CIATEC

ALGORITMOS GENÉTICOS PARA LA SECUENCIACIÓN DE TAREAS DENTRO DE LAS OPERACIONES EN FÁBRICAS DE PRODUCCIÓN INTERMITENTE

Tesis

QUE PARA OBTENER EL GRADO ACADÉMICO DE:

Maestro en Ciencia y Tecnología en la Especialidad de Ingeniería Industrial y de Manufactura.

PRESENTA:

Ing. Jorge Armando Ramos Frutos

Director

Dr. Javier Yáñez Mendiola

Co director.

Dr. Ricardo Pérez Rodríguez

León, Guanajuato, México, noviembre de 2020.

ALGORITMOS GENÉTICOS PARA LA SECUENCIACIÓN DE TAREAS DENTRO DE LAS OPERACIONES EN FÁBRICAS DE PRODUCCIÓN INTERMITENTE

León, Guanajuato, a 24 de febrero de 2021.

Dr. Ricardo Jaime Guerra Sánchez.
Director General.
CIATEC – PICYT.
León, Guanajuato.

Estimado Dr. Guerra:

Los abajo firmantes, miembros del Jurado del Examen de Grado del alumno Jorge Armando Ramos Frutos, una vez leída y revisada su Tesis titulada ***"Algoritmos Genéticos para la secuenciación de tareas dentro de las operaciones en fábricas de producción intermitente"***, aceptamos que la referida tesis revisada y corregida sea presentada por el alumno para aspirar al grado de Maestro en Ciencia y Tecnología en la Especialidad de Ingeniería Industrial y de Manufactura durante el Examen de Grado correspondiente.

Y para que así conste firmamos la presente a los 24 días del mes de febrero del año dos mil veintiuno.

Dr. Jesús Salvador Jaime Ferrer
Presidente.

Mtro. Armando Saldaña Valencia
Secretario.

Dr. Javier Yañez Mendiola.
Vocal.

EG-850-01-F14

León, Guanajuato, a 02 de febrero de 2021.

Coordinación de Posgrado.
PICYT – CIATEC.
Guanajuato.

Los abajo firmantes miembros del Comité Tutorial del alumno *Jorge Armando Ramos Frutos*, una vez leída y revisada la Tesis titulada ***"Algoritmos Genéticos para la secuenciación de tareas dentro de las operaciones en fábricas de producción intermitente"***, aceptamos que la referida tesis revisada y corregida sea presentada por el alumno para aspirar al grado de Maestro en Ciencia y Tecnología en la Especialidad de Ingeniería Industrial y de Manufactura durante el Examen de Grado correspondiente.

Y para que así conste se firma la presente a los 02 días del mes de febrero del año dos mil veintiuno.

Dr. Javier Yañez Mendiola.
Tutor Académico.

Dr. Ricardo Pérez Rodríguez.
Tutor en planta.

Omega No. 201, Col. Industrial Delta, CP. 37545, León, Gto., México.
Tel: +52 (477) 710 0011 atencionaclientes@ciatec.mx www.ciatec.mx

Agradecimientos

Existen personas en la vida que te dan un impulso en la misma para lograr tus sueños. Agradezco a la vida de ponerme en el camino personas tan especiales. Una de esas personas mi abuelo. Un hombre formidable y disciplinado que dejó huella en mi forma de ver la vida. Gracias por tu apoyo siempre, algún día nos volveremos a encontrar y seguiré diciéndote gracias por todas las oportunidades que me diste, por todas las enseñanzas y por creer en mí. Gracias Profesor José Guadalupe Ramos Cerda que, aunque en este año dejaste de ser cuerpo, te quedas en los corazones de muchos.

Agradezco a mis padres por ser parte de cada alegría, de cada tristeza, de cada momento de cólera, les agradezco por estar siempre a mi lado y darme motivos para seguir adelante. Amo a esa mujer que ha sido implacable en la vida a pesar de toda adversidad; mi madre. Amo a ese hombre servicial y de gran actitud ante la vida, siempre alegre. Gracias a mi abuela Ma. Tránsito por cada consejo sabio que me ha otorgado y espero lo siga haciendo por un periodo infinito de la vida. A mi abuela María por ser parte de mi formación como humano. Agradezco a mis hermanos y tí@s por tenderme la mano cuando se requiere de un amigo sincero.

Agradezco a mis asesores por sus enseñanzas y paciencia. Los admiro y procuro seguir sus consejos en la mayoría de mis actividades académicas. Agradezco a CIATEC por brindarme todos los recursos para concluir este ciclo de mi vida profesional. A CONACYT por el apoyo económico y por la oportunidad de generar un pensamiento crítico.

Resumen

La programación de operaciones es un proceso de toma de decisiones que es utilizado en compañías manufactureras. Un problema que se afronta al programar es establecer la secuencia de los trabajos. En el caso del Flow Shop todos los trabajos pasan por todas las estaciones. Las metaheurísticas encuentran una ruta óptima de un conjunto de posibles secuencias de trabajos. Se proponen Algoritmos Genéticos para encontrar el orden en que serán procesados los trabajos en las diferentes estaciones. La calibración de los parámetros del Algoritmo Genético se realizó utilizando la Metodología de Superficie de Respuesta para la optimización de dos respuestas (makespan y tiempo ocioso). Se analizó un problema y se obtuvo que los niveles óptimos de operación del Algoritmo Genético para esas dos variables de respuesta fueron 250 individuos de población inicial, 650 ciclos, 0.3 en probabilidad de cruza y 0.2 en probabilidad de mutación. De esta manera, se tienen mejoras en makespan del 11.37%.

Índice de Contenido

Índice de Tablas

Índice de Figuras

Capítulo 1 Marco Contextual

1.1 Introducción

La programación de operaciones es un proceso de toma de decisiones que es utilizado en compañías manufactureras. En general, se trata de establecer los recursos que serán utilizados para realizar ciertos trabajos en un determinado intervalo de tiempo (Baker & Trietsch, 2009). Un problema que se afronta al programar es establecer la secuencia de los trabajos. En el caso del Flow Shop todos los trabajos pasan por todas las estaciones. El objetivo de la secuenciación es la asignación eficiente de máquinas y otros recursos a los trabajos, y la determinación del orden en el cual cada uno de estos trabajos debe ser procesado. En la secuenciación se trabaja con un problema del tipo $\mathrm{NP}-\mathrm{Díficil}$ y se utilizan heurísticas y metaheurísticas para resolver el problema. Las metaheurísticas encuentran una ruta óptima de un conjunto de posibles opciones. Las metaheurísticas tienen parámetros que deben ser calibrados en su proceso. La calibración de los parámetros se puede realizar utilizando herramientas matemáticas de optimización. Por lo tanto, se debe proponer una metaheurística que otorgue valores óptimos en el problema de secuenciación de tareas en talleres de flujo continuo utilizando parámetros que se calibren con una técnica estadística.

Se ha trabajado con diferentes métodos para la optimización de secuencias en la programación de actividades dentro de fábricas de producción intermitente algunas de ellas son algoritmos de ramificación y acotamiento (Temiz & Erol, 2004) (Melab, Chakroun, Mezmaz, & Tuyttens, 2012) (Gooding, Pekny, & McCroskey, 1994), recocido simulado (Allahverdi, Aydilek, & Aydilek, 2020), búsqueda tabú (Ben & Al-Fawzam, 1998) (Gao, Chen, & Deng, 2011), colonia de hormigas (Yagmahan & Mutlu, 2008) (Engin & Güclü, 2018) (Zheng, Zhou, Xu, & Chen, 2019), Algoritmos de Estimación de Distribuciones (Shao, Pi, & Shao, 2018), Algoritmos Genéticos (Murata, Ishibuchi, & Tanaka, 1996) (Andrade, Silva, & Pessoa, 2019) (Chen & Hao, 2018), entre otros.

Los Algoritmos Genéticos propuestos llevan a Qing y Wang (2012) a utilizar un nuevo operador de cruce basado en la máquina y en el operador tomando la ruta crítica como base para el diseño del algoritmo demostrando la efectividad del algoritmo con la simulación por computadora de problemas semejantes. Wojakowski y Walzolek (2013) en su investigación llegan a la conclusión de que las metaheurísticas más explotadas en términos del problema de secuenciación de tareas son de manera definitiva los Algoritmos Genéticos. Pérez, Pérez y Jiménez para el 2014 presentan un algoritmo genético con población de individuos constante y realizando bajos porcentajes de cruce y altos porcentajes de mutación generada de forma se obtienen mejores resultados reduciendo el tiempo total de fabricación total entre 10%-20%. El nuevo algoritmo incrementa las posibilidades para 57 casos de 205 casos. Muthiah, Rajkumar y Muthukumar (2015) usan un algoritmo genético para minimizar el tiempo en que un pedido tarda en llegar al cliente. Algunos de los desarrolladores de Algoritmos Genéticos utilizan algunas estrategias en las operaciones realizadas por un Algoritmo Genético. Por tal motivo y por la cantidad de parámetros se elige utilizar un AG.

En la mayoría de los casos se seleccionan los parámetros del algoritmo genético (probabilidad de recombinación, probabilidad de mutación, etc.) con base en lo existente en la literatura o de forma aleatoria. Pocas veces se realiza un estudio para la selección de esos parámetros. La selección de los parámetros en el algoritmo genético cambia para cada caso y para cada tipo de operador. Katayama et al. utilizan una probabilidad de cruza de 1.0 y una probabilidad de mutación de 0.05 porque a ellos les resultaba “mejor” en los resultados obtenidos para el problema del “Agente Viajero”. Pongcharoen et al. utilizan diseños experimentales para obtener la mejor configuración de los parámetros y operadores, para obtener los mejores niveles de los parámetros y operadores utilizaron un diseño factorial con tres niveles en los parámetros y 9 niveles en los operadores para cribar. La optimización multiobjetivo de los parámetros que se fijan dentro de la metaheurística depende de las variables de respuesta que se utilicen como indicadores. Jolai et al. utilizan recocido simulado para resolver el problema de secuenciación de actividades en talleres de flujo continuo flexibles. Tratan con dos funciones objetivo,

la primera es la minimización del tiempo de finalización de todos los trabajos y la segunda es el tiempo de retraso. Para determinar el tipo de recocido simulado que será utilizado. Para ello, utilizan una técnica propuesta por ellos, mezclan el método Taguchi y el enfoque de Toma de Decisiones Multi-Objetivo. Concluyen que no se puede decidir por un método para la solución del problema. Villalba y Kumral (2018) realizan una calibración de los parámetros del algoritmo genético utilizado en problemas relacionados con la minería. Ellas utilizan un diseño factorial Box-Behnjen y la Metodología de Superficie de Respuesta para Múltiples Variables y toman como conclusiones las siguientes. Un tamaño de población grande no siempre incrementa el tiempo de solución, el tiempo de solución fue positivo en relación con los índices de mutación y recombinación; a mayores esos índices menores, es el tiempo de solución. El análisis simultáneo del tiempo de solución y las ganancias; en el problema de minería en el que se utilizó, ilustró la compensación entre el tiempo de cálculo aceptable y la conveniencia de obtener ganancias mediante la selección de parámetros de GA. En la literatura no se ha observado una optimización de múltiples criterios de decisión utilizando la Metodología de Superficie de Respuesta. Para esta tesis se considera la optimización de dos respuestas utilizando una función de deseabilidad dentro de la Metodología de Superficie de Respuesta para múltiples respuestas.

El contenido de la investigación se divide en cinco capítulos. El capítulo uno contiene la definición del problema, el estado del arte y una justificación. El capítulo dos contiene la teoría necesaria para el desarrollo del proyecto, desde la definición del problema de secuenciación de tareas hasta la Metodología de Superficie de Respuesta aplicada en múltiples variables de respuesta. En el tercer capítulo se plantean los objetivos y la hipótesis. En el capítulo cuatro se presenta el procedimiento de investigación y por último, en el capítulo cinco, se presentan los resultados.

1.2 Definición del Problema

La secuenciación de actividades en talleres de flujo continuo consiste en asignar un conjunto de n trabajos a un conjunto de m máquinas. Cada trabajo consiste en una secuencia de operaciones y cada trabajo visita de forma ordenada las máquinas en la planta de producción. Cada máquina sólo puede procesar un trabajo a la vez. El tiempo de finalización de todos los trabajos y el tiempo ocioso en el procesamiento de un conjunto de trabajos son criterios utilizados para la evaluación de las secuencias propuestas. Por lo tanto, se debe generar una secuencia que minimice el tiempo de finalización de todos los trabajos y el tiempo ocioso en la maquinaria o estaciones de trabajo. Las metaheurísticas son utilizadas para la solución de problemas de combinación. La secuenciación de actividades en talleres de flujo continuo es un problema de optimización combinatoria. Los Algoritmos Genéticos (AG) son metaheurísticas utilizadas para la solución de problemas de optimización combinatoria que se basan en la generación de poblaciones de individuos (vectores). Los procesos generados dentro del AG son probabilísticos y esto permite que su capacidad de exploración sobre la vecindad de posibilidades incremente. Utilizando AG permite no estancarse en óptimos locales y encontrar óptimos globales por su capacidad de exploración. Por lo que, se puede utilizar un AG para encontrar una secuencia que minimice los tiempos de finalización de todos los trabajos y minimice los tiempos de ocio en la maquinaria. Dentro del AG se configuran los parámetros: población inicial, probabilidad de recombinación, probabilidad de mutación y ciclos. Los valores de estos parámetros se configuran de manera empírica, utilizando técnicas de optimización o utilizando técnicas estadísticas. La Metodología de Superficie de Respuesta (MSR) para dos objetivos es conjunto de técnicas matemáticas y estadísticas que permite conocer los valores en los que los parámetros otorgan una respuesta óptima. La MSR permite conocer las condiciones óptimas de operación del AG. Por lo tanto, al utilizar la MSR para encontrar las condiciones óptimas de operación de un AG multi-objetivo mejora los resultados de problemas de secuenciación de tareas en talleres de flujo continuo en al menos un 5%.

1.3 Estado del Arte

Para este caso se divide el estado del arte en tres partes: la primera parte es la correspondiente a los trabajos realizados en la Secuenciación de Trabajos en Talleres de Flujo Continuo, la segunda parte muestra los trabajos de algoritmos genéticos y la Secuenciación de Trabajos en Talleres de Flujo Continuo y Talleres del Flujo Continuo Híbridos y la tercera parte se muestra la utilización de técnicas estadísticas para la optimización de parámetros en metaheurísticas.

1.3.1 Secuenciación de Trabajos en Talleres de Flujo Continuo

Algoritmos de ramificación y acotamiento (B&B) son los algoritmos determinísticos con mayor utilización. Haouari, Hidri y Ghasrbi en 2006 resuelven un problema de secuenciación para un taller de flujo continuo híbrido de dos estaciones utilizando un B&B efectivo utilizando como criterio el tiempo de terminación total de los trabajos, el método produce soluciones óptimas para problemas mayores a $1{,}000$ trabajos. Temiz y Erol (2004) resuelven el problema de secuenciación de tareas utilizando un algoritmo de ramificación y acotamiento difuso y logran ampliar el panorama de la administración sobre la secuencia de actividades. Melab et al. en 2012 utilizan la unidad de procesamiento gráfico para acelerar la ejecución del algoritmo, logrando la solución del problema y un incremento en la velocidad cien veces mayor. Gooding et al. utilizan como función objetivo la minimización del costo y programación matemática para resolver un problema de secuenciación en una configuración híbrida con máquinas paralelas. A pesar del éxito relativo de los algoritmos exactos. Los algoritmos exactos aún no son capaces de resolver casos medianos y largos y que son demasiado complejos para problemas del mundo real.

Fattahi, Jolai y Arkat en 2009 proponen Recocido Simulado para resolver el Problema de Secuenciación de Tareas en Talleres de Producción Intermitente. Ellos consideran que los trabajos coinciden de forma parcial en tiempo (existen piezas esperando a ser procesadas en cada estación de trabajo) lo que minimiza el tiempo de proceso total e incrementa la utilización de la maquinaria.

El algoritmo desarrollado por Pezzella, Morganti y Ciaschetti (2008) integran diferentes estrategias para la generación de la población inicial, seleccionando a los individuos para la reproducción y reproduciendo nuevos individuos generando

resultados comparables con el mejor algoritmo conocido basado en la Búsqueda Tabú. Los resultados computacionales obtenidos muestran el interés de ambos para obtener una eficiencia y efectividad en el método de resolución.

1.3.2 Algoritmos Genéticos y la Secuenciación de Trabajos en Talleres de Flujo Continuo y Talleres del Flujo Continuo Híbridos

Los Algoritmos Genéticos propuestos llevan a Qing y Wang (2012) a utilizar un nuevo operador de cruce basado en la máquina y en el operador tomando la ruta crítica como base para el diseño del algoritmo demostrando la efectividad del algoritmo con la simulación por computadora de problemas semejantes. Wojakowski y Walzolek (2013) en su investigación llegan a la conclusión de que las metaheurísticas más explotadas en términos del problema de secuenciación de tareas son de manera definitiva los Algoritmos Genéticos. Spanosa, Ponisb, Tatsiopoulosb, Christouc y Rokoub en 2014 proponen un Algoritmo Genético Paralelo Híbrido especializado en las operaciones de cruce y la mutación utilizando los conceptos de generar soluciones basadas en soluciones históricas para el enfoque de optimización combinatoria y búsqueda tabú en particular. Los resultados indican que el algoritmo propuesto es demasiado comparable con algunos de los mejores rendimientos de algoritmos genéticos para el problema. Pérez, Pérez y Jiménez para el 2014 presentan un algoritmo genético con población de individuos constante y realizando bajos porcentajes de cruce y altos porcentajes de mutación generada de forma se obtienen mejores resultados reduciendo el tiempo total de fabricación total entre 10%-20%. El nuevo algoritmo incrementa las posibilidades para 57 casos de 205 casos. Muthiah, Rajkumar y Muthukumar (2015) usan un algoritmo genético para minimizar el tiempo en que un pedido tarda en llegar al cliente. Algunos de los desarrolladores de Algoritmos Genéticos utilizan algunas estrategias en las operaciones realizadas por un Algoritmo Genético. Por tal motivo y por la cantidad de parámetros se elige utilizar un AG.

1.3.3 Técnicas Estadísticas Utilizadas en la Optimización de Parámetros y Operadores de una Metaheurística

En la mayoría de los casos se seleccionan los parámetros del algoritmo genético (probabilidad de recombinación, probabilidad de mutación, etc.) con base en lo existente en la literatura o de forma aleatoria. Pocas veces se realiza un estudio para la selección de esos parámetros. La selección de los parámetros en el algoritmo genético cambia para cada caso y para cada tipo de operador. Katayama et al. utilizan una probabilidad de cruza de 1.0 y una probabilidad de mutación de 0.05 porque a ellos les resultaba “mejor” en los resultados obtenidos para el problema del “Agente Viajero”. Pongcharoen et al. utilizan diseños experimentales para obtener la mejor configuración de los parámetros y operadores, para obtener los mejores niveles de los parámetros y operadores utilizaron un diseño factorial con tres niveles en los parámetros y 9 niveles en los operadores para cribar. Después se realizó otro diseño factorial y el resultado que se obtuvo es que todos los parámetros fueron significativamente diferentes en sus niveles excepto el parámetro de mutación.

