema (generalmente relacionado con asuntos técnicos o mecánicos), abarcando tanto el período de reparación como el de prueba. Este indicador sigue contando el tiempo hasta que el sistema esté completamente funcional nuevamente, sin detener el reloj en ningún momento.

4.1.5. Herramienta computacional Python.

Python es un lenguaje de programación muy utilizado en el desarrollo de aplicaciones web y software. Se destaca por ser uno de los más fáciles de aprender y eficientes, lo que lo convierte en uno de los más populares del mundo. Este lenguaje tiene sus bases en C y C++, con influencia del sistema operativo UNIX.

5. Descripción del problema y modelo matemático

5.1. Descripción del problema

Se considera un secuenciamiento de un sistema de N trabajos en 2 máquinas, donde se van a hacer variaciones dependiendo el tiempo de cada máquina empleado para realizar cada proceso, esto con el fin de optimizar el makespan, adicional a eso, se quiere revisar cuanto tiempo se demora cada máquina en realizar un proceso cuando presenta una falla, es decir, analizar el tiempo de reparación. Para ello se tienen dos máquinas las cuales se encuentran en serie donde los trabajos en cada método varían, dependiendo del ordenamiento más factible para ser procesados los pedidos. Por medio del algoritmo genético y la regla de Johnson se puede validar cuál algoritmo contribuye a la reducción del tiempo de procesamiento en la fabricación de pedidos, erradicando el desperdicio de tiempos, los cuales repercuten directamente en el producto final y tiempos de entrega.

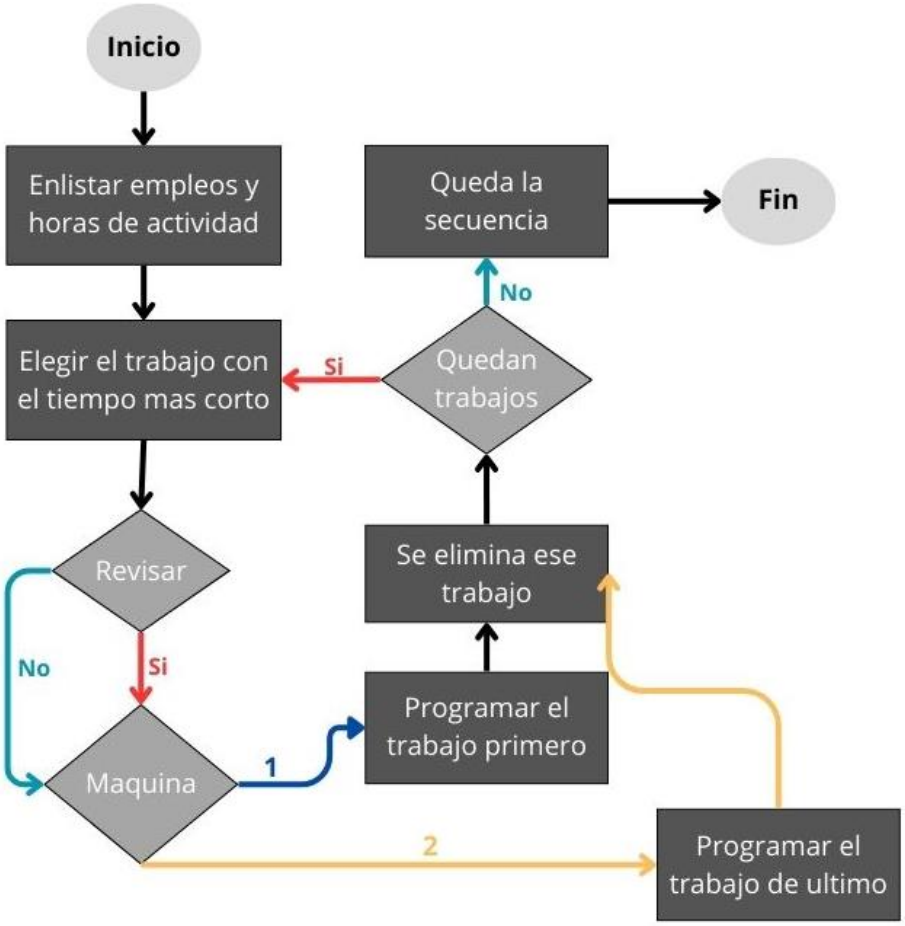
A continuación, por medio de diagramas de flujo se mostrará cómo funciona el algoritmo genético y la regla de Johnson.

- Regla de Johnson

La regla de Johnson es utilizada para minimizar el tiempo ocioso, el tiempo de procesamiento estableciendo la secuenciación de trabajos en dos centros de trabajo (2 máquinas). Dicha regla tiene una serie de pasos a seguir con los cuales se llega a minimizar tiempos en la fabricación de pedidos, estos pasos se encuentran enlistados anteriormente. Con lo anterior, en la figura 16 se observa el diagrama de flujo de la regla de Johnson.

Figura 16.

Diagrama de flujo regla de Johnson



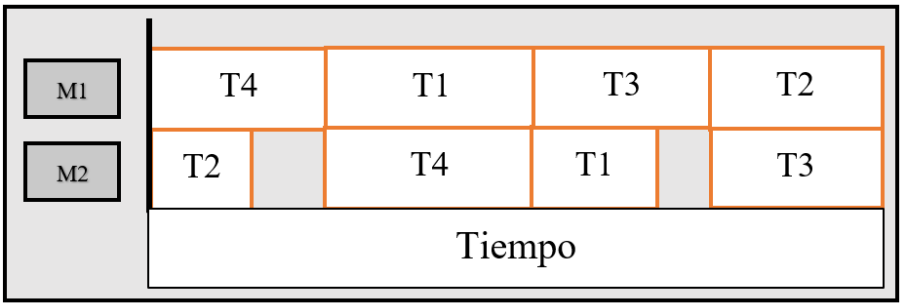
Nota: Esta figura tiene elaboración propia.

En la figura 16 se observa la forma de establecer en la regla de Johnson la mejor secuencia que minimiza el makespan, a la hora de obtener la mejor secuencia, a través del diagrama de Gantt se realiza el dibujo donde en el eje Y se muestra los centros de trabajos (2 máquinas) y en el eje X se muestra los tiempos de procesamiento de cada centro de trabajo.

En la Figura 17 se observa un sistema de secuenciamiento de N trabajos en 2 máquinas, en el que 2 o más trabajos deben procesarse en 2 máquinas según un orden. Por lo anterior, aunque los trabajos lleguen a la primera máquina, el problema de programación comienza a formar tiempos ociosos en la segunda máquina, es por ello, que se debe programar para que las máquinas operen de manera eficiente.

Figura 17.

Diagrama de Gantt para un sistema N/2



- Algoritmo genético

El algoritmo genético está basado en el proceso genético de los organismos vivos para generar soluciones efectivas a problemas reales. Dichos algoritmos parten de una población de individuos, cada uno de ellos representa una solución posible al problema, a cada individuo se le asigna un valor que refleja su eficacia en la resolución del problema, a mayor eficacia, mayor será la probabilidad de que el individuo sea seleccionado para reproducirse. En caso de ser menor la adaptación menor será la probabilidad de ser seleccionado para reproducirse, a lo que se produce una nueva población de posibles soluciones, reemplazando la anterior y verificando la proporción de las mejores características comparándola con la anterior con el fin de propagar las mejores características a través de la población lo que favorece el cruce de individuos mejor adaptados.

Con lo anterior, en la figura 18 se observa el diagrama de flujo del algoritmo genético para obtener la mejor secuencia

Figura 18.

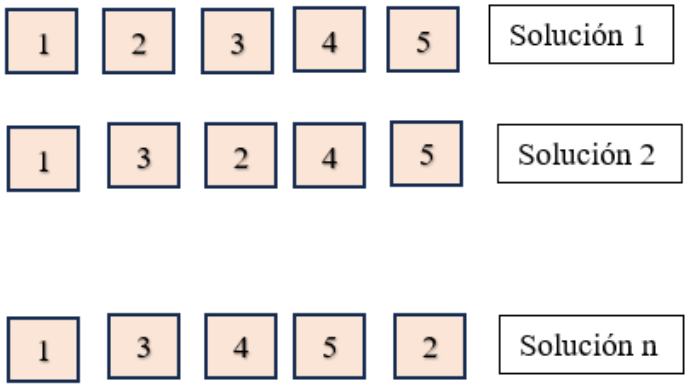
Diagrama de flujo Algoritmo Genético

Nota: Tomado de "Diagrama de flujo algoritmo genético simple. (n.d). ResearchGate"

En la Figura 19 se puede observar cómo el algoritmo genético trabaja los sistemas de secuenciamiento de máquinas, ya que el algoritmo genético consiste en conservar partes de la antigua solución y se le van haciendo cambios o modificaciones.

Figura 19.

Algoritmo genético



El algoritmo genético es ideal para problemas de optimización de grandes magnitudes mediante procesos de interacciones se busca mantener a través de generaciones una población final óptima que satisfaga las restricciones definidas, para ello, se debe definir la estructura del cromosoma que debe quedar a representación de las posibles soluciones del problema. A continuación, se abordarán los operadores genéticos principales y los parámetros que van a dar estructura al problema de optimización

- Función: Es un elemento determinante ya que establece el proceso de evolución de los individuos ya que se evalúan los valores de aptitud del cromosoma o individuo, determinando si es seleccionado o desechado.
- Población inicial: Es un elemento crucial ya que al tener un tamaño considerablemente grande es probable concurrir a un óptimo global, aunque esto dure un buen tiempo.

- Selección: Es encargado de codificar el algoritmo para producir con mayor frecuencia a los individuos con mejores aptitudes. Existen dos métodos de selección que comúnmente son usados como son la selección por ruleta y por torneo, la primera de ella asigna a cada cromosoma o individuo una sección de ruleta que es proporcional a los niveles de aptitud, y la segunda de ellas enfrenta los cromosomas que son seleccionados al azar donde se escoge el cromosoma con el mejor nivel de aptitud para la reproducción.
- Cruce: Es un operador que permite la formación y codificación de nuevas generaciones, permitiendo intercambiar información en el proceso de la reproducción de cromosomas, donde se unen los padres para dar lugar a una nueva generación la cual contiene la misma información genética de ellos.
- Tamaño de población: Tamaño de la población inicial de cromosomas, este representa una posible solución al problema.
- Número de generaciones: Cantidad de generaciones que el algoritmo genético iterará, esta generación consiste en un ciclo completo de selección de padres, cruzamiento, mutación y evaluación de la aptitud de los descendientes.
- Número padres torneo: El número de cromosomas que se seleccionarán en cada torneo para determinar los padres, dicha selección es aleatoria y se escoge el mejor de ellos como padre.
- Probabilidad mutación: Es la probabilidad de que ocurra una mutación en un cromosoma durante la fase de mutación (la mutación es un operador genético que introduce variación en las partes de un cromosoma).

Definidos los operadores que componen el algoritmo genético, a continuación, en la figura 20 se abordará el pseudocódigo de funcionamiento del algoritmo genético, donde de manera detallada se puede observar cómo funciona cada operador.

Figura 20.

Pseudocódigo funcionamiento Algoritmo Genético

```
Iniciar población actual aleatoriamente
MIENTRAS no se cumpla el criterio de terminación
    Crear población temporal vacía
    SI copiar en población temporal mejores individuos
    MIENTRAS población temporal no llena
    Seleccionar padres
    Cruzar padres con probabilidad PC
    SI se ha producido el cruce
        Mutar uno de los descendientes
        Evaluar descendientes
        Añadir descendientes a la población temporal
    SINO
        Añadir padres a la población temporal
    FIN SI
    FIN MIENTRAS
    Aumentar contador generaciones
Establecer como nueva población actual la población temporal
FIN MIENTRAS
```

Nota: Tomado de "Dorado, Gestal, Rivero, Rabuñal, & Pazos (2010)"

La regla de Johnson tiene como objetivo reducir el tiempo de tránsito requerido para completar todos los trabajos, desde el inicio del primer trabajo hasta la culminación del último,

reduce el tiempo ocioso entre los dos centros de trabajo. Como anteriormente se describió, dicha regla presenta una serie de pasos que se deben tener en cuenta para llevar de manera adecuada una solución óptima para el problema a resolver. A continuación, en la figura 21 se abordará de manera más concisa el pseudocódigo del funcionamiento de la regla de Johnson.

Figura 21.

Pseudocódigo funcionamiento Regla de Johnson

Input: datos de instancia
Output: secuencia óptima $\prod$
Begin
Let $\prod = \emptyset, J = \{1, \dots, n\}$;
Let $J_1 = \{j \in J / p_{1,j} \leq p_{2,j}\}$;
Let $J_2 = \{j \in J / p_{1,j} > p_{2,j}\}$;
Construir $\prod_1$ *trabajos de clasificación en* J_1 *aumentando valores de* $p_{1,j}$*;*
Construir $\prod_2$ *trabajos de clasificación en* J_2 *disminuyendo valores de* $p_{2,j}$*;*
$\prod = (\prod_1 \mid \prod_2)$;
Return $\prod$

Nota: Tomado de "Framiñán, Leisten and Ruiz, (2014)"

5.2. Modelo matemático

Función objetivo: Minimizar los tiempos de procesamiento de las máquinas

Parámetros:

- N: Número total de trabajos
- M: Número total de máquinas (es un sistema N/2, por ende, M=2).
- P_{ij}: Tiempo de procesamiento del trabajo i en la máquina j, donde i=1, 2, …, n y j=1, 2

- S_i: Secuencia de máquinas para el trabajo i

Variables:

- C_i: Tiempo de finalización del trabajo i
- C_{max}: Tiempo máximo de terminación
- U_j: Orden de procesamiento de trabajos en la máquina j (j=2)
- T_i: Tiempo en el que el trabajo i es finalizado
- X_{ij}: Variable binaria que indica si el trabajo i es procesado en la máquina j

Restricciones:

- Cada trabajo debe ser procesado exactamente una vez en cada máquina

$$\sum_{j=1}^{m} x_{ij} = 1, \forall i \epsilon \{1,2, \dots, n\}$$

- La secuencia de procesamiento de los trabajos en cada máquina debe respetar el orden de secuencia establecido.

$$C_i + p_{i1} \leq C_j, \forall (i,j)\ \epsilon\ S$$

$$C_i + p_{i2} \leq C_j, \forall (i,j)\ \epsilon\ S$$

Donde S representa el conjunto de pares de trabajos que deben procesarse en orden secuencial.

6. Diseño de los algoritmos

Tabla 2.

Regla de Johnson

Heurística: Regla de Johnson
input: Números aleatorios **output:** Secuencia de trabajos óptima y tiempo de procesamiento.
1. Generar el aleatorio 2. Definir el secuenciamiento para almacenar la secuencia de tareas 3. Encontrar el menor tiempo de secuenciamiento en ambas máquinas para todas las tareas. 4. Encontrar el menor tiempo de procesamiento. 5. Encontrar las tareas con el menor tiempo de procesamiento. 6. Seleccionar aleatoriamente una tarea si hay más de una con el mismo menor tiempo. 7. Si el menor tiempo está en la primera máquina, la tarea se coloca al inicio de la secuencia 8. Si el menor tiempo está en la segunda máquina, la tarea se coloca al final de la secuencia 9. Eliminar la tarea seleccionada de la lista de tareas pendientes.

En la tabla 2 se observa el diseño de la heurística regla de Johnson con el fin de dar con la secuencia óptima y el mejor tiempo de procesamiento.

Tabla 3.

Programación regla de Johnson

```
import random
import numpy as np
import pandas as pd # SUPER REAL JOHNSON alearorios
import numpy as np
import csv
def Find_seq(DataSet):
    l=DataSet.shape[0]
    seq=[]
```

```
    for i in range(l):
        seq.insert(i,0)
    i,j=0,l-1
    while DataSet.empty==False:
        M1_min=DataSet['M1'].min()
        M2_min=DataSet['M2'].min()
        if M1_min>M2_min:
            index=DataSet['M2'].idxmin()
            job=DataSet['Job'][index]
            seq[j]=job
            j-=1
            DataSet = DataSet.drop([index], axis=0)
        else:
            index=DataSet['M1'].idxmin()
            job=DataSet['Job'][index]
            seq[i]=job
            i+=1
            DataSet = DataSet.drop([index], axis=0)
    return seq
def FindInOut_Table(DataSet,seq):
    InOut={'JobSeq':['M1_in','M1_out','M2_in','M2_out']}
    for i in range(DataSet.shape[0]):
        if i==0:
            idx = DataSet[DataSet['Job']==seq[i]].index.values.astype(int)
            InOut[seq[i]]=[0,DataSet['M1'][idx[0]],DataSet['M1'][idx[0]],DataSet['M1
'][idx[0]]+DataSet['M2'][idx[0]]]
        else:
            idx = DataSet[DataSet['Job']==seq[i]].index.values.astype(int)
            M1outTemp=InOut[seq[i-1]][1]+DataSet['M1'][idx[0]];
            if M1outTemp>InOut[seq[i-1]][3]:
                M2inTemp=M1outTemp
            else:
                M2inTemp=InOut[seq[i-1]][3]
            InOut[seq[i]]=[InOut[seq[i-
1]][1],M1outTemp,M2inTemp,M2inTemp+DataSet['M2'][idx[0]]]

    InOutTable=pd.DataFrame.from_dict(InOut, orient='index')
    return InOutTable
```

```
def Calculate_FlowAndIdleTime(InOutTable):
    print("\nTotal Flow Time: ",InOutTable[3][InOutTable.shape[0]-1])
    M1_IdleTime=InOutTable[3][InOutTable.shape[0]-1]-
InOutTable[1][InOutTable.shape[0]-1]
    print("M1 Idle Time: ", M1_IdleTime)
    M2_IdleTime=0
    for i in range(InOutTable.shape[0]-1):
        if i==0:
            M2_IdleTime=InOutTable[2][1]-InOutTable[0][1]

        else:
            M2_IdleTime=M2_IdleTime+(InOutTable[2][i+1]-InOutTable[3][i])
    print("M2 Idle Time: ", M2_IdleTime)
# Crear la lista de diccionarios con los datos de la matriz
# Definir el tamaño de la matriz
num_rows = 5
num_cols = 3
# Establecer la semilla aleatoria para asegurar reproducibilidad
random.seed(42)
# Generar valores enteros aleatorios para la matriz
data = {
    'Job': [chr(65 + i) for i in range(num_rows)],
    'M1': np.random.randint(15, 30, size=num_rows),
    'M2': np.random.randint(10, 20, size=num_rows)
}
# Crear el DataFrame a partir de la lista de diccionarios
dataset = pd.DataFrame(data)
print("\nSize:- ", dataset.shape)S
print("\n", dataset)
seq=Find_seq(dataset)
print("\nSequence: ",seq,"\n")
InOutTable=FindInOut_Table(dataset,seq)
print(InOutTable)
Calculate_FlowAndIdleTime(InOutTable)
```

En la tabla 3 se puede observar la programación utilizada para el desarrollo del proyecto a través de la regla de Johnson, la cual nos ejecuta los tiempos de procesamiento. Los datos utilizados

por la función Random se usan para generar números aleatorios por medio Numpy se hicieron arreglos numéricos y, pandas se hace el respectivo análisis de los datos, separando por comas la secuencia. Para la matriz se definió el tamaño de filas y columnas, se definieron los valores aleatorios para la máquina 1 y la máquina 2 para posteriormente la programación arrojarnos el tiempo de procesamiento.

