SOLUCIÓN DEL PROBLEMA FLOW SHOP HÍBRIDO FLEXIBLE EN MÁQUINAS IDÉNTICAS EMPLEANDO EL ALGORITMO GENÉTICO DE CHU-BEASLEY

YESICA PAOLA MURILLO CORONEL
STACI MARCELA MÉNDEZ CALDERÓN

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
BUCARAMANGA
2016

SOLUCIÓN AL PROBLEMA DE FLOW SHOP HÍBRIDO FLEXIBLE EN MÁQUINAS IDÉNTICAS EMPLEANDO EL ALGORITMO GENÉTICO DE CHU-BEASLEY

YESICA PAOLA MURILLO CORONEL
STACI MARCELA MÉNDEZ CALDERÓN

Trabajo de Grado para optar el título de ingeniero Industrial

Director:
CARLOS EDUARDO DÍAZ BOHORQUEZ
MSc. Ingeniería Industrial

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
BUCARAMANGA
2016

DEDICATORIA

Al niño Jesús de Praga y la Virgen María en su advocación de la Rosa Mística por siempre escuchar mis oraciones.

A mi papi Horacio Murillo que su amor trasciende las fronteras de la vida y la muerte.

A mi mami Paulina Coronel por su esfuerzo diario, apoyo y amor incondicional.

A mis hermanos y mis sobrinos a quienes llevo en mi corazón.

A toda mi familia, especialmente a mi tía Florinda y Pedro por el apoyo en esta etapa universitaria.

A todas las personas que me acompañaron en este camino y lo hicieron más agradable, especialmente a Yeisson.

Yesica Paola Murillo Coronel

.

DEDICATORIA

A Dios por darme la sabiduría, la fortaleza y guiar cada día mi camino. Toda la gloria sea para Él.

A mi padre Eulises Méndez ejemplo de vida y de superación, por su amor, apoyo incondicional y porque siempre se ha esforzado para que en mi casa estemos bien y nunca falte nada.

A mi madre Ligia Calderón por su cariño, amor, comprensión, sacrificio, y sobre todo por creer en mí.

A mis hermanos Julián David y Jonatán Daniel, por ser mi fuente de motivación e inspiración para poder superarme cada día más.

A mi familia quienes con sus oraciones y palabras de aliento estuvieron presentes en mi proceso de formación como ingeniera.

Staci Marcela Méndez Calderón.

AGRADECIMIENTOS

Al profesor MSc. Carlos Eduardo Díaz Bohórquez por su orientación y acompañamiento en el desarrollo de este proyecto.

Al grupo de investigación OPALO.

A nuestros padres y familia.

A nuestros amigos.

Yesica Paola Murillo Coronel *Staci Marcela Méndez Calderón*

TABLA DE CONTENIDO

LISTA DE TABLAS

pág

LISTA DE FIGURAS

pág

LISTA DE ANEXOS

pág

RESUMEN

TITULO: SOLUCIÓN DEL PROBLEMA FLOW SHOP HÍBRIDO FLEXIBLE EN MÁQUINAS IDÉNTICAS EMPLEANDO EL ALGORITMO GENÉTICO DE CHU-BEASLEY*

AUTORES: YESICA PAOLA MURILLO CORONEL, STACI MARCELA MÉNDEZ CALDERÓN**

PALABRAS CLAVE: ALGORITMO GENÉTICO DE CHU-BEASLEY, FLOW SHOP HIBRIDO FLEXIBLE, PROGRAMACIÓN DE PRODUCCIÓN, MAKESPAN.

DESCRIPCIÓN:

El problema de secuenciación de tareas con una configuración productiva tipo Flow Shop Hibrido Flexible, ha sido un tema importante en el área investigativa donde se busca establecer la programación óptima de trabajos en máquinas dentro de un proceso de producción en una industria en general, para esta programación se han empleado diferentes técnicas como lo son las meta-heurísticas, las cuales surgen como alternativa de solución ya que han demostrado aportar resultados de muy buena calidad a problemas complejos, con el fin de que la industria reduzca costos y logre altos niveles de productividad para que las organizaciones mejoren su competitividad. La configuración productiva tipo Flow Shop Hibrido Flexible en máquinas paralelas idénticas que se aborda en esta investigación consiste en la programación de i trabajos los cuales presentan una secuencia lineal a través de k etapas, pero uno o más trabajos puedan saltar una o más etapas durante su procesamiento, es decir, se hace referencia a trabajos que no necesitan ser procesados en todas las etapas del proceso, además en cada etapa existe más de una máquina; también se tiene en cuenta la existencia de tiempos de configuración de las maquinas dependientes de la secuencia de trabajos a procesar. Dadas las características del problema y para solucionarlo de manera óptima se desarrolló e implementó la metaheurística llamada Algoritmo Genético con una pequeña modificación introducida por Chu-Beasley. Para validar el algoritmo se hacen necesarias unas instancias que fueron encontradas en la literatura, y con estas se puede medir tanto la eficiencia computacional como la precisión del algoritmo para obtener los resultados.

* Proyecto de Grado

** Facultad de Ingenierías Físico Mecánicas. Escuela de Estudios Industriales y Empresariales. Director: Msc. Carlos Eduardo Díaz Bohórquez

ABSTRACT

TITLE: SOLVING OF PROBLEM HYBRID FLEXIBLE FLOW SHOP IN MACHINES IDENTICAL USING THE GENETIC ALGORITHM CHU-BEASLEY*

AUTHORS: YESICA PAOLA MURILLO CORONEL, STACI MARCELA MÉNDEZ CALDERÓN**

KEYWORDS: GENETIC ALGORITHM OF CHU-BEASLEY, HYBRID FLEXIBLE FLOW SHOP, PRODUCTION SCHEDULING, MAKESPAN.

DESCRIPTION:

The problem of sequencing tasks with a productive setting type Hybrid Flexible Flow Shop, has been a major issue in the area of research which seeks to establish the optimal scheduling jobs on machines within a production process in an industry in general for this programming have been used different techniques such as meta-heuristics, which arise as an alternative solution as they have proven to provide results of very good quality to complex problems, in order for the industry to reduce costs and achieve high levels of productivity to organizations to improve their competitiveness. The productive configuration type Flow Shop Hybrid Flexible identical parallel machines addressed in this research consists in programming *i* works which have a linear sequence through k stages, but one or more jobs may skip one or more stages during processing, that is to say, referring to jobs that do not need to be processed at all stages of the process, also at each stage there is more than one machine is made; also it takes into account the existence of time-dependent settings machines the job stream processing. Given the nature of the problem and to solve it optimally developed and implemented the metaheuristic called Genetic Algorithm with a small modification by Chu-Beasley. To validate the algorithm some instances were found in the literature are necessary, and these can be measured both computational efficiency and accuracy of the algorithm to obtain the results.

* Thesis Degree Project

** Facultad de Ingenierías Físico Mecánicas. Escuela de Estudios Industriales y Empresariales. Director: Msc. Carlos Eduardo Díaz Bohórquez

INTRODUCCIÓN

La competitividad es, para una organización productiva, la capacidad inmediata y futura de diseñar, producir y vender bienes y servicios cuyos precios y otras cualidades formen un conjunto más atractivo que el de sus competidores,[1] creando así una ventaja competitiva para la organización en términos de: precios, diferenciación del producto, innovación, servicio post-venta, etc. Para enfrentar los retos y suplir las necesidades del mercado, las empresas que consideran necesario crear o mejorar la ventaja competitiva con el fin de asegurar su permanencia en el mercado deben perfeccionar sus sistemas de producción y disponer de herramientas que les permitan obtener mejoras en términos de costo y calidad, que permitan dar respuesta pronta, expedita, flexible y eficiente a las necesidades que la industria impone con gran dinamismo y las cuales definen el rumbo de la industria.

En este entorno las compañías que no gestionan y administran de una forma correcta y oportuna sus recursos tienen dificultades para encontrar estabilidad y permanecer en el mercado, en las empresas industriales o manufactureras uno de los recursos que se deben gestionar acertadamente son las máquinas que se emplean para la transformación de materia prima y que hacen parte del proceso productivo, en este sentido gana importancia la gestión de la producción que implica todo el proceso de planeación desde el largo plazo (planeación estratégica), pasando por la programación táctica o de mediano plazo, y llegando a la programación operativa o de corto plazo, en lo que se conoce como el enfoque jerárquico de la producción, [2] la finalidad de la programación de la

[1] ARIEL SARACHE, William; RAMOS, Rafael y CESPÓN, Roberto. Aplicación de Indicadores para el Diagnóstico de Sistemas de Producción. En *Revista Universidad EAFIT*, [en línea]. 2012, Vol.38, No.126, p. 56-66, [Citado 2016-04-22]. Disponible en: <http://publicaciones.eafit.edu.co/index.php/revista-universidad-eafit/article/view/948>.

[2] DOMINGUEZ MACHUCA, José Antonio, *et al.* Dirección de Operaciones: Aspectos estratégicos. Madrid, España: Mc Graw-Hill, 1995.

producción es la asignación de los recursos que se encuentran disponibles para satisfacer un conjunto de requerimientos y determinar el orden adecuado que genere más beneficios dependiendo de los objetivos que se planteen previamente.

Uno de los sistemas que comúnmente se encuentran en las empresas textiles y ensambladoras es el denominado Flow Shop en donde las máquinas están organizadas de manera que el flujo de trabajos sea unidireccional, en este trabajo se abordará el estudio del FLOW SHOP HÍBRIDO FLEXIBLE en máquinas paralelas idénticas, donde los trabajos *i* siguen presentando una secuencia lineal a través de las etapas *k,* pero uno o más trabajos puedan saltar una o más etapas durante su procesamiento, es decir, se hace referencia a trabajos que no necesitan ser procesados en todas las etapas del proceso, [3] y en cada etapa existe más de una máquina; el cual se acerca a la realidad de numerosas empresas que producen diversos productos o referencias que no necesariamente tienen que ser procesados en todas las máquinas disponibles.

Este tipo de problemas ha sido considerado como NP-HARD[4] y por lo tanto se hace necesario implementar metaheurísticas para encontrar soluciones óptimas en tiempos razonables; en este caso se aplicará una modificación al algoritmo genético propuesta por Chu-Beasley[5] en 1996, atendiendo una de las sugerencias realizadas por Jiménez Morales[6] en el 2012 donde implementa este algoritmo en un sistema Flow Shop Flexible, además esta modificación al

[3] ZANDIEH, M. y KARIMI, N. An adaptive multi-population genetic algorithm to solve the multi-objective group Scheduling problem in Hybrid Flexible Flow Shop with sequence-dependent setup times. En: *Journal of Intelligent Manufacturing.* .2011, Vol.22, No. 6, p 979-989.

[4] PAN, Quan-Ke; RUIZ, Rubén. An estimation of distribution algorithm for lot-streaming flow shop problems with setup times. En *Omega.* [Online]. 2012, Vol. 40, p.166-180. [Citado 2016-04-25]

[5] CHU, P.C. y BEASLEY, J.E..A Genetic Algorithm for the Generalised Assignment Problem. En: *Computers Ops. Res.* [online],1997. Vol. 24, No. 1, p.17-23. [Citado: 2016-17-05].

[6] JIMÉNEZ MORALES, Ángela Patricia. Solución del problema de programación de Flow-shop flexible empleando el algoritmo genético de Chu-Beasley. Pereira, 2012. Tesis (Ingeniería Industrial). Universidad Tecnológica de Pereira. Facultad de Ingeniería Industrial.

algoritmo genético ha sido poco estudiada en nuestro país y de esta manera se realiza un aporte académico y empresarial.

El presente documento se encuentra organizado de la siguiente manera: planteamiento del problema, objetivos, revisión de la literatura, marco teórico, definición del sistema productivo bajo estudio, diseño del modelo matemático, validación del algoritmo, diseño experimental, resultados conclusiones y recomendaciones.

Los aportes de este proyecto son de relevancia para el grupo de investigación OPALO y la escuela de estudios industriales y empresariales en la consolidación y el fortalecimiento de la línea de investigación de programación de operaciones.

Tabla 1. Cumplimiento de objetivos

Cumplimiento de objetivos	
Objetivos específicos	**Numeral relacionado**
Revisar la literatura sobre el problema Flow Shop Híbrido Flexible en máquinas idénticas.	4
Formular el problema a través de un modelo de optimización.	7
Diseñar el algoritmo genético de Chu-Beasley a través de un software de programación para la solución del problema.	8-Anexo 1 y 6
Comparar el algoritmo con instancias encontradas en la revisión de la literatura	9
Realizar un artículo de carácter publicable con los resultados, análisis y conclusiones obtenidas.	Anexo 5

2. PLANTEAMIENTO DEL PROBLEMA

Esta investigación busca estudiar el problema Flow Shop Híbrido Flexible (HFFS), el cual enuncia que un conjunto de n trabajos deben ser procesados en m etapas, donde cada una de las etapas está compuesta por varias máquinas paralelas idénticas, en cada una de las máquinas en cada etapa se puede procesar solo un trabajo a la vez y cada trabajo puede omitir una o más etapas en su ruta de procesamiento[7]; como finalidad se pretende encontrar la programación o secuencia de trabajo adecuada para obtener el mínimo makespan, teniendo en cuenta que el makespan o Cmax hace referencia al tiempo que transcurre entre el inicio del primer trabajo en la primera máquina y la finalización del último trabajo en la última máquina[8]; además se tendrá en consideración aspectos como el tiempo de alistamiento dependiente de la secuencia el cual indica que en cada estación de trabajo se debe realizar una configuración para iniciar el siguiente trabajo teniendo en cuenta cuál es el trabajo realizado con anterioridad en la misma estación.

El sistema productivo objeto de estudio consta de más de dos etapas y los trabajos a realizar tienen en consideración características como los tiempos de alistamiento dependientes de la secuencia por lo tanto el problema enunciado pasa a ser NP-HARD[9], porque la cantidad de soluciones y alternativas que se encuentran tiene un crecimiento exponencial, para estos casos se han creado diversos métodos de solución, en este caso se utilizará una metaheurística basada en el algoritmo genético de Chu-Beasley (AGCB) , el cual es una modificación al

[7] JABBARIZADEH, F.; ZANDIEH, M. y TALEBI, D. Hybrid Flexible Flow shops with sequence-dependent setup times and machine availability constraints. En: *Computers & Industrial Engineering.* [Online]. 2009, Vol.57, p. 949-957. [Citado 2016-04-25].

[8] HOJJATI, S.M.H; SAHRAEYAN, A. Minimizing makespan subject to budget limitation in hybrid flow shop. En: *International Conference on Computers & Industrial Engineering.* [Online]. 2009, p. 18-22. [Citado 2016-04-25].

[9] PAN, Quan-Ke. Op. Cit. p.166-180.

algoritmo genético básico que lo convierten en un algoritmo más eficiente[10], dicho algoritmo será comparado con los resultados obtenidos a través de otras metaheurísticas.

La secuenciación de trabajos en un entorno como el propuesto por el problema HFFS es muy común en la industria, por lo cual se hace importante su análisis con el fin de desarrollar herramientas que permitan a las empresas ofrecer soluciones oportunas a sus problemas de secuenciación y toma de decisiones.

[10] GALLEGO, Ramón; TORO, Eliana y ESCOBAR, Antonio. Técnicas Heurísticas y Metaheurísticas. Universidad Tecnológica de Pereira, Pereira, 2015. p 158.

3. OBJETIVOS

3.1 OBJETIVO GENERAL

Construir un algoritmo genético modificado de Chu-Beasley para la solución del problema de Flow Shop Híbrido Flexible en máquinas idénticas para la reducción del makespan.

3.2 OBJETIVOS ESPECÍFICOS

- Revisar la literatura sobre el problema Flow Shop Híbrido Flexible en máquinas idénticas.

- Formular el problema a través de un modelo de optimización.

- Diseñar el algoritmo genético de Chu-Beasley a través de un software de programación para la solución del problema.

- Comparar el algoritmo con instancias encontradas en la revisión de la literatura.

- Realizar un artículo de carácter publicable con los resultados, análisis y conclusiones obtenidas.

4. REVISIÓN DE LA LITERATURA

José Eduardo Márquez Delgado, Ricardo Lorenzo Ávila Rondón, Miguel Ángel Gómez-Elvira González, Carlos Rafael Herrera Márquez (2012)[11], en el trabajo denominado "Algoritmo genético aplicado al problema de programación en procesos tecnológicos de maquinado con ambiente Flow Shop**"** presentan una metaheurística basada en un algoritmo genético, para resolver problemas de programación de tipo Flow Shop, con el objetivo de minimizar el tiempo de finalización de todos los trabajos.

La solución propuesta se probó con problemas clásicos publicados por otros autores, obteniéndose resultados satisfactorios en cuanto a la calidad de las soluciones encontradas y el tiempo de cómputo empleado.

Se describe el problema a resolver y se asumen las siguientes restricciones:

- Hay solo una máquina de cada tipo.
- Una máquina puede procesar solamente un trabajo en un solo instante.
- Las restricciones tecnológicas son conocidas e invariables.
- No pueden procesarse dos operaciones de un mismo trabajo simultáneamente.
- Cada operación una vez comenzada debe ser completada antes de que otra operación pueda hacerlo en esa misma máquina.
- Cada trabajo incluye una y solo una operación en cada máquina, por lo que todos los trabajos contienen una cantidad de operaciones no mayor al número de máquinas.

[11] MÁRQUEZ DELGADO, José *et al.* Algoritmo Genético aplicado al Problema de programación en procesos tecnológicos de maquinado con ambiente Flow Shop. En *Revista Ciencias Técnicas Agropecuarias* [en línea]. 2012, Vol.21, No. 2, p.70-75.

- Los tiempos de proceso son independientes de la secuencia seguida, lo que excluye tiempos de ajuste en las máquinas según la secuencia de los trabajos considerada o tiempos de transporte entre máquinas.

La solución al Flow Shop Scheduling se presenta mediante el uso de un algoritmo genético simple, donde se seleccionaron 10 instancias de problemas conformados por 20 trabajos en 5 máquinas. El algoritmo desarrollado tiene un buen desempeño, y la solución propuesta mostró resultados de alta calidad, de acuerdo al problema de programación en procesos tecnológicos de maquinado con ambiente Flow Shop.