1.4.4 Optimización Multi-objetivo de los Parámetros en Metaheurísticas

La optimización multiobjetivo de los parámetros que se fijan dentro de la metaheurística depende de las variables de respuesta que se utilicen como indicadores. Jolai et al. utilizan recocido simulado para resolver el problema de secuenciación de actividades en talleres de flujo continuo flexibles. Tratan con dos funciones objetivo, la primera es la minimización del tiempo de finalización de todos los trabajos y la segunda es el tiempo de retraso. Para determinar el tipo de recocido simulado que será utilizado. Para ello, utilizan una técnica propuesta por ellos, mezclan el método Taguchi y el enfoque de Toma de Decisiones Multi-Objetivo. Concluyen que no se puede decidir por un método para la solución del problema. Villalba y Kumral (2018) realizan una calibración de los parámetros del algoritmo genético utilizado en problemas relacionados con la minería. Ellas utilizan un diseño

factorial Box-Behnjen y la Metodología de Superficie de Respuesta para Múltiples Variables y toman como conclusiones las siguientes. Un tamaño de población grande no siempre incrementa el tiempo de solución, el tiempo de solución fue positivo en relación con los índices de mutación y recombinación; a mayores esos índices menores, es el tiempo de solución. El análisis simultáneo del tiempo de solución y las ganancias; en el problema de minería en el que se utilizó, ilustró la compensación entre el tiempo de cálculo aceptable y la conveniencia de obtener ganancias mediante la selección de parámetros de GA.

1.4 Justificación

Dentro de las operaciones de una empresa se encuentra la programación de actividades. Una de las decisiones que se toman al programar operaciones es el orden de la secuencia que siguen las actividades dentro de la planta al ser procesadas. En la secuenciación de tareas se plantean varios objetivos y el programador elige cuál de ellos es el que va acorde a los objetivos generales de la empresa. Algunos de los objetivos son:

- Minimizar el tiempo de terminación de todos los trabajos.
- Minimizar los retrasos en las entregas.
- Minimizar los tiempos de ocio de las estaciones y/o máquinas.
- Minimizar el inventario en proceso.
- Maximizar la utilización de las máquinas.
- Minimizar los costos de producción.
- Maximizar las utilidades.

El AG siempre genera una solución, pero puede que no siempre sea la misma por el carácter probabilístico que hay en sus operadores. En problemas que contienen 20 o más trabajos puede generar soluciones diferentes en cada caso. Para obtener mejores resultados en el AG con respecto a los objetivos ya mencionados, se deben elegir: la cantidad de población inicial, la cantidad de ciclos, la probabilidad de recombinación y la probabilidad de cruza. La MSR es una metodología que se utiliza para elegir los parámetros de operación óptimos dentro de cualquier proceso. Para el caso del AG se utiliza la MSR para determinar los niveles de operación óptimos de los parámetros utilizados en el AG para generar resultados óptimos en los objetivos que son el tiempo de finalización de todos los trabajos y el tiempo ocioso en las estaciones o máquinas.

Utilizando la MSR se dejan de tomar decisiones empíricas sobre los niveles de operación de los parámetros del AG. Al conocer los niveles óptimos de los parámetros se pueden generar secuencias que logren tiempos de finalización

menores y tiempos ociosos menores que utilizando técnicas empíricas o cualitativas. La ventaja de los parámetros del AG es que pueden tomar cualquier valor entero en los ciclos y tamaño de población, y cualquier valor continuo en la probabilidad de recombinación y la probabilidad de mutación. Por lo que se puede utilizar el punto óptimo obtenido en la MSR para generar mejores resultados, dado que la región de operabilidad es la misma que la región de analizada.

Capítulo 2 Marco Teórico

2.1 Programación de Operaciones

La programación de operaciones es un proceso de toma de decisiones que es utilizado en compañías manufactureras. En general, se trata de establecer los recursos que serán utilizados para realizar ciertos trabajos en un determinado intervalo de tiempo (Baker & Trietsch, 2009). Con la programación de actividades se trata de contribuir a la mejorar u optimización de algún índice (tiempo de terminación, retardos en entrega, porcentaje de utilización, inventario en proceso, etc.) (Krajewski, Ritzman, & Malhotra, 2008).

Los recursos y tareas en una organización pueden ser concebidos de diferentes formas. Algunos de los recursos son: máquinas en una fábrica, cajeros en un banco, empleados en una empresa, etc. Las tareas por su parte, pueden ser operaciones de algún proceso, emisión de una factura, actividades de mantenimiento, etc. (Jacobs & Chase, 2014) (Heizer & Render, 2008)

Los procesos o configuraciones productivas se pueden clasificar de la siguiente manera (Ramos, 2001):

- Configuración Productiva por Proyectos.
    - Se utiliza para la fabricación de productos únicos que representan cierta complejidad debida a la cantidad de entradas, volumen o peso del producto, lo cual hace complicado el transporte al finalizar el proceso productivo. En este tipo de configuración, todas las entradas se trasladan al lugar del producto o donde se presta el servicio.
- Configuración Productiva por Lotes.
    - Se utiliza para la fabricación de múltiples productos haciendo uso de las mismas instalaciones, de tal manera que entre referencias se realice un ajuste a las máquinas para hacer posible el procesamiento del siguiente producto. Se divide en configuración de centros de trabajo y en línea.
- Configuración Continua.

- Se ejecutan las mismas operaciones, en las mismas máquinas, para la obtención del mismo producto, con una disposición en cadena o línea, se tiene un alto volumen de producción con costos bajos y cumpliendo plazos de entrega establecidos. Cada operario y/o máquina siempre realiza la misma operación. Las máquinas están programadas para aceptar de manera automática el trabajo que una máquina precedente le suministra a esta también de manera automática.

2.1.1 Secuenciación

Un problema que se afronta al programar es establecer la secuencia de los trabajos. En el caso del Flow Shop todos los trabajos pasan por todas las estaciones.

El objetivo de la secuenciación es la asignación eficiente de máquinas y otros recursos a los trabajos, y la determinación del orden en el cual cada uno de estos trabajos debe ser procesado.

Los siguientes supuestos aparecen de manera frecuente en la literatura sobre secuenciación de trabajos en máquinas:

- Las máquinas siempre están disponibles y nunca dejan de funcionar.
- Cada máquina puede procesar a lo mucho un trabajo a la vez.
- Cualquier trabajo puede ser procesado en una máquina a la vez.
- Los tiempos de preparación de todos los trabajos son cero.
- No se permiten interrupciones.
- Los tiempos de cambio son independientes de los programas y están incluidos en los tiempos de procesado.
- Los tiempos de cambio y las restricciones tecnológicas son deterministas y se conocen de antemano, al igual que con las fechas de entrega.
- El número de estaciones de trabajo m es al menos de 2.
- Todos los trabajos son procesados siguiendo el mismo flujo de producción: estación 1, estación 2, ..., estación m.
- Cada trabajo j requiere un tiempo de procesamiento p_{jk} en la estación k.

Con estos supuestos, se refleja que la teoría se aleja de la realidad dado que los entornos donde se realiza producción son dinámicos y sujetos a una serie de variaciones de carácter estocástico. Estas perturbaciones pueden modificar el estado del sistema productivo y afectar el rendimiento del mismo, entre estas perturbaciones se encuentran eventos relacionados con los recursos como los fallos en máquinas, bajas en los operarios, no disponibilidad de materiales o herramientas, etc. Por otro lado, existen eventos relacionados con los trabajos como son la llegada de órdenes, la cancelación de trabajos, cambios en las fechas de entrega, etc.

En muchos problemas industriales reales estos supuestos no son válidos y es por eso que en los últimos años han aparecido modelos y procedimientos de resolución que relajan uno o varios de los supuestos anteriores.

Las variantes importantes del problema de secuenciación son (Öztop, Fatih, Türsel, & Pan, 2019):

1. **Job Shop Scheduling.** Consiste en un conjunto finito de trabajos, en donde cada trabajo está dividido en operaciones o actividades, estas operaciones serán procesadas o ejecutadas en un determinado número de recursos o máquinas, cada trabajo tiene su propia ruta de actividades a seguir.
2. **Task Scheduling.** Se tiene un conjunto de tareas, cada una de ellas está asociada a una duración determinada de tiempo, el objetivo es programar las tareas en máquinas procurando el orden más apropiado, atendiendo que para que se ejecute una tarea, sus predecesoras deben ser ejecutadas.
3. **Flow Shop Scheduling Problem.** Consiste en un número de trabajos que son procesados en un número de máquinas, cada trabajo en el mismo orden. Se encuentra la siguiente variedad de problemas
    a. **Flow Shop Básico.** Es considerado un caso general del Job Shop diferenciándose de este porque los trabajos a procesar siguen la misma ruta de procesamiento a través de una serie de máquinas organizadas de manera lineal.
    b. **Flow Shop Permutacional.** En esta configuración la secuencia inicial de los trabajos llevada a cabo en la primera etapa se mantiene para el

resto de las etapas de la línea, por lo tanto existen $n!$ secuencias posibles como solución a este problema.

c. **Flow Shop sin y con Buffer.** Un sistema sin buffer o con buffer de capacidad nula es aquel que al finalizar una tarea en determinada máquina, esta no puede avanzar hasta la siguiente máquina en su ruta de procesamiento si hay otra tarea siendo ejecutada en dicha máquina, lo que evita que el trabajo avance y por lo tanto se queda bloqueando la máquina que ya terminó su tarea y así mismo bloqueando el acceso de los trabajos que suceden al trabajo estancado.

d. **Flow Shop Híbrido.** Es un caso especial de Flow Shop en donde en cada etapa puede existir más de una máquina y estas son conocidas como máquinas paralelas, las cuales se clasifican en:

   - **Idénticas:** con tiempos de procesamiento iguales para todas las máquinas.
   - **Uniformes:** cuando los tiempos de procesamiento tienen una relación paramétrica entre ellas.
   - **No relacionadas:** cuando los tiempos de procesamiento no pueden ser expresados mediante una relación paramétrica.

   Mientras que en el Flow Shop básico al sólo existir un recurso por etapa es necesario sólo tomar la decisión de la secuencia de tareas, en el Flow Shop Híbrido se deben tomar dos decisiones: la asignación de los trabajos a las máquinas de cada etapa y la secuencia de los trabajos en las diferentes máquinas.

e. **Flow Shop Flexible.** Al incluir el concepto de flexibilidad se encuentra un problema donde los trabajos aún siguen una secuencia lineal a través de las etapas, pero el sistema tiene la capacidad de permitir que los trabajos puedan saltar una o más etapas durante su procesamiento, es decir, se trata de trabajos que no necesitan ser procesados en todas las etapas del proceso.

Existen tiempos entre el procesamiento de los trabajos, es decir, existe un periodo destinado a la preparación o alistamiento de las máquinas entre la continuación de los trabajos, este tiempo de configuración de cada máquina puede depender del trabajo anterior y es denominado como tiempos de alistamiento dependientes de la secuencia.

2.2 Algoritmos Genéticos

Un algoritmo genético (AG) es una técnica de búsqueda iterativa (Sivanandam & Deepa, 2008) que imita el proceso de evolución biológica de “sobrevivencia del más apto” (Ponce, 2010). Los AG no buscan modelar la evolución biológica sino derivar estrategias de optimización (Ruiz & Vázquez, 2009) (Hillier & Lieberman, 2010). El concepto se basa en la generación de poblaciones de individuos mediante la reproducción de los padres (Mirjalili, 2019).

2.2.1 Codificación del Algoritmo Genético

La codificación es un proceso de representación de los genes de los individuos. El proceso de codificación puede ser realizado utilizando bits, números, árboles, arreglos, listas o algunos otros objetos. La codificación que interesa en este caso está relacionada con la problemática a tratar. Para el caso de la secuenciación de tareas en fábricas de producción intermitente se utiliza la codificación por permutación o codificación real (Sivanandam & Deepa, 2008).

Cada cromosoma es una cadena de números, la cual representa los números en secuencia. En la codificación por permutación, cada cromosoma es una cadena de valores enteros-reales, los cuales representan números en una secuencia. Este tipo de codificación se muestra en la figura 1.

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

Figura 1. Codificación por Permutación

Fuente: Sivanandam y Deepa, 2008

2.2.2 Operadores de Selección

El principal objetivo del operador de selección es obtener las mejores soluciones de una población inicial (Kramer, 2017). Un operador de selección es diseñado en una forma que se obtenga una baja probabilidad de seleccionar una mala solución. Se presentan dos operadores de selección: Selección por torneo y ruleta.

Selección por torneo. Esta es una estrategia de selección que está disponible para ajustar la presión selectiva y la diversidad de población. La estrategia de selección por torneo proporciona presión selectiva (ésta determina en qué grado la reproducción está dirigida por los mejores individuos) mediante la celebración de un torneo entre N individuos.

El mejor individuo del torneo es el que tiene la aptitud más alta. El ganador se coloca en la piscina de apareamiento. El torneo es repetido hasta que la piscina de apareamiento está llena de cromosomas.

Selección por ruleta. La selección por ruleta es una de las técnicas tradicionales de los algoritmos genéticos. El operador consiste en seleccionar un individuo utilizando como criterio una probabilidad proporcional a la aptitud. El principio de la selección por ruleta es una búsqueda lineal por una ruleta con ranuras en la rueda con una proporción de los valores de aptitud de los individuos. Un valor objetivo se fija, el cual es una proporción aleatoria de la suma de las aptitudes en la población. La población cesa hasta que el valor objetivo se alcanza y el individuo que se selecciona es aquél que se encuentra sobre el valor objetivo. La desventaja de esta técnica es que se basa en la aptitud de los individuos y existen posibilidades de quedar en un óptimo local. Se recomienda que la población no se ordene para evitar el error en la selección y poder salir de los óptimos locales (Sivanandam & Deepa, 2008).

El método se implementa como sigue:

- La suma del valor esperado total de los individuos en la población. Será T.
- Repetir N veces:
    - Elegir un aleatorio r entre 0 y T
    - Sumar los valores esperados de los individuos, hasta que la suma sea más grande o igual que r. El individuo que se selecciona para la operación de recombinación es aquél que se encuentra sobre este límite.

La selección por ruleta es fácil de implementar, pero es ruidosa. El índice de evolución depende de la varianza de las aptitudes de la población.

2.2.3 Operadores de Recombinación

La recombinación (cruza) es el proceso de tomar dos padres de la piscina de selección y de ellos producir un hijo. Este proceso viene después de la selección, la población se enriquece con los mejores individuos (Sivanandam & Deepa, 2008). Existen varios operadores de cruza en los individuos de codificación entera, los que se presentan son: un solo punto, PMX, CX y OX.

Recombinación de un sólo punto. Para ilustrar cómo se crean los hijos con el operador de un solo punto (Taha, 2012), considere los cromosomas padres $P_1 = 1-3-5-2-4$ y $P_2 = 5-4-2-3-1$. Suponga que ocurre un cruce de un punto aleatorio en el tercer gen. Los dos primeros genes de $C_1(C_2)$; cromosomas de los descendientes, se construyen intercambiando los dos primeros genes de $P_1(P_2)$. Los últimos tres genes son los que permanecen a partir de $P_1(P_2)$ después de excluir a los primeros dos genes, es decir;

$$\text{Primeros 2 genes de } C_1 = \{5, 4\}$$
$$\text{Primeros 2 genes de } C_2 = \{1, 3\}$$
$$\text{Últimos 3 genes de } C_1 = \{1, 3, 5, 2, 4\} - \{5, 4\}$$
$$\text{Últimos 3 genes de } C_2 = \{5, 4, 2, 3, 1\} - \{1, 3\}$$

Por lo tanto,

$$C_1 = 5-4-1-3-2$$
$$C_2 = 1-3-5-4-2$$

Cruce por emparejamiento parcial (Partially Mapped Crossover; PMX). Cruce por Emparejamiento Parcial (PMX) también conocido como cruce en dos puntos, es un operador utilizado en la solución de problemas de permutación por su capacidad de producir hijos con genes diferentes. En la figura siguiente se puede ver la forma en que se generan los hijos utilizando PMX.

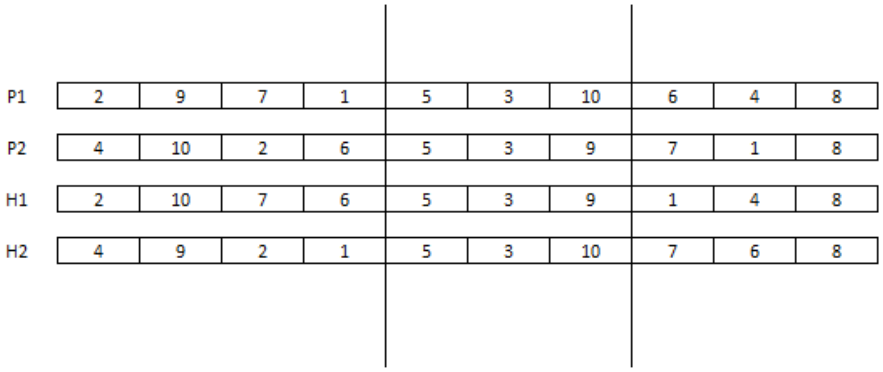

Figura 2. *Recombinación por emparejamiento parcial*

La idea del operador es seleccionar de forma aleatoria dos posiciones por las cuales serán divididos los cromosomas padres. La parte comprendida entre los dos puntos de corte del primer padre la heredará el segundo hijo conservando las posiciones, mientras que la misma parte correspondiente al segundo padre hará parte del cromosoma del primer hijo.

Para terminar la construcción del primer hijo se toma cada gen del padre (no perteneciente a la región entre los dos cortes) y se revisa si ya se encuentra en el cromosoma hijo, en caso de que no se encuentre se heredará dicho valor en la misma posición.

Para aquellas posiciones que aún quedan vacías se buscará qué genes del segundo padre aún no hacen parte de este. El segundo hijo se termina de construir de forma similar.

Cruce Cíclico (Cycle Crossover; CX). Es un operador que garantiza que los hijos generados a partir de él cumplan con exigencia inicial de los cromosomas, permitiendo que estos sigan siendo factibles en la solución.

En la figura siguiente se pueden observar los dos ciclos de los que está compuesto este operador. El primer ciclo inicia con la primera posición del padre número uno, dicho gen será heredado al primer hijo. En el cromosoma del segundo padre se buscará la posición donde se encuentra el gen heredado y esta será la posición del siguiente gen a heredar por el hijo uno, el cual estará en la misma posición que en el cromosoma del padre uno. El ciclo termina cuando el valor del gen en el padre

uno esté en la primera posición del segundo padre (tal como ocurrió con el gen ubicado en la posición nueve, tarea cuatro, que se encuentra en la primera posición del segundo padre).