Tabla 4.

Algoritmo genético

Metaheurística: Algoritmo genético
input: Números aleatorios. **output:** Secuencia de trabajos óptima y tiempo de procesamiento.
1. Definir datos del problema 2. Función para calcular el tiempo total de procesamiento de una secuencia de tareas. 3. Función para generar una población inicial de cromosomas. 4. Función de selección de padres por torneo. 5. Función de cruzamiento (recombinación) de dos padres para producir un hijo. 6. Función de mutación de un cromosoma (secuencia de tareas) 7. Función para ejecutar el algoritmo genético 8. Asignar parámetros del algoritmo genético 9. Ejecutar algoritmo genético 10. Imprimir resultados

En la tabla 4 se observa el diseño de la metaheurística algoritmo genético con el fin de dar con la secuencia óptima y el mejor tiempo de procesamiento.

Tabla 5.

Programación algoritmo genético

```
import random # GENETICO REAL REAL
import pandas as pd

# Definir la función para calcular el tiempo total de procesamiento de una
secuencia de tareas
def calcular_tiempo_total(secuencia, data):
    tiempo_maquina1 = 0
    tiempo_maquina2 = 0
    for tarea in secuencia:
        tiempo_maquina1 += data.loc[tarea, 'M1']
        tiempo_maquina2 = max(tiempo_maquina1, tiempo_maquina2) +
data.loc[tarea, 'M2']
    return tiempo_maquina2

# Función para generar una población inicial de cromosomas
def generar_poblacion_inicial(data, tamaño_poblacion):
    poblacion = []
    for _ in range(tamaño_poblacion):
        secuencia = data.index.tolist()
        random.shuffle(secuencia)
        poblacion.append(secuencia)
    return poblacion

# Función de selección de padres por torneo
def seleccionar_padres(poblacion, num_padres_torneo, data):
    padres = []
    for _ in range(len(poblacion)):
        torneo = random.sample(poblacion, num_padres_torneo)
        mejor_padre = min(torneo, key=lambda x: calcular_tiempo_total(x,
data))
        padres.append(mejor_padre)
    return padres
```

```
# Función de cruzamiento (recombinación) de dos padres para producir un hijo
def cruzar(padre1, padre2):
    punto_corte = random.randint(1, len(padre1) - 1)
    hijo = padre1[:punto_corte] + [tarea for tarea in padre2 if tarea not in
padre1[:punto_corte]]
    return hijo

# Función de mutación de un cromosoma (secuencia de tareas)
def mutar(cromosoma, probabilidad_mutacion):
    if random.random() < probabilidad_mutacion:
        indices_mutacion = random.sample(range(len(cromosoma)), 2)
        cromosoma[indices_mutacion[0]], cromosoma[indices_mutacion[1]] =
cromosoma[indices_mutacion[1]], cromosoma[indices_mutacion[0]]
    return cromosoma

# Función para ejecutar el algoritmo genético
def algoritmo_genetico(data, tamaño_poblacion, num_generaciones,
num_padres_torneo, probabilidad_mutacion):
    poblacion = generar_poblacion_inicial(data, tamaño_poblacion)
    for _ in range(num_generaciones):
        padres = seleccionar_padres(poblacion, num_padres_torneo, data)
        descendientes = []
        for i in range(0, len(padres), 2):
            hijo1 = cruzar(padres[i], padres[i+1])
            hijo2 = cruzar(padres[i+1], padres[i])
            hijo1 = mutar(hijo1, probabilidad_mutacion)
            hijo2 = mutar(hijo2, probabilidad_mutacion)
            descendientes.extend([hijo1, hijo2])
        poblacion = descendientes
    mejor_secuencia = min(poblacion, key=lambda x: calcular_tiempo_total(x,
data))
    mejor_tiempo = calcular_tiempo_total(mejor_secuencia, data)
    return mejor_secuencia, mejor_tiempo
# Establecer la semilla aleatoria para asegurar reproducibilidad
```

```
random.seed(42)
# Datos de ejemplo


# Crear el DataFrame a partir de los datos
dataset = pd.DataFrame(data)
print("\nSize:- ", dataset.shape)
print("\n", dataset)

# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 100
num_padres_torneo = 2
probabilidad_mutacion = 0.5

# Ejecutar el algoritmo genético
mejor_secuencia, mejor_tiempo = algoritmo_genetico(dataset,
tamaño_poblacion, num_generaciones, num_padres_torneo,
probabilidad_mutacion)

# Imprimir resultados
print("Mejor secuencia de tareas:", mejor_secuencia)
print("Tiempo total de procesamiento:", mejor_tiempo)
```

En la tabla 5 se observa la programación utilizada para el algoritmo genético, se definieron los valores aleatorios por medio de la función Random, se generaron funciones como fueron: generar la población inicial, seleccionar padres, cruzamiento de dos padres para producir un hijo, mutación del cromosoma el cual define la secuencia de tareas, función para ejecutar el algoritmo genético. Posterior a esto, se definen los parámetros como fueron tamaño población, número de generaciones, número padres torneo y la probabilidad de mutación; donde se puede imprimir los resultados y arroja el tiempo de procesamiento.

7. Evaluación del algoritmo

Para hacer el análisis del comportamiento de la regla de Johnson y el algoritmo genético en el problema de secuenciamiento de un sistema de N trabajos en 2 máquinas, se desarrollaron los algoritmos en Python a través de la plataforma Google Colab, siendo es la utilizada para hacer la ejecución de los algoritmos usados en el proyecto. Por medio de la programación, se realizó el diagrama de Gantt para mostrar de manera más ilustrativa la secuenciación de los trabajos para cada una de las máquinas. En la figura 22 y la figura 23 se observa el makespan que la programación arrojó los cuales fueron tomados para hacer el análisis respectivo.

Figura 22.

Ejecución makespan regla de Johnson

```
Size:-  (5, 3)

   Job  M1  M2
0   A  17  13
1   B  20  14
2   C  29  13
3   D  16  16
4   E  20  13

Sequence:  ['D', 'B', 'E', 'C', 'A']

            0       1       2       3
JobSeq  M1_in  M1_out  M2_in  M2_out
D           0      16     16      32
B          16      36     36      50
E          36      56     56      69
C          56      85     85      98
A          85     102    102     115

Total Flow Time:  115
M1 Idle Time:  13
M2 Idle Time:  46
```

Figura 23.

Ejecución makespan algoritmo genético

```
Size:-  (5, 3)

   Job  M1  M2
0   A  17  13
1   B  20  14
2   C  29  13
3   D  16  16
4   E  20  13
Mejor secuencia de tareas: [1, 0, 4, 3, 2]
Tiempo total de procesamiento: 115
```

Para los dos algoritmos se hicieron variaciones:

- Inicialmente se definieron valores aleatorios para la matriz donde se hicieron variaciones: en la tabla 6., se observa que para la máquina 1 (M1) y la máquina 2 (M2) los valores aleatorios que se tomaron para la M1 fueron (10-20) unidades de tiempo y la M2 fueron (15-30) unidades de tiempo; en la tabla 7. Se observa que para la máquina 1 (M1) y la máquina 2 (M2) los valores aleatorios que se tomaron para la M1 fueron (15-30) unidades de tiempo y la M2 fueron (10-20) unidades de tiempo. Es importante aclarar que los valores aleatorios se originaron por medio de una distribución rectangular, es decir, se tiene la misma probabilidad de ser seleccionados.

Tabla 6.

Valores aleatorios para la primera muestra de la distribución rectangular.

```
M1': np.random.randint(10, 20, size=num_rows)
M2': np.random.randint(15, 30, size=num_rows)
```

Tabla 7.

Valores aleatorios para la segunda muestra de la distribución rectangular.

```
M1': np.random.randint(15, 30, size=num_rows)
M2': np.random.randint(10, 20, size=num_rows)
```

- El número de trabajos donde se toma la decisión de realizar 3 tamaños clasificándolas: en la tabla 8., se observa unos n los cuales corresponden a pequeña para un n=30, mediana para un n=100 y grande para un n=300, donde cada uno de ellos tuvo un número de corridas de 100 y para el caso de un número de trabajos de n=5000 se tuvo un número de corridas

de 10. Estos n se tomaron con el fin de mirar qué comportamiento tiene la regla de Johnson y el algoritmo genético en cuanto al makespan y tiempos de ejecución.

Tabla 8.

Tamaño con número de corridas

N	Corridas
30	100
100	100
300	100
5000	10

- Se definieron 3 instancias de parámetros para el algoritmo genético donde se varía: en la tabla 9, tabla 10 y tabla 11, se observa la variación en el tamaño población, número generaciones, número padres torneo, probabilidad mutación.

Tabla 9.

Primera instancia de parámetros para el AG

```
tamaño_poblacion = 50
num_generaciones = 50
num_padres_torneo = 1
probabilidad_mutacion = 0.1
```

Tabla 10.

Segunda instancia de parámetros para el AG

```
tamaño_poblacion = 50
num_generaciones = 100
num_padres_torneo = 2
probabilidad_mutacion = 0.5
```

Tabla 11.

Tercera instancia de parámetros para el AG

```
tamaño_poblacion = 50
num_generaciones = 200
num_padres_torneo = 4
probabilidad_mutacion = 0.9
```

- Para concluir en la tabla 12 y la tabla 13 se observa el resumen de lo expuesto anteriormente, con el fin de tener un mejor entendimiento de lo abordado en la programación. Donde se definieron los valores aleatorios de las máquinas, los parámetros del algoritmo genético y el n con su respectivo número de corridas en la programación.

Tabla 12.

Resumen de la primera toma de datos

M1': np.random.randint(10, 20, size=num_rows)
M2': np.random.randint(15, 30, size=num_rows)

# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 50
num_padres_torneo = 1
probabilidad_mutacion = 0.1

N	Corridas
30	100
100	100
300	100
5000	10

# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 100
num_padres_torneo = 2
probabilidad_mutacion = 0.5

N	Corridas
30	100
100	100
300	100
5000	10

# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 200
num_padres_torneo = 4
probabilidad_mutacion = 0.9

N	Corridas
30	100
100	100
300	100
5000	10

Tabla 13.

Resumen de la segunda toma de datos

M1': np.random.randint(15, 30, size=num_rows)
M2': np.random.randint(10, 20, size=num_rows)

# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 50
num_padres_torneo = 1
probabilidad_mutacion = 0.1

N	Corridas
30	100
100	100
300	100
5000	10

# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 100
num_padres_torneo = 2
probabilidad_mutacion = 0.5

N	Corridas
30	100
100	100
300	100
5000	10

# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 200
num_padres_torneo = 4
probabilidad_mutacion = 0.9

N	Corridas
30	100
100	100
300	100
5000	10

7.1. Resultados diseño experimental

El algoritmo realizado nos permite observar que el programa realizado está ejecutando la regla de Johnson y el algoritmo genético del secuenciamiento para un sistema de N trabajos en 2 máquinas planteados en esta investigación.

Es importante aclarar que para el algoritmo genético se tuvieron en cuenta unos parámetros como fueron el tamaño de la población, número de generaciones, número de padres torneo y probabilidad de mutación como se observa anteriormente, los cuales fueron unos supuestos con el fin de mirar variación al momento de cambiar los parámetros mencionados.

Por último, los valores aleatorios no son generados por una distribución normal sino por una distribución rectangular, por lo que cada valor tiene igual probabilidad de salir, los resultados

tienden a ser normales. Esto se da por el teorema de límite central, teniendo un tamaño de muestras grande la forma de la distribución muestral de la media se aproxima a una curva normal, es importante aclarar, que el tamaño de muestra tiene que ser suficientemente grande.

Inicialmente se tomó el aleatorio de la máquina 1 entre 10 y 20 unidades de tiempo, para la máquina 2 se tomó el aleatorio entre 15 y 30 unidades de tiempo.

- El primer análisis que se realizó sobre la programación a la regla de Johnson y el algoritmo genético, para un tamaño de n=30, 100, 300 y 5000.

Para un tamaño de n=30 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=30
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 50
num_padres_torneo = 1
probabilidad_mutacion = 0.1
```

A través de un análisis de datos realizado en Excel, se obtiene que tanto para la regla de Johnson como el algoritmo genético presentan una variabilidad en datos que fueron originados aleatoriamente en la programación, se encuentra una diferencia en los parámetros del algoritmo genético ya que presenta el makespan más alto en comparación a la regla de Johnson. A continuación, a través del histograma de la figura 24 y la figura 25, se puede observar la distribución que tuvo el conjunto de datos en la regla de Johnson y el algoritmo genético donde nos representa la frecuencia que tuvieron los datos y por ello se representa en forma de barras, arrojando mejor el makespan en la regla de Johnson con un mínimo de 615 unidades de tiempo y un máximo de 729 unidades de tiempo con una media de 668.3, para el algoritmo genético un

mínimo de 616 unidades de tiempo y un máximo de 732 unidades de tiempo con una media de 670.09. El número de corridas que se realizaron para la toma de dichos datos fueron de 100, por lo que se aproxima a una curva normal, esto se da por el teorema de límite central dado que a un mayor número de corridas los datos tienden a ser normales, la media se encuentra ubicada en la clase con mayor frecuencia.

Figura 24.

Histograma regla de Johnson para un n=30

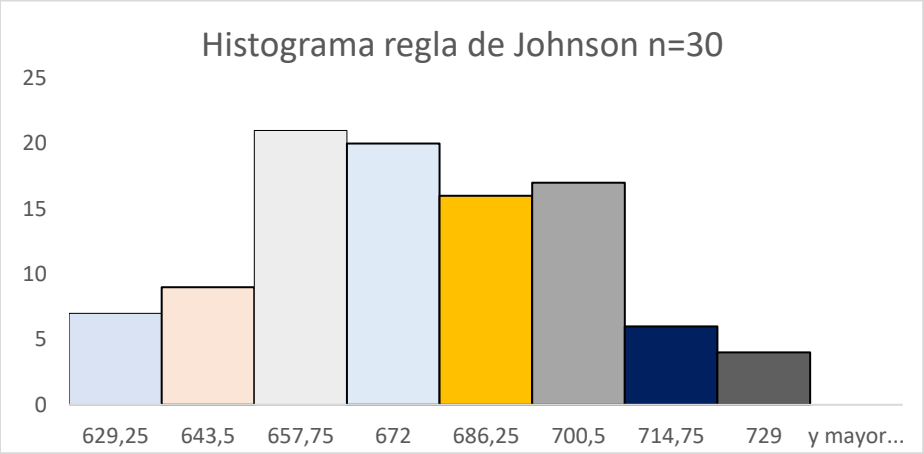
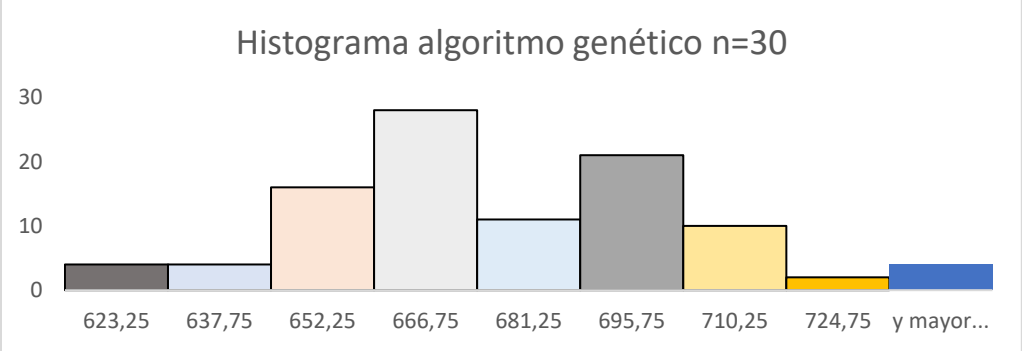


Figura 25.

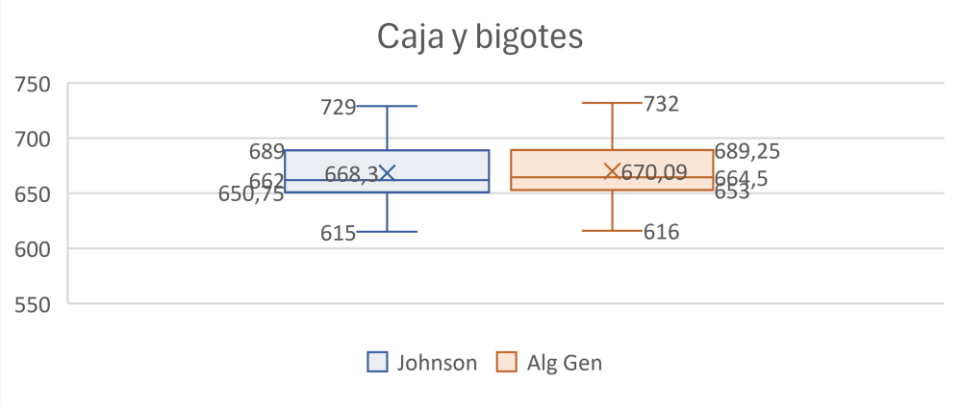
Histograma algoritmo genético para un n=30



En la figura 26 podemos observar que para la regla de Johnson nos presenta un extremo inferior con 615 y un extremo superior con 729, para el caso del algoritmo genético nos da un extremo inferior de 616 y un extremo superior de 732, de lo que podemos concluir que la dispersión es similar, ya que para la regla de Johnson entre el 25% y el 75% de los datos se encuentran entre el cuartil inferior 650.75 y el cuartil superior 689. Para el algoritmo genético los datos se encuentran entre el 25% y 75% es decir, entre el cuartil inferior 653 y el cuartil superior 689.25. Cabe resaltar que la mediana se encuentra cerca al centro de la caja, lo que quiere decir que los valores son más o menos simétricos.

Figura 26.

Caja y bigotes regla de Johnson y algoritmo genético n=30



Para un tamaño de n=100 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=100
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 50
num_padres_torneo = 1
probabilidad_mutacion = 0.1
```

Los datos de la programación muestran una variación entre la regla de Johnson y el algoritmo genético. Esta discrepancia se manifiesta en los makespan, siendo significativamente más altos para el algoritmo genético en contraste con la regla de Johnson. A continuación, en el histograma de la figura 27 se observa el makespan con un mínimo de 2105 unidades de tiempo y un máximo de 2326 unidades de tiempo para la regla de Johnson con una media de 2203.01, en la figura 28 para el caso del algoritmo genético presenta un mínimo con 2110 unidades de tiempo y un máximo de 2327 unidades de tiempo con una media de 2206.42. De lo anterior, se obtuvo que por medio de la regla de Johnson se obtuvo mejor el makespan.