Rubén Ruiz, José Antonio Vázquez Rodríguez (2010)[12], en el artículo titulado "The Hybrid Flow Shop Scheduling Problem" dan una definición sobre el Flow Shop y Flow Shop Híbrido (HFS), los cuales son comunes en ambientes de fabricación donde un conjunto de *n* trabajos es procesado en una serie de *k* etapas, optimizando una determinada función objetivo. Aunque existen una serie de variantes, todos tienen la mayoría de las siguientes características en común:

- El número de etapas de procesamiento es de al menos 2
- Cada etapa k tiene M^K>1 máquinas en paralelo.
- Todos los trabajos se procesan siguiendo el mismo flujo de producción: etapa 1, etapa 2,…, *k*. Un trabajo puede omitir cualquier número de etapas siempre que se procese en al menos una de ellas.
- Cada trabajo *j* requiere un tiempo de procesamiento p_{jk} en la etapa *k*. Se refiere al procesamiento del trabajo *j* en la etapa *k* como operación *ojk*.

[12]RUIZ, Rubén y VASQUÉZ RODRÍGUEZ, José Antonio. The Hybrid Flow Shop Scheduling Problem. En: *European Journal of Operational Research*. 2010, Vol.205, No.1, p.1-18.

En el problema HFS estándar todos los trabajos y máquinas están disponibles en el momento cero, las máquinas en un momento dado son idénticas, cualquier máquina puede procesar sólo una operación a la vez, cualquier trabajo puede ser procesado por una sola máquina a la vez, los tiempos de alistamiento son insignificantes, la capacidad de almacén entre fases es ilimitada y los datos del problema son determinísticos y conocidos de antemano. Aunque la mayoría de los problemas que se revisan en este artículo no cumplen plenamente con los supuestos mencionados anteriormente, principalmente se diferencian en dos o tres aspectos; el problema estándar sirve como "plantilla" a la cual se le agregarán o quitarán supuestos y restricciones para describir diferentes variantes del problema HFS.

Los problemas se clasifican según su configuración de fabricación (α), restricciones y supuestos (β), y la función objetivo considerada (γ). Luego cada problema es descrito con un triplete α/ β/ γ. El parámetro α define la estructura del taller, incluyendo el número de etapas y el número de características de las máquinas por etapa, α se compone por cuatro parámetros. El segundo elemento β, contiene las listas de restricciones y supuestos, adicionales a los del problema estándar. Finalmente realizan una revisión literaria del problema, con tres clases de soluciones más utilizadas en rangos: algoritmos exactos, heurísticas y Metaheurísticas.

Yunior C. Fonseca, Yailen Martínez, Ángel E. Figueredo, Luis A. Pernía (2014)[13], en el artículo “Behavior of the main parameters of the Genetic Algorithm for Flow Shop Scheduling Problem”, realizan una investigación que busca resolver el problema de secuenciación de tareas Flow Shop Scheduling (FSSP)

[13]FONSECA-REYNA, Yunior César, *et al*. Influencia de los parámetros principales de un Algoritmo Genético para el Flow Shop Scheduling. *Rev. cuba cienc. Informat.* 2014, Vol.8, No.1, p. 53-59.

para minimizar el tiempo de finalización de todos los trabajos o makespan a través de un algoritmo genético.

Tienen en cuenta la importancia de encontrar parámetros de control óptimos en un algoritmo genético como: factor de reproducción, factor de mutación y tamaño de la población, a partir de estudios previos de otros autores se admiten desempeños aceptables en estos parámetros, sin embargo se determinan otros parámetros que influyen notablemente en los resultados generados por los algoritmos genéticos en los problemas de optimización como son el operador de selección, el operador de cruzamiento y el uso de elitismo en los operadores mencionados anteriormente; por consiguiente, los autores realizan un estudio estadístico que les permite determinar cuál de los operadores mencionados (selección, cruzamiento y aplicación de elitismo) además de la interacción entre los mismos, tiene mayor influencia sobre el comportamiento y el desempeño de un algoritmo genético.

El algoritmo desarrollado analiza los siguientes operadores:

- **Operadores de selección:** ruleta, torneo, ranking, uniforme, muestreo determinístico y muestreo estocástico.
- **Operadores de cruzamiento**: un punto de cruce, dos puntos de cruce, tres puntos de cruce y Partial Matched Crossover (PMX)
- **Operador de mutación:** de orden

Aplicaron el test de Friedman con el fin de detectar diferencias entre las posibles combinaciones y encontraron que la combinación resultante Dos puntos de cruce-Ranking con Elitismo es la que arroja los mejores resultados al problema estudiado. Finalmente realizaron una aplicación del algoritmo genético con los

parámetros de mayor influencia a problemas clásicos propuestos en la literatura encontrándose que el algoritmo genético obtiene resultados de alta calidad comparados con el conjunto de casos (32 instancias de problemas) propuestos por Erick Taillard con los cuales fue comparado el algoritmo.

Huaping Chen, Shengchao Zhou, Xueping Li, Rui Xu (2014)[14], en el artículo denominado "A Hybrid Differential Evolution Algorithm for a Two-Stage Flow Shop on Batch Processing Machines with Arbitrary Release times and Blocking", analizan la programación de dos BPMs (máquinas de procesamiento por lotes).

Las BPMs se usan comúnmente en instalaciones de fabricación electrónica en las cuales se reúnen placas de circuito impreso (PCB) para productos electrónicos de consumo donde se utilizan dos cámaras para realizar una serie de pruebas de estrés ambiental (ESS), es importante resaltar que el tiempo de procesamiento de un lote está dado por el mayor tiempo de procesamiento entre todos los PCB en ese lote; los lotes de PCB no pueden esperar entre las dos cámaras por lo tanto un lote debe permanecer dentro de la primera cámara bloqueando esta si la segunda cámara se encuentra ocupada. La programación de BPMs incurre en dos decisiones: formar lotes y secuenciarlos; las cuales son interdependientes, porque el tiempo de procesamiento del lote depende de los trabajos en el lote.

El problema se resuelve a través del algoritmo HDDE (Hybrid Discrete Differential Evolution) en el cual los individuos en la población son representados como secuencias de trabajo discreto y la generación de nuevos candidatos se basa la aplicación de operadores de mutación y crossover, se resalta que en el algoritmo

[14] CHEN, Huaping, *et al.* A Hybrid Differential Evolution Algorithm for a Two-Stage Flow Shop on Batch Processing Machines with Arbitrary Release Times and Blocking. En *International Journal of Production Research*. 2014, Vol.52, No.19, p. 5714-5734.

está inmerso un método de búsqueda local para aumentar la capacidad de explotación.

El algoritmo propuesto por los autores fue comparado con un algoritmo genético y con un algoritmo de recocido simulado para lo cual se concluye que el algoritmo HDDE demostró la superioridad en calidad de la solución, robustez y tiempo de la ejecución respecto a los otros dos.

Mohammad Farahmand-Mehr, Parviz Fattahi, Mohammad Kazemi, Hassan Zarei, Ali Piri (2014)[15], en el artículo " An efficient genetic algorithm for a hybrid flow shop scheduling problem with time lags and sequence-dependent setup time" presentan el problema de programación Flow Shop Híbrido con un nuevo enfoque considerando los lapsos de tiempo y los tiempos de configuración dependientes de la secuencia para llevarlos a situaciones reales que se encuentran en las industrias como la de procesamiento de alimentos, química, textil, metalúrgica y fabricación de automóviles.

Una de las características analizadas son los lapsos de tiempo que existen entre el final de la producción de un trabajo en una etapa y el comienzo del procesamiento del trabajo en la siguiente etapa. Un lapso de tiempo indica la duración que debe existir entre dos operaciones sucesivas de un trabajo, teniendo en cuenta que después de completar el procesamiento de un trabajo y antes de iniciar el procesamiento del siguiente trabajo, algún tipo de configuración debe realizarse en una máquina. El intervalo de tiempo que se requiere para la configuración de una máquina para procesar un trabajo, depende tanto del anterior

[15] FARAHMAND-MEHR, Mohammad *et al.* An efficient genetic algorithm for a hybrid Flow shop scheduling problem with time lags and sequence-dependent setup time. En *The International Journal of Advanced Manufacturing Technology*. 2014, Vol.63, p. 337-348.

como del trabajo actual que está siendo procesado, así que los tiempos de configuración son dependientes de la secuencia.

Se formula el problema como un modelo de programación lineal entera, basándose en el problema de vendedor viajero y es solucionado por el método de Branch and Bound. Adicionalmente para evaluar su efectividad y eficiencia se proponen métodos heurísticos, como Johnson, SPT y Palmer; y metaheurísticos como un algoritmo genético; concluyendo que Johnson es el algoritmo preferido entre las otras heurísticas y GA tiene el mejor rendimiento entre todas las demás heurísticas.

Eduardo Salazar, Rene A. Sarzuri (2014)[16], en el artículo "Improved Genetic Algorithm for Total Tardiness Minimization in a Flexible Flow Shop with Sequence Dependent Setup Times" , analizan el problema Flow Shop Flexible (FFS) con tiempos de preparación dependientes de la secuencia a través de un algoritmo genético mejorado en el cual se realiza la generación de la población inicial utilizando vecindades de las heurísticas EDD (Earliest Due Date) y Slack (Holgura); además se implementa un algoritmo de búsqueda de vecindad IP con el fin de mejorar el rendimiento de los algoritmos.

Este estudio se basa en una investigación realizada por Salazar y Figueroa en el 2012 denominada "Minimización de la Tardanza para el Flow Shop Flexible con Setup utilizando Heurísticas Constructivas y un Algoritmo Genético", en el cual fue comparado el algoritmo genético simple con las heurísticas EDD y Slack; como resultado se encontró que las heurísticas EDD y Slack proporcionaban mejores resultados que el algoritmo genético simple a pesar de ser métodos heurísticos simples.

[16] SALAZAR HORNIG, Eduardo y SARZURI GUARACHI, René A. Improved Genetic Algorithm for Total Tardiness Minimization in a Flexible Flow Shop with Sequence-Dependent Setup Times. En *Ingeniare. Rev. chil. ing*. 2015, Vol.23, No.1, p. 118-127.

Los resultados obtenidos mostraron que los algoritmos AG_EDD y AG_SLACK presentan un mejor rendimiento que las heurísticas EDD y SLACK, y el algoritmo genético básico presenta el peor desempeño de todos. Con el fin de mejorar la solución del algoritmo genético se implementa un algoritmo de búsqueda en vecindad IP a los algoritmos AG_EDD Y AG_SLACK que mejora la solución final obtenida para cada estrategia. Al integrar el algoritmo de búsqueda en vecindad IP se observó que los resultados mantienen el mismo orden siendo AG_EDD Y AG_SLACK quienes presentan el mejor desempeño comparado con los demás métodos.

Marcelo Seido Nagano, Hugo Hissashi Miyata, Daniella Castro Araujo (2014)[17], en la investigación llamada "A constructive heuristic for total flow time minimization in a no-wait flow shop with sequence-dependent setup times" abordan el problema de la programación de trabajos Flow Shop sin esperas con secuencia de tiempos de preparación dependientes con el objetivo de minimizar el tiempo de flujo total.

Las heurísticas constructivas BAH y BIH fueron desarrollados para el Flow Shop sin esperas con configuraciones dependientes de la secuencia para el criterio de minimización de tiempos, y la heurística TRIPS fue desarrollado para el Flow Shop sin esperas con tiempos de configuración independientes de la secuencia para el criterio de makespan; TRIPS rompe el problema heurístico en trillizos, reduciendo el esfuerzo computacional requerido para encontrar la solución. Sin embargo, en este artículo, esta heurística es adaptada al problema Flow Shop con configuraciones dependientes de la secuencia realizando unas modificaciones, la principal diferencia es que rompe el problema en cuartetos en lugar de trillizos y se

[17]NAGANO, Marcelo; MIYATA, Hugo y CASTRO, Daniella. A Constructive Heuristic for Total Flow time Minimization in a no –wait Flow shop with Sequence Dependent Setup Times. En *Journal of Manufacturing Systems*. 2015, Vol.36, p. 224-230.

le agrega una fase de construcción adicional, para mejorar su solución. La nueva heurística se llama QUARTS. Al comparar las heurísticas mencionadas se concluye que QUARTS presenta el mejor desempeño y soluciones de alta calidad.

M. Ebrahimi, S.M.T. Fatemi Ghomi, B. Karimi (2014)[18], realizan una investigación denominada ***"***Hybrid Flow Shop Scheduling with Sequence Dependent Family Setup Time and Uncertain Due Dates***"***; en donde estudian el problema de la programación de Flow Shop Híbrido, con máquinas paralelas idénticas, en donde todas las máquinas que se encuentran ubicadas en la misma etapa son iguales y el tiempo de procesamiento de cada trabajo en cada etapa no depende de la máquina que está siendo utilizada.

Tienen en cuenta el problema de la configuración de tiempos dependientes de la secuencia, resaltando como a su vez afecta la minimización del makespan; en la mayoría de los casos estos tiempos de configuración son asimétricos; esto quiere decir que el tiempo de configuración del trabajo i al j es diferente comparado al tiempo de configuración del trabajo j al i. Especialmente, cuando el trabajo procesado se basa en un grupo tecnológico, hay configuración de tiempo dependientes de la secuencia entre grupos de trabajo, todos los tiempos de configuración dependientes de la secuencia se consideran como un básico GT. GT es una teoría gerencial que se propone a partes de un grupo de acuerdo a la similitud de los procesos de producción o características de producción (forma exterior de las partes) o ambas de ellas, es decir, para todas las partes que forman un grupo, se considera solo un tiempo de configuración, así que el tiempo de configuración para todos los trabajos dentro del grupo no se tiene en cuenta,

[18] EBRAHIMI, M.; FATEMI GHOMI. S.M.T y KARIMI, B. Hybrid Flow Shop Scheduling with Sequence Dependt Family Setup Time and Uncertain Due Dates. En *Applied Mathematical Modelling Research* . 2014, Vol.38, No.9-10, p. 2490-2504.

pero es calculado para cada grupo. Con esto, el tiempo total de configuración disminuye y el rendimiento del sistema mejora.

Se plantean dos objetivos para optimizar simultáneamente, la reducción del makespan y la tardanza total; para solucionar el problema propuesto se desarrollan dos algoritmos metaheurísticos basados en el algoritmo genético, estos son: Non-dominated Sorting Genetic Algorithm (NSGAII) y Multi Objective Genetic (MOGA), para evaluar su rendimiento se comparan con Multi-Phase Genetic Algorithm (MPGA) y se encuentra que NSGAII obtiene mejores conjuntos de soluciones en más corto tiempo.

Wen-Chiung Lee, Jen-Ya Wang, Lin-Yo Lee (2015)[19], en el artículo "A Hybrid Genetic Algorithm for an Identical Parallel-Machine Problem with Maintenance Activity" estudian el problema de programación de las actividades de mantenimiento en máquinas idénticas paralelas, atendiendo que las máquinas necesitan mantenimiento o remplazo de piezas de forma preventiva, esto ha llevado a las empresas manufactureras a enfrentarse al problema de encontrar la programación adecuada para realizar los mantenimientos en los puestos de trabajo sin dejar de cumplir con los programas de producción, por lo cual el mantenimiento debe ser realizado dentro de un intervalo de tiempo definido; los autores se enfocaron en determinar las frecuencias, ubicación y horario óptimo de mantenimiento de tal forma que el makespan y el tiempo total de terminación se redujera al máximo. Para resolver este problema utilizan un algoritmo de Branch and Bound para un tamaño pequeño del problema y posteriormente implementan un algoritmo genético híbrido para un tamaño grande del problema.

[19] LEE, Wen-Chiung; WANG, Jen-Ya, y LEE, Lin-Yo. A Hybrid Genetic Algorithm for an Identical Parallel-Machine Problem with Maintenance Activity. En: *The Journal of the Operational Research Society.* [Online]. 2015. Vol. 66, No.11, p. 1906-1918. [Citado 2016-04-18]. Disponible en: http://dx.doi.org/10.1057/jors.2015.19

El problema estudiado se asemeja a dos aplicaciones en el mundo real: industria logística y extrusión de aluminio, las máquinas utilizadas en cada una de las aplicaciones son adquiridas en diferentes fechas y después de un determinado periodo de operación requieren mantenimiento.

El estudio se divide en dos partes: en la primera se prueba la capacidad de los algoritmos: Branch and Bound (B&B), Genético (GA) y Genético con búsqueda local (GA +LS), en tres problemas (n=12, 16, 20) y la segunda parte prueba la velocidad de convergencia y la calidad de la solución de los algoritmos: Genético (GA) y Genético con búsqueda local (GA+LS) con problemas (n=50,100).

Los autores concluyen que el algoritmo Branch and Bound siempre genera los horarios óptimos pero no se puede aplicar a instancias grandes (n>20) debido a su complejidad de tiempo, el algoritmo genético converge muy rápidamente pero la solución que ofrece es de mala calidad y finalmente el algoritmo genético con búsqueda local siempre genera soluciones casi óptimas en tiempos razonables por lo cual es una buena solución para problemas con tamaño n>20.

Sheng-yao Wang, Ling Wang, Min Liu y Ye Xu (2013)[20], en el artículo "An enhanced estimation of distribution algorithm for solving hybrid flow-shop scheduling problem with identical parallel machines" enuncian el clásico problema de Flow Shop Híbrido que tienes tres variantes: el Flow Shop Híbrido con máquinas paralelas idénticas, el Flow Shop Híbrido con máquinas uniformes y el Flow Shop Híbrido con máquinas paralelas no relacionadas; abarcan en la investigación la primer variante del problema, es decir, el Flow Shop Híbrido con

[20] WANG, Sheng-yao *et al.* An enhanced estimation of distribution algorithm for solving hybrid flow-shop scheduling problem with identical parallel machines. En: *The International Journal of Advanced Manufacturing Technology.* 2013, Vol.68, No. 9-12, p. 2043-2056.

máquinas paralelas idénticas en donde un trabajo tiene tiempos de procesamiento idénticos en cada máquina en cualquier etapa.

No tienen en consideración el tiempo de liberación de las máquinas y los tiempos de procesamiento en cada etapa son determinísticos y conocidos. Buscan determinar las tareas de las máquinas de cada etapa y la secuencia de operaciones en todas las máquinas para minimizar el tiempo de finalización de todos los trabajos.

Para solucionar este problema implementan el algoritmo de estimación de distribución (EDA) el cual emplea distribuciones de probabilidad explicitas para optimizar las soluciones; y se compara con los algoritmos PSO Y AIS, demostrando que el algoritmo propuesto EDA genera soluciones de mejor calidad.

***Juan C. López, Jaime A. Giraldo, Jaime A. Arango (2014)[21]*,** en el artículo "Reduction of the Makespan in Scheduling the Production of a Hybrid Flexible Flow Shop", estudia el problema de máquinas paralelas no relacionadas según la clasificación en función de los tiempos de procesamiento de los trabajos en máquinas paralelas y contempla igualmente que el tiempo de preparación de un trabajo puede depender del trabajo inmediatamente anterior.