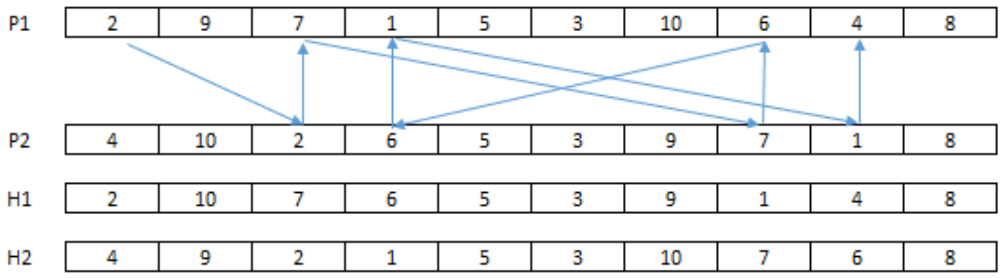

Figura 3. *Recombinación por cruce cíclico*

Una vez finalizado este ciclo se procede a completar el cromosoma del primer hijo; para lo cual se busca qué genes del segundo padre aún no hacen parte del cromosoma y se van colocando en aquellos genes que aún no tienen valor.

Cruce por Orden (Order Crossover; OX). La idea central es obtener una máscara binaria que indicará que gen será heredado de cada padre.

En la figura siguiente se puede ver un ejemplo de este operador. La máscara binaria generada muestra que gen heredará cada hijo de sus padres; aquellas posiciones donde aparece el número uno, indica que el hijo uno heredará dicho gen del padre uno y el segundo hijo lo heredará del segundo padre. Aquellas posiciones donde aparezca un cero en la máscara indicará que este gen se tomará del otro padre (primer hijo tomará el gen del segundo padre y el segundo hijo del primer padre. Como es necesario tener en cuenta que los genes no pueden repetirse se procederá a completar el cromosoma de forma similar al segundo ciclo del operador de cruce cíclico.

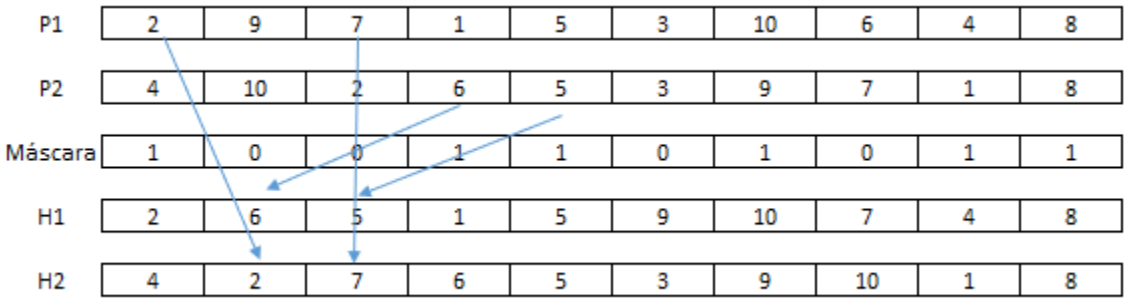

Figura 4. *Recombinación basada en orden*

2.2.4 Operadores de Mutación

La operación de mutación viene después de la operación de recombinación. Esta operación permite a que el algoritmo genético explore otras regiones del espacio de soluciones. Al explorar más el espacio de soluciones disminuye la probabilidad de quedar en un mínimo o máximo local y obtener mejores soluciones. Se presentan dos operadores de mutación para el caso de la codificación del tipo permutación: mutación por intercambio y mutación por inserción.

Mutación por intercambio. Dos posiciones (genes) en el cromosoma (vector) se seleccionan de forma aleatoria y el valor de sus alelos es intercambiado (Kramer, 2017) Esto se ilustra en la figura siguiente.

Figura 5. *Mutación por Intercambio*

Mutación por inserción. Se seleccionan dos alelos de forma aleatoria y el segundo se mueve al lado del primero, recorriendo los alelos que se encuentran entre los puntos selectos a la derecha. El procedimiento se ilustra en la figura siguiente:

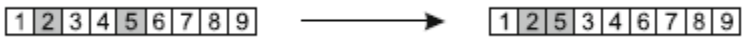

Figura 6. *Mutación por Inserción*

2.2.5 Reemplazo

En la operación de reemplazo se coloca el 50% de los mejores individuos obtenidos de las operaciones de mutación y recombinación en la población inicial para el caso del ciclo 1 y en la población $i-1$ para el caso del segundo o mayores ciclos.

Como se puede ver en los puntos 2.2.1 al 2.2.5 se habla de los procesos que ocurren dentro del AG. Los cuales se numeran a continuación:

1. Generación de la población inicial.
2. Selección de un conjunto de individuos de la población inicial.
3. Recombinación de algunos de los individuos que fueron seleccionados.
4. Mutación de algunos de los individuos seleccionados.
5. Reemplazo de los mejores individuos en la población inicial o la población $i-1$-ésima
6. Repetir los pasos 2 al 5 de acuerdo a los ciclos programados en el algoritmo.

Lo anterior se muestra en el seudocódigo de la figura 7. En el cual se utiliza un ciclo for con la cantidad de ciclos determinados por el usuario.

Inicio del Algoritmo Genético
Definir:
    Individuos de la población inicial, cantidad de trabajos, función objetivo,
    Probabilidad de recombinación, probabilidad de mutación, ciclos.
Codificación
Población Inicial
Evaluación de aptitud
for $i = 1$:ciclos
    Población (sólo si $i \geq 2$)
    Evaluación de la aptitud (sólo si $i \geq 2$)
    Selección
    Recombinación
    Mutación
    Reemplazo (nueva población (50%) de los individuos obtenidos)
End
Tiempo de procesamiento de n trabajos en m máquinas
Fin del Algoritmo Genético

Figura 7. Seudocódigo del Algoritmo Genético empleado para resolver el problema de secuenciación de tareas

2.3 Diseño de Experimentos

El diseño de experimentos es una estrategia para la planificación de experimentos de tal manera que las conclusiones relevantes sean alcanzadas en forma eficiente y económica (Gutiérrez & De la Vara, 2012). Un diseño experimental eficiente te permite obtener conclusiones sobre un fenómeno observado utilizando el mínimo de recursos (Díaz, 2009). Dentro del diseño experimental se encuentran diferentes técnicas que permiten analizar los fenómenos, tales técnicas deben cumplir tres supuestos: normalidad, homogeneidad e independencia (Montgomery, Diseño y Análisis de Experimentos, 2004). A continuación, se describen tres técnicas de diseño experimental que son las utilizadas por la metodología de superficie de respuesta.

2.3.1 Diseño de Experimentos 2^k

Estos diseños son útiles cuando el número de factores a estudiar está entre dos y cinco, rango en el cual su tamaño está entre cuatro y 32 tratamientos. Si el número de factores es mayor que 5 se recomienda utilizar un factorial fraccionado 2^{k-p}. En general, los factoriales en dos niveles sean constituyen el conjunto de diseños de mayor impacto en las aplicaciones, debido a su eficacia y versatilidad (Montgomery, Diseño y Análisis de Experimentos, 2004).

En el diseño 2^k se consideran k factores con dos niveles cada uno, y tiene 2^k tratamientos o puntos de diseño. Las k columnas y 2^k renglones componen la matriz para este diseño, considerando una réplica, se construyen de la siguiente manera: en la primera columna, que corresponde a los niveles del factor A, se alternan los signos $+$ y $-$, empezando con – hasta llegar a los 2^k renglones; en la segunda columna se alternan dos signos menos con dos signos más; en la tercera, se alternan cuatro signos menos y cuatro signos más, y así sucesivamente hasta la k-ésima columna compuesta por 2^{k-1} signos menos seguidos de 2^{k-1} signos más.

En la tabla 1 se muestra la familia de diseños factoriales 2^k. Nótese que el número de tratamientos siempre es potencia de dos.

Tabla 1. Diseño Experimental para Cuatro Factores

Tratamiento	Notación de Yates	*A*	*B*	*C*	*D*
1	(1)	−	−	−	−
2	*a*	+	−	−	−
3	*b*	−	+	−	−
4	*ab*	+	+	−	−
5	*c*	−	−	+	−
6	*ac*	+	−	+	−
7	*bc*	−	+	+	−
8	*abc*	+	+	+	−
9	*d*	−	−	−	+
10	*ad*	+	−	−	+
11	*bd*	−	+	−	+
12	*abd*	+	+	−	+
13	*cd*	−	−	+	+
14	*acd*	+	−	+	+
15	*bcd*	−	+	+	+
16	*abcd*	+	+	+	+

Fuente: Gutiérrez y De la Vara, 2012

Con el diseño factorial completo 2^k se pueden estudiar en total 2^{k-1} efectos siguientes:

$$\binom{k}{1} = k, \text{efectos principales}$$

$$\binom{k}{2} = \frac{k!}{2!\,(k-2)!} = \frac{k(k-1)}{2}, \text{interacciones dobles}$$

$$\binom{k}{3} = \frac{k!}{3!\,(k-3)!}, \text{interacciones triples. Y así hasta}$$

$$\binom{k}{k} = 1 \text{ interacción de los } k \text{ factores}$$

Donde la operación $\binom{k}{r} = \frac{k!}{r!(k-r)!}$ son las combinaciones de los k factores tomados de r en r. Por ejemplo, el diseño factorial 2^5 tiene cinco efectos principales, 10 interacciones dobles, 10 interacciones triples, 5 interacciones cuádruples y una interacción quíntuple, lo cual da un total de $2^5 - 1 = 31$ efectos.

Cada uno de los efectos se estima a partir de su contraste, el cual a su vez se puede obtener construyendo la tabla de signos del diseño. Recordemos que las columnas de signos para los contrastes que definen a los efectos principales están dadas por la matriz del diseño, mientras que la columna de un efecto de interacción se obtiene multiplicando las columnas que señala dicho efecto de interacción. En la tabla 2 se muestra parte de la tabla de signos para un diseño 2^5.

Tabla 2. Tabla de Signos para un Diseño Factorial 2^5

Notación de Yates	A	B	C	D	E	AB	AC	AD	AE	BC	BD	BE	CD	CE	DE
(1)	−	−	−	−	−	+	+	+	+	+	+	+	+	+	+
a	+	−	−	−	−	−	−	−	−	+	+	+	+	+	+
b	−	+	−	−	−	−	+	+	+	−	−	−	+	+	+
ab	+	+	−	−	−	+	−	−	−	−	−	−	+	+	+
c	−	−	+	−	−	+	−	+	+	−	+	+	−	−	+
ac	+	−	+	−	−	−	+	−	−	−	+	+	−	−	+
bc	−	+	+	−	−	−	−	+	+	+	−	−	−	−	+
abc	+	+	+	−	−	+	+	−	−	+	−	−	−	−	+
d	−	−	−	+	−	+	+	−	+	+	−	+	−	+	−
ad	+	−	−	+	−	−	−	+	−	+	−	+	−	+	−
bd	−	+	−	+	−	−	+	−	+	−	+	−	−	+	−
abd	+	+	−	+	−	+	−	+	−	−	+	−	−	+	−
cd	−	−	+	+	−	+	−	−	+	−	−	+	+	−	−
acd	+	−	+	+	−	−	+	+	−	−	−	+	+	−	−
bcd	−	+	+	+	−	−	−	−	+	+	+	−	+	−	−
abcd	+	+	+	+	−	+	+	+	−	+	+	−	+	−	−
e	−	−	−	−	+	+	+	+	−	+	+	−	+	−	−
ae	+	−	−	−	+	−	−	−	+	+	+	−	+	−	−
be	−	+	−	−	+	−	+	+	−	−	−	+	+	−	−
abe	+	+	−	−	+	+	−	−	+	−	−	+	+	−	−
ce	−	−	+	−	+	+	−	+	−	−	+	−	−	+	−
ace	+	−	+	−	+	−	+	−	+	−	+	−	−	+	−
bce	−	+	+	−	+	−	−	+	−	+	−	+	−	+	−
abce	+	+	+	−	+	+	+	−	+	+	−	+	−	+	−
de	−	−	−	+	+	+	+	−	−	+	−	−	−	−	+
ade	+	−	−	+	+	−	−	+	+	+	−	−	−	−	+
bde	−	+	−	+	+	−	+	−	−	−	+	+	−	−	+
abde	+	+	−	+	+	+	−	+	+	−	+	+	−	−	+
cde	−	−	+	+	+	+	−	−	−	−	−	−	+	+	+
acde	+	−	+	+	+	−	+	+	+	−	−	−	+	+	+
bcde	−	+	+	+	+	−	−	−	−	+	+	+	+	+	+
abcde	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Fuente: Gutiérrez y De la Vara, 2012

El contraste de cada efecto se obtiene al multiplicar su columna de signos por la columna de totales expresados en la notación de Yates. Con los contrastes se procede a estimar los efectos mediante la siguiente fórmula.

$$\text{Efecto } A = \frac{1}{n2^{k-1}}[\text{Contraste } A]$$

Para el cual su suma de cuadrados con un grado de libertad está dada por:

$$SC_A = \frac{1}{n2^k}[\text{Contraste } A]^2$$

Donde n es el número de réplicas del experimento.

ANOVA del Diseño Factorial 2^k

La suma de cuadrados totales (SC_T) en el diseño factorial 2^k se calcula como:

$$SC_T = \sum_{i=1}^{n2^k} y_i^2 - \frac{y_.^2}{n2^k}$$

Y tiene $n2^k - 1$ grados de libertad, donde el subíndice i corre sobre el total de observaciones. La suma de cuadrados del error (SC_E) se obtiene por diferencia y tiene $2^k(n-1)$ grados de libertad. Con estas dos sumas de cuadrados y las de los efectos, se procede a escribir la tabla ANOVA siguiendo los esquemas particulares. Cada efecto de interés en el ANOVA es una fuente de variación para la cual se prueba la hipótesis H_0:efecto $= 0$ y H_1:efecto $\neq 0$. Así, cuando se concluye que un efecto está activo, significa que es estadísticamente diferente de cero.

Si en la tabla ANOVA se incluye el total de efectos que se estiman con el factorial completo 2^k, será necesario realizar cuando menos dos réplicas del experimento para estimar una suma de cuadrados del error. Sin embargo, en la mayoría de los casos sólo interesa estudiar los efectos principales y las interacciones dobles. Esto hace que cuando el número de factores es mayor o igual a cuatro ($k \geq 4$), no será estrictamente necesario realizar réplicas.

Cabe agregar que cuando se emplea un diseño factorial 2^k, se supone que la respuesta es aproximadamente lineal en el rango de variación de cada uno de los factores estudiados. No es necesario suponer una linealidad perfecta, pero sí que no haya una curvatura muy grande. De esta manera, dado que cada factor se prueba en dos niveles, no es posible estudiar los efectos de curvatura (efectos del tipo A^2, B^2, etc.), aunque esta exista en el proceso; para estudiar tales efectos se necesitan al menos tres niveles de cada factor. Esto no implica que sea recomendable un diseño factorial con al menos tres niveles en cada factor, sino que en primera instancia se pueden agregar repeticiones (mínimo tres) al centro del diseño factorial 2^k, y con ella se podrá detectar la presencia de curvatura.

Diagrama de Pareto de Efectos

El diagrama de Pareto para los efectos sin estandarizar representa una manera práctica de ver cuáles efectos son los más grandes en cuanto a su magnitud. Se recomienda utilizar el gráfico de Pareto para decidir cuáles efectos mandar al error.

Se dice que el diagrama de Pareto trabaja limpiamente cuando quedan bien delimitados los diferentes grupos de efectos, de los más a los menos importantes (véase la figura 8). En la figura 8, cada concavidad de la línea sobrepuesta a las barras indica las oleadas o rachas que ocurren, y en este caso habría dos posibilidades para construir el error y hacer el análisis de varianza: excluir el primer grupo de menor importancia o también se excluye el segundo grupo de menor importancia. Por otra parte, si las barras del diagrama quedan como escalones de igual tamaño, el principio de Pareto no está trabajando limpiamente, y en esta situación es necesario usar otros criterios que ayuden a dilucidar dónde hacer el corte de exclusión.

Otros Criterios Útiles

Cuando no es claro el Diagrama de Pareto. Se recomienda fijarse en todos los criterios siguientes y no sólo en uno de ellos:

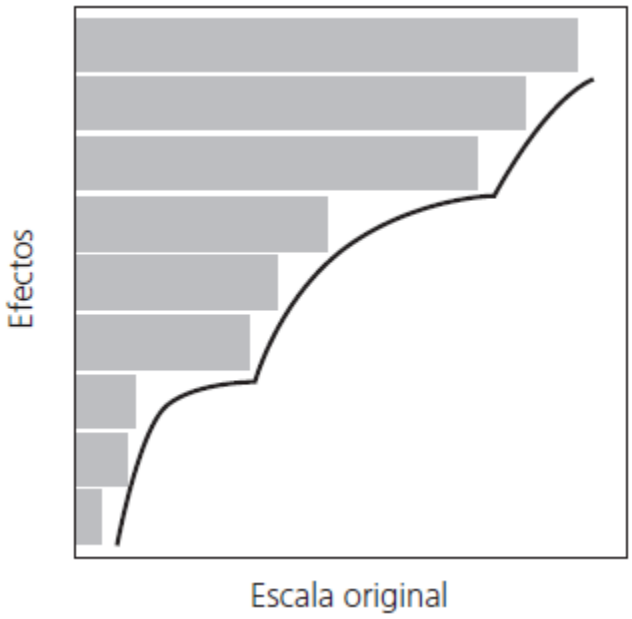

Figura 8. Diagrama de Pareto aplicado en el diseño de experimentos, figura tomada de Gutiérrez y de la Vara, 2012

1. La magnitud del efecto. Si se conoce la desviación estándar σ del proceso, la magnitud del efecto puede indicar si éste se manda al error. De manera específica, en el factorial 2^k con una réplica se compara el efecto observado contra dos veces el error estándar del efecto ($\sigma/\sqrt{2^{k-2}}$) y si el primero es más grande es porque puede ser un efecto real.
2. Si primero se excluyen los efectos que son claramente no significativos de acuerdo con el gráfico de Pareto, se puede lograr un ANOVA preliminar cuya significancia da información útil para excluir o no los efectos restantes. Específicamente, los efectos cuyas significancias en el ANOVA preliminar están alrededor de 0.2 o menores, no necesariamente se excluyen del análisis. Esta decisión es más confiable que cuando dicho ANOVA preliminar ya alcanzó al menos ocho grados de libertad para el error.
3. Los grados de libertad del error deben ser al menos ocho para tener un ANOVA más confiable.
4. El R^2_{aj} del modelo ANOVA preliminar. Cuando se van eliminando los efectos que no son significativos, el estadístico R^2_{aj} crece. En el momento en el que

se elimina un efecto y este estadístico decrece 3% o más, significa que posiblemente ese efecto no debe excluirse.