Figura 27.

Histograma regla de Johnson n=100

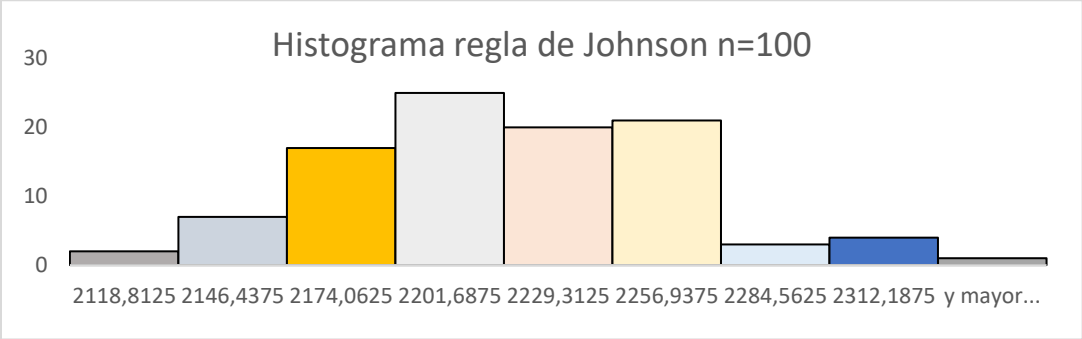
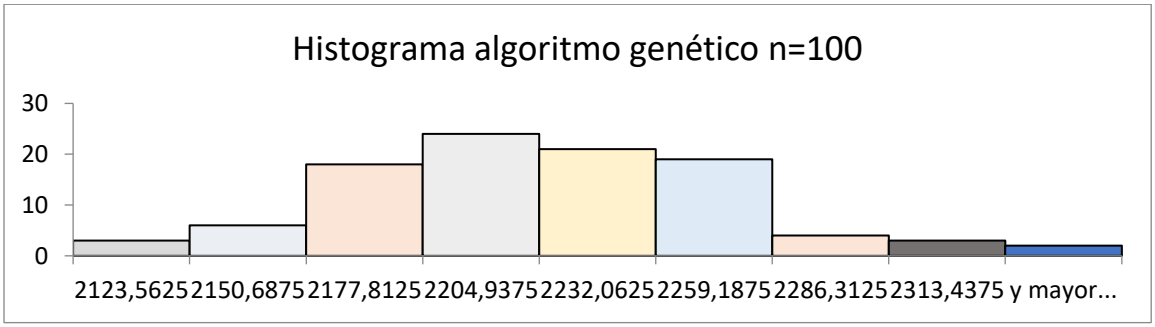


Figura 28.

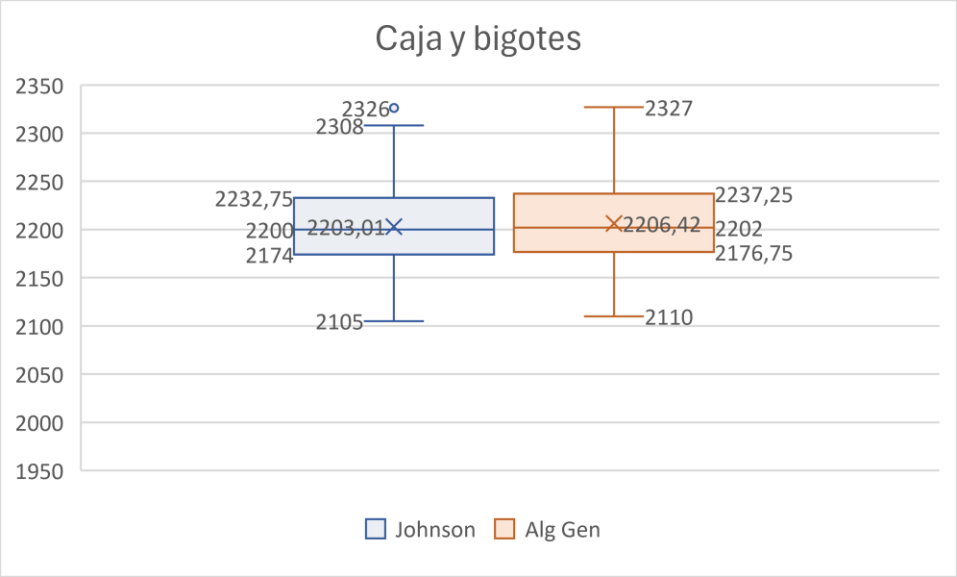
Histograma algoritmo genético n=100



De la figura 29 podemos concluir que la regla de Johnson a comparación del algoritmo genético nos presenta un dato atípico de 2326, esto se da, dado que en la programación los valores son generados aleatoriamente ya que los procesos estocásticos funcionan dando un resultado al azar, donde solo se conoce la distribución, pero no los valores que se van a generar.

Figura 29.

Caja y bigotes regla de Johnson y algoritmo genético n:100



Para un tamaño de n=300 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=300
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 50
num_padres_torneo = 1
probabilidad_mutacion = 0.1
```

A continuación, a través del histograma de la figura 30 y la figura 31 se puede analizar que el makespan para la regla de Johnson presentan un mínimo de 6427 unidades de tiempo y un

máximo de 6783 unidades de tiempo con una media de 6616.25, para el caso del algoritmo genético presentan un mínimo de 6434 unidades de tiempo y un máximo de 6787 unidades de tiempo con una media de 6621.5. De lo anterior, se evidencia que la regla de Johnson presenta mejor el makespan en comparación al algoritmo genético. Los datos originados presentan una variabilidad esta diferencia se da dado que los parámetros del algoritmo genético nos presentan el makespan más alto en comparación a la regla de Johnson.

Figura 30.

Histograma regla de Johnson n=300

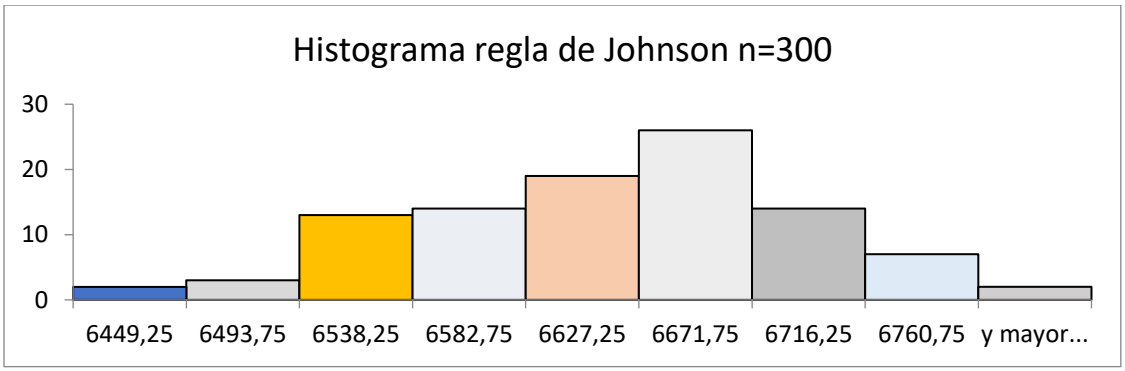
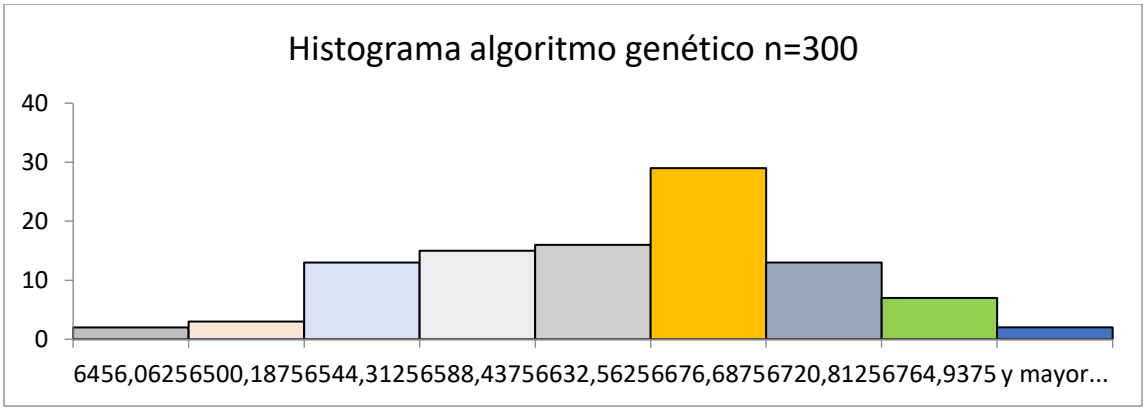


Figura 31.

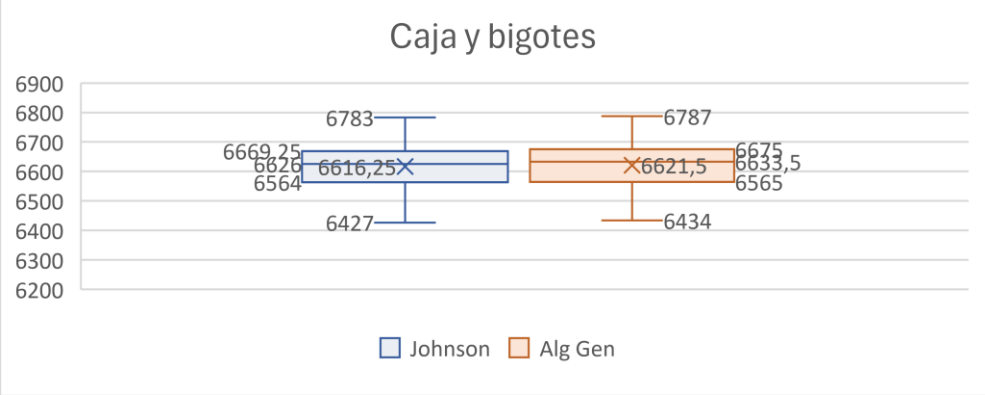
Histograma algoritmo genético n=300



En la figura 32 se puede observar que en la regla de Johnson los valores se encuentran entre el cuartil inferior 6564 y el cuartil superior 6669.25 donde la mediana 6616.25 se encuentra ubicada en este intervalo, en el caso del algoritmo genético los datos se encuentran ubicados entre el cuartil inferior 6565 y el cuartil superior 6675 con una mediana de 6621.5. De lo anterior, se puede concluir que la dispersión es similar entre ambos algoritmos.

Figura 32.

Caja y bigotes regla de Johnson y algoritmo genético n:300



Por último, para un tamaño de n=5000 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=5000
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 50
num_padres_torneo = 1
probabilidad_mutacion = 0.1
```

A través del histograma de la figura 33, se puede observar que la regla de Johnson presenta el makespan con un mínimo de 109602 unidades de tiempo y un máximo de 110389 con una media de 109907.9, en la figura 34 se observa que para el algoritmo genético se tienen el makespan con

un mínimo de 109605 unidades de tiempo y un máximo de 110389 unidades de tiempo con una media de 109909.3. De lo anterior, se encuentra que la regla de Johnson presenta el makespan mínimo diferente al del algoritmo genético. Es importante aclarar que los valores aleatorios no son generados por una distribución normal sino por una distribución rectangular, por lo que cada valor tiene igual probabilidad de salir, si el número de datos que se hubiese tomado para calcular el makespan fuera lo suficientemente mayor, se hubiese obtenido un histograma con una distribución normal, para este caso el número de datos que se tomaron fue 10, por lo que no se observa que distribución toma el histograma.

Figura 33.

Histograma regla de Johnson n=5000

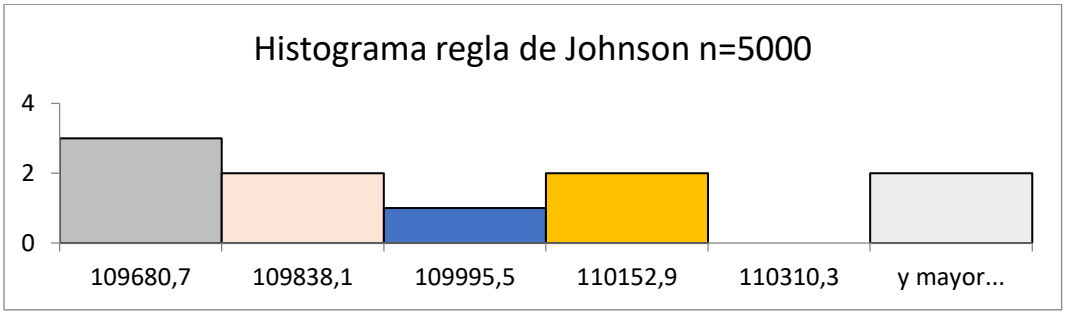
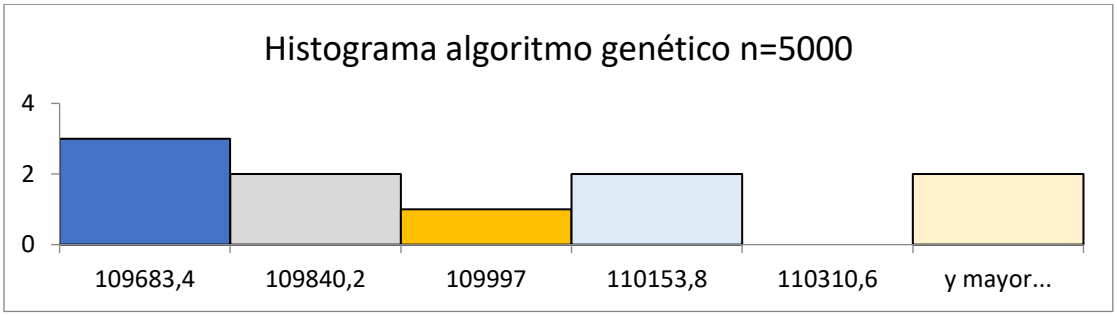


Figura 34.

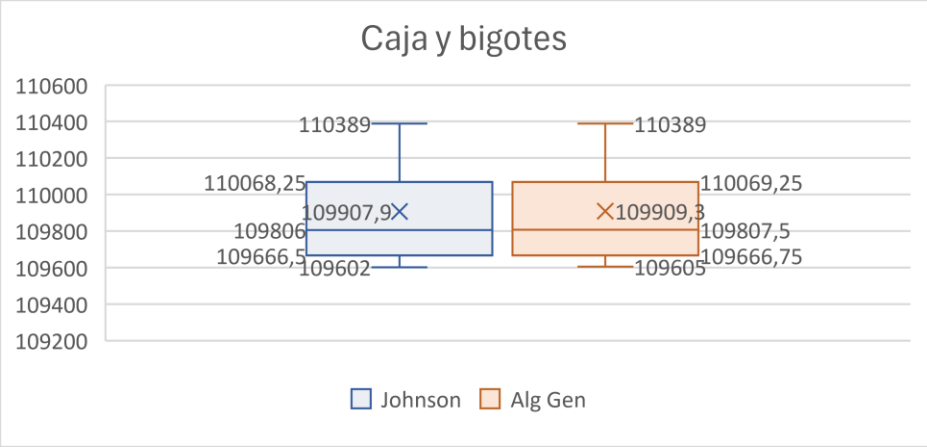
Histograma algoritmo genético n=5000



Se puede observar en la figura 35 los extremos inferiores y superiores de la regla de Johnson y el algoritmo genético, donde se puede evidenciar que para la regla de Johnson el cuartil inferior es 109602 y el cuartil superior es 110068.25 donde en este rango se encuentra ubicada la mediana 109806. Para el algoritmo genético el cuartil inferior es 109666.75 y el cuartil superior es 110069.25 en este intervalo se encuentra almacenada la mediana 109807.5 por lo que se puede concluir que tanto para la regla de Johnson y el algoritmo genético tienen una dispersión similar.

Figura 35.

Caja y bigotes regla de Johnson y algoritmo genético n:5000



- El segundo análisis que se realizó sobre la programación a la regla de Johnson y el algoritmo genético, para un tamaño de n=30, 100, 300 y 5000.