La función objetivo define que el makespan será el máximo tiempo de terminación de todos los trabajos en la última etapa, además observa que el tiempo de procesamiento del trabajo i en la máquina k del proceso j es afectado por el factor de eficiencia E_k, el cual es causado por los problemas de calidad, errores

[21] LÓPEZ, Juan C; GIRALDO, Jaime A. y ARANGO, Jaime A. Reducción del Tiempo de Terminación en la Programación de la Producción de una Línea de Flujo Hibrida Flexible (HFS). En *Información Tecnológica*. 2015, Vol.26, No.3, p. 157-172.

humanos y condiciones de la máquina, este factor de eficiencia se calcula como la diferencia entre el porcentaje de utilización de la máquina y el factor de mantenimiento de está; ocasionando una reducción del nivel de eficiencia de la máquina. En la solución del problema se aplicó un algoritmo genético simple o estándar (AGS) y los resultados fueron comparados con un algoritmo aleatorio, el problema fue aproximado al entorno real de la industria textil por lo cual se trabajó con un sistema compuesto por cinco (5) etapas con diecinueve (19) máquinas: hilandería (3 máquinas), urdizaje (2 máquinas), remetido (3 máquinas), tejeduría (10 máquinas) y acabado (1 máquina). Encontrándose que las mejores soluciones del algoritmo aleatorio son de menor calidad incluso que las peores soluciones encontradas por el algoritmo genético en todos los escenarios analizados.

Solarte Martínez, G. R., Castillo Sanz, a. G., & Rodríguez Gahona, G. (2015)[22]. La investigación presentada en el artículo "Optimization of a vehicular routing using simple genetic Chu-Beasley algorithm" está orientada hacia el área de los algoritmos genéticos en la optimización de recursos y procesos; y se enfoca en un estudio de caso fundamentado en el ruteo vehicular y la optimización del mismo, aplicando el algoritmo genético de Chu-Beasley, ya que este permite una gestión particular de la población objeto de estudio. El algoritmo de Chu-Beasley consiste fundamentalmente, en una modificación del algoritmo genético simple con algunas consideraciones adicionales que permiten una optimización más alta de las posibles soluciones.

La investigación fue aplicada en la ciudad de Bogotá porque cuenta con innumerables zonas industriales, centros de acopio y almacenes; además la extensión territorial exponencial de los últimos años ha ocasionado problemas de

[22] SOLARTE MARTÍNEZ, Guillermo; CASTILLO SANZ, Andrés y RODRIGUEZ GAHONA, Guillermo. Optimization of a vehicular routing using simple genetic Chu-Beasley algorithm. En: *Revista Tecnura*. 2015, Vol.19, No. 44, p. 93-108.

movilidad y trasporte. Se analiza el caso particular de Súper Almacenes Olímpica, quienes deben distribuir sus productos, optimizar los recorridos y garantizar efectividad y eficiencia en las entregas.

Súper Almacenes Olímpica debe hacer varias rutas diarias para llevar los productos a los almacenes, droguerías y restaurantes por toda la ciudad; los productos son entregados por camiones. Para lograr la optimización de la ruta vehicular, se utilizó el algoritmo de Chu-Beasley el cual encontró la mejor ruta y la mejor vía para tiempos de entrega.

Eliana M. Toro Ocampo, Mauricio Granada E. (2005)[23], en el artículo "Método híbrido entre el algoritmo genético de Chu-Beasley y Simulated Annealing para la solución del problema de asignación generalizada", analizan el problema de asignación generalizado (PAG), que es un problema clásico, en el cual hay un número de agentes y un número de tareas. Cualquier agente puede ser asignado para desarrollar cualquier tarea, recurriendo algún coste que puede variar dependiendo del agente y la tarea asignados.

El PAG es solucionado a través de una metodología hibrida entre el algoritmo genético de Chu-Beasley (AG) y el algoritmo Simulated Annealing (SA). El algoritmo genético de Chu-Beasley se caracteriza por mantener constante el tamaño de la población de alternativas de solución, de manera que en cada iteración la población es remplazada por un único descendiente generado. La población inicial en esta investigación no se obtiene de manera aleatoria, sino que se genera a partir de un análisis de sensibilidades usando un algoritmo heurístico

[23] GRANADA, Mauricio y TORO, Eliana. Mauricio. Método Hibrido entre el algoritmo genético de Chu-Beasley y Simulated Annealing para la solución del problema de asignación generalizada. En: *Scientia et Technica.* [En línea], 2005, Vol.2, No.28. [Citado:2016-04-05].

constructivo con el fin de mejorar la calidad de la búsqueda inicial y disminuir el esfuerzo computacional.

La etapa de mutación del AG se aborda a través del SA, este proceso es importante ya que permite una amplia gama de soluciones. Debido al SA en este proceso se invierte un alto esfuerzo computacional para encontrar una modificación sugerida de la alternativa actual, esto implica que el proceso consume mucho más tiempo computacional en cada iteración. Sin embargo, cada mutación mejora drásticamente a la actual, por lo que el número de iteraciones totales generalmente es menor.

El algoritmo propuesto permite obtener múltiples soluciones para el mismo problema y la estrategia de generar una población inicial construyendo una heurística y posteriormente realizar la mutación a través de SA presenta un alto desempeño obteniendo mejores respuestas que las obtenidas por el algoritmo de Chu-Beasley.

Las investigaciones encontradas en la literatura sobre el Flow Shop Híbrido Flexible en máquinas idénticas y con tiempos dependientes de la secuencia de procesamiento evidencian la necesidad de validar otros métodos de solución para este problema, con el fin de encontrar buenas soluciones en tiempos computacionales razonables, para esto se propone el desarrollo de un algoritmo genético que incluye unas modificaciones propuestas por Chu-Beasley[24], dicho algoritmo ha sido objeto de estudio de algunas investigaciones en nuestro país.

[24] CHU, P.C. Op. Cit., p.17-23.

5. MARCO TEÓRICO

5.1 PLANEACIÓN DE LA PRODUCCIÓN

La producción es el conjunto de actividades desarrolladas con la utilización de unos medios o recursos convenientemente seleccionados, organizados y gestionados, para la obtención o adicción de valor de uno o varios productos, a través de un proceso de producción[25], según el enfoque jerárquico de la producción la planificación y programación de la producción se caracterizan por contar con un conjunto de decisiones estructurales interrelacionadas, las cuales permiten definir la actividad productiva de la organización a largo, mediano y corto plazo, estas decisiones se dan en tres grandes áreas o niveles, donde cada nivel tiene sus propias metas y estas restringen a los niveles inferiores respecto a las decisiones a tomar en estos: [26]

- **Nivel estratégico:** decisiones de largo plazo y alto impacto en la supervivencia empresarial en relación con los productos, los procesos y capacidad, se convierten en condiciones fijas o restricciones operacionales de la empresa que responden a la estrategia competitiva de la organización[27].

- **Nivel táctico:** decisiones a mediano plazo, hace referencia a la programación eficiente de materiales y mano de obra atendiendo las restricciones tomadas en el nivel estratégico. Se sustenta en la planeación

[25]CUATRECASAS, Luis. Organización de la producción y dirección de operaciones: sistemas actuales de gestión eficiente y competitiva. Ediciones Díaz De Santos, 2011, p. 16
[26]DOMINGUEZ MACHUCA, José Antonio. Op. Cit. p.142
[27]CHASE, Richard; JACOBS, Robert y AQUILANO, Nicholas. Administración de la producción y operaciones para una ventaja competitiva, Décima edición ed. Mc Graw Hill, México D.F., México, 2005, p. 518

agregada* de los recursos. [28] Es el eslabón que une la planeación de las instalaciones con la programación de operaciones [29]

- **Nivel operativo:** decisiones a corto plazo y de tipo cotidiano, relacionadas con el desarrollo de la planeación y control operacional. Se diseña el programa maestro de producción que es el insumo principal para la secuenciación de pedidos, la programación del día y las actividades de control de la producción.[30]

5.1.1 Tipos de procesos o configuraciones productivas. Se debe decidir por el tipo de distribución productiva que permita ordenar las máquinas teniendo en cuenta factores como la demanda del mercado, el volumen de producción, la automatización y estandarización del proceso; los distintos tipos de procesos o configuraciones productivas son expuestos por Domínguez31 Machuca et al. a partir de la clasificación de Hayes y Wheelwright:

- **Configuración productiva por proyectos:** Se utiliza para la fabricación de productos únicos que representen cierta complejidad debida a la cantidad de entradas, volumen o peso del producto; lo cual impide o hace muy dificultoso su transporte al finalizar el proceso productivo, por eso es necesario que todos los inputs deban trasladarse hasta el lugar donde se elabora el producto o se presta el servicio.

* La planeación agregada no desglosa una cantidad detallada de la producción si no que la planeación se realiza para una medida general o para un número reducido de categorías agregadas de productos.
[28] Ibid., p.518.
[29] SCHROEDER, Roger; Meyer, Susan y RUNGTUSANATHAM, Johnny. *Administración de Operaciones: Conceptos y casos contemporáneos,* Quinta edición. México D.F., Editorial McGraw-Hill, 2008.
[30] BECERRA RODRIGUEZ, Fredy, *et. al.* Gestión de la producción: una aproximación conceptual. Primera Edición. Bogotá. Universidad Nacional de Colombia, Unibiblos, 2008. p.19
[31] DOMINGUEZ MACHUCA, José Antonio, Op. Cit. 142

- **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 en centros de trabajo**: Se procesan lotes pequeños con amplia variedad de productos, los cuales son hechos a medida de las necesidades expresadas por los clientes, es decir, el proceso implementado no está estandarizado. Las máquinas se agrupan en talleres o centros de trabajo de acuerdo a las funciones que realizan. Se distinguen dos situaciones.[32]

    a. Configuración a medida o de talleres: Se procesan lotes pequeños de producto, los cuales son diseñados a medida de las exigencias del cliente y son restringidos por la capacidad técnica de la empresa, se requieren un número reducido de operaciones poco especializadas que pueden ser desarrolladas por un trabajador o un grupo de trabajadores, quienes realizan todo el proceso productivo en los diversos centros de trabajo; para esto deben dominar cada una de las tareas involucradas en el proceso, estos se caracterizan por su alta flexibilidad debido a la no estandarización de los mismos.

    b. Configuración en lote o batch: Se procesan pequeños lotes de producto, los cuales son diseñados a partir de la elección del cliente entre las versiones ofrecidas por el fabricante, aunque ya no es producción a medida en su totalidad si existe un alto nivel de variación y flexibilidad en el proceso.[33] El proceso productivo requiere una mayor cantidad de operaciones, las cuales son más

[32] DOMINGUEZ MACHUCA, José Antonio, Op. Cit. 143
[33] CUATRECASAS, Luis. Op. Cit p.74

especializadas y necesitan maquinaria más enfocada en ciertas operaciones específicas, por lo tanto cada operario realiza las operaciones en su centro de trabajo asignado y la inversión inicial es mayor.

- **Configuraciones en línea:** Se procesan grandes lotes de pocos productos diferentes pero técnicamente homogéneos, haciendo uso de las mismas instalaciones. El proceso productivo de dichos productos tiene una secuencia similar por esto las máquinas se organizan en línea, es decir, una detrás de la otra. Existe poca flexibilidad y poca participación del cliente en el proceso. Como ejemplo se tiene una línea de montaje de un coche donde aunque varié el equipamiento, la motorización o el número de puertas, se trata del mismo modelo[34].

- **Configuración continúa:** 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 producción con costos bajos y cumpliendo los 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.

5.1.2 Secuenciación. Un problema que afronta el programador en taller de flujo consiste en establecer la secuencia de los trabajos, esta secuenciación de trabajos es parte del proceso de control en un sistema de producción y es necesaria cuando un conjunto común de recursos debe ser compartido para fabricar una serie de productos durante el mismo intervalo de tiempo.[35] En el caso

[34]CUATRECASAS, Luis. Op. Cit p.146
[35] SCHOROEDER Roger G. Op. Cit. p. 403

del Flow Shop Híbrido donde se disponen de múltiples recursos para realizar las operaciones el programador debe resolver también el problema de asignación y temporización.

El objetivo de la secuenciación, es la asignación eficiente de máquinas y otros recursos a los trabajos, o a las operaciones contenidas en estos, y la determinación del orden en el cual cada uno de estos trabajos debe ser procesado[36].

Los siguientes supuestos aparecen frecuentemente en la literatura sobre secuenciación de trabajos en máquinas, la mayoría de estas son introducidas por Conway, R. W.,Maxwell, W. L., y Miller, L. W. en 1967 [37]:

- Las máquinas están siempre disponibles y nunca dejan de funcionar.
- Cada máquina puede procesar a lo sumo un trabajo a la vez.
- Cualquier trabajo puede ser procesado únicamente en una máquina a la vez.
- Los tiempos de preparación de todos los trabajos son cero, por ejemplo, todos los trabajos están disponibles al comienzo del proceso.
- No se permiten interrupciones, es decir, cuando una operación ha comenzado debe terminarse antes de empezar otra en la misma máquina.
- 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, y similarmente ocurre con las fechas de entrega.

[36] CASTILLO, Guillermo; FANDIÑO, Oscar. Diseño de un modelo general para la planeación operativa y la secuenciación de actividades en pequeñas empresas con procesos de manufactura. Bucaramanga, 2005. p. 58. Trabajo de grado (Ingeniería Industrial). Universidad Industrial de Santander. Disponible en el catálogo en línea de la Biblioteca de la Universidad Industrial de Santander:< http://tangara.uis.edu.co/biblioweb/>
[37] CONWAY, R. W.; MAXWELL, W. L., y MILLER, L. W. Theory of scheduling. Dover Publications, Inc. New York,1967

Con los anteriores supuestos, se refleja que la teoría se aleja de la realidad dado que los entornos donde se realiza la 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 de 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 válidas y es por eso que en los últimos años han aparecido modelos y procedimientos de resolución que relajan una o varias de los supuestos anteriores.[38]

Las variantes más importantes del problema de secuenciación son:

- **Job Shop Scheduling[39]:** 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.

- **Task Scheduling[40]:** Se tiene un conjunto de tareas, cada una de ellas está asociada a una duración determinada de tiempo, el objetivo es programar

[38] LIU, J. y MACCARTHY, BL. General Heuristic Procedures and Solutions Strategies for FMS Scheduling. En: *International Journal of Production Research.* [Online]. 2015, Vol.37, No.14, p. 3305-3333. [Citado 2016-05-02].
[39] BARD, Jonathan F. y FEO, Thomas A. Operations sequencing in discrete parts manufacturing. En: *Journal of Management Science*. [Online]. 1989, Vol.35. No.2. p. 249-255. [Citado: 11-05-2016].
[40]TUPIA ANTICONA, Manuel Francisco. Un algoritmo GRASP para resolver el problema de la programación de tareas dependientes en máquinas diferentes (task scheduling).Lima, 2005. Tesis (Magister en ingeniería de sistemas). Universidad Nacional Mayor de San Marcos.

las tareas en unas máquinas procurando el orden más apropiado, atendiendo que para que se ejecute una tarea, sus predecesoras necesariamente tienen que ser ejecutadas.

- **Flow Shop Scheduling**: Consiste en un número de trabajos que son procesados en un número de máquinas, cada trabajo debe ser procesado por todas las máquinas en el mismo orden. Se encuentran la siguiente variedad de problemas:

➢ **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[41].

Características del problema de programación de tareas Flow Shop:[42]

- ✓ Cada máquina está disponible continuamente y sin interrupciones.
- ✓ Cada máquina puede procesar una tarea por vez.
- ✓ Cada tarea sólo puede ser procesada por una máquina cada vez.
- ✓ Los tiempos de procesamiento de las tareas en las diferentes máquinas son determinados y fijos.
- ✓ Las tareas tienen la misma opción de ser programadas.
- ✓ Los tiempos de preparación de las operaciones en las distintas máquinas están incluidos en los tiempos de procesamiento.
- ✓ Las operaciones en las máquinas, una vez iniciadas no deben ser interrumpidas.

[41]PINEDO, Michael L. Planning and Scheduling in Manufacturing and Services. Springer Science + Business Media, LLC, New York, 2005.
[42]TORO OCAMPO, Eliana; RESTREPO GRISALES, Yov Steven y GRANADA ECHEVERRI, Mauricio. Algoritmo genético modificado aplicado al problema de secuenciamiento de tareas en sistemas de producción lineal - Flow Shop. En: *Scientia Et Technica*. [en línea]. 2006, Vol. XII, No.30, p. 285-290. [Citado: 2016-05-12].

El problema Flow Shop con más de dos etapas está clasificado como un problema NP-Hard, debido a que su tiempo de ejecución aumentará exponencialmente con el tamaño del problema.[43]

- **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 etapas de la línea, por lo tanto existen n! secuencias posibles como solución a este problema.[44]

- **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 termino su tarea y así mismo bloqueando el acceso de los trabajos que suceden al trabajo estancado.[45]

- **Flow shop híbrido:** Es un caso especial del 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[46]:

  - ✓ Idénticas: con tiempos de procesamiento iguales para todas las máquinas

[43] ÁLVAREZ MARTÍNEZ, David; TORO OCAMPO, Eliana y GALLEGO RENDÓN, Ramón. Estudio computacional con técnicas heurísticas basadas en recocidos para resolver el problema de secuenciación de tareas. En: *Ingeniería & Desarrollo*. Universidad del Norte. [en línea], 2009, Vol.25, p.154-179. [Citado: 2016-05-11].
[44] PAN, Quan-Ke y RUIZ, Rubén. An effective iterated greedy algorithm for the mixed no-idle permutation flow shop scheduling problem. En: *Omega*. [online]. 2014, Vol.44, p. 41-50. [Citado: 2016-05-12].
[45] MATÍA DEPRIT, Javier. Optimización de la secuenciación de tareas en taller mediante algoritmos genéticos. Madrid, 2007. Tesis (Ingeniería Industrial). Universidad Pontificia Comillas.
[46]CEVIKCAN, E.; DURMUSLOGLU, M. y BASKAK, M. Integrating parts design characteristics and scheduling on parallel machines. En: *Expert Systems with Applications*. [Online]. 2011, Vol.38, No. 10, p.13232-13253. [Citado: 2016-05-12]. Disponible en: doi:10.1016/j.eswa.2011.04.140

- ✓ 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 solo existir un recurso por etapa únicamente es necesario tomar la decisión del secuenciamiento de tareas, por el contrario en el Flow Shop Híbrido se deben domar dos decisiones: la asignación de los trabajos a las máquinas de cada etapa y el secuenciamiento de los trabajos en las diferentes máquinas.[47]

- **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[48].