Colapsar o Proyectar el Diseño

Cuando en el mejor ANOVA se detecta que un factor particular no es significativo, ya que su efecto principal y todas las interacciones en las que interviene no son importantes, entonces en lugar de mandar al error este factor y sus interacciones, otra posibilidad es colapsar o proyectar el diseño, lo cual consiste en eliminar completamente del análisis a tal factor, con lo que el diseño factorial 2^k original se convierte en un diseño completo con un factor menos $(k-1)$ y con dos repeticiones en cada punto. Al haber repeticiones en el diseño 2^{k-1} resultante de la colapsación, entonces se puede estimar el CM_E y construir la tabla del análisis de varianza de manera usual. En general, si se pueden omitir h factores, los datos se convierten en un diseño factorial 2^{k-h} con 2^h repeticiones en cada punto.

El efecto de colapsar un diseño factorial 2^3 se representa en la figura 9.

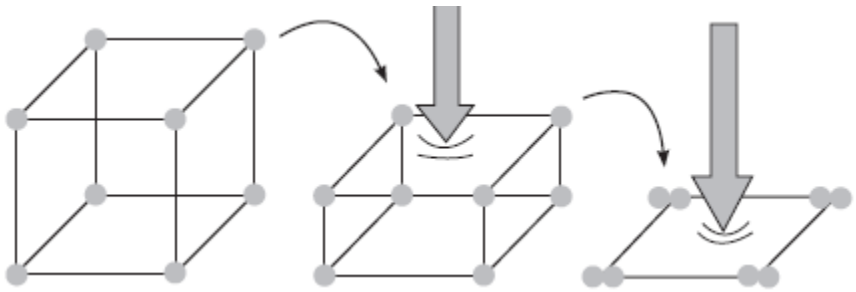

Figura 9. Colapso de un diseño experimental 2^3, figura tomada de Gutierrez y de la Vara, 2012

2.3.2 Metodología de Superficie de Respuesta

Después de una primera etapa experimental quizá sea necesario desplazar la región experimental en una dirección adecuada, o bien, explorar en forma más detallada la región experimental inicial. La forma de realizar ambas cosas, son parte de la llamada metodología de superficie de respuesta (MSR).

La MSR es una estrategia experimental y de análisis que permite resolver el problema de encontrar las condiciones de operación óptimas de un proceso. La

región experimental es el espacio delimitado por los rangos de experimentación utilizados con cada factor. La región de Operabilidad está definida por el conjunto de condiciones donde el equipo o proceso puede ser operado. En la figura 10, se presenta un ejemplo donde se muestra la región experimental y la región de Operabilidad.

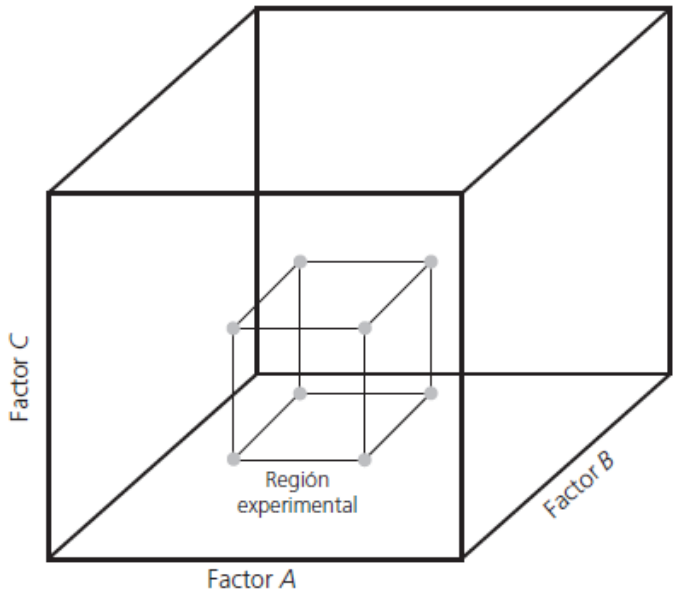

Figura 10. Región experimental y región de operabilidad de un diseño factorial 2^3, figura tomada de Gutiérrez y de la Vara, 2012

El punto óptimo que interesa encontrar pudiera localizarse en cualquier lugar de la región de Operabilidad, dentro o fuera de la región experimental.

En diseños factoriales completos el mejor tratamiento es el "tratamiento ganador", desde el punto de vista estadístico, de entre todos los que se probaron en el estudio. En cambio, el punto óptimo implica que es la mejor combinación posible en toda la región de Operabilidad. Determinar el punto óptimo requiere de una estrategia más completa, que incluye la posibilidad de realizar varios experimentos en forma secuencial y el uso de otras técnicas de análisis.

En la figura 11 se muestra la diferencia entre punto óptimo y mejor tratamiento. Las curvas de nivel o isolíneas en esta figura representan el "verdadero

comportamiento" de la respuesta y tiene un punto óptimo localizado en el centro de la elipse más pequeña. Cada curva de nivel representa puntos sobre la montaña que tienen la misma altura. El problema es encontrar la combinación (x_{01}, x_{02}) que da por resultado óptimo del proceso.

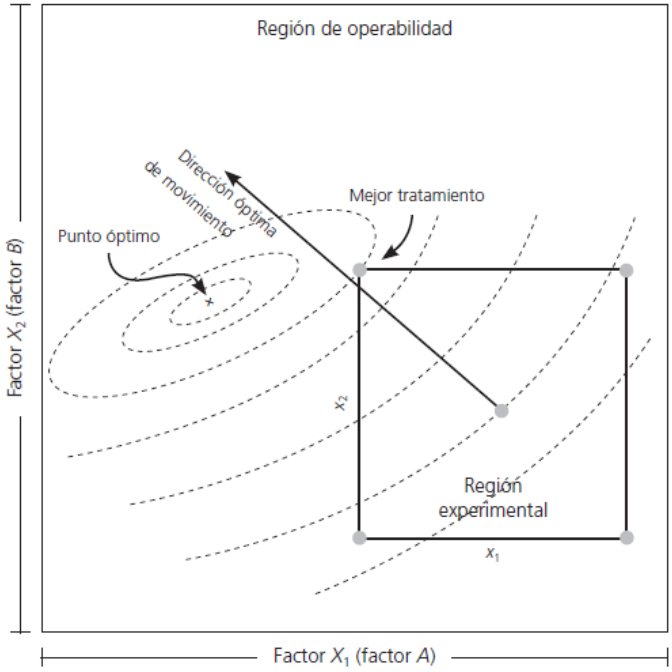

Figura 11. Traslación de la región experimental para encontrar el óptimo de la región de operabilidad, figura tomada de Gutiérrez y de la Vara, 2012

Se debe analizar si realizar el esfuerzo por encontrar el óptimo es rentable. En ocasiones el óptimo se encuentra cerca del mejor tratamiento. Y se debe analizar si es necesario correr más experimentos para obtener el óptimo tomando en cuenta los costos que se generan por la realización del o los experimentos. En muchas ocasiones tiene mayor factibilidad operativa tomar un punto cercano al óptimo porque las condiciones del proceso lo exigen.

La MSR implica tres aspectos: diseño, modelo y técnica de optimización. El diseño y el modelo se piensan al mismo tiempo, y dependen del tipo de comportamiento que se espera en la respuesta. De manera específica, el modelo puede ser de primero o segundo orden.

El aspecto diseño implica que para optimizar un proceso se debe aplicar el diseño de experimentos, en particular aquellos que sirven para ajustar un modelo de regresión lineal múltiple.

El aspecto modelo utiliza el análisis de regresión lineal múltiple, junto con sus elementos básicos que son: parámetros del modelo, modelo ajustado, significancia del modelo, prueba de falta de ajuste, residuos, predichos, intervalos de confianza para predichos y coeficiente de determinación.

Por último, el aspecto optimización está formado por algunas técnicas matemáticas que sirven para explorarlo a fin de obtener información sobre el punto óptimo. Conviene recordar técnicas como: derivadas de funciones, multiplicadores de Lagrange, operaciones con matrices, valores y vectores propios y sistemas de ecuaciones simultáneas.

En la figura 12 se presenta un esquema de la metodología de superficie de respuesta, donde se distinguen tres etapas en la búsqueda del punto óptimo, que son: cribado, búsqueda I o de primer orden y búsqueda II o de segundo orden.

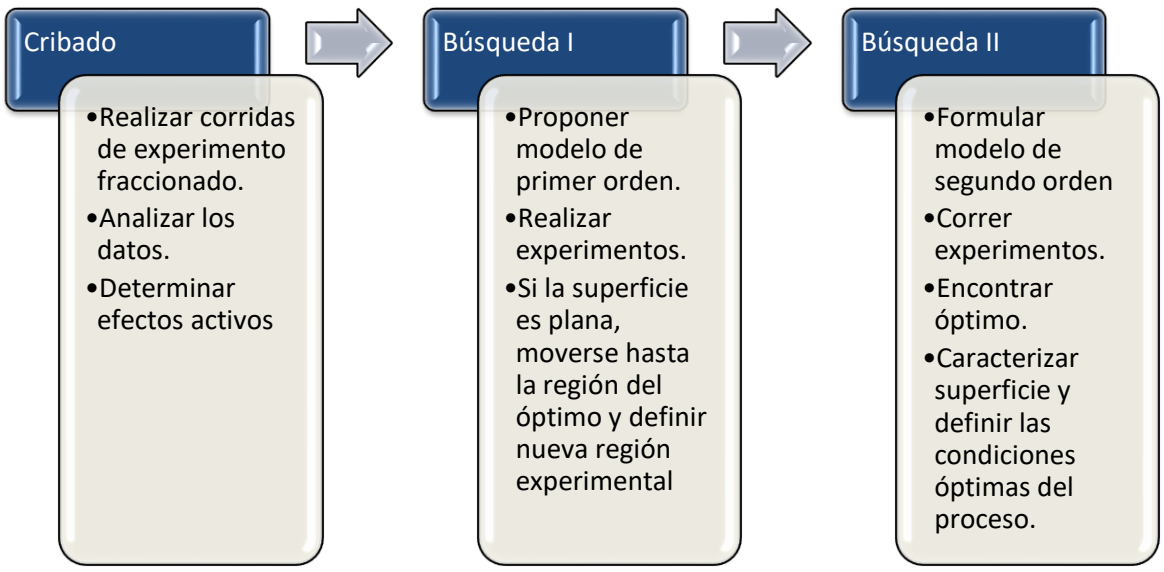

Figura 12. Metodología de Superficie de Respuesta

La MSR se representa en la figura 13, en la cual se supone ya rebasada la fase de cribado y se presentan las dos etapas posteriores considerando dos variables de proceso. La realidad del proceso está representada por las curvas de nivel, y el punto óptimo deseado se encuentra en el centro de la superficie más pequeña marcado con una cruz. En la práctica no se conoce a priori dónde se encuentra este punto debido a que la realidad se desconoce, no obstante, la MSR es una buena estrategia para llegar a éste.

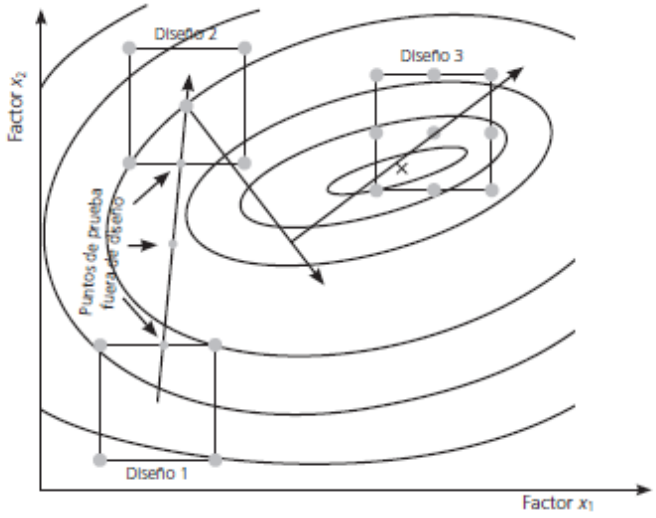

Figura 13. Últimas dos etapas de la Metodología de Superficie de Respuesta, figura tomada de Gutiérrez y de la Vara, 2012

Las superficies de respuesta se caracterizan ajustando un modelo a los datos experimentales. Los modelos utilizados en MSR son básicamente polinomios. Si se tienen k factores, el modelo de primer orden está dado por:

$$Y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \varepsilon$$

Y el modelo de segundo orden:

$$Y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum_{i=1}^{k} \sum_{<j=1}^{k} \beta_{ij} x_i x_j + \varepsilon$$

La forma específica que toma la superficie en un modelo de segundo orden depende de los signos y magnitudes de los coeficientes en el modelo. En las figuras 14, 15 y 16 se muestran las tres formas básicas, que son: superficie de máximo, superficie de mínimo y superficie con punto silla; en ese orden.

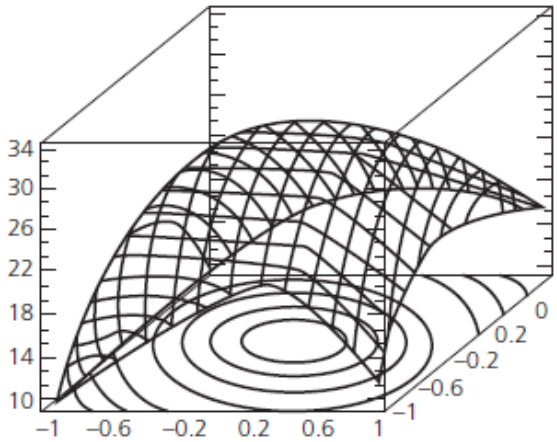

Figura 14. Superficie de máximo, figura tomada de Gutiérrez y de la Vara, 2012

Para más de dos factores las superficies de respuesta no se pueden graficar completas de una sola vez porque se encuentran en cuatro dimensiones o más. Para $k > 2$ el modelo de primer orden representa un hiperplano y el segundo orden constituye un hiperelipsoide o hiperboloide. Sin embargo, para $k = 3$ factores es posible graficar la superficie haciendo las tres gráficas con dos factores cada vez, manteniendo el tercero constante.

En superficie de respuesta se prefieren modelos jerárquicos (tiene los términos más simples que componen los términos de mayor orden que están en el modelo), ya que tienen un comportamiento más estable y suave que facilita la exploración de las superficies que representan. Esto implica la eliminación de efectos o términos del modelo debe ser menos estricta que en el análisis de varianza.

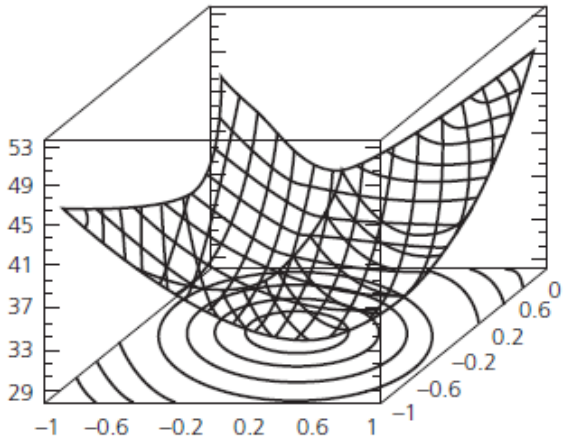

Figura 15. Superficie de mínimo, figura tomada de Gutiérrez y de la Vara, 2012

Si el modelo no explica un mínimo de 70% del comportamiento de la respuesta en términos del R^2_{aj}, no se recomiendo utilizarlo para fines de optimización porque su calidad de predicción es mala. La técnica de optimización a utilizar depende del tipo de modelo ajustado y existen básicamente tres métodos, que son:

1. Escalamiento ascendente
2. Análisis canónico
3. Análisis de cordillera

El escalamiento ascendente es para el modelo de primer orden y las otras dos técnicas son para el modelo de segundo orden.

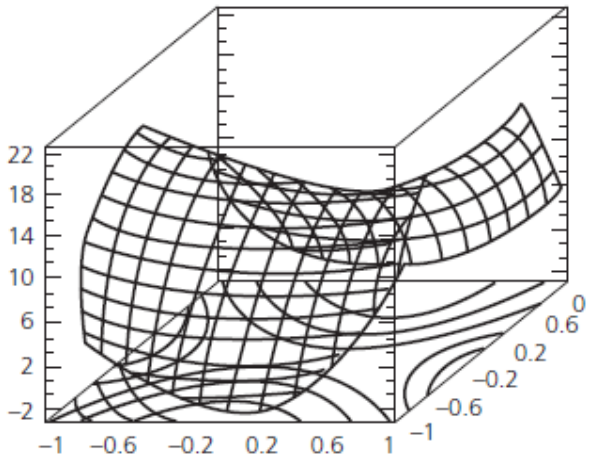

Figura 16. Superficie de silla, figura tomada de Gutiérrez y de la Vara, 2012

El escalamiento ascendente se aplica cuando el tipo de variable de respuesta de interés es del tipo: mientras más grande es mejor, se busca encontrar condiciones que maximicen esa variable, entonces se tiene un escalamiento ascendente; pero si lo que interesa es minimizar porque se tiene una variable del tipo: mientras más pequeña mejor, se trata de escalamiento descendente.

La técnica de optimización de escalamiento se aplica cuando se cree que se está lejos de la condición óptima, por lo que será necesario determinar una dirección en la cual experimentar fuera de la región inicial.

A partir del conocimiento que ya se tiene del problema es preciso seleccionar los niveles de los factores para determinar la región de exploración.

Para obtener la trayectoria es necesario decidir una longitud de paso en unidades codificadas y reales. Se recomienda utilizar un paso de movimiento unitario (en unidades codificadas) en el factor con mayor influencia, con lo que se asegura que los pasos en los factores restantes serán de menor amplitud y proporcionales a sus coeficientes.

Un paso unitario equivale a moverse en el factor correspondiente a intervalos de un medio de su rango de prueba en unidades originales**.** Una longitud de paso mayor se considera agresiva y puede tener riesgos, mientras que una longitud de un paso menor se considera una forma conservadora de proceder, que en algunos casos se está experimentando a nivel proceso puede ser buena opción.

El análisis canónico se aplica un diseño de segundo orden cuando se quiere explorar con más amplitud una región experimental y/o cuando se espera que el punto óptimo ya esté cerca (probablemente cerca de la región experimental).

La mejor estrategia será encontrar, primero, los coeficientes de la ecuación canónica que indican el tipo de superficie observada y sólo si esta es del tipo que interesa (por ejemplo, máximo), entonces se procede a localizar las coordenadas del punto estacionario. Si la superficie encontrada no es del tipo deseado, se sigue el análisis de cordillera descrito posteriormente.