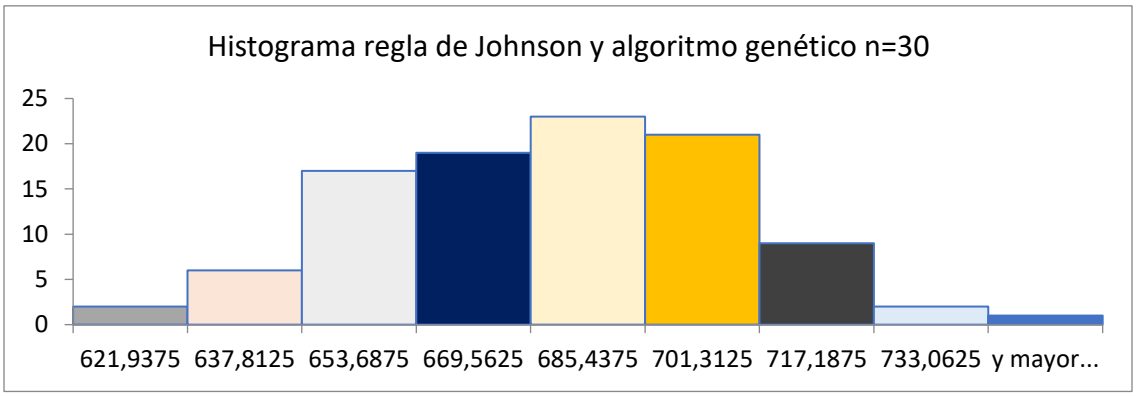
Para un tamaño de n=30 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=30
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 100
num_padres_torneo = 2
probabilidad_mutacion = 0.5
```

Por medio de un análisis de datos realizado en Excel, nos arrojó que para la regla de Johnson como el algoritmo genético no se presenta una variabilidad en datos los cuales fueron originados aleatoriamente en la programación. A continuación, a través del histograma de la figura 36 se puede observar que el makespan para el algoritmo genético y la regla de Johnson fueron los mismos, obteniendo el makespan mínimo de 614 unidades de tiempo y un máximo de 741 unidades de tiempo con una media de 674.12. Se puede observar que la media se encuentra ubicada en la clase con mayor frecuencia, el tamaño de muestras que se tomaron para realizar el histograma fue 100, como anteriormente se comentó, teniendo un tamaño de muestras grandes, se aproxima a una curva normal.

Figura 36.

Histograma regla de Johnson y algoritmo genético n=30



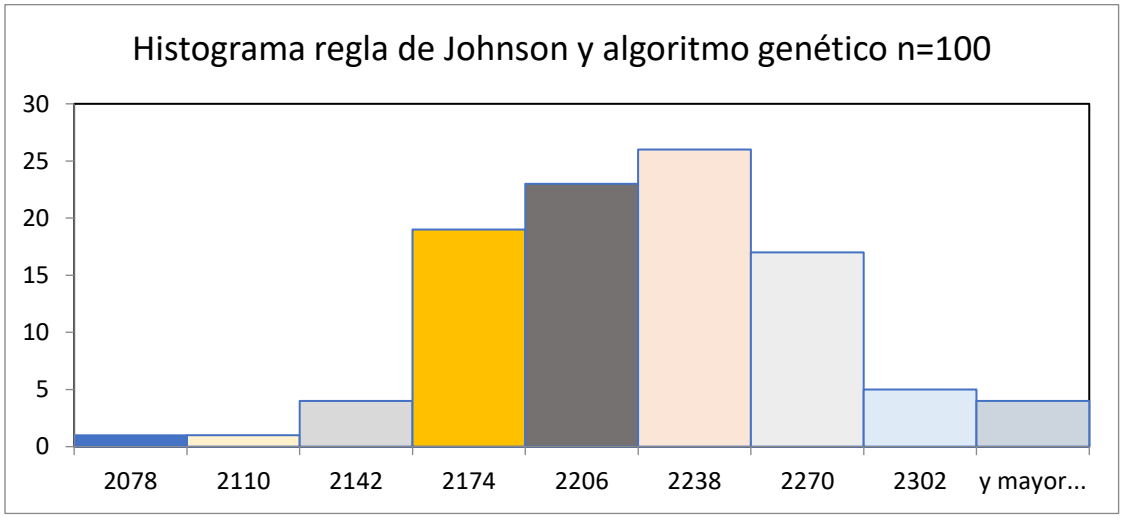
Para un tamaño de n=100 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=100
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 100
num_padres_torneo = 2
probabilidad_mutacion = 0.5
```

Se evidencia que tanto la regla de Johnson como el algoritmo genético muestran una uniformidad notable en conjuntos de datos generados aleatoriamente durante la programación. A través del histograma de la figura 37 se observa lo anteriormente expuesto ya que el makespan para el algoritmo genético y la regla de Johnson fueron los mismos, obteniendo el makespan mínimo de 2062 unidades de tiempo y un máximo de 2318 unidades de tiempo con una media de 2208.37, el número de la muestra nos permite observar la forma que toma el histograma aproximándose a una curva normal.

Figura 37.

Histograma regla de Johnson y algoritmo genético n=100



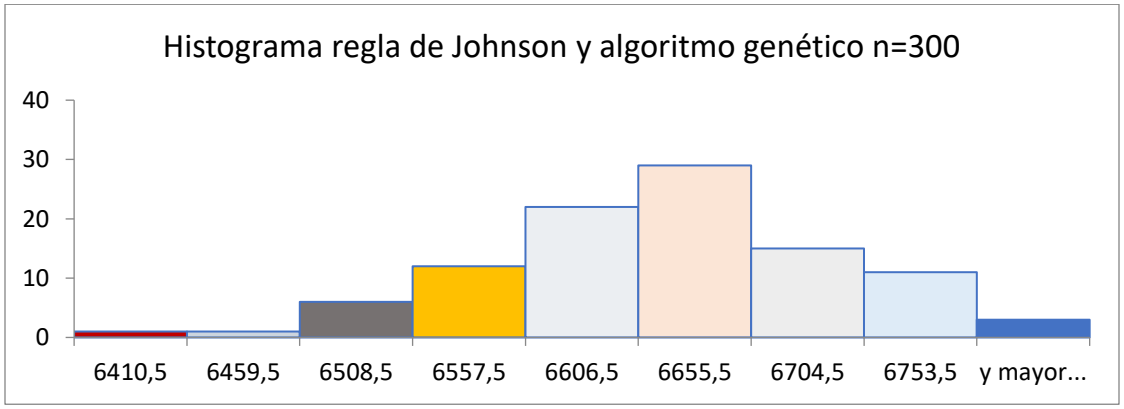
Para un tamaño de n=300 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=300
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 100
num_padres_torneo = 2
probabilidad_mutacion = 0.5
```

A través de la toma realizada para la obtención del makespan se pudo observar que no se presenta una variabilidad en datos, por ello, en la figura 38 los tiempos de procesamiento para el algoritmo genético y la regla de Johnson fueron los mismos, obteniendo el makespan mínimo de 6386 unidades de tiempo y un máximo de 6778 unidades de tiempo con una media de 6615.04, obteniendo así el mismo histograma para la regla de Johnson y el algoritmo genético. La forma que tomaron los datos se aproxima a una curva normal, ya que el tamaño muestral fue de 100, esto se da por el teorema de límite central, dado que mayor número de corridas la curva tiende a ser una curva normal.

Figura 38.

Histograma regla de Johnson y algoritmo genético n=300



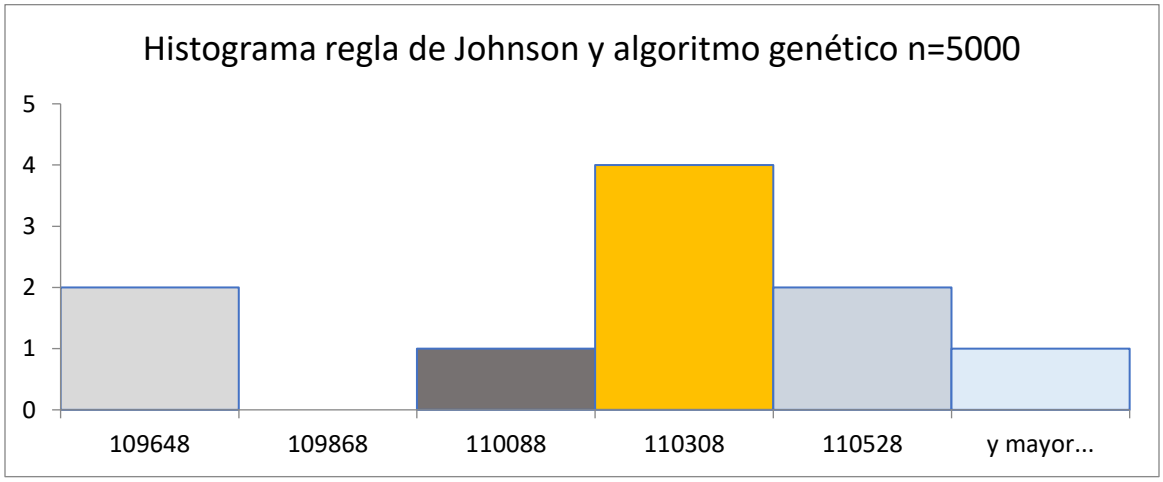
Para un tamaño de n=5000 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=5000
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 100
num_padres_torneo = 2
probabilidad_mutacion = 0.5
```

Por medio del histograma de la figura 39 se observa que el makespan para el algoritmo genético y la regla de Johnson fueron los mismos, obteniendo el makespan mínimo de 109538 unidades de tiempo y un máximo de 110638 unidades de tiempo con una media de 110111.3, obteniendo así el mismo histograma para la regla de Johnson y el algoritmo genético. Cabe resaltar que la media se encuentra ubicada en la clase con mayor frecuencia, la muestra tomada para el makespan fue 10 por lo que no se observa que distribución toma el histograma, si la muestra fuera mayor hubiese tomado aproximadamente igual a una curva normal.

Figura 39.

Histograma regla de Johnson y algoritmo genético n=5000



- El tercer análisis que se realizó sobre la programación a la regla de Johnson y el algoritmo genético, para un tamaño de n=30, 100, 300 y 5000.

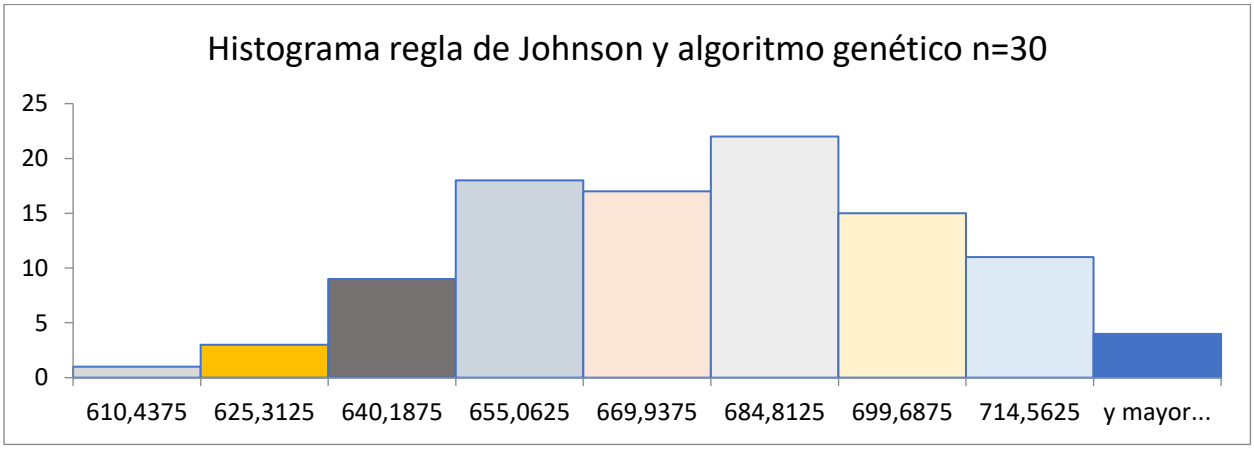
Para un tamaño de n=30 en el algoritmo genético y la regla de Johnson, a continuación, se observa que dichos parámetros se cambian ya que se quiere ver que rendimiento tienen cada uno de ellos:

```
n=30
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 200
num_padres_torneo = 4
probabilidad_mutacion = 0.9
```

Con un análisis de datos realizado en Excel, se obtuvo que para la regla de Johnson como el algoritmo genético no presentan una diferencia en los datos. La figura 40 muestra el makespan para el algoritmo genético y la regla de Johnson fueron los mismos, obteniendo el makespan mínimo de 603 unidades de tiempo y un máximo de 722 unidades de tiempo con una media 669.82 siendo así el mismo histograma para la regla de Johnson y el algoritmo genético. La clase con mayor frecuencia se encuentra cerca a la media, el tamaño muestral fue grande lo que nos permite ver la distribución del histograma aproximándose una curva normal.

Figura 40.

Histograma regla de Johnson y algoritmo genético n=30



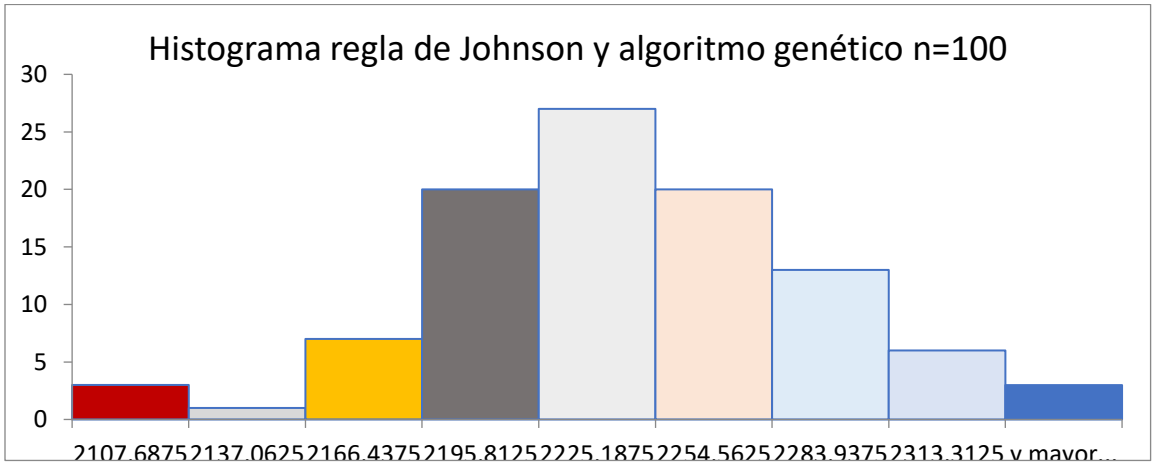
Para un tamaño de n=100 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=100
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 200
num_padres_torneo = 4
probabilidad_mutacion = 0.9
```

En la figura 41 se puede observar que el makespan para el algoritmo genético y la regla de Johnson fueron los mismos, obteniendo el makespan mínimo de 2093 unidades de tiempo y un máximo de 2328 unidades de tiempo con una media de 2216.86, siendo así el mismo histograma para la regla de Johnson y el algoritmo genético. De la figura se puede abordar que en la clase con mayor frecuencia se encuentra la media, también podemos deducir que la distribución que toma el histograma se aproxima a una curva normal, ya que el tamaño muestral es grande y nos permite observar mejor el comportamiento de los datos.

Figura 41.

Histograma regla de Johnson y algoritmo genético n=100



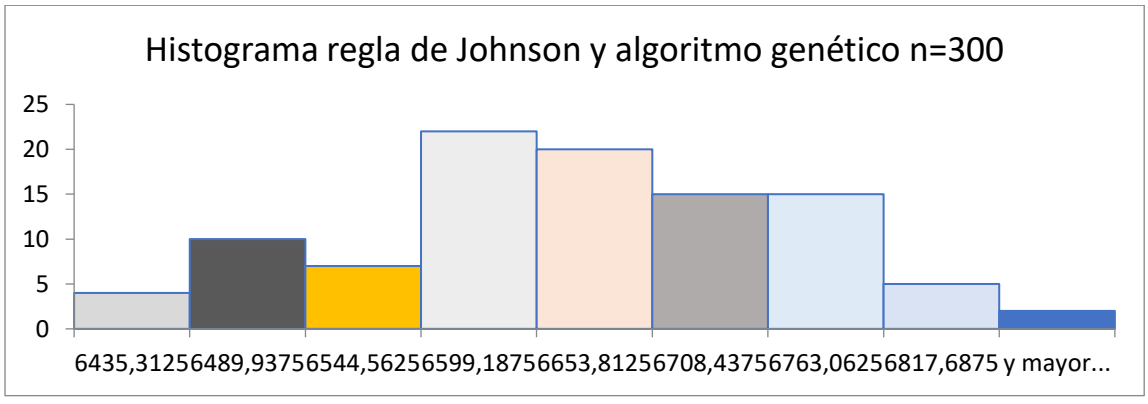
Para un tamaño de n=300 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=300
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 200
num_padres_torneo = 4
probabilidad_mutacion = 0.9
```

La regla de Johnson como el algoritmo genético no presentan una diferencia en los datos, lo anterior se puede observar en el histograma de la figura 42 ya que el makespan para el algoritmo genético y la regla de Johnson fueron los mismos, obteniendo el makespan mínimo de 6408 unidades de tiempo y un máximo de 6845 unidades de tiempo con una media de 6622.01, obteniendo así el mismo histograma para la regla de Johnson y el algoritmo genético. Por medio de la figura 42 observamos que la media se encuentra ubicada en la clase con mayor frecuencia, el tamaño muestral es grande lo que conlleva a decir que la distribución tomada por los datos obtenidos es aproximadamente igual a una curva normal.

Figura 42.

Histograma regla de Johnson y algoritmo genético n=300



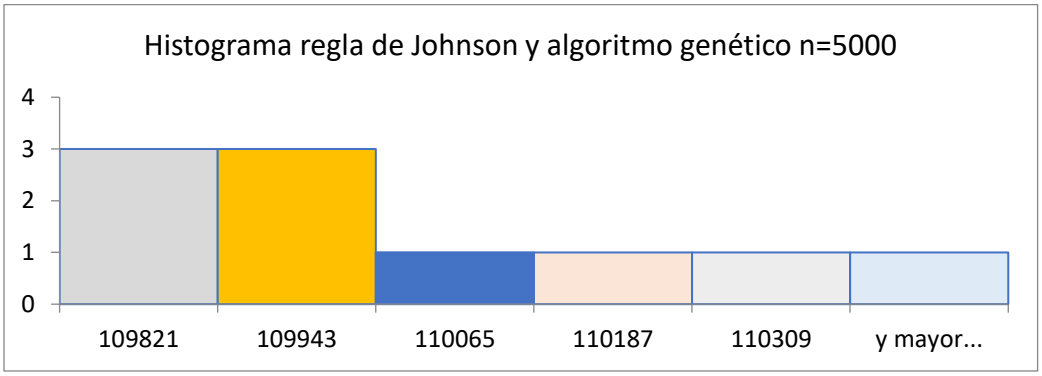
Para un tamaño de n=5000 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=5000
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 200
num_padres_torneo = 4
probabilidad_mutacion = 0.9
```

En el histograma de la figura 43 se puede observar que el makespan para el algoritmo genético y la regla de Johnson fueron los mismos, obteniendo el makespan mínimo de 109760 unidades de tiempo y un máximo de 110370 unidades de tiempo con una media de 109972.7, obteniendo así el mismo histograma para la regla de Johnson y el algoritmo genético. Los valores aleatorios son generados por la distribución rectangular los cuales tienen la misma probabilidad de salir, el tamaño de la muestra fue pequeño, ya que no nos permite definir que distribución toma el histograma.

Figura 43.

Histograma regla de Johnson y algoritmo genético n=5000



Para concluir, se observó que en los tres análisis el que tuvo mejor desarrollo fue la regla de Johnson para el caso de la primera instancia del algoritmo genético en el cual se obtuvo mejor el makespan la regla de Johnson, para la segunda y tercera instancia se obtuvieron diferentes el

makespan siendo la regla de Johnson la que obtuvo el menor tiempo. En los histogramas se observó que los tamaños muestrales grandes nos arrojan una distribución aproximadamente igual a la curva normal.

Por consiguiente, se tomó el aleatorio de la máquina 1 entre 15 y 30 unidades de tiempos, para la máquina 2 se tomó el aleatorio entre 10 y 20 unidades de tiempos, de manera contraria al caso anterior.

- El primer análisis que se realizó sobre la programación a la regla de Johnson y el algoritmo genético, para un tamaño de n=30, 100, 300 y 5000.

Para un tamaño de n=30 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=30
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 50
num_padres_torneo = 1
probabilidad_mutacion = 0.1
```

De los datos arrojados por la programación se observa que la regla de Johnson como el algoritmo genético presentan variabilidad en los datos. El histograma de la figura 44 y la figura 45 se observa que el makespan para la regla de Johnson tienen un mínimo de 610 unidades de tiempo y un máximo de 726 unidades de tiempo con una media de 671.18 y para el algoritmo genético un mínimo de 611 unidades de tiempo y un máximo de 727 unidades de tiempo con una media de 673.13. De la figura 44 y 45 se observa que las medias se encuentran ubicadas en la clase con mayor frecuencia, también se puede concluir que la distribución que toman los histogramas es

aproximadamente igual a una curva normal ya que el tamaño muestral es grande y nos permite dar dicha conclusión a través del teorema de limite central.