Existen tiempos entre el procesamiento de los trabajos, es decir, existe un periodo de tiempo destinado a la preparación o alistamiento de las máquinas entre la continuación de dos trabajos[49], este tiempo de configuración en cada máquina puede depender del trabajo inmediatamente

---

47 GOMEZ GASQUET, Pedro. Programación de la producción en un taller de flujo hibrido sujeto a incertidumbre: arquitectura y algoritmos. Aplicación a la industria cerámica. Valencia, 2010. Tesis (Doctorado en Gestión de la Cadena de Suministro e Integración Empresarial). Universidad Politécnica de Valencia.

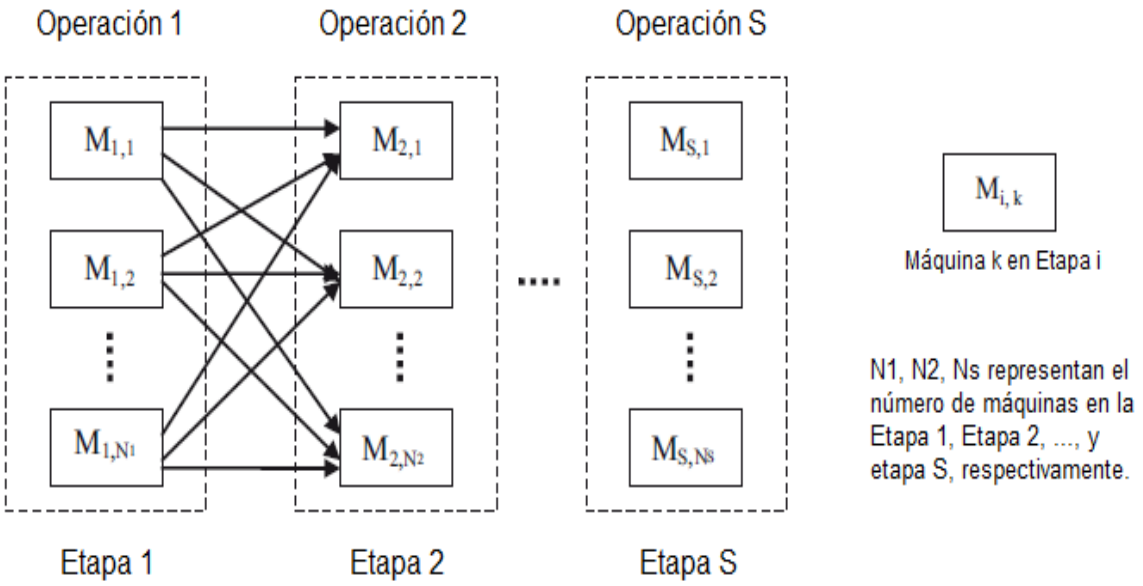
48 ZANDIEH,M. y KARIMI, N. An adaptive multi-population genetic algorithm to solve the multi-objective group Scheduling problem in Hybrid Flexible Flow Shop with sequence-dependent setup times. En: *Journal of Intelligent Manufacturing*, [online].2011, Vol.22, No. 6 [Citado 2016-04-22], p 979-989

49ÁNGEL-BELLO, Francisco, *et al* A heuristic approach for a scheduling problem with periodic maintenance and sequence-dependent setup times. En: *Computers & Mathematics with Applications*. [online]. 2011, Vol. 61, No. 4, pp.797-808.[Citado:12-05-2016].

anterior y es denominado como tiempos de alistamiento dependientes de la secuencia;[50]la unión del problema Flow Shop Híbrido Flexible con una configuración con tiempos dependientes de la secuencia da origen al problema SDST/HFFS[51] que considera líneas de flujo en el que concurren varias etapas con múltiples máquinas donde existe la posibilidad de que no todos los trabajos deban ser procesados en todas las etapas, considerando que existe un tiempo de configuración o setup de las máquinas entre dos trabajos diferentes y que este setup depende de la secuencia de trabajos a procesar.

En la figura 1 se ilustra un sistema Flow Shop Híbrido Flexible.

Figura 1. Ambiente Flow Shop Híbrido Flexible



Fuente: GONZÁLES PALACIO, Ángela. Diseño de una metodología de programación de producción para la reducción de costos en un Flow shop híbrido flexible mediante el uso de algoritmos genéticos. Aplicación a la industria textil. En: Universidad Nacional de Colombia. 40 p.

[50] JABBARIZADEH, F. Op. Cit. p.949

[51] DUPUY, Germán, *et al.* Métodos Metaheurísticos aplicados a procesos de gestión en una empresa. Laboratorio de Investigación en Sistemas Inteligentes. XVII Workshop de Investigadores en Ciencias de la Computación, Salta, 2015. Disponible en : http://hdl.handle.net/10915/45319

5.2 OPTIMIZACIÓN

Es una disciplina fundamental en campos de la ciencia tales como la informática, la inteligencia artificial o la investigación operativa. En otras comunidades científicas, la definición de optimización es imprecisa, y se relaciona con la idea de "hacerlo mejor". El concepto de optimización se concibe como el proceso de intentar encontrar la mejor solución posible a un problema de optimización, generalmente en un tiempo limitado.[52]

Se considera un problema de optimización como un problema en el que hay varias posibles soluciones y alguna forma clara de comparación entre ellas.

Desde un punto de vista matemático, ciertos problemas pueden ser matemáticamente modelados representando lo más cercano a la vida real, para los cuales existen modelos bien definidos. Un problema de optimización P se puede formular como una lista de 3 elementos P= (f, SS, F) definida como sigue:

$$P = \begin{cases} opt: \ f(x), Funcion\ Objetivo \\ s.a \\ x \in F \ \ c\ SS \ \ Restricciones \end{cases}$$

Donde f es la función a optimizar (maximizar o minimizar), F es el conjunto de soluciones factibles y SS es el espacio de soluciones.

Este tipo de problemas se pueden dividir de forma natural en dos categorías: aquellos en los que la solución esta codificada mediante valores reales y aquellos cuyas soluciones están codificadas por valores enteros. Dentro de la segunda

[52] DUARTE MUÑOZ, Abraham; PANTIGRO FERNÁNDEZ, Juan José y GALLEGO CARRILLO, Micael; Metaheurísticas. Universidad Rey Juan Carlos. Editorial DYKINSON.S.L., 2007, Madrid. p.1.

categoría se encuentran un tipo particular de problemas denominados problemas de optimización combinatoria.

En un problema de optimización combinatoria el objetivo es encontrar el máximo o el mínimo de una determinada función sobre un conjunto finito de soluciones. No se exige ninguna condición o propiedad sobre la función objetivo o la definición del conjunto de soluciones.[53] Dada la finitud de soluciones, las variables deben ser discretas, restringiendo su dominio a una serie finita de valores. Habitualmente, el número de elementos de soluciones es muy elevado, haciendo impracticable la evaluación de todas sus soluciones para determinar el óptimo.

En los últimos años se ha presentado un crecimiento notable en el desarrollo de procedimientos heurísticos para la solución de problemas combinatorios. Estos presentan como particularidad que siempre existe un algoritmo exacto que permite obtener la solución óptima. Este método consiste en la exploración de forma exhaustiva del conjunto de soluciones. Este algoritmo suele ser extremadamente ineficiente, ya que para la mayoría de los problemas de interés que se pueden encontrar en el ámbito de la optimización combinatoria el tiempo que emplearía en encontrar una solución crece de forma exponencial con el tamaño del problema.

5.3 COMPLEJIDAD COMPUTACIONAL

En la década de los sesenta se elaboran los primeros cimientos de la Teoría de la Complejidad Computacional con la clasificación de lenguajes y funciones según los estudios de J.Hartmanis, P.M. Lewis y R.E. Stearns; la teoría de la complejidad computacional forma parte de la teoría de la computación, la cual analiza los

[53] MARTÍ, Rafael. Procedimientos Metaheurísticos en Optimización Combinatoria. Departamento de Estadísticas e Investigación de Operaciones. Facultad de Matemática. Universidad de Valencia. 2003.

recursos requeridos para la resolución de un problema, la teoría de la complejidad estudia la eficiencia de los algoritmos en función de los recursos que utiliza en un sistema computacional, estos usualmente son dos:

- Espacio: cantidad de memoria utilizada para resolver un problema
- Tiempo: número de pasos de ejecución de un algoritmo para resolver un problema.

En el análisis interesa como se lleva el cálculo y su complejidad en función de los recursos que utiliza, se dice que el cálculo es complejo si es difícil de realizar, entendiendo a la complejidad como la cantidad de recursos necesarios para realizar un cálculo, es decir, un cálculo difícil requiere más recursos para su solución que un cálculo de menor dificultad, en este sentido se encuentran dos tipos de complejidad: complejidad temporal que se evidencia cuando un cálculo requiere mucho tiempo para su solución y complejidad espacial si el cálculo a realizar requiere mucho espacio de almacenamiento, de esta manera se determina que un algoritmo que resuelve un problema pero tarda mucho tiempo no será catalogado como de utilidad, igualmente sucede si el algoritmo necesita gran cantidad de memoria para su ejecución y solución.[54] Para conocer el grado de complejidad de un problema se hace referencia al modelo computacional de la máquina de Turing[55], la cual fue propuesta en 1936 y continua utilizándose hoy en día para estudiar la potencia de los procesos algorítmicos, mediante esta se llega a obtener una clasificación de los problemas en base al nivel de complejidad presente para resolverlos.[56]

[54] CORTEZ, Augusto. Teoría de la complejidad computacional y teoría de la computabilidad. En *Rev. investig. sist. inform*., [En línea], 2004, Vol. 1, No. 1, p.102-105. [Citado 2016-05-04].
[55] BROOKSHEAR, J. Glenn. Introducción a la computación. Décimo primera edición. Pearson Educación, S.A., Madrid, 2012, p.594-595
[56] CRUZ CHÁVEZ, Marco Antonio; MORENO BERNAL, Pedro y PERALTA ABARCA, Jesús del Carmen. Aplicación de la teoría de la complejidad en optimización combinatoria. En *Inventio*. [En línea]. 2014, Vol.20, p.35-41. [Citado 2016-05-05]. Disponible en: http://goo.gl/QDGTGt.

El principal objetivo de la complejidad computacional es la clasificación de los problemas dependiendo la cantidad de recursos necesarios para alcanzar la solución.

5.3.1 Clasificación de la complejidad computacional Los problemas de acuerdo a su complejidad se dividen en las siguientes clases:

- **CLASE P:** Son problemas que tienen una solución óptima mediante tiempo polinomial, el tiempo de ejecución del algoritmo aumenta proporcionalmente a las instancias de entrada. [57]

- **CLASE NP:** Son problemas para los cuales no se ha encontrado un algoritmo que en tiempo polinómico los resuelva pero si se ha logrado obtener una solución en ocasiones tomando espacios de tiempo extensos para instancias pequeñas, es decir, son soluciones sub óptimas,[58] por lo tanto son problemas verificables en tiempo polinómico. Los problemas P pertenecen a la clase NP, puesto que siempre se puede verificar o comprobar su solución en tiempo polinómico[59]

- **NP-COMPLETOS:** Son problemas que pertenecen a la clase NP[60] por lo tanto no es conocida su solución óptima, a diferencia de los problemas NP no existe manera para encontrar su solución exacta ni siquiera para instancias pequeñas debido a la gran cantidad de variables[61]; pero si es

[57] GAREY M; JONSON D. Local Search in Combinatorial Optimization. Editorial John Wiley & Sons, Inglaterra, 1997.
[58]DUARTE MUÑOZ, Abraham. Op. Cit. p.9
[59] PAPADIMITRIOU, Christos y KLEINGBER, Jon. Computability and Complexity. En: *Cornell University. Department of Computer Science*. [Online], 2004, p.8. [Citado: 2016-05-05].
[60] COOK, Stephen. The Complexity of Theorem Proving Procedures, Proceedings 3rd ACM Symposium Theory of Computing [online], 1971, p.151-158. [Citado 2016-05-05].
[61]BRASSAR Gilles y BRATLEY Paul. Fundamental of algorithmics. Editorial Prentice Hall, New Jersey - USA, 1996.

posible determinar si un valor es solución al problema a través de un algoritmo polinómico.

- **NP-DUROS:** Son problemas en donde no existe un algoritmo polinómico que permita verificar la solución. Esta clase es un subconjunto de los problemas NP.[62]

En la figura 2 se evidencia la relación entre los niveles de complejidad.

Figura 2. Relación entre los problemas P, NP, NP-Completo y NP-Duro

Fuente: DUARTE MUÑOZ, Abraham. Metaheurísticas. Editorial Dykinson. Madrid. 2008. 9 p.

5.4 HEURISTICAS

Cuando las técnicas de optimización exactas en su intento de encontrar la solución óptima de un problema de la vida real que necesita resolverse no producen el resultado deseado, debe recurrirse al uso de técnicas que no garantizan la obtención de la solución óptima global, pero que permiten encontrar soluciones subóptimas de buen rendimiento y buena calidad; estas técnicas se denominan heurísticas. Un algoritmo heurístico es por lo tanto una estrategia de búsqueda de soluciones que utiliza un conjunto de procedimientos que encuentran soluciones de buena calidad de manera simple y rápida.

[62] DUARTE MUÑOZ, Abraham; Op. Cit. p. 12

En los últimos años se ha producido un crecimiento importante en el desarrollo de procedimientos heurísticos, el auge que experimentan los métodos heurísticos se debe a la necesidad de disponer de herramientas que permitan ofrecer soluciones rápidas a problemas reales. Los algoritmos heurísticos son de diferente naturaleza, y se pueden agrupar en dos categorías, las cuales se describen a continuación.[63]

5.4.1 Métodos constructivos Consisten en ir adicionando, generalmente uno a uno, componentes individuales de la solución hasta encontrar una solución factible en particular. Se realiza mediante un proceso iterativo en donde se añaden elementos a una estructura inicialmente vacía, que representa a la solución. A este método pertenecen:

- Algoritmo goloso: construye paso a paso una solución factible a partir de una semilla y finaliza cuando no existe una solución vecina de mejor calidad.

- Algoritmo de Descomposición: divide el problema principal en sub problemas más sencillos con la finalidad de simplificar el proceso de solución. Posteriormente las soluciones obtenidas de los sub problemas son combinadas hasta tener la solución al problema original.

- Algoritmos de reducción: identifican propiedades o características específicas que contienen las soluciones para convertirlas en restricciones del problema con el propósito de limitar el espacio de soluciones y de esta forma reducir el tamaño del problema.

[63] MATÍA DEPRIT, Javier. Op. Cit. pp. 23

- Algoritmos de manipulación del modelo: buscan relajar el problema original y usan la solución del modelo relajado para ayudar a encontrar una buena solución del problema original. En este algoritmo se puede linealizar, eliminar o adicionar restricciones a conveniencia.

5.4.2 Métodos de búsqueda Producen mejores soluciones a partir de una solución factible. Entre los métodos de búsqueda se encuentran:

- Algoritmo de búsqueda usando vecindad: mejora progresivamente la solución factible inicial, mediante una transición adecuada pasan a una solución vecina que mejora la solución actual.

- Heurísticas primales: Se utilizan cuando el modelo matemático del problema es conocido y este incluye variables enteras.

Los métodos heurísticos tienen su principal limitación en su incapacidad para escapar de óptimos locales. Para solucionar este problema, se introducen otros algoritmos de búsqueda más inteligentes denominados metaheurísticas que evitan, en la medida de lo posible este problema[64].

5.5 METAHEURÍSTICAS

En los últimos años han aparecido una serie de métodos bajo el nombre de metaheurísticos con el propósito de encontrar soluciones mejores y de mayor calidad que las heurísticas mediante el uso de estrategias de búsqueda exitosas.

[64] DUARTE MUÑOZ, Abraham; Op. Cit. pp.3

De acuerdo con Osman y Kelly[65] “los procedimientos Metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los Metaheurísticos pueden verse como un marco general para crear nuevos algoritmos híbridos de mayor alcance, combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos.”

Las metaheurísticas utilizan el concepto de vecindad y, a diferencia de las heurísticas, poseen mecanismos que les permiten escapar de soluciones óptimas locales y hallar la mejor solución; esta mejor solución encontrada es denominada incumbente y es la respuesta del algoritmo una vez que termina el proceso de búsqueda[66].

A continuación se nombran algunas características que describen a las metaheurísticas:[67]

- Son algoritmos paso a paso, basados en un principio sencillo y claro.
- Adecuados para resolver problemas NP-complejos, y para los que se desea alcanzar soluciones subóptimas de alta calidad.
- Son autónomos en decidir si pasan o no a la mejor solución vecina y no finalizan el proceso de búsqueda cuando no hay un mejor vecino.
- Cada metaheurística tiene su propia estrategia para seleccionar las propuestas de solución que debe evaluar.

[65] OSMAN, Ibrahim y KELLY, James. Meta-heuristics: Theory and Applications. Kluwer Academic Publishers. 1996.
[66] GALLEGO, Ramón; Op. Cit. p.25
[67] Ibid., p.25

- Pueden ser adaptables y robustos. Deben ser capaz de adaptarse a diferentes contextos de aplicación o modificaciones importantes del modelo y poco sensible a pequeñas alteraciones del mismo.
- Generalmente dan por finalizado el proceso de búsqueda cuando la solución actual no mejora después de un cierto número de iteraciones.

A pesar que no existen características que sean comunes a todas las metaheurísticas y que permitan realizar una clasificación exacta de las mismas, se separan las metaheurísticas en dos grandes bloques[68]: metaheurísticas trayectoriales y metaheurísticas poblacionales.

5.5.1 Metaheurísticas trayectoriales Se utiliza el termino trayectoria porque el proceso de búsqueda que desarrollan estos métodos se caracterizan por una trayectoria en el espacio de soluciones. Es decir, que partiendo de una solución inicial, son capaces de generar un camino o trayectoria en el espacio de búsqueda a través de operaciones de movimiento. Las características de la trayectoria proporcionan información sobre el comportamiento del algoritmo y de su efectividad. Las metaheurísticas trayectoriales son:

- Búsqueda tabú: propuesta por F. Glover consiste en un procedimiento de mejora a través de una búsqueda local dotado de una memoria adaptativa, es decir, la memoria se actualiza en función del tiempo y en la búsqueda sensible que le permite no quedar atrapado en óptimos locales.