El punto estacionario es el punto $(x_{10}, x_{20}, \dots, x_{k0})$ en el espacio de factores, sobre el cual el plano es tangente a la superficie tiene pendiente igual a cero. Por ejemplo, si la superficie tiene un máximo, el punto estacionario es justo el punto donde se ubica el máximo. De aquí que el punto estacionario sea un candidato natural o un punto óptimo, que resulta "electo" sólo cuando es del tipo que interesa y se encuentra dentro de la región experimental. Podría pasar que, aunque se está buscando un máximo, el punto estacionario sea un punto mínimo o un punto silla, en cuyo caso evidentemente no se trataría del óptimo buscado.

Suponga que ya se realizaron los tres primeros pasos del análisis canónico y que, por lo tanto, ya se tiene ajustado un modelo de segundo orden:

$$\hat{Y} = \hat{\beta}_0 + \sum_{i=1}^{k} \hat{\beta}_i x_i + \sum_{i=1}^{k} \hat{\beta}_{ii} x_i^2 + \sum_{i=1}^{k} \sum_{<j=1}^{k} \hat{\beta}_{ij} x_i x_j$$

para el cual se quiere encontrar el punto estacionario. El punto se localiza derivando al modelo con respecto a la variable x_i, igualando a cero y resolviendo en forma simultánea todas las ecuaciones. Todo esto se facilita si el modelo se reescribe en notación matricial como:

$$\hat{Y} = \hat{\beta}_0 + x'b + x'Bx$$

donde $x' = (x_1, x_2, \dots, x_k)$ es cualquier punto en la región de Operabilidad del proceso, en unidades codificadas; el vector b son los coeficientes de la parte lineal (efectos principales) del modelo y la matriz B son los coeficientes de las interacciones y de los términos cuadráticos puros. Esto es:

$$b = \begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_k \end{pmatrix}$$

$$B = \begin{pmatrix} \hat{\beta}_{11} & \hat{\beta}_{12}/2 & \hat{\beta}_{13}/2 & \cdots & \hat{\beta}_{1k}/2 \\ \hat{\beta}_{12}/2 & \hat{\beta}_{22} & \hat{\beta}_{23}/2 & \cdots & \hat{\beta}_{2k}/2 \\ \hat{\beta}_{13}/2 & \hat{\beta}_{23}/2 & \hat{\beta}_{33} & \cdots & \hat{\beta}_{3k}/2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{\beta}_{1k}/2 & \hat{\beta}_{2k}/2 & \hat{\beta}_{3k}/2 & \cdots & \hat{\beta}_{kk} \end{pmatrix}$$

Derivando el modelo dado con respecto al vector x e igualando a cero, se obtiene:

$$\frac{\partial \hat{Y}}{\partial x} = b + 2Bx = 0$$

Resolviendo para x se llega a que el punto estacionario está dado por:

$$x_0 = -\frac{B^{-1}b}{2}$$

2.3.3 Metodología de Superficie de Respuesta Múltiple

El problema en la optimización simultánea radica en que, por lo general, los óptimos individuales no caen en la misma combinación de los factores de control $(X_1, X_2, \dots, X_k)$. Esto hace necesario buscar una solución de compromiso, en la que todas las variables tengan un nivel satisfactorio. A esa solución compromiso la llamaremos óptimo simultáneo. Por ejemplo, en la figura siguiente se muestran dos superficies de dos respuestas en la misma región experimental. Si en ambas respuestas interesa el mínimo, claro que los óptimos individuales se encuentran en extremos opuestos de la región experimental. Por lo tanto, habrá que buscar otra combinación (punto) donde las dos variables sean al mismo tiempo lo más pequeñas posible. En la figura 17 se muestra dónde está tal óptimo simultáneo.

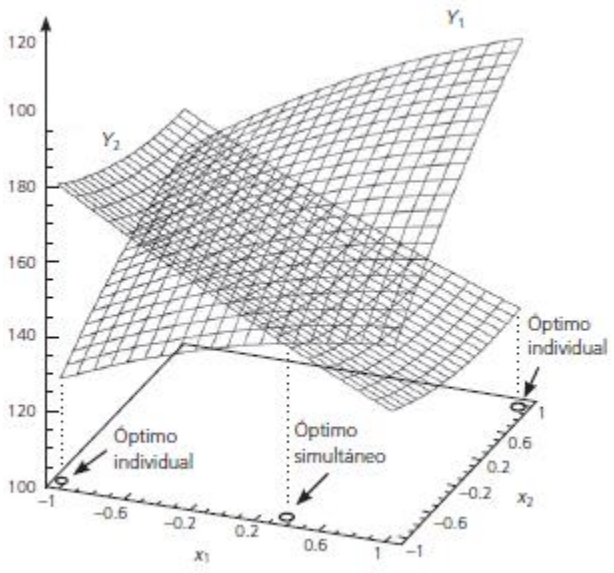

Figura 17. Superficie de Respuesta Múltiple en una misma región experimental, figura tomada de Gutiérrez y de la Vara, 2012

Se presentan dos métodos de optimización simultánea, uno gráfico y el otro analítico, basado en una función de deseabilidad. Ambos métodos proveen soluciones consistentes tanto con los datos observados en el experimento como con la información que se les provee. Asimismo, son intuitivos y flexibles en el sentido que permiten balancear de diversas formas la importancia relativa de las respuestas. Para aplicar estos métodos es importante contar con el software apropiado.

Para ambos casos se supone que el comportamiento de cada una de las m respuestas a optimizar se describe adecuadamente por un modelo de segundo orden en términos de estos k factores de control. Es decir, para empezar la optimización simultánea se deben tener los m modelos estimados, dados por:

$$\hat{Y}_l = \hat{\beta}_{0l} + \sum_{i=1}^{k} \hat{\beta}_{il} x_i + \sum_{i=1}^{k} \hat{\beta}_{iil} x_i^2 + \sum_{i=1}^{k} \sum_{<j=1}^{k} \hat{\beta}_{ijl} x_i x_j \,, \qquad l = 1, 2, \ldots, m$$

No necesariamente tienen que ser modelos cuadráticos completos, pero se recomienda que sean modelos jerárquicos, es decir, por cada interacción o término cuadrático en el modelo, éste incluya los términos más simples que se pueden

formar con los factores involucrados. Es preciso verificar que cada modelo cumpla con los supuestos tradicionales de normalidad, la varianza constante y la independencia de los residuos, y que el coeficiente de determinación de cada modelo sea de al menos el 70%. Tanto el método gráfico como el método de la función de deseabilidad requieren del conocimiento de especificaciones para cada una de las variables de respuesta.

El método de análisis de deseabilidad fue propuesto por Harrington en 1965 y después fue mejorado por Derringer y Suich 1980 y Derringer 1994. Consiste en definir una función en el espacio de factores que estima la deseabilidad global (DG) del producto en cada punto; de tal forma que convierte el problema de optimización multivariado en un problema de optimización univariado. Basta maximizar DG para obtener el punto óptimo buscado. Para definir la DG, primero se transforma cada respuesta predicha $\hat{y}_i(x)$ en un valor de deseabilidad individual $d_i(x)$ que cae en el intevalo $[0, 1]$. De esta manera, $d_i(x)$ mide la deseabilidad en el punto $x = (x_1, x_2, \dots, x_k)$ de la variable y_i. La transformación $d_i(x)$ se hace en términos de las especificaciones y del valor objetivo de cada y.

En particular, si la variable y_i tiene por especificaciones inferior y superior a EI_i y ES_i y su valor objetivo o nominal es T_i, se define la transformación d_i como:

$$d_i(x) = \begin{cases} \left[\dfrac{\hat{y}_i(x) - EI_i}{T_i - EI_i}\right]^s, & \text{si } EI_i \leq \hat{y}_i(x) \leq T_i \\ \left[\dfrac{\hat{y}_i(x) - ES_i}{T_i - ES_i}\right]^t, & \text{si } T_i < \hat{y}_i(x) \leq ES_i \\ 0, & \text{si } \hat{y}_i(x) < EI_i \text{ o } \hat{y}_i(x) > ES_i \end{cases}$$

Donde s y t son exponentes que sirven para elegir la forma de la transformación deseada por el experimentador para cada y. Se asignan valores grandes (digamos $s, t \geq 10$) si se quiere que la deseabilidad d_i sólo tome valores grandes cuando $\hat{y}_i$ caiga cerca de su valor objetivo. Se asignan valores pequeños para s y t $(s, t \leq 0.1)$ si se quiere que cualquier valor de $\hat{y}_i$ dentro del intervalo $[EI_i, ES_i]$ sea igualmente deseable (véase la figura 18).

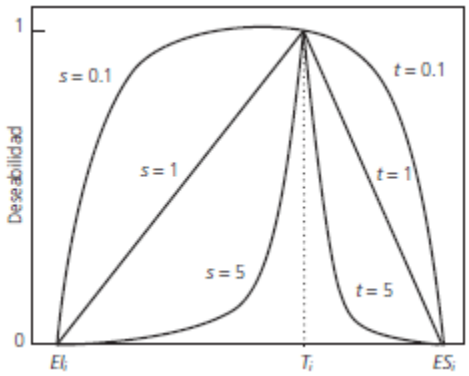

Figura 18. Cambio de la deseabilidad cambiando los valores de s y t en un intervalo (EI_i, ES_i), figura tomada de Gutiérrez y de la Vara, 2012

El valor por omisión de estos exponentes es uno, lo cual sugiere un incremento lineal de la deseabilidad hacia el valor objetivo. Si la respuesta tiene especificaciones de un solo lado, lo que se hace es tomar el valor objetivo T_i igual al valor a partir del cual se considera que no hay ganancia adicional en la calidad de la respuesta. Es decir, en la función dada por la ecuación anterior desaparece una de las restricciones y la figura anterior se reduce a uno de los lados con respecto al valor objetivo T_i.

Una vez calculadas la m deseabilidades individuales sobre el punto x, la deseabilidad global (DG) de x es definida por la media geométrica ponderada:

$$DG(x) = \left(d_1^{w_1} \times d_2^{w_2} \times \ldots \times d_m^{w_m}\right)^{\frac{1}{\sum w_i}}$$

Donde los pesos w_i son constantes que permiten balancear la importancia relativa de cada variable; mientras más grande es el peso dado a una variable con respecto a las restantes, más grande será la exigencia para que el punto óptimo global beneficie a tal variable. Si todas son igualmente, $w_i = 1$ para $i = 1, 2, \ldots, m$, y la DG toma la forma siguiente:

$$DG = \sqrt[m]{d_1 \times d_2 \times \ldots \times d_m} = (d_1 \times d_2 \times \ldots \times d_m)^{\frac{1}{m}}$$

Si d_i es igual a uno significa que la respuesta correspondiente predicha $\hat{y}_i$ toma su valor máximo deseable. Si $d_i = 0$, la respuesta $\hat{y}_i$ predice un valor inaceptable y en este caso la deseabilidad global es cero ($DG = 0$), lo cual significa que todo producto es inaceptable, independientemente de los valores de las respuestas restantes. Esto último explica el uso de la media geométrica en la deseabilidad global DG

Nótese que los exponentes s y t definidos en la transformación de la deseabilidad se pueden introducir como parte de los pesos w_i, pero es importante elegir los exponentes y los pesos de manera separada, ya que los primeros definen la forma de la función de deseabilidad que se quiere para cada respuesta individual y los segundos precisan la importancia relativa entre las respuestas. El punto óptimo simultáneo es el punto $x_0 = (x_{10}, x_{20}, \dots, x_{k0})$ sobre el cual la función $DG(x_0)$ es máxima. Para encontrar este máximo se recurre a algún método numérico.

Una desventaja del método gráfico y del método de deseabilidad es que no consideran la aleatoriedad de los predichos $\hat{y}_i$, lo cual puede causar que dos respuestas con deseabilidad similares no predigan la misma cantidad de producto fuera de especificaciones en los intervalos de predicción correspondientes.

Capítulo 3 Objetivos e Hipótesis

3.1 Objetivos

Minimizar los tiempos de finalización de todos los trabajos a procesar y los tiempos de ocio en las secuencias de trabajos de talleres de flujo continuo utilizando un Algoritmo Genético calibrado por la Metodología de Superficie de Respuesta para obtener mejoras en los resultados de hasta 5%.

De forma matemática el objetivo se escribe con las siguientes expresiones matemáticas:

$$Función\ Objetivo\colon\quad Min\, C_{max} = C(J_{n,m})$$

Sujeto a:

$$\sum_{j=1}^{n} z_{j,i} = 1, \qquad 1 \le i \le n$$

$$\sum_{i=1}^{n} z_{j,i} = 1, \qquad 1 \le j \le n$$

$$s_{1,1} = 0$$

$$s_{1,i} + \sum_{j=1}^{n} t_{1,j} z_{j,i} = s_{1,i+1}, \qquad 1 \le i \le n-1$$

$$s_{r,1} + \sum_{j=1}^{n} t_{r,j} z_{j,1} = s_{r+1,1}, \qquad 1 \le r \le m-1$$

$$s_{r,i} + \sum_{j=1}^{n} t_{r,j} z_{j,i} = s_{r+1,i}, \qquad 1 \le r \le m-1, \qquad 2 \le i \le n$$

$$s_{r,i} + \sum_{j=1}^{n} t_{r,j} z_{j,i} = s_{r,i+1}, \qquad 2 \le r \le m, \qquad 1 \le i \le n-1$$

$$z_{j,i} \in \{0,1\}, \qquad 1 \le j \le n, \qquad 1 \le i \le n$$

$$s_{r,i} \ge 0, \qquad 1 \le r \le m, \qquad 1 \le i \le n$$

$$t_{i,j} \geq 0 \; Tiempos\ de\ proceso\ por\ estaci$$

Siendo $C(J_{n,m})$ el Tiempo Total de Proceso de la tarea i ubicado en la última máquina del proceso de producción. Una variable de decisión binaria $z_{j,i}$ toma el valor de 1 si el trabajo j es asignado en la posición $i \in J$ en la secuencia, y 0 de otra manera. En adición, la variable de decisión no negativa $s_{r,i}$ indica el tiempo de inicio del trabajo en posición i en la máquina r. $t_{i,j}$ es el tiempo de procesamiento de la tarea i en la máquina j.//

Objetivo Específico

- Comparar diferentes operadores utilizados en el Algoritmo Genético para definir los operadores con mejores resultados en los problemas de secuenciación de tareas en fábricas de producción intermitente.

3.2 Hipótesis

Utilizando la Metodología de Superficie de Respuesta para calibrar los parámetros de un Algoritmo Genético que genera y evalúa respuestas en problemas de secuenciación de tareas dentro de las operaciones en fábricas de producción intermitente se obtienen resultados con mejoras no mayores al 5%.

Capítulo 4 Procedimiento de Investigación

Dentro de las actividades realizadas para la obtención del algoritmo genético con parámetros óptimos para la solución del problema de secuenciación en talleres de flujo continuo son las siguientes:

1. Programación del algoritmo para la obtención de la población inicial con alelos aleatorios y codificación por permutación. Se presenta el código generado con MATLAB en el Anexo 1 – A.
2. Programación de los algoritmos de selección: selección por ruleta y selección por torneo (Anexos 2-A y 2-B).
3. Programación de los algoritmos de recombinación: recombinación de un solo punto, recombinación por emparejamiento parcial, recombinación cíclica, recombinación por orden (Anexos 3-A, 3-B, 3C y 3D).
4. Programación de los algoritmos de mutación: mutación por intercambio y recombinación por inserción que se encuentran en los anexos 4-A y 4-B.
5. Comparación de 16 posibles casos que se obtienen de los tipos de selección, recombinación y mutación:
    a. Selección: ruleta y torneo.
    b. Recombinación: Emparejamiento parcial, cíclica, de orden y de un solo punto.
    c. Mutación: Inserción e intercambio.
6. Selección de los tres mejores algoritmos con base en tres criterios y una función objetivo.
    a. Criterios: media, desviación estándar y el mínimo obtenido.
    b. Función objetivo: makespan (Programación mostrada en el anexo 5-A).
7. Selección del mejor algoritmo con base en seis criterios y dos funciones objetivo.
    a. Criterios: media, desviación estándar y el mínimo obtenido; de cada función objetivo.

b. Funciones objetivo: makespan y tiempo ocioso de las estaciones/máquinas (Programación mostrada en el anexo 5-B).

8. Optimización de los parámetros del algoritmo utilizando la metodología de superficie de respuesta para una sola variable.
9. Optimización de los parámetros del algoritmo utilizando la metodología de superficie de respuesta para dos variables, utilizando la función de deseabilidad.

Capítulo 5 Resultados

Como se mencionó en el marco teórico se realizaron las operaciones de selección, recombinación y mutación dentro del algoritmo genético. Para la generación de la población inicial se generaron las posiciones de los vectores de forma aleatoria, sin repetir ningún número natural dentro de lo que se denomina cromosoma. Se utilizaron dos operadores de selección, cuatro de recombinación y uno de mutación. Los cuales se numeran en la tabla 1. En la primera columna se muestran las tres operaciones realizadas en el AG, en la segunda columna se mencionan los operadores desarrollados para esas operaciones y en la tercera columna se muestra la cantidad de operadores utilizados por operación.

Tabla 3. Operadores de selección, recombinación y mutación utilizados.

Operación	Operador	Cantidad
Selección	Ruleta y Torneo	2
Recombinación	Un Punto, PMX, OX y CX	4
Mutación	Inserción e Intercambio	2

De estos operadores se generan 16 formas posibles de combinarlos. Para comparar esas 16 combinaciones se resolvieron 29 problemas; utilizando como criterio el tiempo de terminación de todos los trabajos y se generaron 8 réplicas por problema, generando un total de 3 712 corridas experimentales en una máquina con procesador Intel(R) Core(TM) i3-6006U CPU 2.00 GHz, memoria RAM 4.00 GB y el software MATLAB. Al comparar las posibles combinaciones de acuerdo con tres criterios (media, mínimo y desviación estándar) se obtienen los resultados mostrados en la figura 19. En las etiquetas del eje de las abscisas se muestran abreviaciones que constan de cinco letras, las primeras dos letras corresponden a las letras iniciales de la palabra “Algoritmo Genético”, la tercera letra indica el operador de selección (Torneo, y Ruleta), la cuarta letra aborda el operador de recombinación utilizado (Un punto, PMX, CX y OX) y la última letra muestra el operador de mutación (Inserción e Intercambio).

Figura 19. Comparación de las combinaciones de operadores utilizados en el AG.

La combinación que salió victoriosa es la que utiliza los operadores de selección por torneo, recombinación de un punto y mutación por intercambio. Aunque se tomaron las tres mejores combinaciones de operadores y se agregaron tres criterios de decisión más basados en el tiempo ocioso de las máquinas y/o estaciones. Para esto se resolvieron 22 problemas de la literatura. Se generaron 15 réplicas por problema y por combinación. En total, se hicieron 990 corridas experimentales tomando seis criterios de selección (media makespan, mínimo makespan, desviación estándar makespan, media tiempo ocioso, mínimo tiempo ocioso y desviación estándar tiempo ocioso). La figura 20 muestra los resultados en términos de makespan de los 22 problemas resueltos con los tres algoritmos. Los algoritmos se ven muy parejos en los mínimos obtenidos y si se tomara una decisión sólo con la forma gráfica no se podría decidir con certeza. La figura 21 muestra los mismos 22 problemas pero ahora con el tiempo ocioso como función objetivo. En los problemas 22 y 21 se muestra una diferencia considerable del algoritmo “AGTUS” con los otros dos. Aunque no es posible determinar de forma visual cuál de las tres combinaciones es la mejor. Tomando en cuenta los criterios ya mencionados, la combinación que obtuvo mejores resultados fue la “AGTUS”. Por lo tanto, se decide tomar esa combinación para optimizar los factores que influyen en los resultados del AG.