Figura 44.

Histograma regla de Johnson n=30

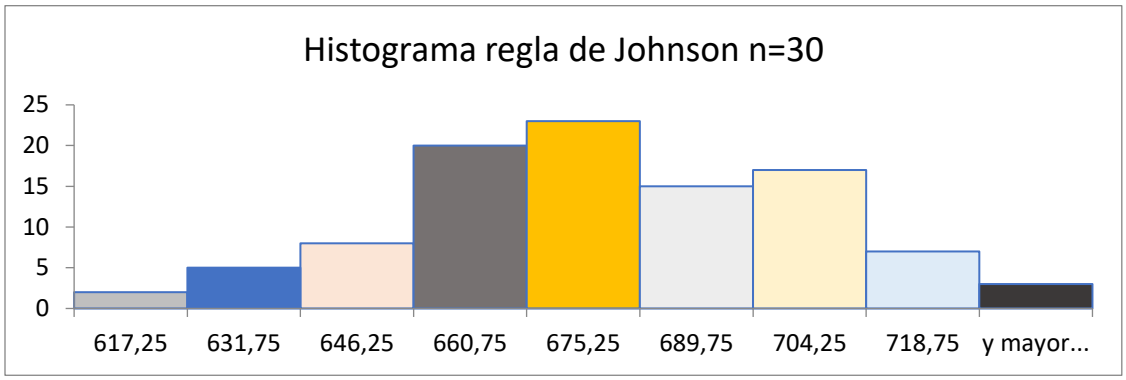
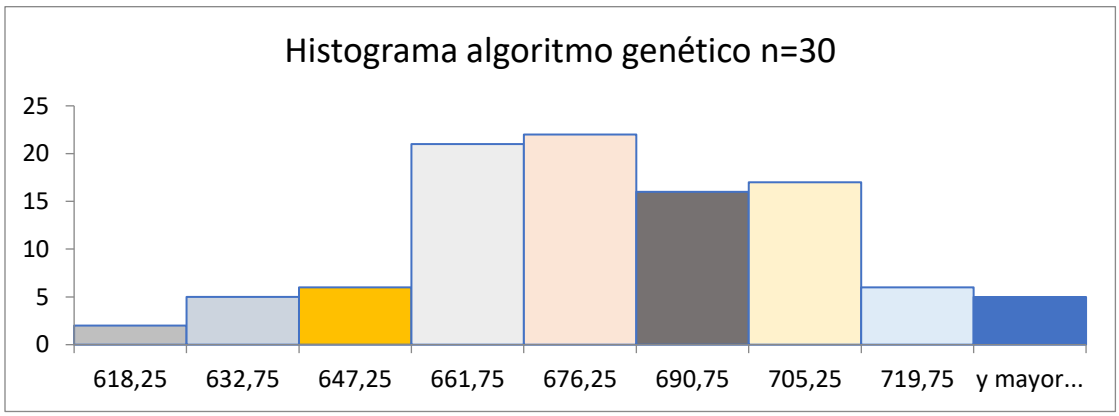


Figura 45.

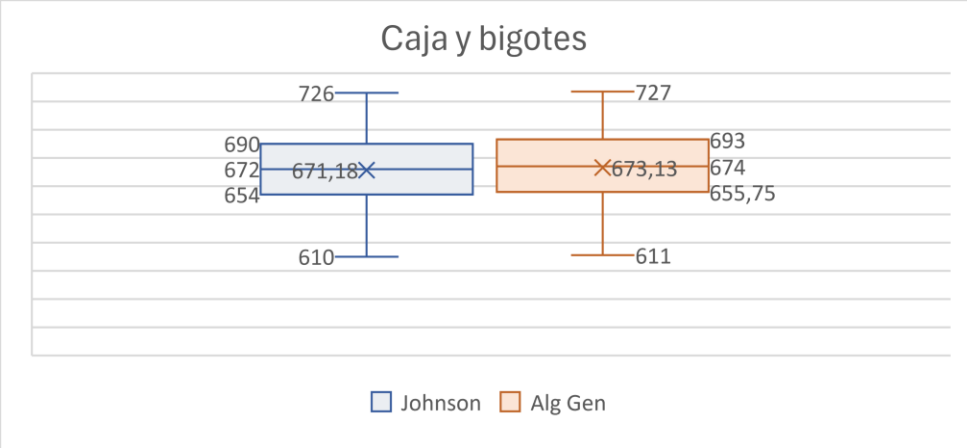
Histograma algoritmo genético n=30



De la figura 46 se observa que la mediana se encuentra ubicada cerca a la mitad de la caja, lo que indica que los valores son más o menos simétricos tanto para la regla de Johnson como para el algoritmo genético, los datos se encuentran almacenados entre el 25% y el 75% de la caja, es decir el cuantil inferior 654 y el cuartil superior 690.

Figura 46.

Caja y bigotes para los algoritmos n:30



Para un tamaño de n=100 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=100
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 50
num_padres_torneo = 1
probabilidad_mutacion = 0.1
```

En la figura 47 y la figura 48 el makespan para la regla de Johnson tienen un mínimo de 2116 unidades de tiempo y un máximo de 2358 unidades de tiempo con una media de 2221.73 y para el algoritmo genético un mínimo de 2116 unidades de tiempo y un máximo de 2362 unidades de tiempo con una media de 2222.83. En las figuras mencionadas anteriormente se detalla que las medias se encuentran ubicadas en la clase con mayor frecuencia y la distribución que toma los histogramas se aproxima a una curva normal, dado que el tamaño muestral es grande.

Figura 47.

Histograma regla de Johnson

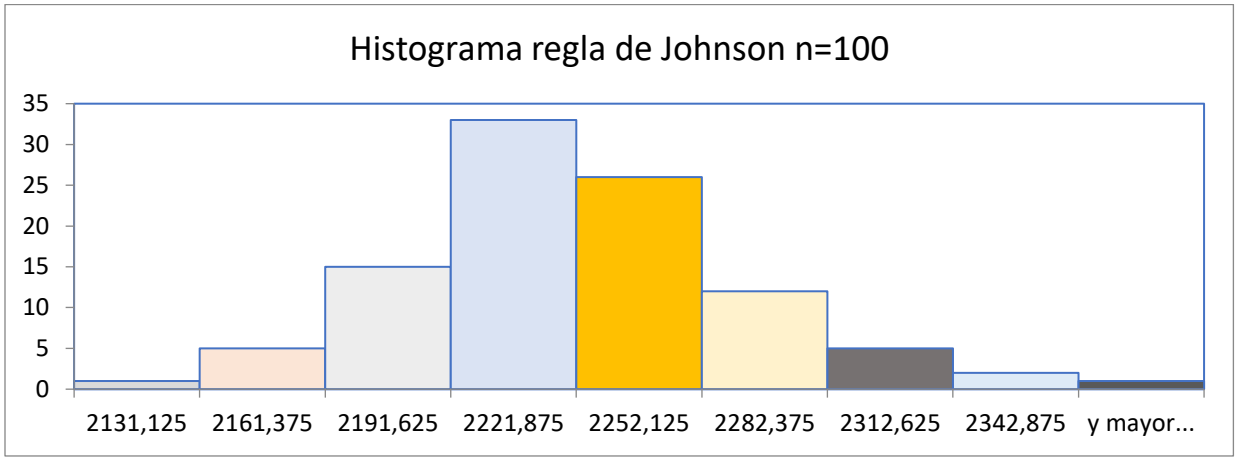
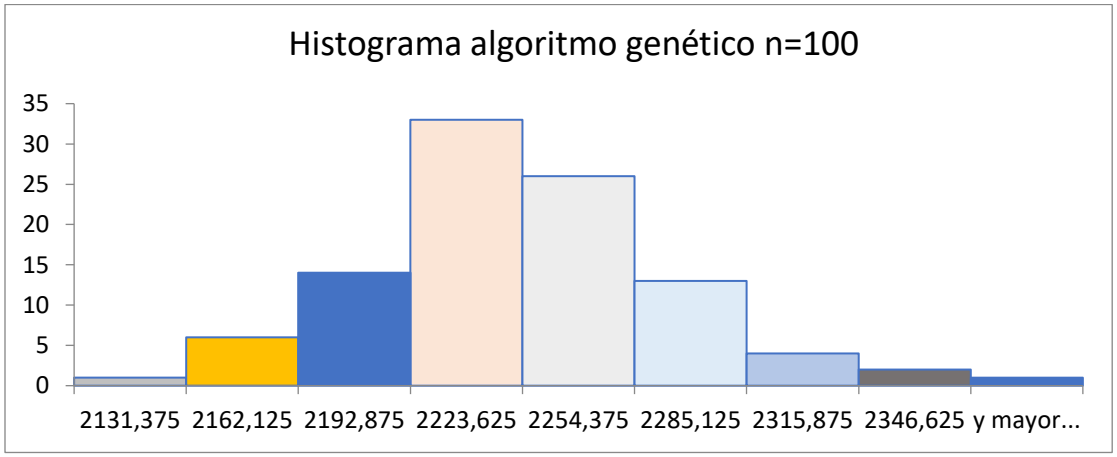


Figura 48.

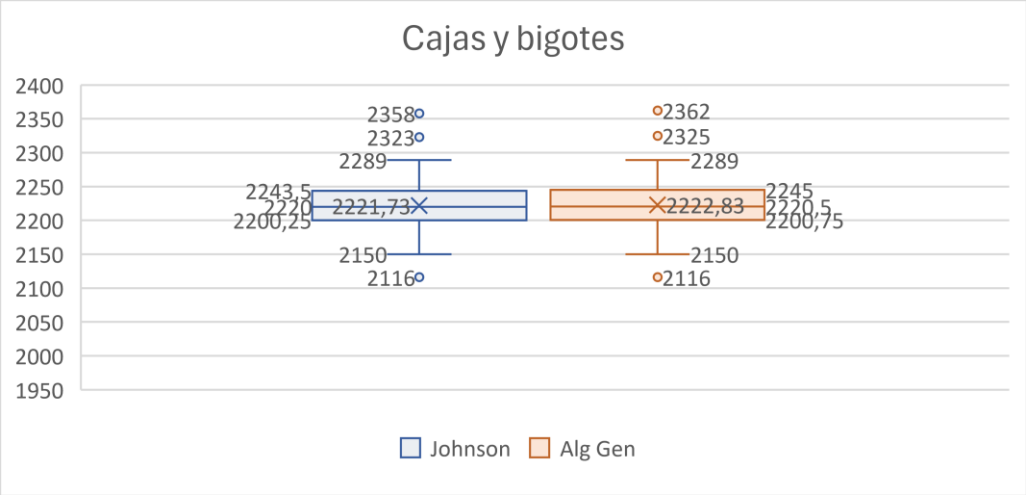
Histograma algoritmo genético n=100



En la figura 49 se observa que entre el 25% y el 75% de los valores de los datos se encuentran en este rango, es decir, entre el cuartil inferior y superior (2200.25;2243.5). Se encuentran unos valores atípicos los cuales están por encima del extremo superior o por debajo del extremo inferior.

Figura 49.

Caja y bigotes para los algoritmos n:100



Para un tamaño de n=300 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=300
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 50
num_padres_torneo = 1
probabilidad_mutacion = 0.1
```

De lo abordado por la programación se puede concluir que la regla de Johnson como el algoritmo genético presentan variabilidad en los datos. En los histogramas de la figura 50 y la figura 51 se observa que el makespan para la regla de Johnson tienen un mínimo de 6411 unidades de tiempo y un máximo de 6776 unidades de tiempo con una media de 6603.1 y para el algoritmo genético un mínimo de 6411 unidades de tiempo y un máximo de 6782 unidades de tiempo con una media de 6604.78.

Figura 50.

Histograma regla de Johnson n=300

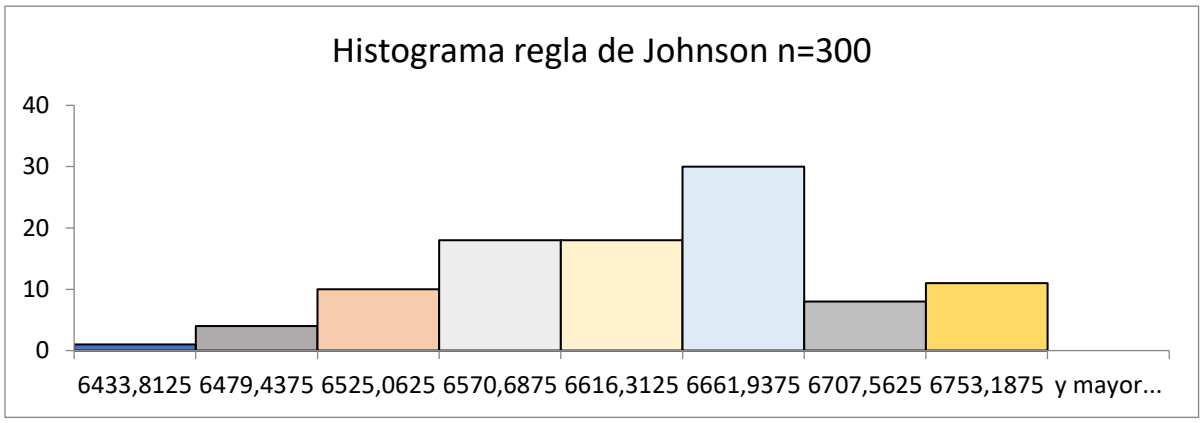
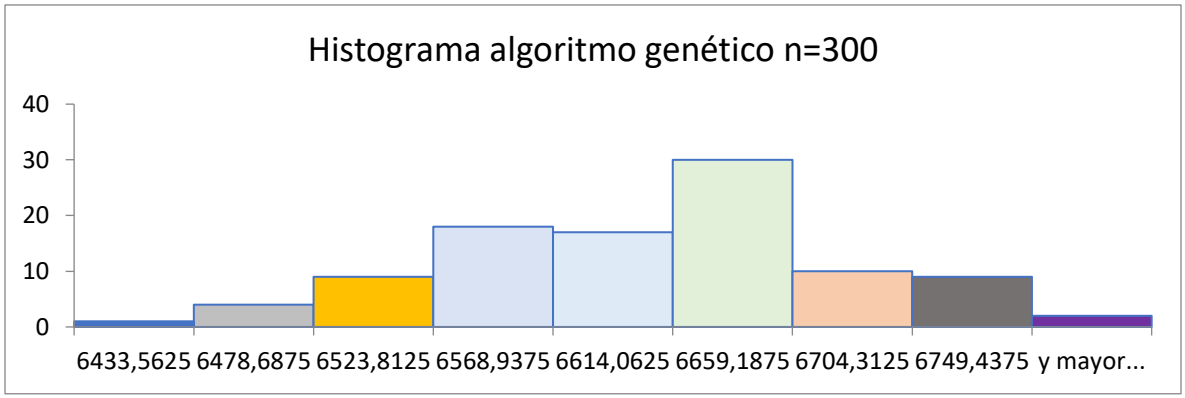


Figura 51.

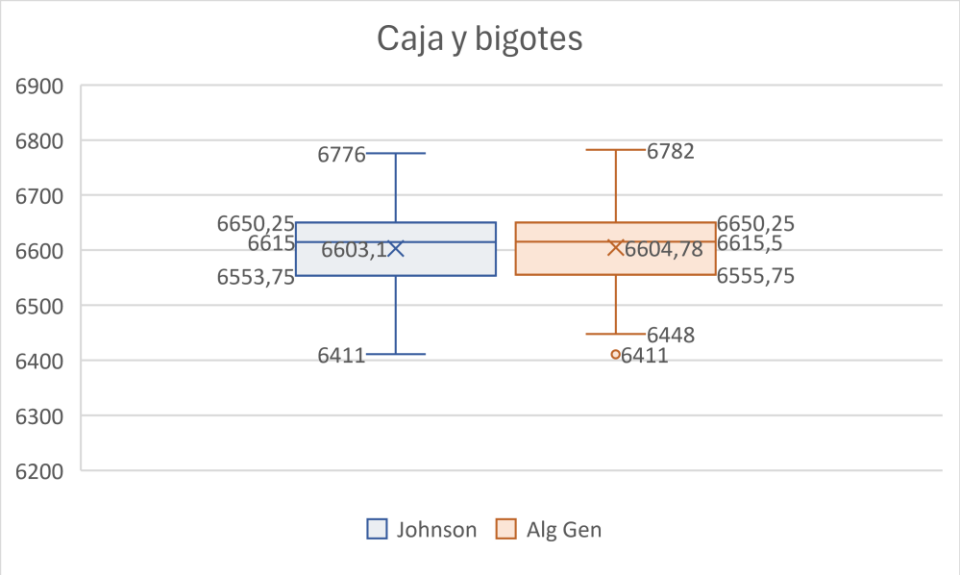
Histograma algoritmo genético n=300



Se puede observar en la figura 52 que para la regla de Johnson nuestro extremo inferior corresponde a 6411 y el extremo superior a 6776, entre el 25% y el 75% se encuentran los valores de los datos. Para el algoritmo genético se puede observar que debajo del extremo inferior correspondiente a 6448, se encuentra un valor atípico, ya que los valores en la programación son generados aleatoriamente. Los valores de los datos se encuentran almacenados entre el 25% y el 75% es decir al cuantil inferior y superior.