- Búsqueda de vecindad variable: propuesta por P. Hansen y N. Mladenovic, es una metaheurística que intenta evitar quedar atrapada en óptimos locales cambiando la estructura de la vecindad donde se realiza la búsqueda.

[68] BLUM, Christian y ROLI, Andrea. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. En: *ACM Computing Surveys*. [online], 2003, Vol. 35, No.3, p.268-308. [Citado: 2016-05-16].

- Búsqueda local guiada: Se basa en la modificación de la función objetivo a través de penalizaciones con el fin de quedar atrapada en óptimos locales, modificando el perfil del espacio de búsqueda.

- Métodos multi-arranque: conjunto de procedimientos que intentan evitar quedar atrapados en óptimos locales re-arrancando de nuevo el procedimiento, lo que permite la diversificación de la estrategia de búsqueda. Es un algoritmo iterativo en el cada iteración tiene dos fases. En la primera fase se genera una solución y en la segunda fase, la solución generada se mejora[69].

- GRASP: desarrollado por T. Feo y M. Resende, es un procedimiento multi-arranque en el que cada arranque corresponde a una iteración. Cada iteración tiene dos fases bien diferenciadas: la fase de construcción, la cual se encarga de obtener una solución factible de alta calidad a través de una heurística constructiva y la fase de mejora, que se basa en la optimización local de la solución obtenida en la primera fase a través de un algoritmo de búsqueda local.

- Recocido simulado: se basa en la analogía entre un proceso de optimización combinatoria y un proceso termodinámico, conocido como recocido. Es un algoritmo aleatorizado de búsqueda local por vecindades, donde se aceptan movimientos que mejoran la función objetivo, y se aceptan, con una probabilidad dependiente de un parámetro controlable (llamado temperatura). Al esquema que controla la temperatura se le llama esquema de enfriamiento (Cooling Schedule) que influye en la calidad de la solución final.

[69] MARTÍ, Rafael. Op. Cit. p.34

5.5.2 Metaheurísticas poblacionales Se caracterizan por trabajar en cada iteración del algoritmo con un conjunto de soluciones denominado población. La eficiencia de la exploración depende fuertemente de cómo se manipule dicha población. Las metaheurísticas poblacionales son:

- Búsqueda dispersa: tiene sus orígenes algorítmicos en el uso de la programación entera aplicada a problemas de secuenciación de tareas, donde se proponían una serie de estrategias para combinar las reglas de decisión y restricciones. La información sobre la calidad o el atractivo de un conjunto de reglas, restricciones o soluciones se puede utilizar mediante la combinación de estas, de tal forma que, dadas dos soluciones de un problema, se podría obtener una nueva solución mediante su combinación, de modo que mejore a las que la originaron[70].

- Algoritmos evolutivos: Se basan en la idea neo-darwiniana de evolución de las especies, es decir, los individuos que tienen una mejor adaptación al medio (por encima de la media) se caracterizan por tener una probabilidad más elevada de vivir más tiempo, con lo que tendrán más posibilidades de generar descendencia que herede sus buenas características. En cambio, los individuos con peor adaptación al medio, tienen menos probabilidad de sobrevivir, por lo que tendrán menos oportunidades de generar descendencia y probablemente acaben extinguiéndose.

[70] GLOVER, Fred y LAGUNA, Manuel. Tabu search. Kluwer Academic Publishers, Norwell Massachusetts,1997.

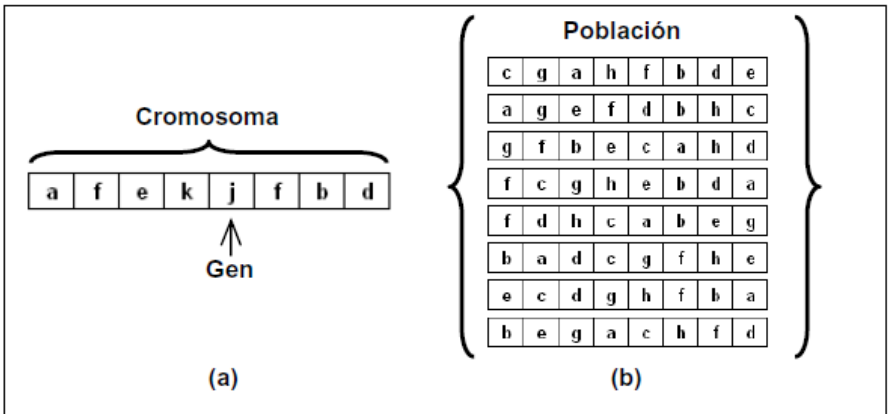
5.6 ALGORITMO GENÉTICO

Los algoritmos genéticos fueron introducidos inicialmente por John Holland[71] en 1975, estos algoritmos integran la teoría de la evolución natural con el mundo computacional dando paso a sistemas artificiales que constituyen una clase de métodos de búsqueda adecuada para la solución de problemas complejos de optimización;[72] según GOLDBERG[73] los algoritmos genéticos son "algoritmos de búsqueda basados en los mecanismos de selección natural y genética natural. Combinan la supervivencia de los más compatibles entre las estructuras de cadenas, con una estructura de información ya aleatorizada, intercambiada para construir un algoritmo de búsqueda con algunas de las capacidades de innovación de la búsqueda humana", de esta forma se tiene que los algoritmos genéticos se identifican por la evolución de una población o solución inicial, a través de iteraciones, las cuales permiten que los individuos más aptos se reproduzcan encontrando mejores soluciones debido a que los individuos más débiles son descartados y por lo tanto logrando el mejoramiento de las características de los descendientes de cada generación[74].

La población es el conjunto de soluciones del problema, la cual parte desde una población inicial que evoluciona de generación en generación. La población está compuesta por individuos o cromosomas los cuales están conformados por una cadena de genes, que codifican un elemento particular de la solución[75].

[71] HOLLAND, John Henry. Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan Press; Pittsburgh, 1975.
[72] RENNER, Gábor y EKÁRT, Anikó. Genetic algorithms in computer aided design. En: *Computer-Aided Design*. [online], 2003. Vol. 35, No. 8, p. 709-726. [Citado 12-05-2016].
[73] GOLDBERG, David. Genetics Algorithms in Search, Optimization and Machine Learning. Addison Wesley Logman Publishing Co., Boston, 1989.
[74] SALAZAR HORNIG, Eduardo Op. Cit. p.120.
[75] Ibid., p.120

Figura 3. Representación de la población en un algoritmo genético



Fuente: VÉLEZ, Mario Cesar y MONTOYA, José Alejandro. Metaheurísticos: una alternativa para la solución de problemas combinatorios en administración de operaciones. En: Rev.EIA.Esc.Ing.Antioq. 2007. 105 p.

El proceso algorítmico básico del algoritmo genético es:[76]

1. Inicio: generación de población inicial de n cromosomas (soluciones posibles para el problema).
2. Fitness: calcular la aptitud f (x) de cada cromosoma x de la población.
3. Prueba: si la condición de terminación está satisfecha, se para el algoritmo, se devuelve la mejor solución de la población actual y se va al paso 7.
4. Nueva población: se crea una nueva población repitiendo los siguientes pasos, hasta que se cumpla la condición de parada.
    - Selección: se selecciona dos cromosomas padres, de una población, según su fitness (cuanto mejor es el fitness, mayor es la probabilidad de ser seleccionado).
    - Cruce: los padres se cruzan para formar a un nuevo descendiente (hijos). Si no se realiza cruce alguno, el descendiente es la copia exacta de los padres.
    - Mutación: el nuevo descendiente muta (en alguna posición de su cromosoma).

[76] CORTEZ VÁSQUEZ, Augusto. *et al.* Sistema de apoyo a la generación de horarios basado en algoritmos genéticos. En: *Revista de Investigación de Sistemas e Informática.* [en línea], 2010. Vol. 7, No.1, p.37-55. [Citado: 2016-05-15].

5. Sustituir: la nueva población generada es aplicada para otra iteración del algoritmo.
6. Bucle: se va al paso 2.
7. Fin del algoritmo

Figura 4. Secuencia del algoritmo genético básico

Fuente: Adaptado de SAN MARTÍN CONTRERAS, Oscar. Optimización de la Producción Para Problemas de 'Flow-Shop' Multiobjetivo Mediante la Utilización de Metaheurísticas. Universidad del Bio Bio. Chile. 2006. 41 p.

En cada una de las iteraciones del algoritmo se crean nuevos individuos que aportan nuevas soluciones al problema mediante la aplicación de los operadores: selección, cruce y mutación. La nueva población se crea mediante la fusión de las mejores soluciones de la población y las nuevas mejores soluciones, es decir, se comparan las peores soluciones de la población con las soluciones hijas y si estas son mejores las remplazan en la población.[77]

[77] MARTÍ, Rafael y GERHARD, Reinelt. The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization. Applied Mathematical Sciences Springer, Berlin, 2011

El algoritmo genético contiene cinco elementos básicos[78]:

- Codificación
- Generación de la población inicial
- Diseño de la función de ajuste (o *fitness*)
- El diseño de los operadores genéticos
- La condición de terminación

5.6.1 Codificación[79] La elección de la codificación dependerá del problema a resolver pues puede darse la situación en la que la resolución de un caso sea más óptimo el uso de una codificación específica. Algunas formas de codificación son:

- Codificación binaria: el cromosoma es una cadena de bits (0 o 1)
- Codificación numérica: el cromosoma es una cadena de números que representan un numero en una secuencia (se utiliza en problemas donde hay que ordenar algo)
- Codificación por valor directo: el cromosoma es una cadena de valores relacionados con el problema a estudiar, pudiendo ser desde números decimales, cadena de caracteres o una combinación de ellos, necesita desarrollar nuevas técnicas de reproducción y mutación para la solución del problema especifico
- Codificación de árbol: cada cromosoma es un árbol con ciertos objetos, se utiliza en programación genética

[78] ZHANG, Songyan. Large-scale flow shop scheduling based on genetic algorithm. En: *2nd International Conference on Education Technology and Computer.* [online], 2010. p.308-310. [Citado: 2016-05-17].
[79] ARRANZ DE LA PEÑA, Jorge y PARRA TRUYOL, Antonio. Algoritmos Geneticos. Universidad Carlos III, 2007.

5.6.2 Generación de la población inicial El algoritmo genético genera su población a través de métodos determinísticos o en la mayoría de los casos de manera aleatoria; otro factor importante es definir el tamaño de la población,[80]para lo cual se debe tener en cuenta que un tamaño de la población muy pequeño puede hacer que el algoritmo converja muy rápidamente y un tamaño de la población muy grande hará que se desperdicien recursos computacionales[81]

5.6.3 Diseño de la función de ajuste o fitness La función de aptitud o fitness permite valorar la aptitud de los individuos y debe tomar siempre valores positivos, esta determina la capacidad de sobrevivencia y de reproducir descendencia de cada individuo, en muchos casos es la misma función objetivo[82]. Cada solución candidata es evaluada según su valor fitness que indica la calidad de la solución representada y aquellas que califiquen mejor tendrán mayor probabilidad de sobrevivir.[83] El valor del fitness depende de lo bien que el cromosoma resuelve el problema, siendo penalizado por su incapacidad para satisfacer las restricciones del problema.[84]

5.6.4 Operadores genéticos Los operadores genéticos se aplican a individuos seleccionados de la población actual con el fin de crear una nueva población. Los operadores de cruce y mutación deben diseñarse con cuidado ya que su elección contribuye al rendimiento de todo el algoritmo genético.[85] La forma más simple de algoritmo genético implica tres tipos de operadores: selección, cruce y mutación[86].

[80]JABBARIZADEH, F. Op. Cit. p.952
[81] CERVANTES POSADA, Mariamar. Nuevos Métodos Meta Heurísticos para la Asignación Eficiente, Optimizada y Robusta de Recursos Limitados. Valencia, 2010. Tesis (Doctor en Informática). Universidad Politécnica de Valencia. Departamento de Sistemas Informáticos y Computación.
[82] SANHUEZA, Raúl, *et al.* Aplicación de algoritmos genéticos al problema de la planificación de sistemas eléctricos de distribución. En: *Revista Facultad de Ingeniería.* [online], 1999, Vol.6. p.55-63
[83] LOPEZ VARGAS, Juan Camilo. Op. Cit. p.24
[84] MITCHELL, Melanie. An Introduction to Genetic Algorithms. Quinta edición, A Bradford Book The MIT Press, Cambridge, Massachusetts,1999, p.7
[85] RENNER, Gábor. Op. Cit. p.712
[86] MITCHELL, Melanie. Op. Cit. p. 8

- **Operador de selección:** Este operador selecciona a los individuos de la población que tendrán descendencia. Los individuos con mejores valores de fitness tienen mayor posibilidad de reproducirse que los individuos con valores menos buenos. Mientras un individuo tenga un valor más alto más veces será seleccionado para reproducirse, sin embargo los individuos con valores bajos también deben tener la posibilidad de ser seleccionados para reproducirse debido a que su material genético puede ser útil para encontrar buenas soluciones[87]. Los esquemas de selección más comúnmente utilizados: [88]

  - **Selección por ruleta:** la probabilidad de ser seleccionados es proporcional a la aptitud o fitness, por lo tanto los mejores individuos son los que tienen mayores posibilidades de ser elegidos puesto que al hacer girar la ruleta cada cromosoma tendrá una parte de esta en función del valor que cada uno tenga.

  - **Selección por torneo**: se comparan p individuos escogidos al azar y se selecciona el que tenga mejor aptitud. El que tenga mayor puntuación se reproduce y sustituye su descendencia al que tiene menor puntuación.

  - **Selección por ranking:** los individuos son ordenados de acuerdo a su ranking de fitness y su selección se basa en su puesto en ese ranking.

[87] RENNER, Gábor. Op. Cit. p.713
[88] CERVANTES POSADA, Mariamar. Op. Cit. 36

OTROS ESQUEMAS DE SELECCIÓN [89]

- **Selección elitista**: copia el mejor cromosoma o alguno de los mejores en la nueva población evitando que estos se pierdan tras la aplicación de los operadores de cruce y mutación.

- **Selección por estado estacionario:** la descendencia de los individuos seleccionados en cada generación vuelve a la población genética prexistente remplazando a algunos de los miembros menos aptos de la anterior generación.

- **Operador de cruce:** Después de realizar la selección de los cromosomas se procede a realizar el cruce entre dos de estos cromosomas con el fin intercambiar su material genético y mejorar sus valores fitness[90]. Los métodos de cruce más utilizados son: [91]

  - **Cruce de un punto:** Después de ser seleccionados los cromosomas padres cada uno se corta en un punto aleatorio generando dos segmentos: cabeza y cola en cada uno de ellos; se intercambian las colas y se generan los nuevos descendientes.

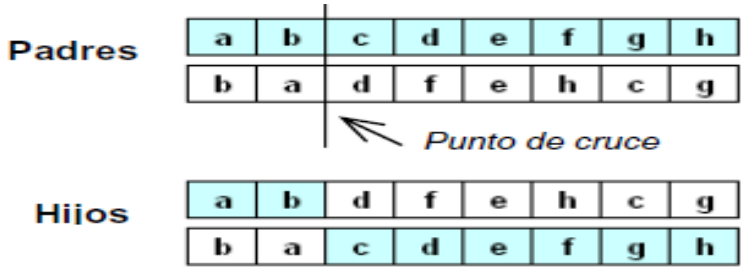
  - **Cruce de dos puntos:** los padres se cortan en dos puntos generando tres secciones en cada uno de los cromosomas.

  - **Cruce uniforme:** cada gen de la descendencia puede pertenecer a cualquiera de los padres. Se genera una máscara de cruce binario donde si el valor de una posición es 1 el gen se copia del padre y si

[89] ARRANZ DE LA PEÑA, Jorge. Op. Cit. p.3
[90] MITCHELL, Melanie. Op. Cit. p.8
[91] GESTAL POSE, Marcos. Introducción a los Algoritmos Genéticos. Universidad de la Coruña. p.9.

es 0 de la madre, estas restricciones pueden variar para crear una mezcla de genes de cada uno de los padres.

Figura 5. Representación gráfica cruce de un punto



Fuente: VÉLEZ, Mario Cesar y MONTOYA, José Alejandro. Metaheurísticos: una alternativa para la solución de problemas combinatorios en administración de operaciones. En: Rev.EIA.Esc.Ing.Antioq. 2007. 105 p.

- **Operador de mutación:** Este operador genera un nuevo individuo a través de variaciones en un cromosoma de manera que reordena al azar algunos de los bits del mismo. La mutación puede ocurrir en cada posición de bit en una cadena con cierta probabilidad, por lo general muy pequeña[92]. Se introduce para evitar la convergencia prematura aun óptimo local mediante el muestreo al azar de nuevos puntos en el espacio de búsqueda.[93] Los principales operadores de mutación son: [94]

  - **Flip bit:** se utiliza con codificación binaria, por lo cual este operador cambia el valor de 0 a 1 y viceversa.
  - **Intercambio entre actividades consecutivas:** se intercambia una actividad con la siguiente siempre que no tengan relación de precedencia.

[92] MITCHELL, Melanie. Op. Cit. p.8
[93] MARTÍ, Rafael. Op. Cit. p. 80
[94] CERVANTES POSADA, Mariamar. Op. Cit. p.38

- **Movimiento a la derecha:** se intercambia una actividad con una que este a su derecha.
- **Mutación aleatoria:** se modifica al azar la lista de actividades.

5.6.5 Condición de terminación El proceso iterativo del algoritmo puede finalizar cuando se cumpla una condición predeterminada[95]:

- Se ha alcanzado un número específico de generaciones.
- Que la aptitud de un candidato supere cierto valor definido.
- La población es demasiado homogénea, es decir, las configuraciones son similares y no existe más evolución.

5.7 ALGORITMO GENÉTICO DE CHU BEASLEY

El algoritmo genético de Chu-Beasley consiste en una modificación al algoritmo genético básico. Los pasos a seguir propuestos por Chu-Beasley en 1996 son:[96]

1. Generar una población inicial de N soluciones construidas al azar. Las soluciones iniciales pueden violar algunas de las restricciones del problema.

2. Decodificar la estructura solución para obtener el valor de la aptitud. Convencionalmente se debe tener en cuenta no sólo el costo de una solución, sino también el grado de inviabilidad de una solución. En lugar de penalizar la función cuando una solución no es factible se generan dos valores asociados a cada solución:

[95] GALLEGO, Ramón. Op. Cit. p.130
[96] CHU, P.C. Op. Cit. p.17-23

- Fitness: es igual al valor de la función objetivo
- Unfitness: es una medida de inviabilidad, cuando esta es 0 se dice que es una solución factible. Se calcula de la siguiente forma: $u_k = \sum_{i \in I} \max \left[0, (\sum_{j \in J, s_{kj}=i} r_{ij}) - b_i\right]$, donde r_{ij} es el recurso i para el trabajo j, y b_i es la cantidad de recursos i disponibles.