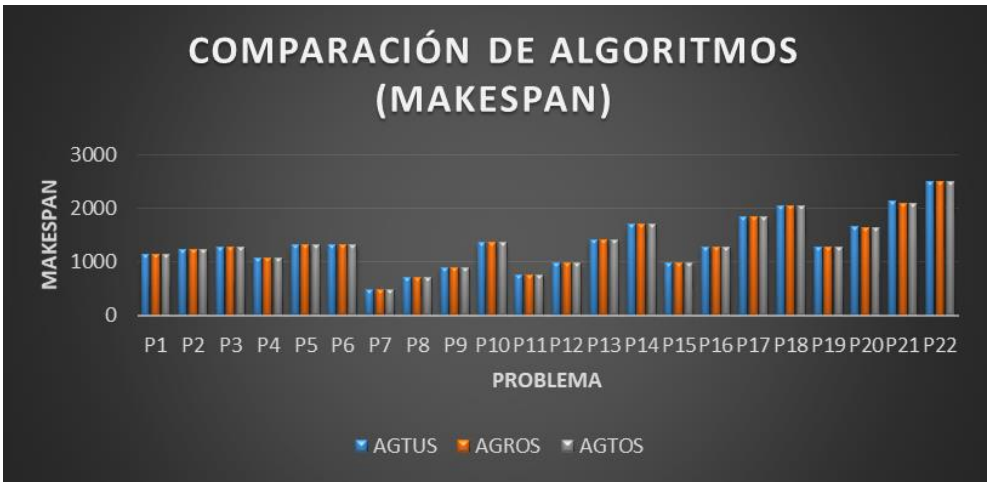

Figura 20. Comparación de las tres mejores combinaciones (makespan).

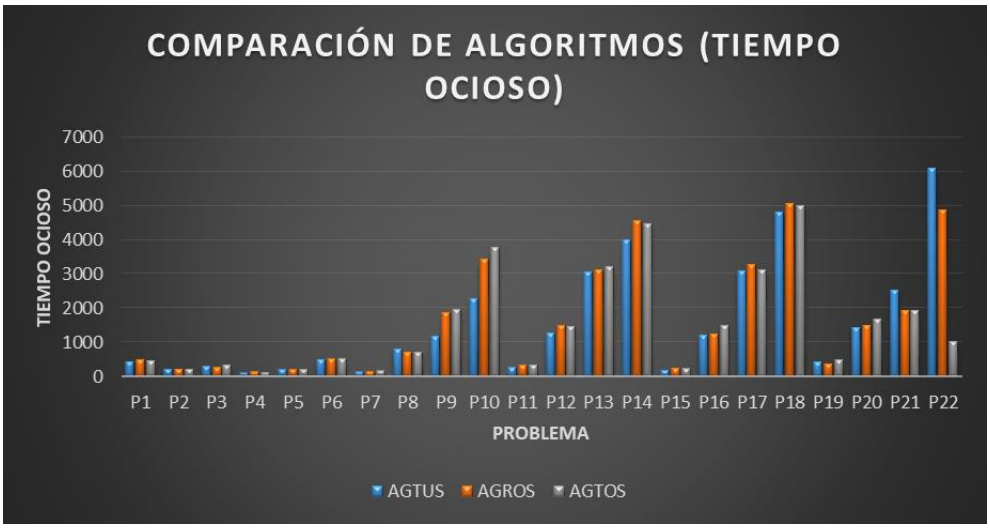

Figura 21. Comparación de las tres mejores combinaciones (tiempo ocioso)

La optimización de los factores se realizó para respuestas múltiples. Se utilizó la Metodología de Superficie de Respuesta. Primero se realizó un diseño de experimentos factorial 2^4 utilizando como respuesta el makespan. Los cuatro factores considerados son las probabilidades de recombinación y de mutación, la cantidad de ciclos y el tamaño de la población inicial. El análisis de varianza se muestra en la tabla 4. De acuerdo con los valor-P se tienen dos factores significativos: factor mutación y el de ciclos. Se toma el factor de ciclos porque no

está muy alejado de 0.05 que es el valor común para tomar la decisión de aceptar o rechazar la hipótesis nula.

Tabla 4. Análisis de Varianza para el Makespan

Fuente	*Suma de Cuadrados*	*Gl*	*Cuadrado Medio*	*Razón-F*	*Valor-P*
A:Poblacion_inicial	0.03	1	0.03	0.00	0.9930
B:Ciclos	1314.61	1	1314.61	3.44	0.0700
C:Factor_Cruza	655.641	1	655.641	1.72	0.1967
D:Factor_Mutacion	3072.0	1	3072.0	8.04	0.0067
AA	45.2256	1	45.2256	0.12	0.7324
AB	26.2813	1	26.2813	0.07	0.7943
AC	64.4113	1	64.4113	0.17	0.6833
AD	1046.53	1	1046.53	2.74	0.1047
BB	181.576	1	181.576	0.48	0.4941
BC	345.845	1	345.845	0.90	0.3464
BD	340.605	1	340.605	0.89	0.3500
CC	317.731	1	317.731	0.83	0.3666
CD	25.92	1	25.92	0.07	0.7957
DD	1.26563	1	1.26563	0.00	0.9544
Falta de ajuste	7308.56	10	730.856	1.91	0.0671
Error puro	17966.1	47	382.259		
Total (corr.)	32712.4	71			

En la figura 22 se muestra el análisis de Pareto para los 4 factores y las interacciones dobles de los mismos. Se puede comprobar que el factor de mutación es significativo y el factor de ciclos está muy cerca de serlo. El factor de ciclos está dentro de un bloque de factores conformado por el factor de ciclos, la interacción del factor de mutación y el tamaño de población. Por lo tanto, se puede decir que el modelo se conforma por los factores sin interacción (se toma también el factor de tamaño de la población por jerarquía) y por la interacción de la mutación con la población inicial. Los demás factores se eliminan del modelo.

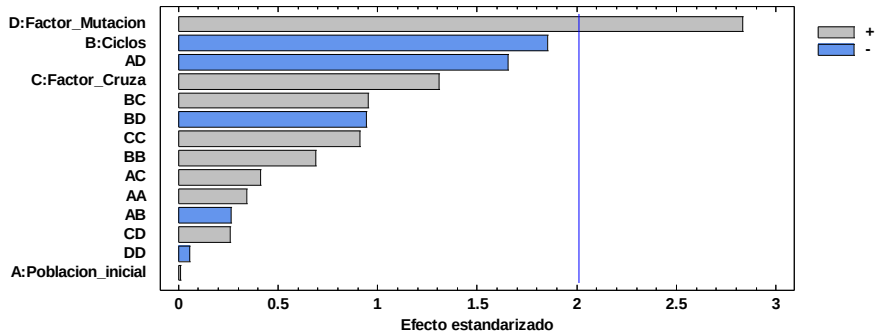

Figura 22. Diagrama de Pareto Estandarizado para Makespan

Al eliminar los factores que no son significativos se analiza el modelo. La figura 23 muestra el efecto que tienen los factores sin interacción con la respuesta. El objetivo del AG es minimizar el makespan. De acuerdo con el objetivo se tiene que a cualquier nivel de cantidad de población inicial se obtiene un menor makespan, a mayor cantidad de ciclos se obtiene un menor makespan y a menores son los factores de recombinación y de mutación se obtiene menor makespan.

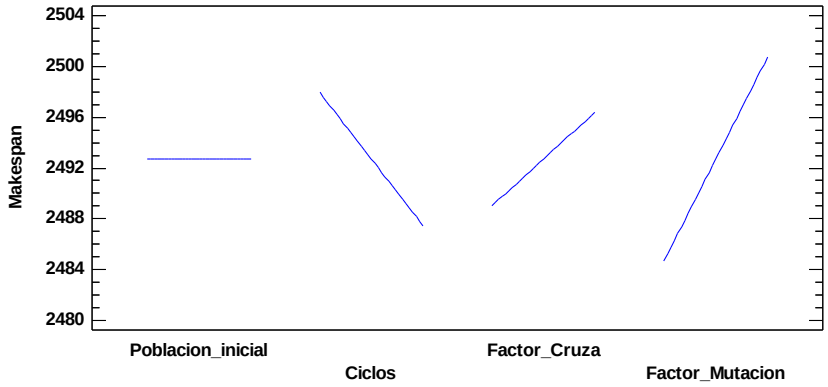

Figura 23. Efectos principales del modelo utilizado para el makespan

En la figura 24 se muestra la interacción doble considerada. Y se observa que al trabajar la población inicial con un nivel alto y la mutación con un nivel alto se obtienen mejores resultados para el makespan. Con el análisis de las figuras 5 y 6 se concluye que los niveles para los factores son: Población inicial (alto), Mutación (bajo), Ciclos (alto) y recombinación (bajo).

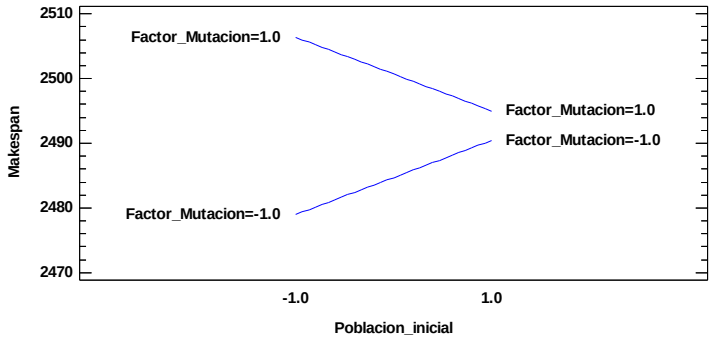

Figura 24. Interacción doble entre población inicial y el factor de mutación

Los residuos muestran una distribución normal al apegarse a la recta de la distribución normal teórico mostrada en la figura 25. Con esto se muestra que los residuos se distribuyen de forma normal. La superficie de respuesta para esta respuesta se encuentra en la figura 26. Se grafican sólo los factores de mutación y recombinación contra la respuesta (makespan) en unidades de tiempo.

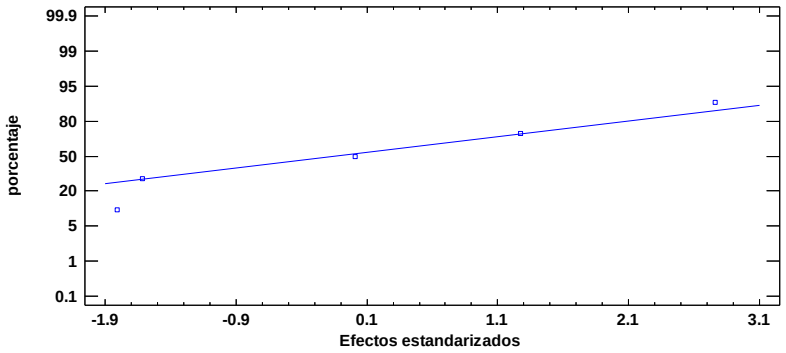

Figura 25. Gráfica de probabilidad normal para el Makespan

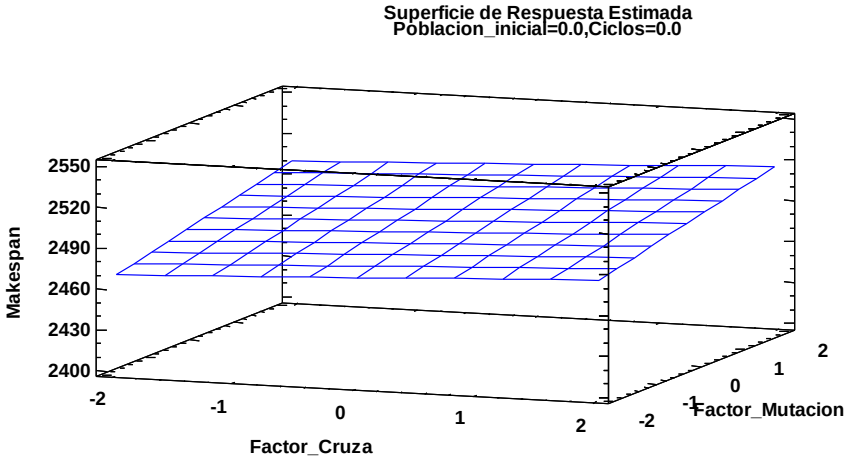

Figura 26. Superficie de respuesta para el makespan

En el tiempo ocioso se realizó; al igual que en el makespan, un diseño factorial 2^4 con puntos axiales y al centro. El análisis de varianza se muestra en la tabla 5. Para el tiempo ocioso se tienen 4 factores significativos; tres más que en el makespan. Los factores significativos son: población inicial, factor de cruza y factor de mutación y la interacción de la población inicial con la población inicial. En este caso los ciclos no son significativos.

Tabla 5. Análisis de Varianza para el tiempo ocioso

Fuente	*Suma de Cuadrados*	*Gl*	*Cuadrado Medio*	*Razón-F*	*Valor-P*
A:Poblacion_inicial	1.42554E6	1	1.42554E6	11.34	0.0028
B:Ciclos	6655.23	1	6655.23	0.05	0.8201
C:Factor_Cruza	1.16333E6	1	1.16333E6	9.26	0.0060
D:Factor_Mutacion	617939.	1	617939.	4.92	0.0372
AA	632144.	1	632144.	5.03	0.0353
AB	128271.	1	128271.	1.02	0.3234
AC	79441.0	1	79441.0	0.63	0.4351
AD	470838.	1	470838.	3.75	0.0659
BB	369816.	1	369816.	2.94	0.1003
BC	74401.5	1	74401.5	0.59	0.4499
BD	77126.3	1	77126.3	0.61	0.4418
CC	42611.3	1	42611.3	0.34	0.5663
CD	239121.	1	239121.	1.90	0.1817
DD	21003.3	1	21003.3	0.17	0.6867
bloques	5383.76	1	5383.76	0.04	0.8379
Falta de ajuste	3.32711E6	34	97856.2	0.78	0.7498
Error puro	2.76522E6	22	125692.		
Total (corr.)	1.1446E7	71			

La figura 27 muestra el análisis de Pareto. Se comprueba lo obtenido en el Análisis de Varianza y puede que el bloque que contiene a las interacciones AD y BB sea tomado en cuenta para el modelo, pero en este caso se determina que no es necesario.

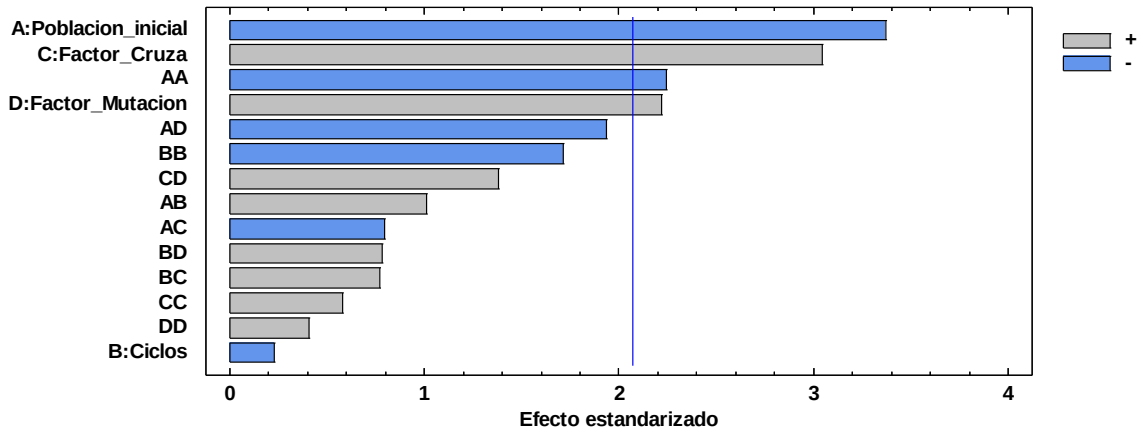

Figura 27. Diagrama de Pareto Estandarizado para tiempo ocioso

El gráfico de efectos principales mostrado en la figura 28 muestra en qué niveles trabajan mejor los efectos principales. En la población inicial se trabaja mejor con

un nivel alto, en los factores de recombinación y de mutación se trabaja mejor con niveles bajos. Se puede ver un efecto de curvatura en la población inicial.

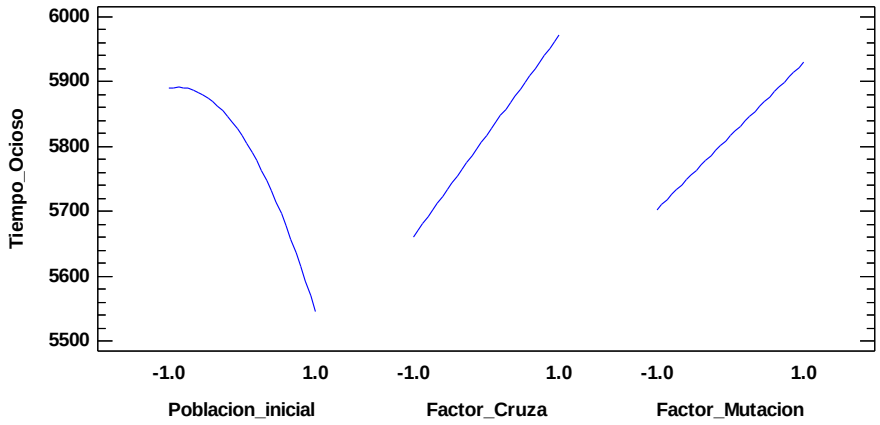

Figura 28. Efectos principales para el tiempo ocioso

La figura 29 muestra la superficie de respuesta para el tiempo ocioso. Puede verse que se tiene curvatura en la superficie, pero es un máximo y lo que se requiere es minimizar el tiempo ocioso porque el tiempo ocioso implica gastos indirectos de fabricación.

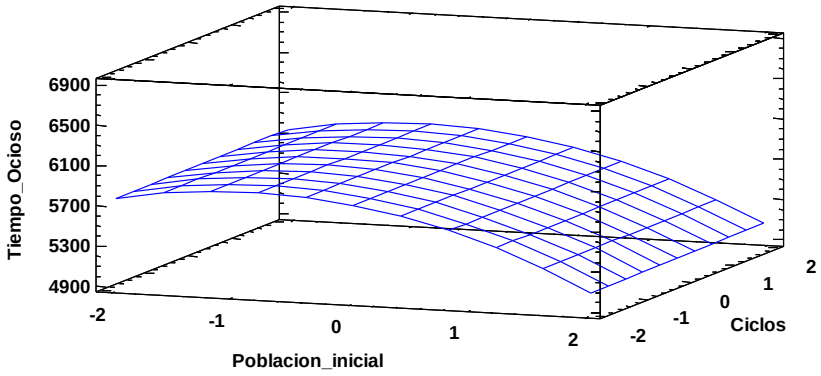

Figura 29. Superficie de Respuesta para tiempo ocioso

Por último, se verifican los supuestos. Con la figura 30 se prueba la normalidad en los residuos y el valor de la prueba de Durbin Watson es mayor a 0.05 (0.8798), por lo que no existe correlación de los residuos.