Figura 52.

Caja y bigotes para los algoritmos n:300



Para un tamaño de n=5000 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=5000
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 50
num_padres_torneo = 1
probabilidad_mutacion = 0.1
```

El histograma de la figura 53 y la figura 54 observamos que el makespan para la regla de Johnson tienen un mínimo de 109545 unidades de tiempo y un máximo de 110356 unidades de tiempo con una media de 109974.7 y para el algoritmo genético un mínimo de 109546 unidades de tiempo y un máximo de 110356 unidades de tiempo con una media de 109975.1. Es importante aclarar que por el tamaño muestral no se puede afirmar que distribución toma el histograma.

Figura 53.

Histograma regla de Johnson n=5000

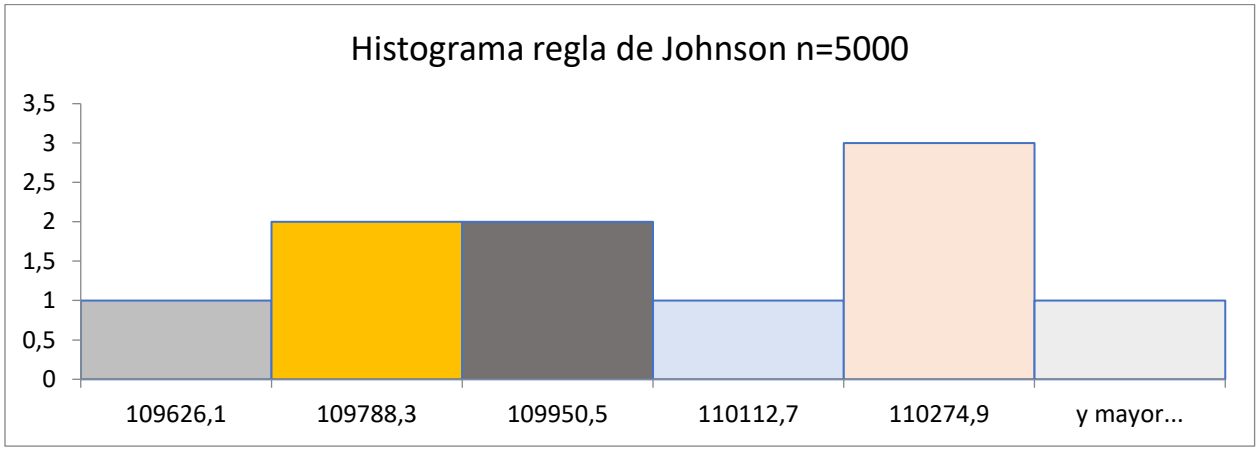
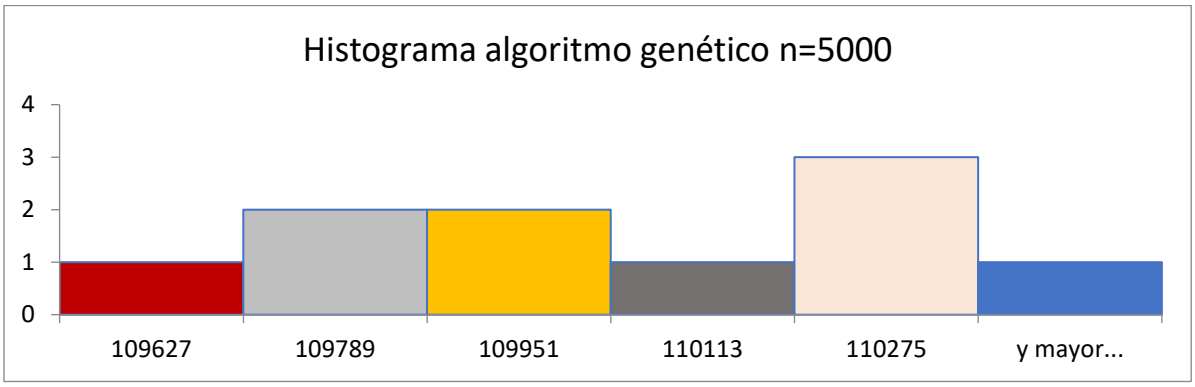


Figura 54.

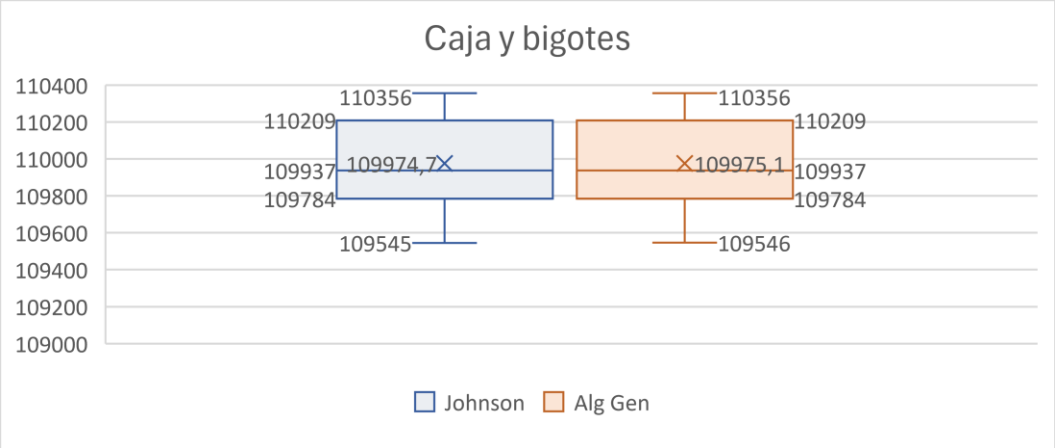
Histograma algoritmo genético n=5000



Con el fin de tener un mejor entendimiento de los histogramas, en la figura 55 se puede observar el diagrama caja y bigotes, donde para el algoritmo genético y la regla de Johnson la mediana se encuentra cerca a la mitad de la caja, lo que quiere decir que los valores son más o menos simétrico, los valores de los datos para juntos algoritmos se encuentran entre el cuartil inferior y el cuartil superior.

Figura 55.

Caja y bigotes para los algoritmos n:5000



- El segundo análisis que se realizó sobre la programación a la regla de Johnson y el algoritmo genético, para un tamaño de n=30, 100, 300 y 5000.

Para un tamaño de n=30 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

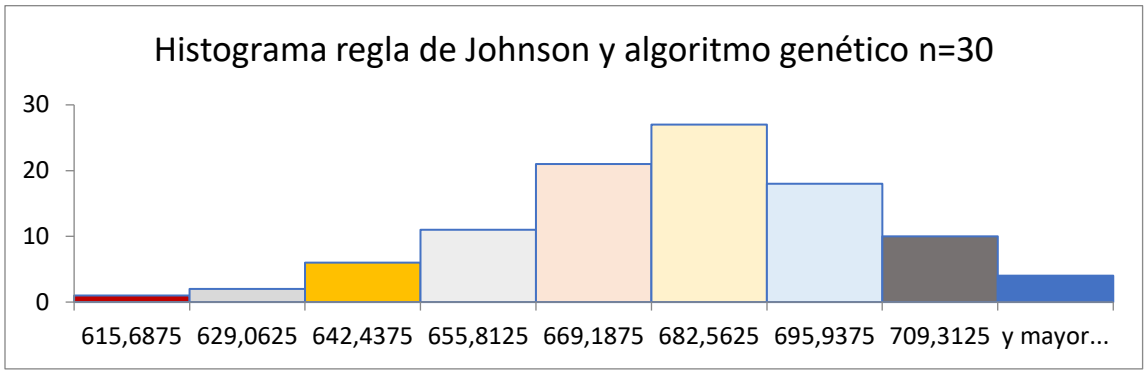
```
n=30
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 100
num_padres_torneo = 2
probabilidad_mutacion = 0.5
```

De la toma de datos concluimos que para la regla de Johnson y el algoritmo genético no se tiene una diferencia en el makespan es por ello por lo que en el histograma de la figura 56 se observan que los tiempos son iguales, teniendo un mínimo de 609 unidades de tiempo y un máximo de 716 unidades de tiempo con una media de 673.33. De la figura se concluye que la media se

encuentra ubicada en la clase con mayor frecuencia y presenta una distribución aproximadamente igual a una curva normal.

Figura 56.

Histograma regla de Johnson y algoritmo genético n=30



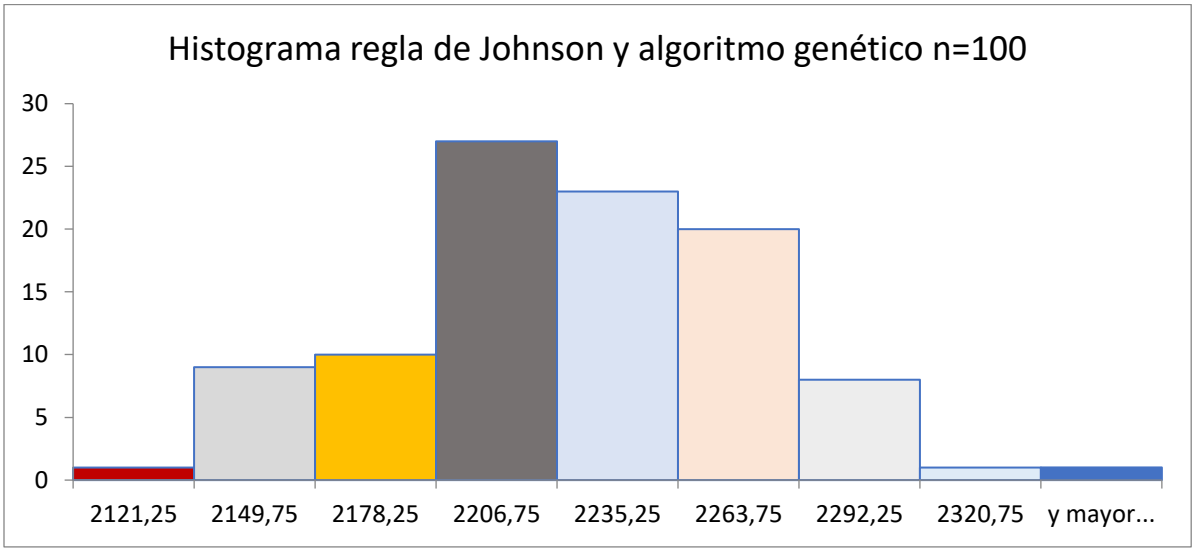
Para un tamaño de n=100 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=100
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 100
num_padres_torneo = 2
probabilidad_mutacion = 0.5
```

La figura 57 nos arroja que el makespan para la regla de Johnson y el algoritmo genético, siendo estos los mismos, teniendo un mínimo de 2107 unidades de tiempo y un máximo de 2335 unidades de tiempo con una media de 2211.66. La media se encuentra en la clase con mayor frecuencia, tomando así una curva normal ya que el tamaño muestral es grande.

Figura 57.

Histograma regla de Johnson y algoritmo genético n=100



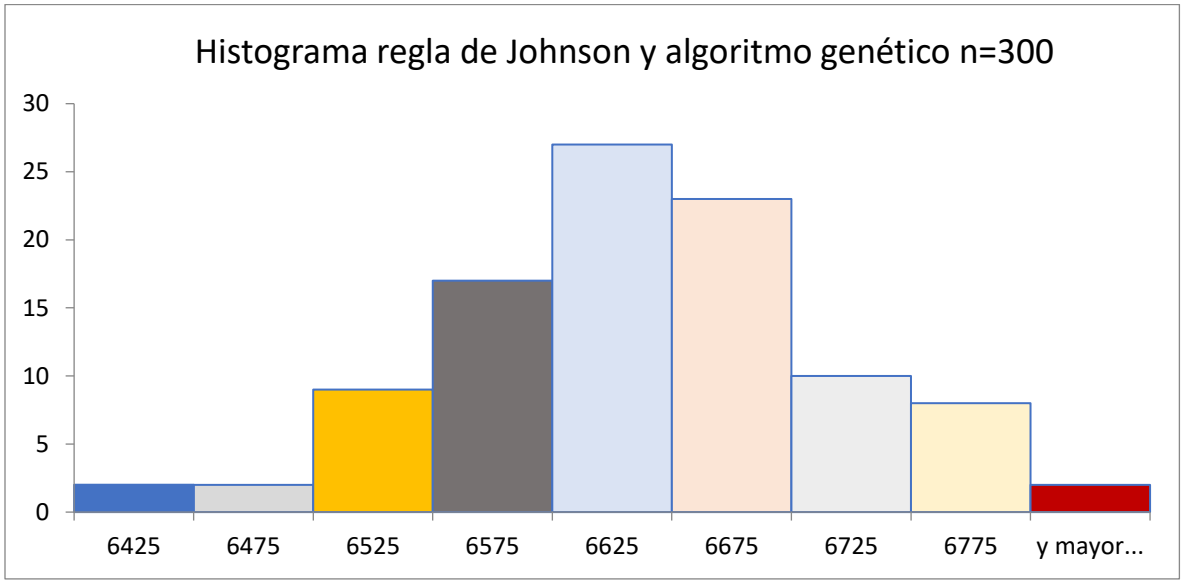
Para un tamaño de n=300 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=300
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 100
num_padres_torneo = 2
probabilidad_mutacion = 0.5
```

A través del histograma de la figura 58 se puede observar que el makespan para la regla de Johnson y el algoritmo genético son los mismos, teniendo un mínimo de 6400 unidades de tiempo y un máximo de 6800 unidades de tiempo con una media de 6614.75 la cual se encuentra ubicada en la barra número 5, siendo esta la clase con la mayor frecuencia tomando una distribución aproximadamente igual a una curva normal.

Figura 58.

Histograma regla de Johnson y algoritmo genético n=300



Para un tamaño de n=5000 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

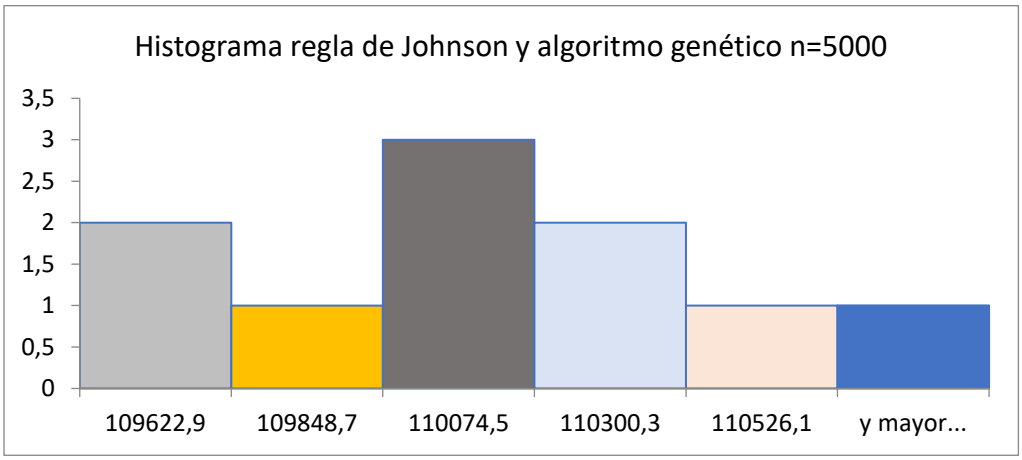
```
n=5000
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 100
num_padres_torneo = 2
probabilidad_mutacion = 0.5
```

Los datos arrojados de la programación fueron usados para realizar un análisis de datos los cuales se encuentran almacenados en el histograma de la figura 59 donde se observa que el makespan para la regla de Johnson y el algoritmo genético son los mismos, teniendo un mínimo de 109510 unidades de tiempo y un máximo de 110639 unidades de tiempo con una media de 110022.5. Es importante aclarar que cuando se tiene un tamaño muestral pequeño no se puede

concluir que distribución toma el histograma, ya que carece de datos para poder afirmar o suponer la forma que toman los datos.

Figura 59.

Histograma regla de Johnson y algoritmo genético n=5000



- El tercer análisis que se realizó sobre la programación a la regla de Johnson y el algoritmo genético, para un tamaño de n=30, 100, 300 y 5000.

Para un tamaño de n=30 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

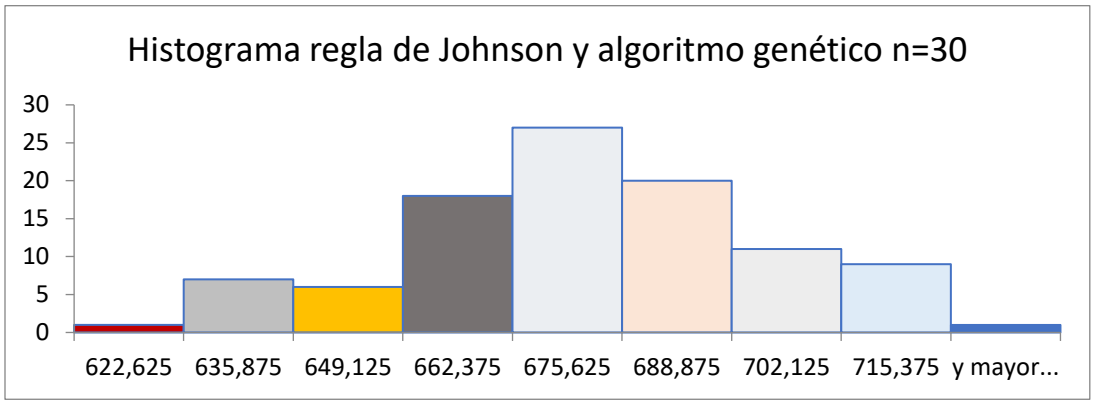
```
n=30
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 200
num_padres_torneo = 4
probabilidad_mutacion = 0.9
```

Los datos obtenidos en el programación se encuentran ubicados en el histograma de la figura 60 donde se observa que el makespan para la regla de Johnson y el algoritmo genético son los mismos, los cuales tienen un mínimo de 616 unidades de tiempo y un máximo de 722 unidades

de tiempo con una media de 671.32 la cual se encuentra ubicada en la barra número 5 dándole una forma a la distribución aproximada a una curva normal, dado que el tamaño muestral es grande.

Figura 60.

Histograma regla de Johnson y algoritmo genético n=30



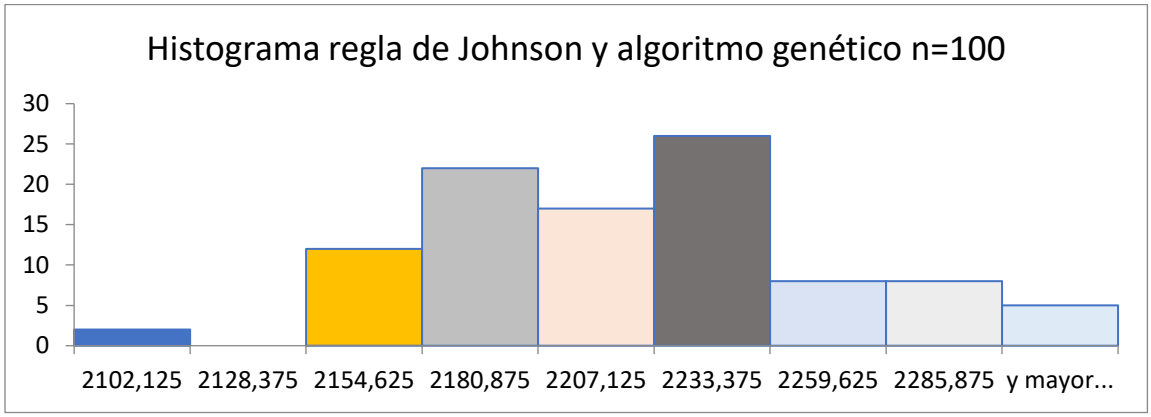
Para un tamaño de n=100 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=100
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 200
num_padres_torneo = 4
probabilidad_mutacion = 0.9
```

De la obtención de los datos arrojados en la programación podemos decir que el makespan para la regla de Johnson y el algoritmo genético son los mismos, dichos datos se encuentran almacenados en el histograma de la figura 61 donde tienen un mínimo de 2089 unidades de tiempo y un máximo de 2299 unidades de tiempo con una media 2202.46.