3. Seleccionar dos soluciones padres para la reproducción a través del método de torneo binario.

4. Aplicar un operador de cruce de un punto a los padres y así generar dos soluciones hijas, posteriormente aplicar el operador de mutación para el intercambio de dos genes seleccionados al azar en cada uno de los cromosomas hijos.

5. El individuo de la población con el valor más alto de unfitness se sustituye por uno de las soluciones hijas. Se debe tener en cuenta que una solución hija duplicada o con estructura idéntica a cualquiera de las estructuras de la población no puede entrar a esta.

6. Los pasos 3-5 se repiten hasta que las solucione hijas no duplicadas se han generado sin mejorar la mejor solución encontrada hasta el momento.

Este algoritmo es considerado un algoritmo muy competitivo para evaluar sistemas de gran tamaño y de gran complejidad matemática. El algoritmo genético de Chu-Beasley tiene como prioridad garantizar la diversidad entre los cromosomas que conforman la población durante todo el proceso, remplazando solo un cromosoma por cada iteración, bajo unas condiciones de factibilidad establecidas.[97]

[97]LONDOÑO POSSO, Julián; HINCAPIÉ ISAZA, Ricardo y GALLEGO RENDÓN, Ramón. Planeamiento de Redes de Baja Tensión, utilizando un Modelo Trifásico. En: *Ciencia e Ingeniería Neogranadina.* [en línea], 2011, Vol. 21, No. 2, p. 41-56.[Citado:2016-05-17].

El algoritmo debe satisfacer algunas características[98]:

- La población inicial debe ser totalmente heterogénea.
- El proceso de selección es por torneo.
- La recombinación genera dos hijos; uno de ellos será seleccionado a través de cualquier criterio.
- Solo se sustituye un individuo en la población en cada ciclo generacional.
- El cromosoma que ingresa a la población debe ser diferente a los demás(criterio de diversidad controlada)
- Es un algoritmo elitista porque el descendiente solo puede ingresar a la población si presenta una combinación función objetivo-infactibilidad mejor calidad que otra solución.

5.7.1 Conceptos y particularidades[99].

- **CRITERIO DE DIVERSIDAD:** es la distancia mínima de separación entre los individuos; si los individuos están codificados de forma binaria el criterio de diversidad seria la cantidad de posiciones del vector que contiene valores diferentes cuando se comparan las soluciones. Este criterio garantiza una mejor exploración del espacio de soluciones debido a que no permite soluciones repetidas.

- **CRITERIO DE ASPIRACIÓN**: es el mecanismo utilizado para no rechazar una solución que es mejor que las soluciones de la población pero que no cumple con el criterio de diversidad.

- **POBLACIÓN INICIAL:** La población inicial debe cumplir con la condición de que todos los individuos que la conforman deben ser diferentes, y en

[98] SOLARTE MARTÍNEZ, Guillermo. Op. Cit. p.97
[99] GALLEGO, Ramón. Op. Cit. p.154

algunos casos se exige que cumplan con un cierto criterio de diversidad (distancia mínima entre soluciones).

- **FUNCIÓN OBJETIVO**: es la función de costo y permite verificar el valor de la solución de mejor calidad, se utiliza para implementar los mecanismos de selección y sustitución cuando la población es factible.

- **FUNCIÓN DE INFACTIBILIDAD:** es el nivel de no cumplimiento de las restricciones y se utiliza para sustituir un individuo en la población cuando aparecen soluciones no factible.

- **SELECCIÓN:** se realiza selección por torneo, para lo cual se desarrollan dos torneos para elegir a cada uno de los padres quienes tienen que ser diferentes.

- **CRUCE:** Se definen p puntos de recombinación y se generan dos descendientes. Posteriormente se escoge a uno de los descendientes de forma aleatorio, por ruleta o torneo, para que ingrese a la población y el otro es eliminado.

- **MUTACIÓN**: se seleccionan de forma aleatoria n posiciones con el propósito de alterar su contenido, ya sea aumentándolo o disminuyéndolo.

- **MEJORIA LOCAL DE UN INDIVIDUO:** puede hacerse de dos formas: mejorando la función objetivo o disminuyendo la infactibilidad. Para eliminar o reducir la infactibilidad, el algoritmo puede auxiliarse de un algoritmo heurístico o constructivo basado en sensibilidad, el cual identifica elementos atractivos para ser adicionados o retirados con el propósito de hacer factible la solución.

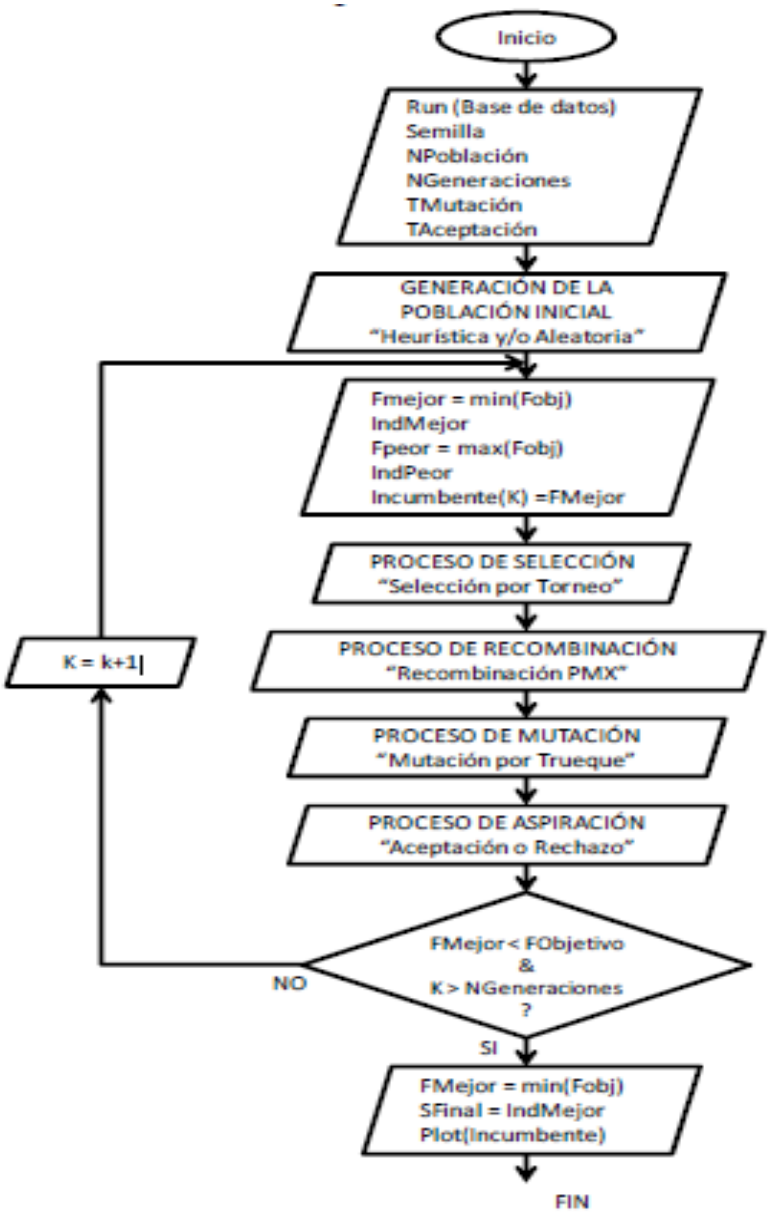
- **MODIFICACIÓN DE LA POBLACIÓN:** El descendiente debe sustituir al individuo de la población de peor calidad, siempre que el descendiente sea de mejor calidad y cumpla el criterio de diversidad establecido. Se presentan las siguientes alternativas:

    1. Un descendiente infactible y una población donde existen soluciones infactibles. En este caso el descendiente sustituirá al individuo más infactible de la población solo si este último es más infactible que el descendiente.
    2. Un descendiente infactible y una población donde no existen soluciones infactibles. En este caso el descendiente es eliminado
    3. Un descendiente factible y una población donde existen soluciones infactibles. En este caso el descendiente debe sustituir al individuo más infactible
    4. Un descendiente factible y una población donde solo existen soluciones factibles. En este caso el descendiente sustituirá al individuo con peor función objetivo, solo si este individuo de la población posee peor función objetivo que el descendiente.

    El descendiente debe respetar el criterio de diversidad para que la sustitución se realice, excepto en los casos donde se aplica el criterio de aspiración. De esta manera las soluciones de una población sólo serán eliminadas cuando aparezcan soluciones hijas con mejor calidad.

- **CRITERIO DE PARADA:** Será interrumpido si la incumbente (mejor solución encontrada durante el proceso) no mejora después de un número especificado de iteraciones o si se satisface el número máximo de ciclos generacionales.

Figura 6. Diagrama de flujo del algoritmo genético de Chu-Beasley.



Fuente: JIMÉNEZ MORALES, Ángela Patricia. Solución del problema de programación de Flow Shop Flexible empleando el algoritmo genético de Chu-Beasley. Universidad Tecnológica de Pereira. Pereira, Colombia. 2012. 100 p.

6. SISTEMA PRODUCTIVO BAJO ESTUDIO

6.1 DESCRIPCIÓN

El sistema productivo a analizar corresponde a un Flow Shop Hibrido Flexible (HFFS), el cual es un caso particular de la configuración Flow Shop, donde se tiene que un conjunto de *N* trabajos deben ser procesados en *M* etapas, donde cada etapa está compuesta por varias máquinas paralelas idénticas y cada trabajo puede omitir una o más etapas en su ruta de procesamiento[100]; otras características del problema de programación de tareas Flow Shop en general son[101]:

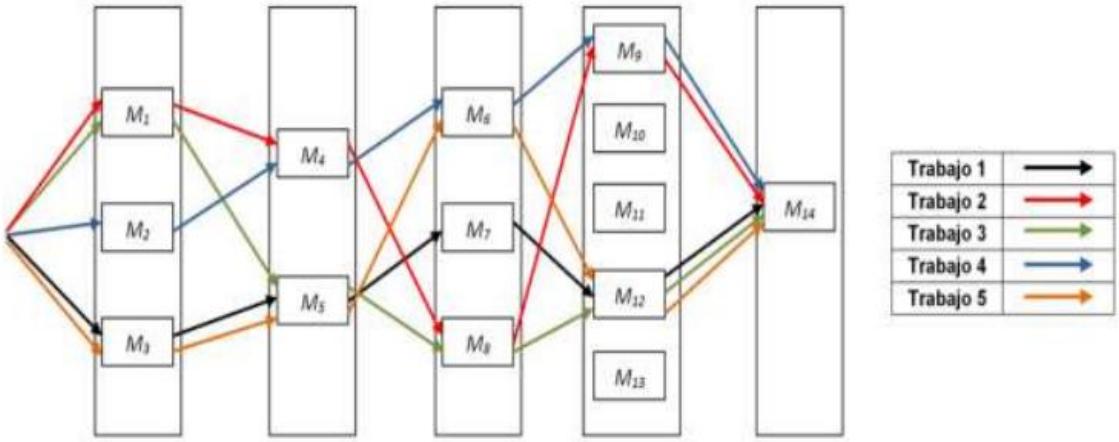
- ✓ Cada máquina está disponible continuamente y sin interrupciones.
- ✓ Cada máquina puede procesar un trabajo por vez.
- ✓ Cada trabajo sólo puede ser procesado por una máquina cada vez.
- ✓ Los tiempos de procesamiento de las tareas en las diferentes máquinas son determinados y fijos.
- ✓ Las tareas tienen la misma opción de ser programadas.
- ✓ Las operaciones en las máquinas, una vez iniciadas no deben ser interrumpidas.

Las tareas o productos siguen un flujo unidireccional a lo largo del sistema productivo, como lo describe la figura.

[100] JABBARIZADEH, F.; ZANDIEH, M. y TALEBI, D. Hybrid Flexible Flow shops with sequence-dependent setup times and machine availability constraints. En: *Computers & Industrial Engineering.* [Online]. 2009, Vol.57, p. 949-957. [Citado 2016-04-25].

[101] TORO OCAMPO, Eliana; RESTREPO GRISALES, Yov Steven y GRANADA ECHEVERRI, Mauricio. Algoritmo genético modificado aplicado al problema de secuenciamiento de tareas en sistemas de producción lineal - Flow Shop. En: *Scientia Et Technica*. [en línea]. 2006, Vol. XII, No.30, p. 285-290. [Citado: 2016-05-12].

Figura 7. Sistema productivo bajo estudio.



Fuente: LOPEZ, Juan; GIRALDO, Jaime y ARANGO, Jaime. Reducción del Tiempo de Terminación en la Programación de la Producción de una Línea de Flujo Híbrida Flexible (HFS). En: Información tecnológica. 2015. 165 p.

Existen tiempos entre el procesamiento de los trabajos, es decir, existe un periodo de tiempo destinado a la preparación o alistamiento de las máquinas entre la continuación de dos trabajos[102], este tiempo de configuración en cada máquina puede depender del trabajo inmediatamente anterior y es denominado como tiempos de alistamiento dependientes de la secuencia;[103]la unión del problema Flow Shop Híbrido Flexible con una configuración con tiempos dependientes de la secuencia da origen al problema SDST/HFFS[104] que considera líneas de flujo en el que concurren varias etapas con múltiples máquinas donde existe la posibilidad de que no todos los trabajos deban ser procesados en todas las etapas, considerando que existe un tiempo de configuración o *setup* de las máquinas entre dos trabajos diferentes y que este tiempo de configuración depende de la secuencia de trabajos a procesar.

[102]ÁNGEL-BELLO, Francisco, *et al* A heuristic approach for a scheduling problem with periodic maintenance and sequence-dependent setup times. En: *Computers & Mathematics with Applications*. [Online]. 2011, Vol. 61, No. 4, pp.797-808.[Citado:12-05-2016].
[103] JABBARIZADEH, F. Op. Cit. p.949
[104] DUPUY, Germán, *et al.* Métodos Metaheurísticos aplicados a procesos de gestión en una empresa. Laboratorio de Investigación en Sistemas Inteligentes. XVII Workshop de Investigadores en Ciencias de la Computación, Salta, 2015. Disponible en : http://hdl.handle.net/10915/45319

7. FORMULACIÓN DEL MODELO MATEMATICO

Se deben tener en cuenta las siguientes consideraciones:

- Las maquinas en cada etapa son idénticas.
- Las maquinas pueden procesar todos los trabajos.
- Cada trabajo puede ser procesado en cada etapa en solo una máquina.
- Una maquina no puede procesar más de un trabajo a la vez.
- Los tiempos de alistamiento son dependientes de la secuencia.
- No hay límites de trabajos a ser asignados a una máquina.
- Todas las máquinas están disponibles en el instante $\boldsymbol{t = 0}$.
- El tiempo de procesamiento de cada trabajo depende de la etapa.
- Una vez terminado el procesamiento de un trabajo en una etapa, este debe ser asignado a cualquiera de las máquinas disponibles en la etapa siguiente, en el caso de no existir máquinas disponibles la maquina actual se bloquea, así que no se considera espera, sin embargo se debe respetar la secuencia.
- En una misma etapa se pueden procesar varias tareas simultáneamente, pero siempre teniendo en cuenta la secuencia.

7.1. MODELO DE PROGRAMACIÓN LINEAL ENTERA MIXTA

A continuación se presentan cada uno de los conjuntos, parámetros y variables consideradas en el modelo matemático teniendo como base el modelo presentado en el 2010 por Gómez Gasquet.[105]

[105] GÓMEZ GASQUET, Pedro. Programación de la producción en un taller de flujo hibrido sujeto a incertidumbre: arquitectura y algoritmos. Aplicación a la industria cerámica. Valencia, 2010. Tesis (Doctorado en Gestión de la Cadena de Suministro e Integración Empresarial). Universidad Politécnica de Valencia

- **Conjuntos**

j, i: Subíndices para los trabajos. *i=1,2,3….N+1* *j=0,1,2….N*

k: Subíndice para las etapas *k=1,2,3….K*

m: Subíndice para las máquinas, depende de la etapa. *m=1,2,3…M*

- **Parámetros**

$S(i, m, k)$: Tiempo de alistamiento del trabajo i en la máquina m, si es el primero que se programa en la máquina m en la etapa k.

$P(i, k)$: Tiempo de procesamiento o de ejecución del trabajo i en la etapa k.

$ST(i, j, m, k)$: Tiempo de preparación de la máquina m en la etapa k para pasar de realizar el trabajo j al trabajo i.

$M(i, k)$: Matriz que determina si el trabajo i es procesado en la etapa k.

- **Variables**

Continuas

$TI(i, m, k)$: Momento de inicio del trabajo i en la máquina m en la etapa k.

$TF(i, m, k)$: Momento de finalización del trabajo i en la máquina m en la etapa k.

$TP(i, m, k)$: Suma de los tiempos de preparación y de procesamiento del trabajo i en la máquina m en la etapa k.

$TPr(i, m, k)$: Tiempo de preparación del trabajo i en la máquina m en la etapa k.