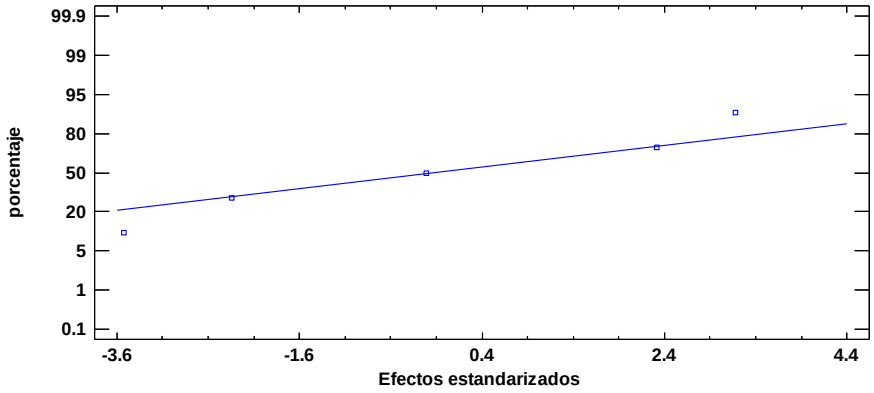

Figura 30. Gráfico de porcentaje contra efectos estandarizados para verificar la normalidad de los residuos

Por último, para calibrar los parámetros del AG selecto se utiliza la Metodología de Superficie de Respuesta para múltiples respuestas utilizando una función de deseabilidad.

La superficie de respuesta para la función de deseabilidad se muestra en la figura 31. La tabla 4 muestra la deseabilidad obtenida y los niveles de los factores con los que se optimiza.

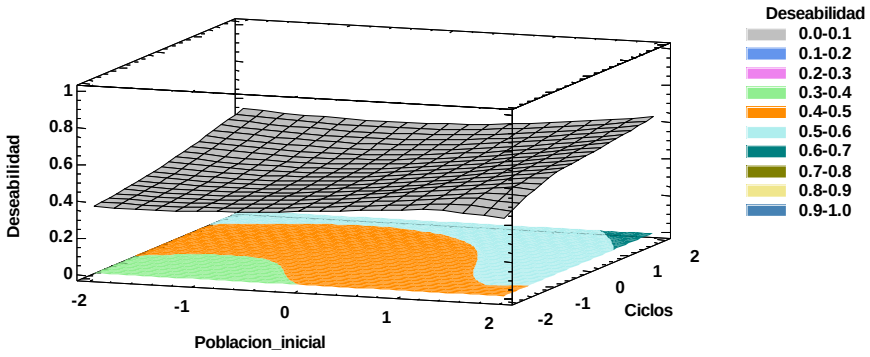

Figura 31. Superficie de Respuesta de la función de deseabilidad.

Tabla 6. Niveles óptimos de los factores

Factor	*Bajo*	*Alto*	*Óptimo*
Poblacion_inicial	-2.0	2.0	-1.99996
Ciclos	-2.0	2.0	0.509986
Factor_Cruza	-2.0	2.0	-0.253375
Factor_Mutacion	-2.0	2.0	-1.99794

Respuesta	*Óptimo*
Makespan	2455.52
Tiempo_Ocioso	4980.0

Conclusiones

Al establecer niveles bajos de población inicial, factor de recombinación y factor de mutación y un nivel alto en la cantidad de ciclos se obtiene makespan y tiempo ocioso óptimos, dos variables de respuesta. Se pueden establecer trabajos a futuro derivados de esta investigación, algunos de los cuales son:

- En el AG se utiliza un bucle for, pero es posible utilizar un bucle $while$ en el cual se tome como condición un error relativo menor a cierto valor establecido y calcular la velocidad con que se converge a la solución.
- Es posible proponer utilizar otras técnicas de optimización multivariable para obtener los parámetros óptimos de operación del AG o un intervalo de los parámetros óptimos tomando en cuenta una muestra representativa de la población.
- Se puede optimizar la función de deseabilidad utilizando una metaheurística con enfoque en datos continuos.
- Emplear la metodología propuesta para encontrar los puntos de operación óptimos en el AG utilizado en el “Flow Shop” para el “Flexible Hybrid Flow Shop”.

Estos son algunos de los posibles trabajos a futuro derivados de esta investigación.

Anexos

Anexo 1-A. Código para la población inicial

```
% Función para la población inicial
%r cantidad de individuos de la poblacion inicial
%t cantidad de trabajos a secuenciar

function [poblacion]=pobinicial(r,t)
    for i=1:r
        prob=rand(1,t);  %Generación de números Aleatorios entre 0 y 1

            k=1;  %Se inicia la variable k que es la que va a colocar los
jobs en orden
            w=0;  %Se inicia la variable w que es el contador del ciclo
while
            while w<t

                mini=10000;   %Número muy grande que se plantea

                for j=1:t
                    if prob(j)<mini
                        mini=prob(j); %Se encontró el menor número
                        l=j; %Extraer la ubicación
                    end
                end

                secuencia(l)=k;  %Colocas el job correspondiente
                prob(l)=1000; %Número muy grande

                k=k+1; %Trabajo k + 1
                w=w+1; %Contador del ciclo while

            end

            poblacion(i,:)=[secuencia];  %Almacenamiento de las
secuencias
     end

pobinicial=poblacion(i,:);
end
```

Anexo 2-A. Código del operador de selección por ruleta

```
%Función para selección por ruleta
%r individuos de la población inicial
%t cantidad de trabajos
%num cantidad de individuos a generar por la selección
%fitness función de aptitud
function [seleccion]=selruleta(r,t,num,fitness,poblacion)

    Sumafitness=0;

    for f=1:r
        Sumafitness=Sumafitness+fitness(f,t); %Se suman los tiempos de
terminación de cada uno de los individuos
    end

    for k=1:num %Ciclo for para generar dos individuos por ruleta
       aleatorio = randi([1 fix(Sumafitness)],1); %Aleatorio entre 1 y el
acumulado
       Sumafitness=0;
       for f=1:r
        Sumafitness=Sumafitness+fitness(f,t); %Se suman los tiempos de
terminación de cada uno de los individuos
         if (Sumafitness>=aleatorio) %Comparación entre la suma acumulada
y el numero generado
            seleccion=poblacion(f,:);
           break
         end
       end

        sel(k,:)=seleccion;
    end

    seleccion=sel;
end
```

Anexo 2-B. Código del operador de selección por torneo.

```
%Función de selección por torneo binario
%r población inicial
%num cantidad de seleccionados
%fitness función de aptitud
%poblacion población inicial

function [sel]=seltorneo(r,num,fitness,poblacion)

    porc=num/r;

    sel=fix(r*porc);

    for s=1:sel
        q=randi([1 r]);
        k=randi([1 r]);
        if fitness(q,end)<=fitness(k,end) %Torneo binario

                seleccionados=poblacion(q,:);

        else

                seleccionados=poblacion(k,:);


        end
        seleccion(s,:)=seleccionados;
    end


    sel=seleccion;

end
```

Anexo 3-A. Código del operador de recombinación de un solo punto

```
%Cruzas
function [cruzas]=unpunto(num,cruza,sel,t)
    for s=1:fix(num/2)
                          q=fix(randi([1 fix(num/2)])); %Número aleatorio
entre 1 y el número de seleccionados para elegir el primer padre
                          r=fix(randi([(fix(num/2)+1) num])); %Número
aleatorio entre 1 y el número de seleccionados para elegir el segundo
padre
                          prob=rand();
                          if prob>=cruza
                              joba=sel(q,:);
                              jobb=sel(r,:);

                              a1=randi([1 (fix(t/2)+1)]); %Número aleatorio
entre 1 y la cantidad de tareas a procesar
                              a2=randi([(fix(t/2)+1) t]); %Número aleatorio
entre 1 y la cantidad de tareas a procesar

                              dif=abs(a1-a2);
                              % subsec1=zeros(1,t);
                              for p=1:t
                                  if p>=a1 && p<=a2
                                     subsecu1(p)=joba(p);
                                  else
                                     subsecu1(p)=0;
                                  end
                              end

                              subsecu1(subsecu1==0)=[];

                              if length(subsecu1)==t
                                  cross=subsecu1;
                              else
                                   for p=1:t
                                      if jobb(p)==subsecu1
                                          subsecu2(p)=0;
                                      elseif jobb(p)~=subsecu1
                                          subsecu2(p)=jobb(p);
                                      end
                                   end
                                  subsecu2(subsecu2==0)=[];
                                  cross=[subsecu1 subsecu2];
                              end

                          else
                              cross=sel(q,:);
                          end
                          cruzas(s,:)=cross;
                          cross=zeros(1, t);
                          subsecu1=zeros(1, t);
                          subsecu2=zeros(t,t);
            end
end
```

Anexo 3-B. Código del operador de recombinación PMX.

```
%PMX en este PMX
function [cruzas]=PMXF(num,cruza,sel,t)
[g,h] = size(sel);
    for s=1:fix(num/2)
            p = randi([1,fix(g/2)]); %Número aleatorio entre 1 y el
número de seleccionados para elegir el primer padre
            q = randi([fix(g/2)+1, g]); %Número aleatorio entre 1 y el
número de seleccionados para elegir el segundo padre
            prob=rand();
            if prob>=cruza
                father_1=sel(p,:);
                father_2=sel(q,:);

                m = randi([2,fix(t/2)-1]); %Primera posición de corte
                n = randi([fix(t/2)+1, t-1]); %Segunda posición de corte

                children_1 = zeros(1,t);
                children_2 = zeros(1,t);

                children_1(m+1:n-1) = father_2(m+1:n-1);  %Colocar los
trabajos entre los dos puntos del padre 2 en el hijo 1
                children_2(m+1:n-1) = father_1(m+1:n-1);  %Colocar los
trabajos entre los dos puntos del padre 1 en el hijo 2

                %hijo_1: toma cada gen del padre (no perteneciente a la
región entre los dos cortes) y se revisa si ya se encuentra en el
cromosoma hijo, en caso que no se encuentre se heredará dicho valor en la
misma posición.toma cada gen del padre (no perteneciente a la región
entre los dos cortes) y se revisa si ya se encuentra en el cromosoma
hijo, en caso que no se encuentre se heredará dicho valor en la misma
posición.
                for i = 1:m
                    if children_1(i) == father_1(i)
                              children_1(i) = 0;
                        else
                            children_1(i)=father_1(i);
                    end

                    for j = (m+1):(n-1)
                        if children_1(i)==children_1(j)
                            children_1(i)=0;
                        end
                    end

                end

                for i = n:t
                    if children_1(i) == father_1(i)
                              children_1(i) = 0;
                        else
                            children_1(i)=father_1(i);
                    end
```

```
                    for j = (m+1):(n-1)
                        if children_1(i)==children_1(j)
                            children_1(i)=0;
                        end
                    end

                end

                %hijo 2: toma cada gen del padre (no perteneciente a la
región entre los dos cortes) y se revisa si ya se encuentra en el
cromosoma hijo, en caso que no se encuentre se heredará dicho valor en la
misma posición.toma cada gen del padre (no perteneciente a la región
entre los dos cortes) y se revisa si ya se encuentra en el cromosoma
hijo, en caso que no se encuentre se heredará dicho valor en la misma
posición.
                for i = 1:m
                    if children_2(i) == father_2(i)
                              children_2(i) = 0;
                        else
                            children_2(i)=father_2(i);
                    end

                    for j = (m+1):(n-1)
                        if children_2(i)==children_2(j)
                            children_2(i)=0;
                        end
                    end

                end

                for i = n:t
                    if children_2(i) == father_2(i)
                              children_2(i) = 0;
                        else
                            children_2(i)=father_2(i);
                    end

                    for j = (m+1):(n-1)
                        if children_2(i)==children_2(j)
                            children_2(i)=0;
                        end
                    end

                end

                %Obtener los trabajos del padre 2 que todavía no contiene
el hijo 1
                for i=1:t
                    for j=1:t
                        if children_1(j)==father_2(i)
                            father_2(i)=0;
                        end
                    end
                end
```

```
                %Obtener los trabajos del padre 1 que todavía no contiene
el hijo 2
                for i=1:t
                    for j=1:t
                        if children_2(j)==father_1(i)
                            father_1(i)=0;
                        end
                    end
                end

                %Colocar los trabajos

                father_1(father_1==0)=[];
                father_2(father_2==0)=[];

                %Colocar los trabajos restantes en las posiciones vacías
hijo 1
                for i=1:t
                    for j=1:length(father_2)
                        if children_1(i)==0
                            children_1(i)=father_2(j);
                            father_2(j)=[];
                        end
                    end
                end

                %Colocar los trabajos restantes en las posiciones vacías
hijo 1
                for i=1:t
                    for j=1:length(father_1)
                        if children_2(i)==0
                            children_2(i)=father_1(j);
                            father_1(j)=[];
                        end
                    end
                end

            else
                children_1 = sel(p,:);
                children_2 = sel(q,:);
            end

            cruzas_1(s,:)=children_1;
            cruzas_2(s,:)=children_2;

    end
    cruzas = [cruzas_1; cruzas_2];
end
```

Anexo 3-C. Código del operador de recombinación CX.

```
%función CX

function [cruzas]=CMXF(num,cruza,sel,t)

    for s=1:fix(num/2)
            [fila, columna] = size(sel);
            p = randi([1,fila/2]); %Número aleatorio entre 1 y el número
de seleccionados para elegir el primer padre
            q = randi([fix(fila/2)+1, fila]); %Número aleatorio entre 1 y
el número de seleccionados para elegir el segundo padre
            prob=rand();
            if prob>=cruza
                father_1=sel(p,:);
                father_2=sel(q,:);

                children_1 = zeros(1,t);
                children_2 = zeros(1,t);


                %Proceso para el ciclo del CMX del padre 1 y el hijo 1
                to_break=0;
                for i=1:t
                    for j=1:t
                        for k=1:t
                            if i==1 %Cuando se inicia el proceso el
cromosoma del padre en la posición 1 es igual al cromosoma del hijo en la
posición 1
                                children_1(i)=father_1(i);
                            elseif children_1(i-1)==father_2(k) & i==2
                                children_1(k)=father_1(k);
                            elseif children_1(j)==father_2(k) & i>2 &
children_1(k)==0 & children_1(j)~=0
                                children_1(k)=father_1(k);
                            elseif children_1(j)==father_2(1) &
children_1(j)~=0
                                to_break=1;
                                break
                            end
                        end
                    end
                        if to_break==1
                            break
                        end
                end

                %Proceso para el ciclo del CMX del padre 2 y el hijo 2
                to_break=0;
                for i=1:t
                    for j=1:t
                        for k=1:t
                            if i==1 %Cuando se inicia el proceso el
cromosoma del padre en la posición 1 es igual al cromosoma del hijo en la
posición 1
```

```
                                children_2(i)=father_2(i);
                            elseif children_2(i-1)==father_1(k) & i==2
                                children_2(k)=father_2(k);
                            elseif children_2(j)==father_1(k) & i>2 &
children_2(k)==0 & children_2(j)~=0
                                children_2(k)=father_2(k);
                            elseif children_2(j)==father_1(1) &
children_2(j)~=0
                                to_break=1;
                                break
                            end
                        end
                    end
                        if to_break==1
                            break
                        end
                end


                %Obtener los trabajos del padre 1 que todavía no contiene
el hijo 1
                for i=1:t
                    for j=1:t
                        if children_1(j)==father_1(i)
                            father_1(i)=0;
                        end
                    end
                end

                %Obtener los trabajos del padre 2 que todavía no contiene
el hijo 2
                for i=1:t
                    for j=1:t
                        if children_2(j)==father_2(i)
                            father_2(i)=0;
                        end
                    end
                end

                %Colocar los trabajos

                father_1(father_1==0)=[];
                father_2(father_2==0)=[];

                %Colocar los trabajos restantes en las posiciones vacías
hijo 1
                for i=1:t
                    for j=1:length(father_1)
                        if children_1(i)==0
                            children_1(i)=father_1(j);
                            father_1(j)=[];
                        end
                    end
                end
```

```
                %Colocar los trabajos restantes en las posiciones vacías
hijo 2
                for i=1:t
                    for j=1:length(father_2)
                        if children_2(i)==0
                            children_2(i)=father_2(j);
                            father_2(j)=[];
                        end
                    end
                end

                else
                children_1 = sel(p,:);
                children_2 = sel(q,:);
                end

                cruzas_1(s,:)=children_1;
                cruzas_2(s,:)=children_2;

        end
        cruzas = [cruzas_1; cruzas_2];
end
```

Anexo 3-D. Código del operador de recombinación OX.

```
%Función de Order Crossover

function [cruzas]=OrderCross(num,cruza,sel,t)
[g, h] = size(sel);
    for s=1:fix(num/2)
            p = randi([1,fix(g/2)]); %Número aleatorio entre 1 y el
número de seleccionados para elegir el primer padre
            q = randi([fix(g/2)+1, g]); %Número aleatorio entre 1 y el
número de seleccionados para elegir el segundo padre
            prob=rand();
            if prob>=cruza
                father_1=sel(p,:);
                father_2=sel(q,:);

                children_1 = zeros(1,t);
                children_2 = zeros(1,t);

                mascara=zeros([1 t]);

                for i=1:t
                    point=rand();
                    if point>=0.5
                        mascara(i)=1;
                    end
                end

                for i=1:t
                    if mascara(i)==1
                        children_1(i)=father_1(i);
                        children_2(i)=father_2(i);
                    end
                end

                %Obtener los trabajos del padre 2 que todavía no contiene
el hijo 1
                for i=1:t
                    for j=1:t
                        if children_1(j)==father_2(i)
                            father_2(i)=0;
                        end
                    end
                end

                %Obtener los trabajos del padre 1 que todavía no contiene
el hijo 2
                for i=1:t
                    for j=1:t
                        if children_2(j)==father_1(i)
                            father_1(i)=0;
                        end
                    end
                end
```

```
                %Colocar los trabajos

                father_1(father_1==0)=[];
                father_2(father_2==0)=[];

                %Colocar los trabajos restantes del padre 2 en las
posiciones vacías hijo 1
                for i=1:t
                    for j=1:length(father_2)
                        if children_1(i)==0
                            children_1(i)=father_2(j);
                            father_2(j)=[];
                        end
                    end
                end

                %Colocar los trabajos restantes del padre 1 en las
posiciones vacías hijo 2
                for i=1:t
                    for j=1:length(father_1)
                        if children_2(i)==0
                            children_2(i)=father_1(j);
                            father_1(j)=[];
                        end
                    end
                end
                else
                children_1 = sel(p,:);
                children_2 = sel(q,:);
                end

                cruzas_1(s,:)=children_1;
                cruzas_2(s,:)=children_2;

        end
        cruzas = [cruzas_1; cruzas_2];
end
```