Figura 61.

Histograma regla de Johnson y algoritmo genético n=100



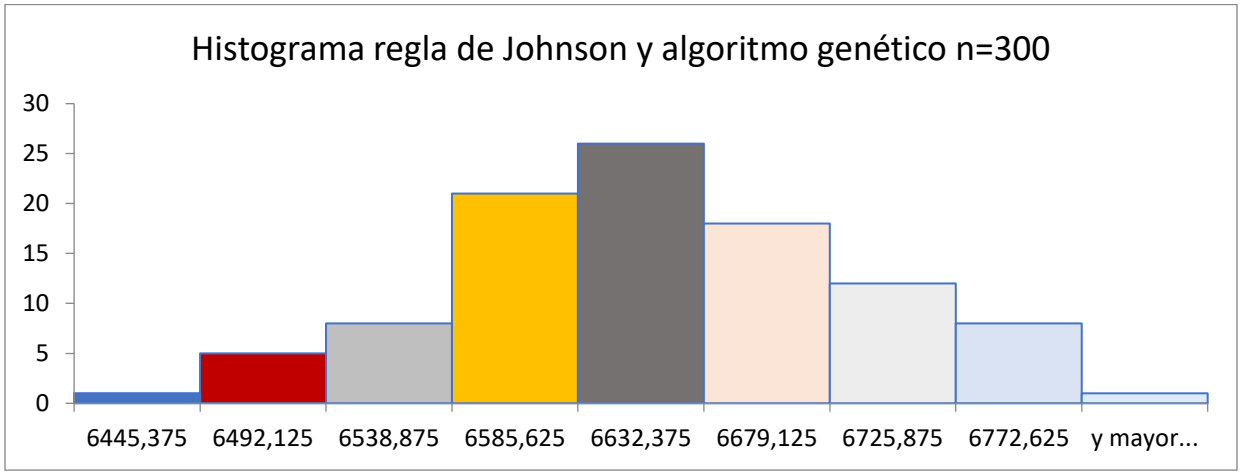
Para un tamaño de n=300 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=300
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 200
num_padres_torneo = 4
probabilidad_mutacion = 0.9
```

Se obtuvo de la programación que el makespan son los mismos para ambas reglas, en el histograma de la figura 62 se observa que tienen un mínimo de 6422 unidades de tiempo y un máximo de 6796 unidades de tiempo con una media de 6615.75 ubicada en la clase con mayor frecuencia, con el tamaño muestral la distribución toma aproximadamente igual a una curva normal.

Figura 62.

Histograma regla de Johnson y algoritmo genético n=300



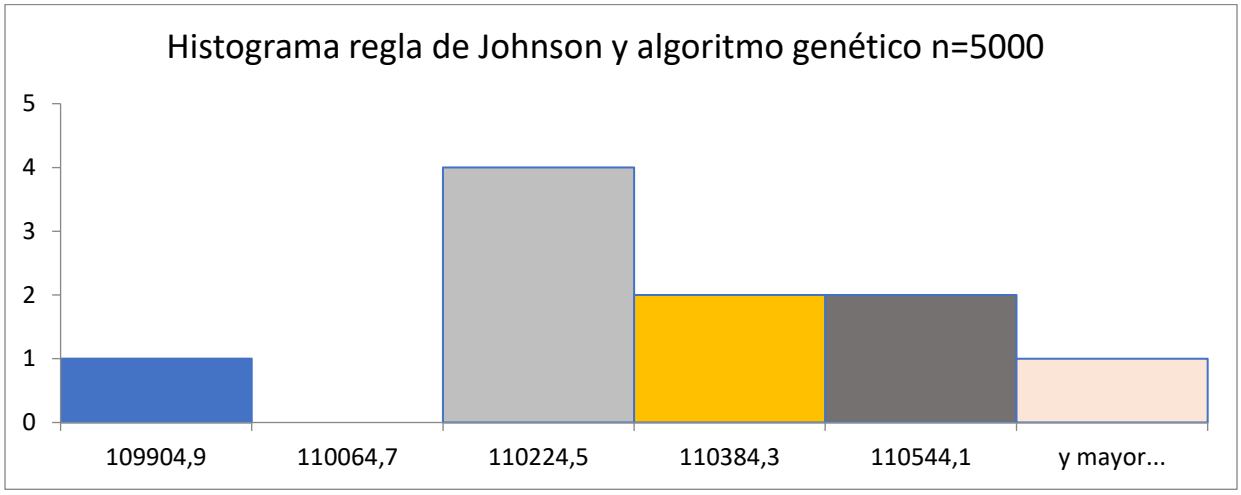
Para un tamaño de n=5000 en el algoritmo genético y la regla de Johnson, con los parámetros observados a continuación se obtiene que:

```
n=5000
# Parámetros del algoritmo genético
tamaño_poblacion = 50
num_generaciones = 200
num_padres_torneo = 4
probabilidad_mutacion = 0.9
```

El histograma de la figura 63 presentó el makespan analizados de los datos arrojados por la programación los cuales tienen un mínimo de 109825 unidades de tiempo y un máximo de 110624 unidades de tiempo con una media de 110223.556, se concluye que por un tamaño muestral pequeño no se puede afirmar que tipo de distribución toman los datos.

Figura 63.

Histograma regla de Johnson y algoritmo genético n=5000



Tiempos de ejecución

Los tiempos de ejecución es el tiempo que tarda la programación para darnos un valor (el makespan) tanto para el algoritmo genético como para la regla de Johnson, dado que al momento de ejecutar la programación se demora para arrojarnos el makespan, por ello se realizó el análisis acerca del tiempo de ejecución tanto para el algoritmo genético como para la regla de Johnson. A partir de lo anterior, se realizó toma de los tiempos de cómputo ya que fue otro factor importante a la hora de realizar dichos análisis, para este caso el tiempo de ejecución se tomó para 30 corridas en el caso de n=30, 100 y 300, para el caso de n=5000 fueron tomadas para 10 corridas.

Para los aleatorios de la máquina 1 (10-20) unidades de tiempo y máquina 2 (15-30) unidades de tiempo, los resultados obtenidos se presentan en la tabla 14:

Tabla 14.

Resumen tiempo de ejecución primer aleatorio

	Regla de Johnson			Algoritmo genético		
N	T. ejecución mín	T. ejecución máx	Tiempo prom	T. ejecución mín	T. ejecución máx	Tiempo prom
30	0	0	0	1	3	2
100	0	0	0	4	6	5
300	0	1	0,5	14	25	19,5
5000	8	9	8,5	550	556	553
N	T. ejecución mín	T. ejecución máx	Tiempo prom	T. ejecución mín	T. ejecución máx	Tiempo prom
30	0	0	0	5	8	6,5
100	0	0	0	18	21	19,5
300	0	0	0	59	63	61
5000	8	20	14	1545	1667	1606
N	T. ejecución mín	T. ejecución máx	Tiempo prom	T. ejecución mín	T. ejecución máx	Tiempo prom
30	0	0	0	21	23	22
100	0	1	0,5	74	83	78,5
300	0	1	0,5	225	233	229
5000	8	23	15,5	5012	6003	5507,5

Para los aleatorios de la máquina 1 (15-30) unidades de tiempo y máquina 2 (10-20) unidades de tiempo, se obtuvieron los siguientes resultados los cuales se muestran en la tabla 15:

Tabla 15.

Resumen tiempo de ejecución segundo aleatorio

	Regla de Johnson			Algoritmo genético		
N	T. ejecución mín	T. ejecución máx	Tiempo prom	T. ejecución mín	T. ejecución máx	Tiempo prom
30	0	0	0	1	2	1,5
100	0	0	0	4	6	5
300	0	0	0	9	10	9,5
5000	8	9	8,5	550	565	557,5
N	T. ejecución mín	T. ejecución máx	Tiempo prom	T. ejecución mín	T. ejecución máx	Tiempo prom
30	0	0	0	5	8	6,5
100	0	0	0	18	20	19
300	0	1	0,5	57	60	58,5
5000	9	20	14,5	1559	1668	1613,5
N	T. ejecución mín	T. ejecución máx	Tiempo prom	T. ejecución mín	T. ejecución máx	Tiempo prom
30	0	0	0	21	33	27
100	0	0	0	60	79	69,5
300	0	1	0,5	226	232	229
5000	8	18	13	4990	6001	5495,5

Cabe resaltar que dichos tiempos fueron pasados a segundos para poder llevar un mejor análisis de la toma de datos, de lo anterior se obtiene que los mayores tiempos se emplearon en las corridas más grandes como fue el caso de n=5000 para los tres parámetros del algoritmo genético, siendo esto uno de los factores que más tiempo tomó a la hora de la toma de datos. No se obtuvo una diferencia considerable al cambiar el parámetro de los números aleatorios entre máquinas, los valores en el primer aleatorio como en el segundo fueron casi los mismos.

Makespan

Para los aleatorios de la máquina 1 (10-20) unidades de tiempo y máquina 2 (15-30) unidades de tiempo y para los aleatorios de la máquina 1 (15-30) unidades de tiempo y máquina 2 (10-20) unidades de tiempo, se tomaron para los n=30, 100, 300 y 5000 donde se tomaron los el makespan máximo y mínimo con el fin de calcular el promedio lo que nos permitieron encontrar un mejor desempeño en la regla de Johnson y el algoritmo genético.

Tabla 16.

Makespan promedio primer aleatorio

	Regla de Johnson			Algoritmo genético		
N	Makespan mín	Makespan máx	Makespan prom	Makespan mín	Makespan máx	Makespan prom
30	615	729	668,3	616	732	670,09
100	2105	2326	2203,01	2110	2327	2206,42
300	6427	6783	6616,25	6434	6787	6621,5
5000	109602	110389	109907,9	109605	110389	109909,3
N	Makespan mín	Makespan máx	Makespan prom	Makespan mín	Makespan máx	Makespan prom
30	614	741	674,12	614	741	674,12
100	2062	2318	2208,37	2062	2318	2208,37
300	6386	6778	6615,04	6386	6778	6615,04
5000	109538	110638	110111,3	109538	110638	110111,5
N	Makespan mín	Makespan máx	Makespan prom	Makespan mín	Makespan máx	Makespan prom
30	603	722	669,82	603	722	669,82
100	2093	2328	2216,86	2093	2328	2216,86
300	6408	6845	6622,01	6408	6845	6622,01
5000	109760	110370	109972,7	109760	110370	109972,7

Tabla 17.

Makespan promedio segundo aleatorio

	Regla de Johnson			Algoritmo genético		
N	Makespan mín	Makespan máx	Makespan prom	Makespan mín	Makespan máx	Makespan prom
30	610	726	671,18	611	727	673,13
100	2116	2358	2221,73	2116	2362	2222,83
300	6411	6776	6603,1	6411	6782	6604,78
5000	109545	110356	109974,7	109546	110356	109975,1
N	Makespan mín	Makespan máx	Makespan prom	Makespan mín	Makespan máx	Makespan prom
30	609	716	673,33	609	716	673,33
100	2107	2335	2211,66	2107	2335	2211,66
300	6400	6800	6614,75	6400	6800	6614,75
5000	109510	110639	110022,5	109510	110639	110022,5
N	Makespan mín	Makespan máx	Makespan prom	Makespan mín	Makespan máx	Makespan prom
30	616	722	671,32	616	722	671,32
100	2089	2299	2202,46	2089	2299	2202,46
300	6422	6796	6615,75	6422	6796	6615,75
5000	109825	110624	110223,5556	109825	110624	110223,5556

De lo anterior se puede observar que se encuentra una diferencia en el promedio del makespan entre el algoritmo genético y la regla de Johnson, como se puede observar en la tabla 16 y tabla 17, en el primer parámetro del algoritmo genético se encuentra diferencia en el makespan ya que a menor probabilidad la elección en el makespan es más alto. En el segundo y tercer parámetro se observan el mismo makespan para la regla de Johnson y el algoritmo genético. Los valores aleatorios generados por la distribución rectangular tienen la misma probabilidad de salir, donde los valores tienden a ser normales.

Análisis de varianza

A través del análisis de varianza realizado para los tiempos de ejecución, se quiere saber si en el caso de: máquina 1 con aleatorios de 10-20 unidades de tiempo y la máquina 2 con aleatorios de 15-30 unidades de tiempo; la instancia de parámetros del algoritmo genético: tamaño de

población 50, número de generaciones 50, número de padres torneo 1 y probabilidad mutación 0.1 para cada tamaño de n, cuál de las hipótesis es verdadera, ya que la hipótesis nula es cuando las medias son iguales y la hipótesis alternativa es cuando las medias son diferentes.

Nota: *Si el valor P es menor al nivel de significación (5%), se rechaza la hipótesis nula*

Tabla 18.

Análisis de varianza tiempos de ejecución

Primer aleatorio			
N	Media Johnson	Media AG	Valor P
30	668,3	670,09	0,63030474
100	2203,01	2206,42	0,585217577
300	6616,25	6621,5	0,626789415
5000	109907,9	109909,3	0,991699978

Se puede observar que el valor p en ninguno de los n es menor a 0.05, es decir, la hipótesis nula no se rechaza, ya que estadísticamente las medias para el algoritmo genético y la regla de Johnson son iguales.

A través del análisis de varianza, se quiere saber si en el caso de: máquina 1 con aleatorios de 15-30 unidades de tiempo y la máquina 2 con aleatorios de 10-20 unidades de tiempo; la instancia de parámetros del algoritmo genético: tamaño de población 50, número de generaciones 50, número de padres torneo 1 y probabilidad mutación 0.1 para cada tamaño de n, cuál de las hipótesis es verdadera, ya que la hipótesis nula es cuando las medias son iguales y la hipótesis alternativa es cuando las medias son diferentes.

Tabla 19.

Análisis de varianza tiempos de ejecución

Segundo aleatorio			
N	Media Johnson	Media AG	Valor P
30	671,18	673,13	0,590276455
100	2221,73	2222,83	0,851117757
300	6603,1	6604,78	0,876272546
5000	109974,7	109975,1	0,997320334

Se puede observar que el valor p en ninguno de los n es menor a 0.05, es decir, la hipótesis nula no se rechaza, ya que estadísticamente las medias para el algoritmo genético y la regla de Johnson son iguales.

Por último, es importante afirmar que para las demás instancias del algoritmo genético no se realizó dicho análisis de datos, dado que las medias para la regla de Johnson y el algoritmo genético son las mismas, por ende, el valor p no se pudo obtener ya que es una división entre cero.

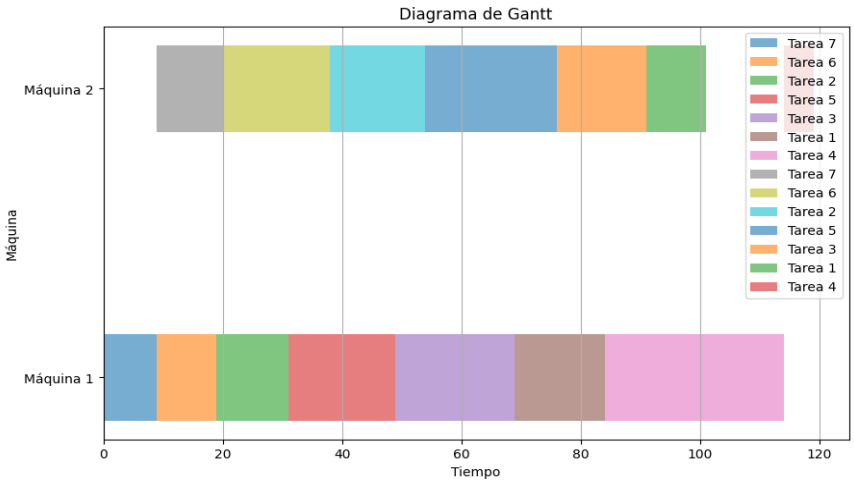
Diagrama de Gantt

Con el fin de comprobar la secuencia, se muestra a través del diagrama de Gantt que la regla está funcionando de manera eficiente, dado que el orden de las máquinas se respeta, el proceso B no comienza antes de haber terminado el proceso A, una máquina no puede tener una orden sin haber terminado las demás.

Para ello por medio del diagrama se quiso mirar que la programación se está ejecutando de manera correcta, ya que a través del diagrama de Gantt se corrobora que cuando se tienen una secuencia X, bien sea dada por el algoritmo genético o por la regla de Johnson, se respete el funcionamiento en serie, y calcular adecuadamente el makespan total de las órdenes.

Figura 64.

Diagrama de Gantt



Como se muestra en la figura 64 se observa cómo ingresando los tiempos por las dos máquinas, nos da una secuencia óptima validando la regla de Johnson.

Tiempo de reparación

Con el fin de obtener un análisis detallado de cuanto es el tiempo que conlleva realizar una reparación a una máquina cuando presenta una falla para el algoritmo genético y la regla de Johnson. Es importante aclarar que para el mantenimiento correctivo los datos para la programación fueron aleatorios por una distribución normal donde para la máquina 1 se tomó una media de 100 con una desviación estándar de 3 y para la máquina 2 una media de 120 con una desviación estándar de 2, para modificar de forma intermitente algunos valores que sean el doble de la media se le suma el factor de aleatoriedad basado en la media, como se muestra en la figura 65.

Figura 65.

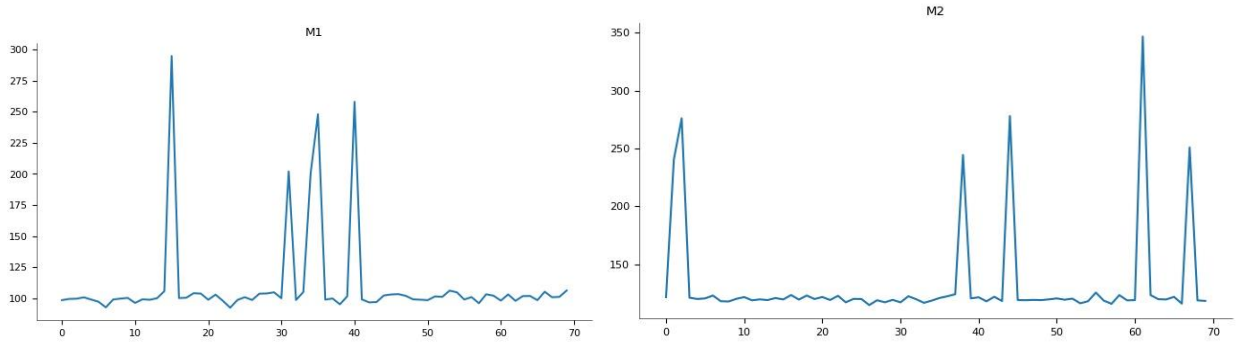
Datos mantenimiento correctivo