Binarias

$$Y(i, m, k): \left\{ \begin{matrix} 1 \text{ si el trabajo i es asignado a la máquina m e n la etapa k} \\ 0 \text{ En caso contrario} \end{matrix} \right\}$$

$X(j,i,m,k)$: {1 si el trabajo j es procesado antes del trabajo i en la máquina m en la etapa k / 0 En caso contrario}

$X(0,i,m,k)$: {1 si el trabajo i es el primero en ser procesado en la máquina m en la etapa k / 0 En caso contrario}

$X(j,N+1,m,k)$: {1 si el trabajo j es el ultimo en ser procesado en la maquina m en la etapa k / 0 En caso contrario}

- **Modelo Matemático**

$$Minimizar\ Z = max\ TF_{(i,m,k)} \qquad (1)$$

Sujeto a:

$$\sum_{m=1} Y_{(i,m,k)} \leq 1 \qquad \forall\, i,k; \qquad (2)$$

$$\sum_{j=0} X_{(j,i,m,k)} - Y_{(i,m,k)} = 0 \qquad \forall\, i,m,k;\ j \neq i; \qquad (3)$$

$$\sum_{i=1} X_{(j,i,m,k)} - Y_{(i,m,k)} = 0 \qquad \forall\, j,m,k;\ j \neq i; \qquad (4)$$

$$\sum_{i=1} X_{(0,i,m,k)} = 1 \qquad \forall\, m,k;\ j \neq i; \qquad (5)$$

$$\sum_{j=1} X_{(j,N+1,m,k)} = 1 \qquad \forall\, m,k;\ j \neq i; \qquad (6)$$

$$\sum_{m=1} Y_{(i,m,k)} = M_{(i,k)} \qquad \forall\, i,k; \qquad (7)$$

$$TF_{(i,m,k)} \geq TI_{(i,m,k)} + TP_{(i,m,k)} \qquad \forall\, i,m,k; \qquad (8)$$

$$TPr_{(i,m,k)} \geq S_{(i,m,k)} \times X_{(0,i,m,k)} \qquad \forall\, i,m,k; \qquad (9)$$

$$TPr_{(i,m,k)} \geq ST_{(i,m,k)} \times X_{(j,i,m,k)} \qquad \forall\, j,i,m,k;\ j \neq 0\ ;j \neq i \qquad (10)$$

$$TP_{(i,m,k)} \geq (TPr_{(i,m,k)} + P_{(i,k)}) \times Y_{(i,m,k)} \qquad \forall\, i,m,k; \qquad (11)$$

$$TI_{(i,m,k)} \geq \sum_m TF_{(i,m,k-1)} \qquad \forall\, i,m,k;\ k \neq 1 \qquad (12)$$

$$TI_{(i,m,k)} \geq TF_{(j,m,k)} \times X_{(j,i,m,k)} \qquad \forall\, i,m,k;\ j \neq i \qquad (13)$$

1. Define la función objetivo del problema, la cual corresponde a la minimización del makespan.

2. Asegura que cada trabajo en cada etapa sea asignado máximo a una máquina.

3. Una vez asignado el trabajo a una máquina, este debe tener un predecesor que puede ser cualquier trabajo o la situación 0 (Inicio de la secuenciación).

4. Una vez asignado un trabajo a una máquina esta debe tener un trabajo sucesor, puede ser cualquiera de los otros trabajos o la situación final de la máquina.

5. Existe un trabajo secuenciado en primer lugar para cada máquina en cada etapa.

6. Existe al menos un trabajo antes de finalizar la secuencia en cada máquina en cada etapa.

7. Contempla si un trabajo i debe ser procesado en la etapa k, teniendo en cuenta que existen trabajos que saltan etapas.

8. El tiempo de finalización de cada trabajo en cada máquina en cada etapa es igual al momento de inicio más la suma del tiempo de preparación empleado y el tiempo de procesamiento correspondiente.

   El tiempo de preparación empleado presenta dos escenarios presentados en las ecuaciones (9) y (10)

9. Contempla el tiempo de preparación empleado cuando el trabajo i es el primer trabajo en procesarse en la máquina m en la etapa k.

10. Contempla el tiempo de preparación empleado cuando el trabajo i es procesado después del trabajo j en la máquina m en la etapa k.

11. Sumatoria del tiempo de preparación empleado y del tiempo de procesamiento correspondiente al trabajo i en la máquina m en la etapa k, teniendo en cuenta que el trabajo fue asignado a esa máquina en esa etapa.

El tiempo de inicio de cada trabajo está condicionado a dos escenarios presentados en las ecuaciones (12) y (13)

12. El tiempo de inicio del trabajo i en la máquina m en la etapa k es mayor o igual al tiempo de finalización del trabajo i en la etapa anterior.(Excepto para la primera etapa).

13. El tiempo de inicio del trabajo i en la máquina m en la etapa k es mayor o igual al momento terminación del trabajo anterior en la misma máquina en la etapa k.

8. DISEÑO DEL ALGORITMO

8.1 REPRESENTACIÓN DEL CROMOSOMA

Se define un vector de permutaciones, en el cual la longitud del mismo corresponde a la cantidad de tareas o trabajos, garantizando que ninguna tarea se repita y que todas sean terminadas. La posición que cada trabajo ocupa en el cromosoma indica la secuencia en que pasaran a las máquinas en cada etapa. A continuación se muestra la representación para un problema con 7 tareas o trabajos:

Figura 8. Representación del cromosoma

De este modo el primer trabajo en ser procesado será el trabajo número 5, seguidamente será el trabajo número 3 y así sucesivamente hasta finalizar con el procesamiento del trabajo número 4

8.2 GENERACIÓN DE LA POBLACIÓN INICIAL

La población inicial del algoritmo se genera a través de una matriz de tamaño *m* x *n* donde *m* es el tamaño de la población y *n* el número de trabajos, cada solución creada en la población inicial es una secuencia permutacional generada de manera aleatoria como se describió en el literal Representación del cromosoma.

El algoritmo no permite que se creen dos soluciones iguales, es por esto que cada solución de la población inicial es comparada con sus antecesoras para de esta manera determinar que no se repita y obtener mayor diversidad de soluciones; si

la solución generada se encuentra repetida esta es sustituida por una nueva solución que también debe ser evaluada para determinar que sea única.

8.3 CALCULO DE LA FUNCIÓN OBJETIVO O MAKESPAN

Como primer paso se realiza la asignación aleatoria de las máquinas disponibles en cada etapa teniendo en cuenta que no es posible asignar la misma máquina dos veces seguidas excepto en aquellas etapas donde solo existe una máquina disponible. Posteriormente se calcula el tiempo de preparación de cada trabajo en cada etapa si este es el primero en procesarse en cada máquina o si es dependiente de la secuencia.

Finalmente se calcula el tiempo de terminación del procesamiento en cada etapa atendiendo que este es la suma del tiempo de preparación y el tiempo de procesamiento; es importante resaltar que el makespan o tiempo de terminación se calcula para cada trabajo en cada etapa a partir del valor máximo entre el tiempo de finalización del mismo trabajo en la etapa anterior y el tiempo de terminación del trabajo anterior en la misma máquina.

8.4 PROCESO DE SELECCIÓN

El proceso de selección es el encargado de seleccionar los cromosomas que serán padres y darán origen a las dos soluciones hijas. El operador de selección utilizado es torneo.

Se seleccionan de manera aleatoria dos cromosomas de la población, entre los cuales se comprara el valor de makespan correspondiente a cada uno de ellos, y se selecciona aquel que posea un makespan inferior. Este proceso se realiza dos veces para dar origen a las dos padres.

8.5. PROCESO DE CRUCE

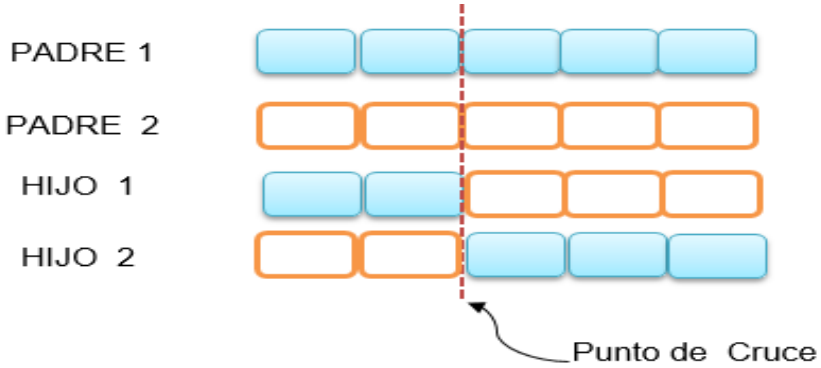
Este proceso inicia con la selección del punto de cruce de manera aleatoria; es decir, el punto de cruce es un número entero entre 1 y n-1, donde n es el número de trabajos. Seleccionado este punto los descendientes se crean de la siguiente manera:

- Hijo 1: desde la posición 1 del padre 1 hasta la posición determinada por el punto de cruce en el padre 1 y desde la posición determinada por el punto de cruce en el padre 2 más uno hasta la posición final del padre 2

- Hijo 2: desde la posición 1 del padre 2 hasta la posición determinada por el punto de cruce en el padre 2 y desde la posición determinada por el punto de cruce en el padre 1 más uno hasta la posición final del padre 1.

Al realizarse el proceso de cruzamiento se debe tener en cuenta que la naturaleza del problema no permite que se repita algún número en la secuencia y es muy posible que al momento de generarse los hijos existan trabajos repetidos y por consecuencia trabajos olvidados; para solucionar esta anomalía de los cromosomas hijos se diseñó en el algoritmo un proceso de identificación de trabajos ya secuenciados y trabajos olvidados; los trabajos olvidados o faltantes ingresan al cromosoma a reemplazar a los trabajos que se repiten, esta reubicación de trabajos se realiza de manera aleatoria.

Figura 9. Proceso de cruce

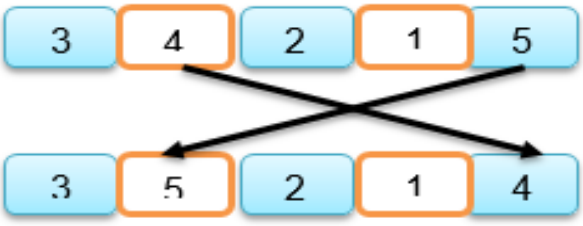


8.6 MUTACIÓN

Obtenidos los hijos se genera un numero aleatorio entre 0 y 1, si este número es menor a la tasa de mutación establecida previamente en los parámetros ajustables del algoritmo se ejecuta el proceso de mutación de lo contrario los hijos pasan este proceso sin modificaciones.

Se aplica la mutación por intercambio, es decir, se seleccionan aleatoriamente dos posiciones del cromosoma y se intercambia la información genética en esos dos puntos.

Figura 10. Mutación



8.7 SELECCIÓN DE DESCENDIENTE

Con las dos soluciones hijas ya establecidas se realiza un proceso de selección de una sola de ellas para ingresar a la población; el proceso de selección del descendiente se realiza por ruleta donde el hijo con mejor makespan tiene mayor probabilidad de ser escogido mediante la generación de un número aleatorio entre 0 y 1.

8.8 PROCESO DE ACEPTACION

Para determinar si el descendiente es apto para ingresar a la población se aplican los siguientes criterios:

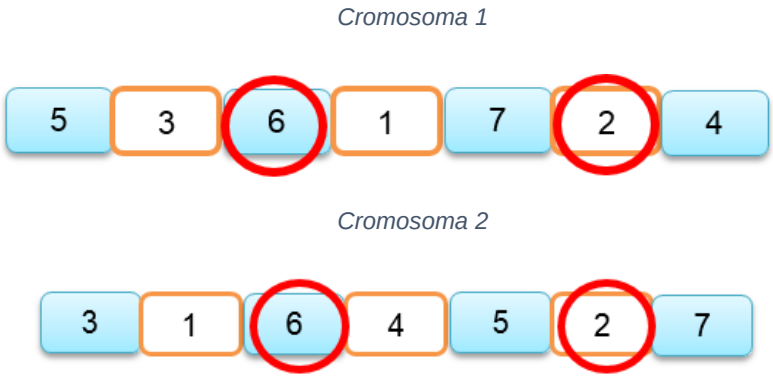
- **CRITERIO DE UNICIDAD**

  Se verifica que no exista una solución dentro de la población idéntica al descendiente, de ser así el descendiente se descarta y no puede ingresar a la población.

- **CRITERIO DE DIVERSIDAD**

  Se verifica que la distancia mínima de separación del descendiente con cada una de las soluciones de la población cumpla con la diversidad aceptada establecida en los parámetros iniciales. A continuación la representación de este criterio entre dos cromosomas con diversidad de 2 genes.

Figura 11. Diversidad Aceptada de 2 genes



Entre los dos cromosomas representados existen 2 genes donde el trabajo secuenciado es el mismo, es decir tiene una diversidad de 2 genes. Entre mayor sea el numero denominado diversidad más parecidas serán las soluciones propuestas. Si este número es menor se asegura una mejor exploración del espacio de soluciones.[106]

Si el descendiente cumple con los dos criterios ya mencionados se acepta su ingreso a la población y el descendiente reemplaza al cromosoma con mayor makespan en la población.

- **CRITERIO DE ASPIRACION**

  Se aplica este criterio cuando el descendiente no cumple con el criterio de diversidad pero su makespan es mejor que el makespan de la incumbente, es decir, de la solución con mejor makespan; por lo tanto se da la posibilidad para que este haga parte de la población lo cual conlleva a la eliminación de la solución con peor makespan.

[106] [106] GALLEGO, Ramón. Op. Cit. p.154

8.9 CRITERIOS DE PARADA

El algoritmo detendrá su ejecución con el cumplimiento de alguno de los dos criterios de parada siguientes:

1. Si la cantidad de iteraciones alcanza el número máximo de iteraciones definido en los parámetros iniciales.
2. Si después de 100 iteraciones consecutivas no existe mejora.

8.10 PARAMETROS Y MATRICES

Para la ejecución del algoritmo se deben tener en cuenta algunos parámetros iniciales y las matrices de tiempos de alistamiento y de procesamiento; el algoritmo programado tal como se puede consultar en el Anexo 5 contempla los siguientes parámetros iniciales y matrices:

Parámetros iniciales:

- Cantidad de trabajos a procesar
- Número de etapas
- Cantidad de máquinas por etapa.

Matrices:

- **MTP:** Matriz de tiempos de procesamiento de los trabajos en cada etapa. Tamaño: *n* x *k,* donde n son los trabajos y k el número de etapas.
- **MTA:** Matriz de tiempos de alistamiento de los trabajos en cada etapa cuando estos son los primeros en ser procesados en cada máquina. Tamaño *n* x *k,* donde n son los trabajos y k el número de etapas.

- **MTAS:** Matriz de tiempos de alistamiento relacionada con la secuencia de trabajos que se esté ejecutando. Tamaño: Número de secuencias disponibles x etapa.
- **Mtas:** Matriz de secuencia de los trabajos, esta matriz describe el número de secuencia correspondiente teniendo en cuenta la identificación del trabajo actual y anterior. Tamaño: *n* x *n*. Nota: está conformada por una diagonal principal de 0 debido a que no existe tiempo de preparación para ejecutar dos veces seguidas el mismo trabajo.

En el Anexo 3 se encuentra un ejemplo de cada una de las matrices anteriormente expuestas, este anexo se elaboró para ejemplificar la ejecución del algoritmo.

La salida del algoritmo muestra cual es la mejor solución encontrada de la siguiente manera.

SOLUCIÓN: 4 5 3 2 1

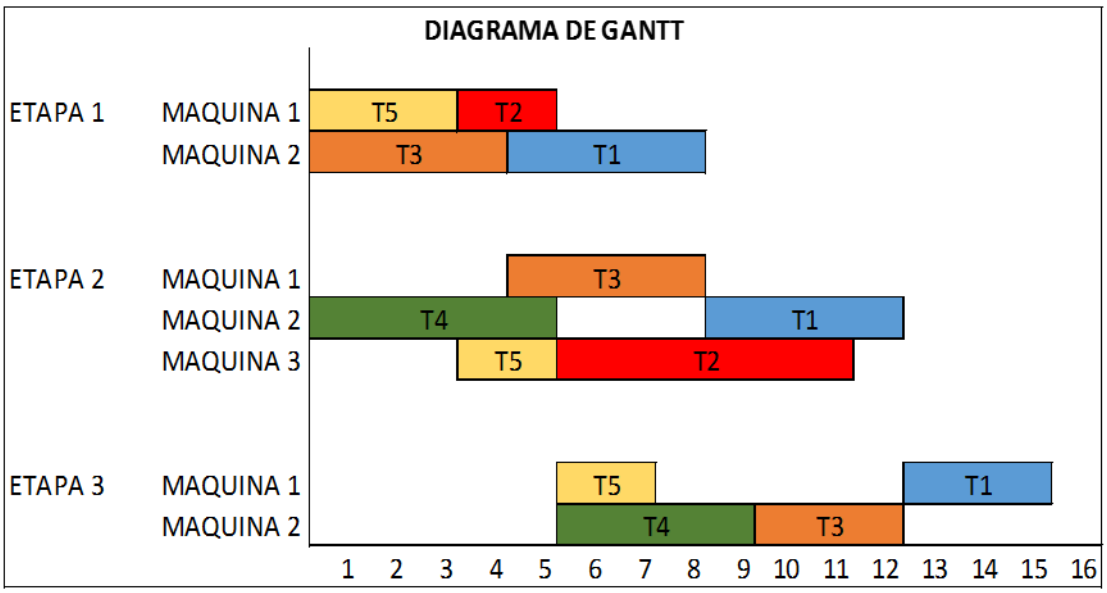
Además, también proporciona la asignación de maquinaria por etapa para cada uno de los trabajos.

Tabla 2. Asignación de maquinaria por etapa

ASIGNACIÒN DE MAQUINARIA					
	TRABAJOS				
ETAPAS	**4**	**5**	**3**	**2**	**1**
1	0	1	2	1	2
2	2	3	1	3	2
3	2	1	2	0	1

A continuación, se presenta el diagrama de Gantt que representa el Makespan del cromosoma ya mencionado; en el cual se encuentra la línea de tiempo que siguen todos los trabajos los cuales terminan de ser procesados en 15 unidades de tiempo para el ejemplo del anexo 3.

Figura 12. Diagrama de Gantt



En el anexo 4 se encuentra el diagrama de flujo empleado en la programación, el cual tiene en cuenta todas las anteriores observaciones o parámetros.

En el anexo 1 se encuentra el código de programación del algoritmo con el respectivo análisis de cada una de las funciones creadas; además en el archivo adjunto o anexo 5 se encuentra el programa ejecutable en el lenguaje Matlab.

9. VALIDACIÓN DEL ALGORITMO

El algoritmo genético modificado de Chu-Beasley fue programado en Matlab y se probó en un computador Intel ® Core ™ i5- 3632 QM CPU 3.2 GHz, Memoria RAM de 8 GB, con un sistema operacional Windows 7 de 64 bits.

En la validación se utilizaron las instancias disponibles en http://soa.iti.es/instancias-problemas, utilizadas por Naderi, Ruiz y Zandieh[107] en el 2010, en donde los autores presentan tres variaciones del algoritmo Búsqueda Local (ILS). En la tabla 2 se muestran las características del conjunto de instancias.

Tabla 3. Características de las instancias

FAMILIA	CANTIDAD DE PROBLEMAS	No. DE TRABAJOS	No. De ETAPAS
20X2	80	20	2
20X4	80	20	4
20X8	80	20	8
50X2	80	50	2
50X4	80	50	4
50X8	80	50	8
80X2	80	80	2
80X4	80	80	4
80X8	80	80	8
120X2	80	120	2
120X4	80	120	4
120X8	80	120	8

[107] NADERI, B; RUIZ, Rubén y ZANDIEH M. Algorithms for a realistic variant of flow shop scheduling. En: *Computers Ops. Res.* [online],2010. Vol. 36, p.236-246. [Citado: 2016-17-08].