Anexo 4-A. Código del operador de mutación por intercambio.

```
%Función Mutación por intercambio
%probmuta probabilidad de mutación
%r individuos de la poblacion inicial
%t cantidad de trabajos}
%cruzas vector de vectores de cruza
function [mutas]=swapmuta(probmuta,r,num,t,cruzas)

    muta=zeros(1,t);
    aleat=rand();

    for i=1:num/2
            a=randi([1 t],1);  %Número aleatorio entre uno y el total de
trabajos para elegir la primera posición de muta
            b=randi([1 t],1);  %Número aleatorio entre uno y el total de
trabajos para elegir la segunda posición de muta
            if probmuta<=aleat && a~=b
                for j=1:t
                    if (j~=a && j~=b)
                        muta(i,j)=cruzas(i,j);
                    elseif j==b
                        muta(i,b)=cruzas(i,a);
                    elseif j==a
                        muta(i,a)=cruzas(i,b);
                    end
                end
            else
                muta(i,:)=cruzas(i,:);
            end
            mutas1(i,:)=muta(i,:);
    end

    mutas=mutas1;
end
```

Anexo 4-B. Código del operador de mutación por inserción.

```
%Función Mutación por inserción
%probmuta probabilidad de mutación
%r individuos de la poblacion inicial
%t cantidad de trabajos}
%cruzas vector de vectores de cruza
function [mutas]=insercionmuta(probmuta,r,num,t,cruzas)

    muta=zeros(1,t);
    aleat=rand();

    for i=1:num/2
        a=randi([1 t],1);  %Número aleatorio entre uno y el total de
trabajos para elegir la primera posición de muta
        b=randi([1 t],1);  %Número aleatorio entre uno y el total de
trabajos para elegir la segunda posición de muta7
        c=a+1; %Inserción al lado de a
        if probmuta<=aleat && a~=b
            for j=1:t
                if a<b
                     c=a+1; %Inserción al lado de a
                    if j~=b && j~=c
                        muta(i,j)=cruzas(i,j);
                    elseif j==b
                        muta(i,b)=cruzas(i,c);
                    elseif j==c
                        muta(i,c)=cruzas(i,b);
                    end
                end
                if b<a
                     c=b+1; %Inserción al lado de b
                    if j~=a && j~=c
                        muta(i,j)=cruzas(i,j);
                    elseif j==a
                        muta(i,a)=cruzas(i,c);
                    elseif j==c
                        muta(i,c)=cruzas(i,a);
                    end
                end
            end
        else
            muta(i,:)=cruzas(i,:);
        end
        mutas1(i,:)=muta(i,:);
end

mutas=mutas1;
end
```

Anexo 5-A. Función objetivo “makespan”.

```
%Función de aptitud "makespan"
%r cantidad de individuos de la poblacion inicial
%t cantidad de trabajos a secuenciar
%ma cantidad de estaciones de trabajo
%tiempo matriz de tiempos con r filas y ma columnas
% vector de vectores de la población

function [fitness]=aptitud(r,t,ma,tiempo, poblacion) %Son necesarios
cuatro parámetros que se muestran arriba

    for i=1:r

    acummaq=zeros(1,ma); %Comienzo de los tiempos acumulados de la
máquina
    acumtra=zeros(1,t); %Comienzo de los tiempos de los trabajos



    for j=1:t
        if j==1 %Condición necesaria para el primer trabajo secuenciado
            for m=1:ma %Se generan los tiempos para el primer trabajo
secuenciado
                acumtra(j)=acumtra(j)+tiempo(poblacion(i,j),m); %Se
comienzan a acumular los tiempos de los trabajos que se obtienen de la
matriz de tiempos
                acummaq(m)=acummaq(m)+acumtra(j); %Se comienzan a
acumular los tiempos de las máquinas, solo para el primer trabajo
            end
        else   %Si la fila es mayor a uno se considera lo siguiente
            for m=1:ma  %Se comienza el contador de las máquinas, pero
ahora se considera a partir de la segunda fila
                if acumtra(j)>=acummaq(m) %Condición de los tiempos si el
tiempo del trabajo es mayor o igual altiempo de la maquina se hace
acumula como sigue
                    acumtra(j)=acumtra(j)+tiempo(poblacion(i,j),m);
%Tiempo acumulado del trabajo j-ésimo más el tiempo de la matriz
                    acummaq(m)=acumtra(j); %Se asigna el tiempo de la
máquina igual al tiempo del trabajo
                else %En caso contrario se acumula como sigue
                    acumtra(j)=acummaq(m)+tiempo(poblacion(i,j),m);
                    acummaq(m)=acumtra(j);
                end
            end
        end
    end

    fitness(i,:)=acumtra; %Muestra las terminaciones de cada uno de las
configuraciones
    end


end
```

Anexo 5-B. Función objetivo “tiempo ocioso”

```
%Función objetivo de tiempo ocioso o idletime
%Parámetros de entrada
%t = trabajos
%r = pobladores
%ma = máquinas
%tiempo = matriz de tiempos

function [totalidletime] = idletime(r,t,ma,tiempo, poblacion)
    for i=1:r

        acummaq=zeros(1,ma); %Comienzo de los tiempos acumulados de la
máquina
        acumtra=zeros(1,t); %Comienzo de los tiempos de los trabajos
    %     sumas = zeros(t+1,ma+1);
        for j=1:t
            if j==1 %Condición necesaria para el primer trabajo
secuenciado
                for m=1:ma %Se generan los tiempos para el primer trabajo
secuenciado
                    acumtra(j)=acumtra(j)+tiempo(poblacion(i,j),m); %Se
comienzan a acumular los tiempos de los trabajos que se obtienen de la
matriz de tiempos
                    acummaq(m)=acummaq(m)+acumtra(j); %Se comienzan a
acumular los tiempos de las máquinas, solo para el primer trabajo

                end
                sumas (j,:) = acummaq;
            else   %Si la fila es mayor a uno se considera lo siguiente
                for m=1:ma  %Se comienza el contador de las máquinas,
pero ahora se considera a partir de la segunda fila
                    if acumtra(j)>=acummaq(m) %Condición de los tiempos
si el tiempo del trabajo es mayor o igual altiempo de la maquina se hace
acumula como sigue
                        acumtra(j)=acumtra(j)+tiempo(poblacion(i,j),m);
%Tiempo acumulado del trabajo j-ésimo más el tiempo de la matriz
                        acummaq(m)=acumtra(j); %Se asigna el tiempo de la
máquina igual al tiempo del trabajo
                    else %En caso contrario se acumula como sigue
                        acumtra(j)=acummaq(m)+tiempo(poblacion(i,j),m);
                        acummaq(m)=acumtra(j);

                    end

                end
                sumas(j,:) = acummaq; %Matriz para obtener los tiempos
ociosos



            end

        end
        col_cer=zeros(t,1);
```

```
        fila_ceros = zeros(1,ma+1);
        sumasn = [sumas col_cer; fila_ceros];
          for x=1:ma-1  %Ciclos para obtener la matriz de tiempos ociosos
que son los positivos dentro de ella
              for y=1:t
                 if y>1
                     idle(y-1,x) = sumasn(y,x)-sumas(y-1,x+1);

                 end
              end
          end
          idletime = zeros(t-1,ma-1);
          for x=1:ma-1
             for y=1:t-1
                if idle(y,x)>0
                    idletime(y,x)=idle(y,x);
                end
             end
          end
          idletime=sum(sum(idletime));
          totalidletime(i)=idletime;




    end
end
```

Anexo 6-A. Algoritmo Genético con la siguiente combinación (Selección-Torneo, Recombinación-PMX, Mutación-Intercambio)

```
%Algoritmo Genético cruza PMX, selección torneo e Swap

clear all
clc
close all

%t=jobs
%r=población inicial
%cruza=probabilidad de cruza
%Generación de población inicial
%ma=número de estaciones de trabajo
%tiempo=matriz de tiempos
%p= porcentaje de reemplazos
%
t=20;
ma=5;
% tiempo=xlsread('Casos.xlsx','Ej1','A1:E20');
% tiempo=xlsread('Casos.xlsx','Ej2','A1:E20');
% tiempo=xlsread('Casos.xlsx','Ej3','A1:E20');
% tiempo=xlsread('Casos.xlsx','Ej4','A1:E20');
% tiempo=xlsread('Casos.xlsx','Ej5','A1:E20');
% tiempo=xlsread('Casos.xlsx','Ej6','A1:E20');

% t = 10;
% ma = 5;
%
%
% %Caso 1
% fileID = fopen('VFR10_5_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);
%

% t = 10;
% ma = 10;
%
% %Caso 2
% fileID = fopen('VFR10_10_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=10;
% ma=15;
%
% % Caso 3
% fileID = fopen('VFR10_15_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);
```

```
% t=10;
% ma=20;
%
% %Caso 4
% fileID = fopen('VFR10_20_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=20;
% ma=5;
%
% %Caso 5
% fileID = fopen('VFR20_5_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=20;
% ma=10;
%
% %Caso 6
% fileID = fopen('VFR20_10_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=20;
% ma=15;
%
% %Caso 7
% fileID = fopen('VFR20_15_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=20;
% ma=20;
%
% %Caso 8
% fileID = fopen('VFR20_20_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=30;
% ma=5;
%
% %Caso 9
% fileID = fopen('VFR30_5_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=30;
% ma=10;
```

```
%
% %Caso 10
% fileID = fopen('VFR30_10_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=30;
% ma=15;
%
% %Caso 11
% fileID = fopen('VFR30_15_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=30;
% ma=20;
%
% % Caso 12
% fileID = fopen('VFR30_20_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=40;
% ma=5;
%
% %Caso 13
% fileID = fopen('VFR40_5_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=40;
% ma=10;
%
% %Caso 14
% fileID = fopen('VFR40_10_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=40;
% ma=15;
%
% %Caso 15
% fileID = fopen('VFR40_15_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=40;
% ma=20;
%
% %Caso 16
```

```
% fileID = fopen('VFR40_20_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=50;
% ma=5;
%
% %Caso 17
% fileID = fopen('VFR50_5_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=50;
% ma=10;
%
% %Caso 18
% fileID = fopen('VFR50_10_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=50;
% ma=15;
%
% %Caso 19
% fileID = fopen('VFR50_15_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);
%
% t=50;
% ma=20;
%
% %Caso 20
% fileID = fopen('VFR50_20_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=60;
% ma=5;
%
% %Caso 21
% fileID = fopen('VFR60_5_1_Gap.txt','r');
% formatSpec = '%f';
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

% t=60;
% ma=10;
%
% %Caso 22
% fileID = fopen('VFR60_10_1_Gap.txt','r');
% formatSpec = '%f';
```

```
% sizetime = [t, ma];
% tiempo = fscanf(fileID, formatSpec,sizetime);

t=60;
ma=15;

%Caso 23
fileID = fopen('VFR60_15_2_Gap.txt','r');
formatSpec = '%f';
sizetime = [t, ma];
tiempo = fscanf(fileID, formatSpec,sizetime);

r=100;
num=r;
cruza=0.8;
probmuta=0.2;
% p=input('Defina el porcentaje de individuos a reemplazar:  ');
ciclos=500;
% calidad2=zeros(1,ciclos);
poblacion=pobinicial(r,t);
for h=1:ciclos
%         poblacion1=reemplazos;
        fitness=aptitud(r,t,ma,tiempo, poblacion); %Función de aptitud

        sel=seltorneo(r,num,fitness,poblacion); %Función de selección

        cruzas1=PMXF(num,cruza,sel,t); %Función de cruza
        cruzas=cruzas1;

        mutas=swapmuta(probmuta,r,num,t,cruzas); %Función de mutas

        posreemplazos=[cruzas; mutas]; %posibles reemplazos que es la
combinacion de las mutas y las cruzas

        %Reemplazos
        [q,k]=size(posreemplazos);
        fitness2=aptitud(q,t,ma,tiempo, posreemplazos); %Función de
aptitud para evaluar los posibles reemplazos
        tiempo_ocioso = idletime(q,t,ma,tiempo, posreemplazos);
        cromapt=[max(fitness2')']; %aptitud de cada cromosoma


        for j=1:q
            x=min(cromapt);%posición del valor mínimo
            for i=1:q
               if cromapt(i)==x
                   cromapt(i)=100000;
                   reemplazo(j,:)=posreemplazos(i,:);
                   break
               end

            end
        end

        reemplazos=reemplazo;
```

```
        poblacion1=poblacion;

        for i=1:fix(r/2) %Obtener la mitad de los reemplazos
            reemplazo1(i,:)=reemplazos(i,:);
        end

        for i=1:fix(r/2) %Obtener la mitad de la poblacion anterior
            poblacion2(i,:)=poblacion1(i,:);
        end

        poblacion=[reemplazo1; poblacion2]; %Obtener una nueva población
inicial


       %Resolver lo del algoritmo de Whitley para terminar el primero
        %Para graficar

        maximos(h,:)=max(fitness2');     %vector de los maximos del
fitness2 de la iteración i-ésima
        calidad1(h)=min(maximos(h,:));
        calidad(h)=mean(maximos(h,:));
       calidad2(h) = min(tiempo_ocioso);

end

d=find(calidad(end));
secuenciaelegida=poblacion(d,:);

disp('La secuencia elegida es:  ');
disp(secuenciaelegida);

disp('Con una media del makespan (min) de');
% disp(mean(calidad1));
disp(mean(calidad))

disp('Con una media del tiempo ocioso (min) de: ');
disp(mean(calidad2));
```

Bibliografía

1. Allahverdi, A., Aydilek, H., & Aydilek, A. (2020). No-wait flowshop scheduling problem with separate setup times to minimize total tardiness subject to makespan. *Applied Mathematics and Computation*, 1-16.
2. Andrade, C., Silva, T., & Pessoa, L. (2019). Minimizing flowtime in a flowshop scheduling problem with a biased random-key genetic algorithm. *Expert Systems with Applications*, 67-80.
3. Baker, K., & Trietsch, D. (2009). *Principles of sequencing and scheduling.* New Jersey: John Wiley & Sons.
4. Ben, M., & Al-Fawzam, M. (1998). A tabu search approach for the flow shop scheduling problem. *European Journal of Operational Research*, 88-95.
5. Chen, W., & Hao, Y. (2018). Genetic algorithm-based design and simulation of manufacturing flow shop scheduling problem. *International Journal of Simulation Modelling*, 702-711.
6. Díaz, A. (2009). *Diseño estadístico de experimentos.* Antioquía: Universidad de Antioquía.
7. Engin, O., & Güclü, A. (2018). A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems. *Applied Soft Computing*, 166-176.
8. Gao, J., Chen, R., & Deng, W. (2011). An efficient tabu search algorithm for the distributed permutation flowshop scheduling problem. *International Journal of Production Research*, 641-651.
9. Gooding, W., Pekny, J., & McCroskey, P. (1994). Enumerative approaches to parallel flowshop scheduling via problem transformation. *Computers and Chemical Engineering*, 909-927.
10. Gutiérrez, H., & De la Vara, R. (2012). *Análisis y diseño de experimentos.* México: McGraw Hill.

11. Heizer, J., & Render, B. (2008). *Dirección de la producción y de operaciones. Decisiones tácticas.* Madrid: Pearson Educación.

12. Hillier, F., & Lieberman, G. (2010). *Introducción a la Investigación de Operaciones.* México: McGraw Hill.

13. Jacobs, F., & Chase, R. (2014). *Administración de operaciones. Producción y cadena de suministro.* México: McGraw Hill.

14. Jolai, F., Asefi, H., Rabiee, M., & Ramezani, P. (2013). Bi-objective simulated annealing approaches for no-wait two-stage flexible flow shop scheduling problem. *Scientia Iranica*, 861-872.

15. Jolai, F., Asefi, H., Rabiee, M., & Ramezani, P. (2013). Bi-objective simulated annealing approaches for no-wait two-stage flexible flow shop scheduling problem. *Scientia Iranica*, 861-872.

16. Krajewski, L., Ritzman, L., & Malhotra, M. (2008). *Administración de operaciones; Procesos y cadenas de valor.* México: Pearson Educación.

17. Kramer, O. (2017). *Genetic algorithm essentials.* Oldemburg: Springer.

18. Melab, N., Chakroun, M., Mezmaz, M., & Tuyttens, D. (2012). A GPU-accelerated Branch-and-Bound Algorithm for the Flow-Shop Scheduling Problem. *International Conference on Cluster Computing* (págs. 10-17). Beijing : IEEE.

19. Mirjalili, S. (2019). *Evolutionary algorithms and neural networks. Theory and applications.* Brishbane: Springer.

20. Montgomery, D. (2004). *Diseño y Análisis de Experimentos.* México: Limusa-Wiley.

21. Montgomery, D. (2004). *Diseño y Análisis de Experimentos.* Tempe: Limusa Wiley.

22. Murata, T., Ishibuchi, H., & Tanaka, H. (1996). Multi-objective genetic algorithm and its applications to flowshop scheduling. *Computers and Industrial Engineering*, 957-968.

23. Öztop, H., Fatih, M., Türsel, D., & Pan, Q.-K. (2019). Metaheuristic algorithms for the hybrid flow shop scheduling problem. *Computers and Operations Research*, 177-196.

24. Ponce, P. (2010). *Inteligencia Artificial con aplicaciones a la ingenieria.* México: Alfaomega.

25. Ramos, J. (2001). *Optimización de operaciones en la línea de producción para incrementar la productividad y disminuir el desperdicio.* Monterrey: Universidad de Nuevo León.

26. Ruiz, R., & Vázquez, J. (2009). The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 1-18.

27. Shao, Z., Pi, D., & Shao, W. (2018). Estimation of distribution algorithm with path relinking for the blocking flow-shop scheduling problem. *Engineering Optimization*, 894-916.

28. Sivanandam, S., & Deepa, S. (2008). *Introduction to Genetic Algorithms.* Berlin: Springer.

29. Taha, H. (2012). *Investigación de Operaciones.* México: McGraw Hill.

30. Temiz, I., & Erol, S. (2004). Fuzzy branch-and-bound algorithm for flow shop scheduling. *Journal of Intelligent Manufacturing*, 449-454.

31. Yagmahan, B., & Mutlu, M. (2008). Ant colony optimization for multi-objective flow shop scheduling problem. *Computers and Industrial Engineering*, 411-420.

32. Zheng, X., Zhou, S., Xu, R., & Chen, H. (2019). Energy-efficient scheduling for multi-objective two-stage flow shop using a hybrid ant colony optimisation algorithm. *International Journal of Production Research*, 4103-4120.