```
    # Modificar de forma intermitente algunos valores para que sean el doble de la media definida
    for columna, media in zip(['M1', 'M2'], [media_m1, media_m2]):
        for i in range(num_filas):
            if np.random.random() < probabilidad_modificar:
                # Modificar el valor de la celda para que sea el doble de la media definida
                df.at[i, columna] = media * 2+random.uniform(0, 1)*media

    return df

# Ejemplo de uso
num_filas = 10  # Número de filas en la matriz
media_m1 = 100  # Media inicial para M1
media_m2 = 120  # Media inicial para M2
probabilidad_modificar = 0.10  # Probabilidad de modificar un valor (15%)
```

Con el fin de mirar el análisis de los datos obtenidos para la máquina 1 y la máquina 2, el comportamiento de los datos se observa en la figura 66.

Figura 66.

Comportamiento de los datos en la máquina 1 y máquina 2



A continuación, se presenta en la tabla 20 se muestran los datos obtenidos.

Tabla 20.

Promedio makespan reparación máquina

Probabilidad modificar (que ocurra falla)	Tiempo ejecución regla Johnson	Tiempo ejecución algoritmo genético
0%	6093,9564	6648,8581
10%	7041,8847	7071,1304
25%	8402,53376	8481,69714
50%	10443,59496	10852,3374

A través del análisis de varianza realizado para el promedio del makespan en la reparación de la máquina, se quiere saber, cuál de las hipótesis es verdadera, ya que la hipótesis nula es cuando los promedios de los tiempos de ejecución son iguales y la hipótesis alternativa es cuando los promedios de los tiempos de ejecución son diferentes.

Nota: *Si el valor P es menor al nivel de significación (5%), se rechaza la hipótesis nula*

Tabla 21.

Análisis de varianza tiempo de ejecución mantenimiento correctivo.

Probabilidad modificar (que ocurra falla)	Tiempo ejecución regla Johnson	Tiempo ejecución algoritmo genético	Valor P
0%	6093,9564	6648,8581	0.31425
10%	7041,8847	7071,1304	0,59439
25%	8402,53376	8481,69714	0,25113
50%	10443,59496	10852,3374	0,00001

Se puede observar que el valor p es menor a 0.05 en la probabilidad de modificar del 50%, es decir, la hipótesis nula se rechaza, ya que los promedios de los tiempos de ejecución para el algoritmo genético y la regla de Johnson no son iguales.

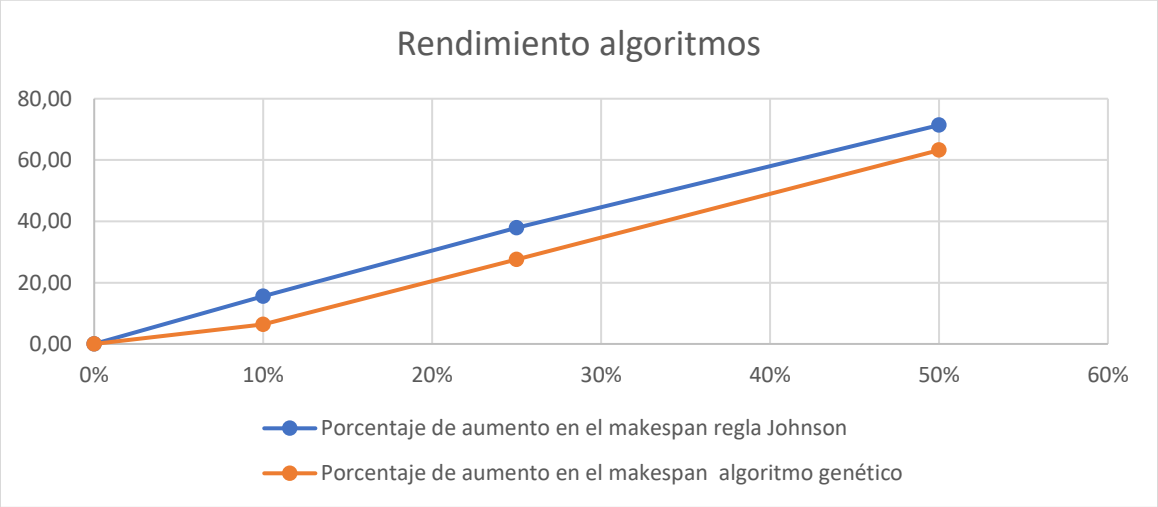
Para los casos donde la probabilidad de modificar es de 0%, 10% y 25%, el valor p es mayor a 0.05, es decir, la hipótesis nula no se rechaza, ya que estadísticamente los promedios de los tiempos de ejecución para ambos algoritmos son iguales.

Dicho promedio se tomó para un número de ordenes de 50 con un número de corridas de 100 respectivamente, para cada probabilidad de modificar, las cuales fueron 0%, 10%, 25% y 50%. Se sacaron los tiempos promedio de cada algoritmo con el fin de dar el rendimiento que tiene cada uno de ellos, a través de la siguiente ecuación:

$$\Delta\% = \frac{(t_2 - t_1)}{t_1}$$

Figura 67.

Rendimiento regla de Johnson y algoritmo genético



De la figura 67 podemos observar el rendimiento que presentan los algoritmos, obteniendo el makespan en la regla de Johnson un mejor rendimiento comparado con el algoritmo genético. Dado que los tiempos de ejecución son menores en la regla de Johnson, por lo que podemos inferir que el tiempo de ejecución de la regla de Johnson cuando la probabilidad de modificar esta en 0% es un valor menor al algoritmo genético.

8. Resultados

Se tuvieron en cuenta tres tamaños de n para obtener los resultados: grande, mediano y pequeño, de este modo se explica que los parámetros de entrada que se varían para n son: los valores aleatorios para cada máquina y los parámetros para el algoritmo genético

Se puede concluir que para los n pequeños el makespan fueron iguales, para los n de valores superiores el makespan en la regla de Johnson es por lo menos igual o mejor al algoritmo genético, lo único que nos varió fue al momento de la toma de los tiempos de ejecución donde la regla de Johnson fue la más rápida comparada con el algoritmo genético. Para los tiempos de ejecución y el makespan la regla de Johnson nos da una mejor solución comparado con el algoritmo genético, de lo anterior se concluye que estadísticamente no se encuentra una diferencia entre los dos métodos.

Por ello, a través de análisis de datos se validó lo obtenido de la programación, en los parámetros del algoritmo genético obtuvimos el makespan más alto con respecto a la regla de Johnson. Por lo anterior, para el caso de n=30, 100, 300 y 5000 con los parámetros del algoritmo genético tamaño_poblacion 50, num_generaciones 50, num_padres_torneo 1, probabilidad_mutacion 0.1 fue donde se encontró la diferencial dando así la regla de Johnson la mejor solución ya que un menor tiempo son soluciones más efectivas o cercanas al óptimo.

Con la ayuda del diagrama de Gantt, se pudo comprobar que los tiempos de generados por la secuencia, eran acordes con dicho diagrama, los cuales mostraban que eran consecuentes con la regla definida para el sistema en serie N/2.

A través del análisis de varianza realizado para los tiempos de ejecución donde máquina 1 con aleatorios de 10-20 unidades de tiempo y la máquina 2 con aleatorios de 15-30 unidades de

tiempo y viceversa, se obtuvo que estadísticamente las medias para el algoritmo genético y la regla de Johnson son iguales.

Por último, para el análisis de varianza realizado para el promedio del makespan en la reparación de la máquina se obtuvo que para la probabilidad de modificar del 50%, los promedios de los tiempos de ejecución para el algoritmo genético y la regla de Johnson no son iguales. Para el caso donde la probabilidad de modificar es de 0%, 10% y 25% la hipótesis nula no se rechaza, ya que estadísticamente los promedios de los tiempos de ejecución para ambos algoritmos son iguales.

9. Conclusiones

Posterior a la revisión de literatura se logra concluir que a medida que transcurren los años se evidencia un crecimiento de publicaciones donde se observa el interés de investigar acerca del secuenciamiento de un sistema de N trabajos en 2 máquinas, siendo esta actividad una de más importantes para las empresas ya que representa costos y esfuerzos, por esto los procesos depende en gran medida entregar los pedidos a tiempo, sin retardos y en óptimas condiciones. Con el objetivo de incrementar la productividad de las empresas o compañías.

Tras el análisis, se puede deducir que para la regla de Johnson en el desarrollo de algoritmos a la hora de ejecutar dicha programación el tiempo de ejecución al momento de dar un resultado es cuestión de segundos ya sea para un N de 30, 100, 300 y 5000. Sin embargo, al momento de ejecutar la programación para el algoritmo genético el tiempo de ejecución cambia de manera notoria como es el caso de un N de 5000 demorando un tiempo prolongado mayor a una hora para dar el makespan.

Tras el análisis, se puede deducir que en los parámetros del algoritmo genético al variar la probabilidad de mutación se observa que a menor probabilidad el algoritmo genético presenta el makespan más alto comparado con la regla de Johnson, ya que hace una selección en el makespan. Esto se da para el análisis de N:30, 100, 300 y 5000.

En el análisis de los tiempos de ejecución del tiempo de reparación para cada uno de los algoritmos, se observó que para las 4 instancias en las que se varía la probabilidad de modificar, las cuales fueron:0 %, 10%, 25% y 50% se observa que, para todos los casos, la regla de Johnson obtuvo un menor porcentaje en el makespan.

10. Recomendaciones

Se recomienda a los futuros investigadores continuar trabajando en el tema de investigación secuenciamiento de un sistema de N trabajos en 2 máquinas siendo este uno de los temas más retadores y complicados, estos problemas se han venido tratando con el paso de los años debido a la importancia que producen en las pequeñas y grandes empresas, modificando las operaciones de los procesos de producción se ven resultados en la reducción de costos, reducción de tiempos de fabricación a través de la erradicación de actividades innecesarias sin afectar los procesos de fabricación, dicho problema busca optimizar la utilización de los recursos. Se puede abordar este problema con otro tipo de heurística o metaheurísticas con el fin de minimizar el makespan.

Es importante abordar el tema realizando variaciones de los tamaños de n ya que nos permite mirar el makespan en comparación a los métodos que se implementen, en el caso del algoritmo genético es bueno realizar variaciones en los parámetros ya que se determine el rendimiento de este, observando la variabilidad que se tienen en ellos; ya que en el proyecto se obtuvo que el algoritmo genético con un probabilidad de mutación menor, se encuentra el makespan más alto en comparación a la otra regla.

Para futuras investigaciones se recomienda una línea de investigación con más máquinas, por ejemplo, secuenciamiento para un sistema N/3 (N trabajos en 3 máquinas) a partir de la regla de Johnson y el algoritmo genético, siendo este un gran desafío a nivel computacional al realizar un aumento en el número de máquinas, ya que la regla de Johnson se puede utilizar hasta 3 máquinas abordando lo ya mencionado durante todo el proyecto.

Referencias Bibliográficas

Aguilar, J. (2017). Resolución computacional de un problema de optimizacion combinatorio híbrido. *Ciencia e Ingeniería, 38*(2), 99-106.

Carranza, A. (06 de 09 de 2023). *Scribd.* Obtenido de https://es.scribd.com/document/87479288/Secuenciacion-n-Trabajos-Multiples-Centros-de-Trabajo#

Chase, R. B., Jacobs, E. R., & Aquilano, N. J. (2009). *Administración de operaciones: producción y cadena de suministros* (Duodécima ed.). Mexico: McGraw-Hill / Interamericana editores.

Correa, J. H. (2010). Aplicación de la heurística de palmer en la secuenciación de n tareas en m máquinas: un caso de estudio. *Scientia et technica, 3(46),*, 175-177.

French, S. (1982). *Sequencing and scheduling: An introduction to the mathematics of the job-shop.* Chichester, West Sussex: E. Horwood.

Garcia, J. P. (2006). *Universidad Politécnica de VALENCIA.* Recuperado el 5 de Septiembre de 2023, de http://personales.upv.es/jpgarcia/LinkedDocuments/ApuntesOptimizacionCombinatoria.pdf

Garduño Juárez, R. (2018 de Octubre de 2018). *Conogasi*. Obtenido de https://conogasi.org/articulos/algoritmos-geneticos/

Gen, M., & Cheng, R. (18 de 12 de 2000). *Genetic algorithms & engineering optimization* (1 ed.). (Wiley-Interscience, Ed.) New York, USA. doi:10.1002/9780470172261

Gupta, J. N., Krüger, K., Lauff, V., Werner, F., & Sotskov, Y. N. (1 de 9 de 2002). Heuristics for hybrid flow shops with controllable processing times and assignable due dates. (Elsevier, Ed.) *Computers & Operations Research, 29*, 1417 - 1439. doi:https://doi.org/10.1016/S0305-0548(01)00040-5

Jimenez Morales, Á. P. (2012). SOLUCIÓN DEL PROBLEMA DE PROGRAMACIÓN DE FLOW-SHOP FLEXIBLE EMPLEANDO EL ALGORITMO GENÉTICO DE CHU-BEASLEY. *Revista UCP*, 1-167.

Juan-Muns, J. G. (10 de 2016). *Secuenciación del mantenimiento preventivo periódico de maquinaria para minimizar el coste global asociado al tiempo entre intervenciones.* Obtenido de chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://upcommons.upc.edu/bitstream/handle/2117/98407/TFM_Josep_Guindulain.pdf?sequence=1&isAllowed=y

López, B. S. (17 de junio de 2019). *Ingenieria Industrial Online*. Obtenido de https://www.ingenieriaindustrialonline.com/produccion/regla-de-johnson/

López, J. C. (7 de Octubre de 2010). *Adictos al trabajo*. Obtenido de https://www.adictosaltrabajo.com/2010/10/07/jgap/

M. R. Garey, D. S. (1976). *The complexity of flowshop and jobshop scheduling* . Mathematics of operations research .

Melián, B., Moreno Perez, J., & Moreno Vega, M. (1 de 7 de 2003). Metaheuristics: A global view. *INTELIGENCIA ARTIFICIAL, 7*(19). doi:http://dx.doi.org/10.4114/ia.v7i19.713

Naupari Quiroz , R. E., & Rosales Geronimo , G. K. (2010). *Universidad Nacional Mayor de San Marcos, Perú*. Obtenido de

https://cybertesis.unmsm.edu.pe/bitstream/handle/20.500.12672/15141/Naupari_qr.pdf?sequence=1&isAllowed=y

Numerentur.org. (06 de 09 de 2023). Obtenido de https://numerentur.org/complejidad-computacional/

Paéz Becerra, S. E., & Mogollón Carreño, D. J. (11 de noviembre de 2022). Obtenido de https://noesis.uis.edu.co/items/6ca96ad9-c84c-4806-9f50-02f9f57d06a8/full

Pérez Pérez , E., Pérez Castillo , I., & Jiménez Bahri , M. (2014). Algoritmo genético para secuenciación de pedidos en taller de mecanizado con máquinas en paralelo, recirculación y tiempos de preparación. *Ingeniería Industrial. Actualidad y Nuevas Tendencias, IV*(12), págs. 38-53. Obtenido de http://www.redalyc.org/articulo.oa?id=215037911004

Pinedo, M. L. (2016). *Scheduling: theory, algorithms and systems.* Springer.

Salazar Hornig , E., & Medina S, J. C. (1 de Enero de 2013). Minimización del makespan en máquinas paralelas idénticas con tiempos de preparación dependientes de la secuencia utilizando un algoritmo genético Makespan Minimization for The Identical Machine Parallel Shop with Sequence Dependent Setup Times Using a Ge. *Ingeniería Investigación y Tecnología, 14*(1), 43-51. doi:10.1016/s1405-7743(13)72224-8

Silva, P. P., Rivero, D. P., & Bolivar, J. E. (30 de Abril de 2013). *Aplicación de una heurística constructiva en programación secuencial para asignación de varios trabajos a varias máquinas en paralelo.* Obtenido de Scientia et technica: https://dialnet.unirioja.es/descarga/articulo/4269539.pdf

Vidal Esmorís, A. (1 de Julio de 2013). *Universidad de Santiago de Compostela.* Obtenido de http://eio.usc.es/pub/mte/descargas/ProyectosFinMaster/Proyecto_782.pdf

Visualizing scientific landscapes. (2023). *Visualizing scientific landscapes*. Obtenido de Visualizing scientific landscapes: https://www.vosviewer.com/