Según estas instancias existen 12 familias que están agrupadas por cantidad de trabajos y número de etapas, en cada familia propuesta existen 80 problemas para su análisis, de los cuales para esta investigación fue escogido aleatoriamente un problema de cada una de las familias. La cantidad de máquinas disponibles en cada familia para cada estación varía de acuerdo a cada problema. Para la ejecución se determinó variar el número de iteraciones, el tamaño de la población y la diversidad aceptada de la siguiente manera:

- Número de iteraciones: 1000-1500
- Tamaño de la población: 50-100
- Diversidad aceptada: 2-5

Se ejecutaron las ocho combinaciones posibles de acuerdo a la variación de los parámetros presentada anteriormente, para cada una de las combinaciones posibles se ejecutó el algoritmo 30 veces de acuerdo a las investigaciones de Naderi, Ruiz y Zandieh[108] en el 2010 y Branz [109] en el 2013.

A continuación se presentan las tablas donde se compara el Makespan encontrado con el algoritmo genético modificado de Chu-Beasley y el mejor conocido reportado en el portal http://soa.iti.es/instancias-problemas que corresponden a las encontradas en la investigación de Naderi, Ruiz y Zandieh denominada Algorithms for a realistic variant of flowshop scheduling[110], también se presenta la media y desviación estándar para cada ejecución del algoritmo.

[108] NADERI, B; RUIZ, Rubén y ZANDIEH M. Op. Cit. p.236

[109] BRAZ MOREIRA, Neuma Eufrazio. Heurísticas para Minimização do Makespan na Classe de Problemas de Sequenciamento Flowshop Híbrido e Flexível com Tempo de Setup Dependente da Sequência: Com e Sem Eventos de Quebra de Máquinas. Belo Horizonte, 2013. Trabajo de Grado (Maestria en Modelamiento Matemático y Computacional). Centro federal de educação Tecnológica de minas gerais. [Citado: 2016-18-08]. Disponible en: <http://www.files.scire.net.br/atrio/cefet-mgppgmmc_upl//THESIS/176/neumaeufraziobrazmoreira.pdf>.

[110] NADERI, B; RUIZ, Rubén y ZANDIEH M. Op. Cit. p.237

- Población: tamaño de la población.
- Diversidad: diversidad aceptada (como se expone en el criterio de diversidad).
- Iteraciones: número de iteraciones.
- MSC: mejor solución conocida o reportada.
- Media: Media del makespan.
- Desviación: desviación estándar del makespan.
- % DIF= porcentaje de diferencia entre la mejor solución y el mínimo makespan encontrado por el AGCB

Tabla 4. INSTANCIA 20x2

INSTANCIA 20X2							
POBLACION	**DIVERSIDAD**	**ITERACIONES**	**MSC**	**MIN AGCB**	**MEDIA**	**DESVIACION**	**% DIF**
100	2	1000	634	658	682,9	9,0	3,79
100	5	1000	634	652	688,6	14,2	2,84
100	2	1500	634	657	673,7	9,5	3,63
100	5	1500	634	660	683,2	10,5	4,10
50	2	1000	634	661	679,6	10,6	4,26
50	5	1000	634	648	683,0	14,1	2,21
50	2	1500	634	651	672,0	10,0	2,68
50	5	1500	634	663	676,8	7,4	4,57

En la instancia 20X2 se evidencia que el algoritmo propuesto no alcanza en ninguna combinación a la mejor solución, sin embargo el mínimo makespan se encuentra con una población de 50 individuos, diversidad de 5 genes y 1000 iteraciones, y con una diferencia del 2,21%. En todas las combinaciones el makespan no supera el 5% de diferencia con la mejor solución.

Tabla 5. INSTANCIA 20X4

INSTANCIA 20X4							
POBLACION	**DIVERSIDAD**	**ITERACIONES**	**MSC**	**MIN AGCB**	**MEDIA**	**DESVIACION**	**% DIF**
100	2	1000	1322	1450	1571,6	51,7	9,68
100	5	1000	1322	1494	1593,4	40,8	13,01
100	2	1500	1322	1407	1493,4	40,7	6,43
100	5	1500	1322	1455	1561,9	42,3	10,06
50	2	1000	1322	1401	1533,2	55,9	5,98
50	5	1000	1322	1416	1562,0	57,6	7,11
50	2	1500	1322	1365	1492,5	50,9	3,25
50	5	1500	1322	1458	1545,8	47,5	10,29

En la instancia 20X4 el algoritmo propuesto no alcanza en ninguna combinación a la mejor solución, sin embargo el mínimo makespan se encuentra con una población de 50 individuos, diversidad de 2 genes y 1500 iteraciones, además presenta una diferencia del 3,25%. Es de resaltar que la desviación en todas las combinaciones es considerable además que el porcentaje de diferencia alcanza un máximo de 13%.

Tabla 6.INSTANCIA 20X8

INSTANCIA 20X8							
POBLACION	**DIVERSIDAD**	**ITERACIONES**	**MSC**	**MIN AGCB**	**MEDIA**	**DESVIACION**	**% DIF**
100	2	1000	933	1048	1073,1	16,2	12,33
100	5	1000	933	1042	1079,9	19,0	11,68
100	2	1500	933	1026	1056,8	16,1	9,97
100	5	1500	933	1026	1062,0	17,4	9,97
50	2	1000	933	1018	1056,7	18,2	9,11
50	5	1000	933	1045	1080,3	20,4	12,00
50	2	1500	933	1025	1050,0	14,7	9,86
50	5	1500	933	1020	1059,1	17,8	9,32

En la instancia 20X8 el algoritmo propuesto no alcanza en ninguna combinación a la mejor solución, sin embargo el mínimo makespan se encuentra con una población de 50 individuos, diversidad de 5 genes y 1500 iteraciones, además presenta una diferencia del 9,32%. El porcentaje de diferencia alcanza un máximo de 12,33%.

Tabla 7. INSTANCIA 50X2

INSTANCIA 50X2							
POBLACION	**DIVERSIDAD**	**ITERACIONES**	**MSC**	**MIN AGCB**	**MEDIA**	**DESVIACION**	**% DIF**
100	2	1000	1437	1410	1443,0	16,6	-1,88
100	5	1000	1437	1391	1459,0	32,7	-3,20
100	2	1500	1437	1407	1430,1	12,5	-2,09
100	5	1500	1437	1401	1434,9	13,5	-2,51
50	2	1000	1437	1397	1432,7	12,7	-2,78
50	5	1000	1437	1405	1433,3	14,9	-2,23
50	2	1500	1437	1385	1417,2	13,7	-3,62
50	5	1500	1437	1396	1421,6	15,8	-2,85

En la instancia 50X2 el algoritmo propuesto alcanza a la mejor solución en todas las combinaciones, el mínimo makespan se encuentra con la combinación de tamaño de la población de 50 individuos, diversidad de 2 genes y 1500 iteraciones, con una mejora del 3,62% respecto a la mejor solución. En las combinaciones se encontró una desviación mínima de 12 y máxima de 33.

Tabla 8. INSTANCIA 50X4

INSTANCIA 50X4							
POBLACION	**DIVERSIDAD**	**ITERACIONES**	**MSC**	**MIN AGCB**	**MEDIA**	**DESVIACION**	**% DIF**
100	2	1000	2447	2590	2635,13	26,38	5,84
100	5	1000	2447	2581	2642,67	29,59	5,48
100	2	1500	2447	2584	2584,80	30,68	5,60
100	5	1500	2447	2552	2600,07	26,53	4,29
50	2	1000	2447	2534	2605,90	30,82	3,56
50	5	1000	2447	2561	2620,90	28,13	4,66
50	2	1500	2447	2524	2577,77	32,09	3,15
50	5	1500	2447	2502	2579,30	31,49	2,25

En la instancia 50X4 el algoritmo propuesto no alcanza a la mejor solución en ninguna de las combinaciones, el mínimo makespan se encuentra con la combinación de tamaño de la población de 50 individuos, diversidad de 5 genes y 1500 iteraciones, con una diferencia del 2,25% respecto a la mejor solución. En todas las combinaciones el porcentaje de diferencia es inferior al 6%.

Tabla 9. INSTANCIA 50X8

INSTANCIA 50X8							
POBLACION	**DIVERSIDAD**	**ITERACIONES**	**MSC**	**MIN AGCB**	**MEDIA**	**DESVIACION**	**% DIF**
100	2	1000	1053	1400	1455,1	26,2	32,95
100	5	1000	1053	1395	1469,7	23,3	32,48
100	2	1500	1053	1364	1430,1	28,5	29,53
100	5	1500	1053	1378	1435,1	27,5	30,86
50	2	1000	1053	1374	1448,8	33,2	30,48
50	5	1000	1053	1353	1445,7	38,1	28,49
50	2	1500	1053	1413	1459,7	25,0	34,19
50	5	1500	1053	1351	1428,0	25,3	28,30

En la instancia 50X8 el algoritmo propuesto no alcanza a la mejor solución en ninguna de las combinaciones, el mínimo makespan se encentra con la combinación de tamaño de la población de 50 individuos, diversidad de 5 genes y 1500 iteraciones, con una diferencia del 28,3% respecto a la mejor solución. Todas las combinaciones tienen un porcentaje de diferencia considerable entre el 28% y 34%.

Tabla 10. INSTANCIA 80X2

INSTANCIA 80X2							
POBLACION	**DIVERSIDAD**	**ITERACIONES**	**MSC**	**MIN AGCB**	**MEDIA**	**DESVIACION**	**% DIF**
100	2	1000	2086	2260	2295.9	26,2	8,34
100	5	1000	2086	2272	2301.7	23,3	8,92
100	2	1500	2086	2239	2279.2	28,5	7,33
100	5	1500	2086	2236	2277.3	27,5	7,19
50	2	1000	2086	2226	2275.8	33,2	6,71
50	5	1000	2086	2247	2284.0	38,1	7,72
50	2	1500	2086	2229	2272.8	25,0	6,86
50	5	1500	2086	2210	2268.9	25,3	5,94

En la instancia 80X2 el algoritmo propuesto no alcanzo a la mejor solución en ninguna de las instancias, el mínimo makespan se encuentra con la combinación tamaño de población 50, diversidad de 5 genes y 1500 iteraciones, con una diferencia de 5,94%. La desviación se encuentra entre 25 y 38.

Tabla 11. INSTANCIA 80X4

INSTANCIA 80X4							
POBLACION	**DIVERSIDAD**	**ITERACIONES**	**MSC**	**MIN AGCB**	**MEDIA**	**DESVIACION**	**% DIF**
100	2	1000	2276	2002	2197,7	172,1	-12,04
100	5	1000	2276	1977	2233,3	194,2	-13,14
100	2	1500	2276	1859	2091,3	172,5	-18,32
100	5	1500	2276	1853	2047,1	156,1	-18,59
50	2	1000	2276	1879	2283,2	246,4	-17,44
50	5	1000	2276	1819	2209,1	235,3	-20,08
50	2	1500	2276	1834	2223,2	240,6	-19,42
50	5	1500	2276	1850	2197,7	202,6	-18,72

En la instancia 80X4 el algoritmo propuesto alcanzo la mejor solución en todas las instancias, el mínimo makespan se encuentra con la combinación de tamaño de la población de 50 individuos, diversidad de 5 genes y 1000 iteraciones, con una mejora del 20,08% respecto a la mejor solución. En todas las combinaciones se encontró una desviación bastante considerable.

Tabla 12. INSTANCIA 80X8

INSTANCIA 80X8							
POBLACION	**DIVERSIDAD**	**ITERACIONES**	**MSC**	**MIN AGCB**	**MEDIA**	**DESVIACION**	**% DIF**
100	2	1000	2689	2985	3063,6	34,8	11,01
100	5	1000	2689	3014	3084,3	29,2	12,09
100	2	1500	2689	2970	3038,4	32,1	10,45
100	5	1500	2689	2939	3037,6	45,0	9,30
50	2	1000	2689	3001	3068,2	42,0	11,60
50	5	1000	2689	2998	3058,2	31,8	11,49
50	2	1500	2689	2979	3030,4	31,4	10,78
50	5	1500	2689	2949	3033,5	34,8	9,67

En la instancia 80X8 el algoritmo propuesto no alcanzo a la mejor solución en ninguna de las instancias, el mínimo makespan se encuentra con la combinación de tamaño de población de 100 individuos, diversidad de 5 genes y 1500 iteraciones, con una diferencia de 9,30%.

Tabla 13. INSTANCIA 120X2

INSTANCIA 120X2							
POBLACION	**DIVERSIDAD**	**ITERACIONES**	**MSC**	**MIN AGCB**	**MEDIA**	**DESVIACION**	**% DIF**
100	2	1000	3356	2694	2954,0	210,1	-19,73
100	5	1000	3356	2750	2986,3	219,7	-18,06
100	2	1500	3356	2676	2867,7	178,6	-20,26
100	5	1500	3356	2649	2881,5	217,6	-21,07
50	2	1000	3356	2698	3099,2	317,0	-19,61
50	5	1000	3356	2589	3089,6	327,0	-22,85
50	2	1500	3356	2630	3117,5	344,5	-21,63
50	5	1500	3356	2545	3002,9	359,7	-24,17

En la instancia 120X2 el algoritmo propuesto supero a la mejor solución en todas las instancias, el mínimo makespan se encuentra con la combinación de tamaño de población de 50 individuos, diversidad de 5 genes y 1500 iteraciones, con una mejora del 24,17% respecto a la mejor solución. En todas las iteraciones la desviación es bastante considerable.

Tabla 14. INSTANCIA 120X4

INSTANCIA 120X4							
POBLACION	**DIVERSIDAD**	**ITERACIONES**	**MSC**	**MIN AGCB**	**MEDIA**	**DESVIACION**	**% DIF**
100	2	1000	5771	6255	6312,5	36,7	8,39
100	5	1000	5771	6206	6314,4	52,5	7,54
100	2	1500	5771	6172	6236,6	31,1	6,95
100	5	1500	5771	6146	6238,5	41,7	6,50
50	2	1000	5771	6166	6243,6	48,5	6,84
50	5	1000	5771	6154	6252,0	48,3	6,64
50	2	1500	5771	6110	6201,0	53,5	5,87
50	5	1500	5771	6078	6213,0	55,4	5,32

En la instancia 120X4 el algoritmo propuesto no alcanza a la mejor solución en ninguna de las combinaciones, el mínimo makespan se encuentra con la combinación de tamaño de la población de 50 individuos, diversidad de 5 genes y 1500 iteraciones, con una diferencia de 5,32%.

Tabla 15. INSTANCIA 120X8

INSTANCIA 120X8							
POBLACION	**DIVERSIDAD**	**ITERACIONES**	**MSC**	**MIN AGCB**	**MEDIA**	**DESVIACION**	**% DIF**
100	2	1000	3442	4333	4414,9	33,8	25,89
100	5	1000	3442	4315	4397,2	39,0	25,36
100	2	1500	3442	4293	4360,6	43,2	24,72
100	5	1500	3442	4263	4358,0	43,9	23,85
50	2	1000	3442	4279	4366,6	44,5	24,32
50	5	1000	3442	4254	4375,7	51,3	23,59
50	2	1500	3442	4271	4357,1	43,4	24,08
50	5	1500	3442	4231	4339,4	54,6	22,92

En la instancia 120X8 el algoritmo propuesto no alcanza a la mejor solución en ninguna de las combinaciones, el mínimo makespan se encuentra con la combinación de tamaño de la población de 50 individuos, diversidad de 5 genes y 1500 iteraciones, con una diferencia de 22,92%.

10. DISEÑO EXPERIMENTAL

Para identificar los efectos que tienen los parámetros sobre la variable respuesta makespan, se realiza un diseño factorial 2^k , teniendo en cuenta los parámetros tamaño de la población, diversidad de genes e iteraciones, los cuales son variables de entrada del Algoritmo Genético de Chu-Beasley, se eligen estos factores como los posibles efectos principales debido a que estos mismos fueron utilizados en la validación.

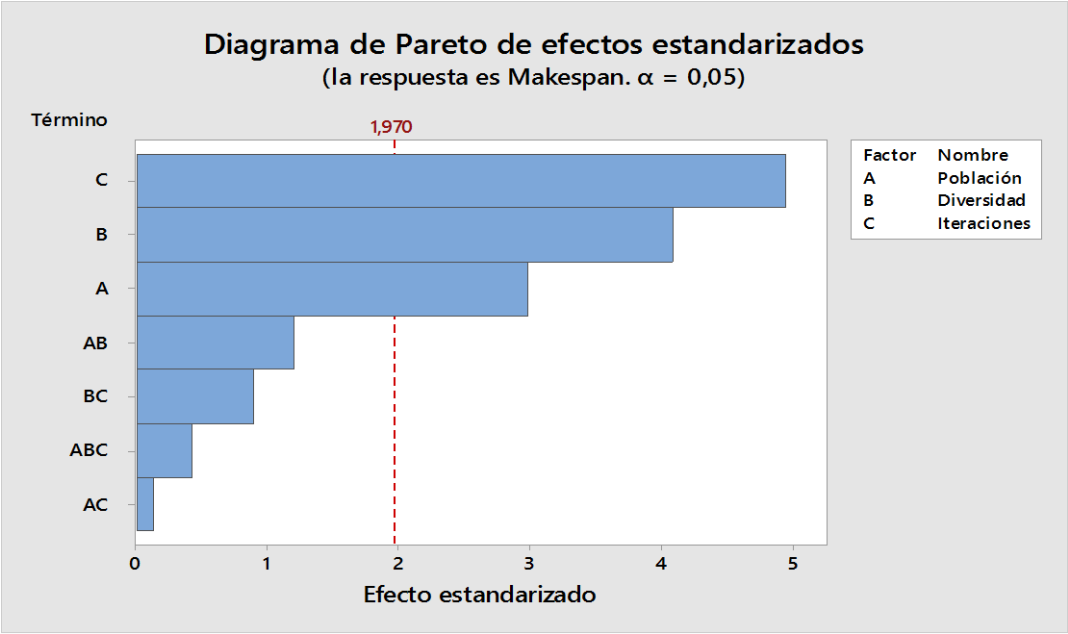
Para cada una de las instancias se realizó un diseño de experimentos, que fueron creados y analizados en el software estadístico Minitab 17, con 3 factores (A, B,C) y dos niveles (+1, -1), siendo 2^3 el diseño seleccionado. Los niveles utilizados para cada parámetro se muestran en la tabla 16.

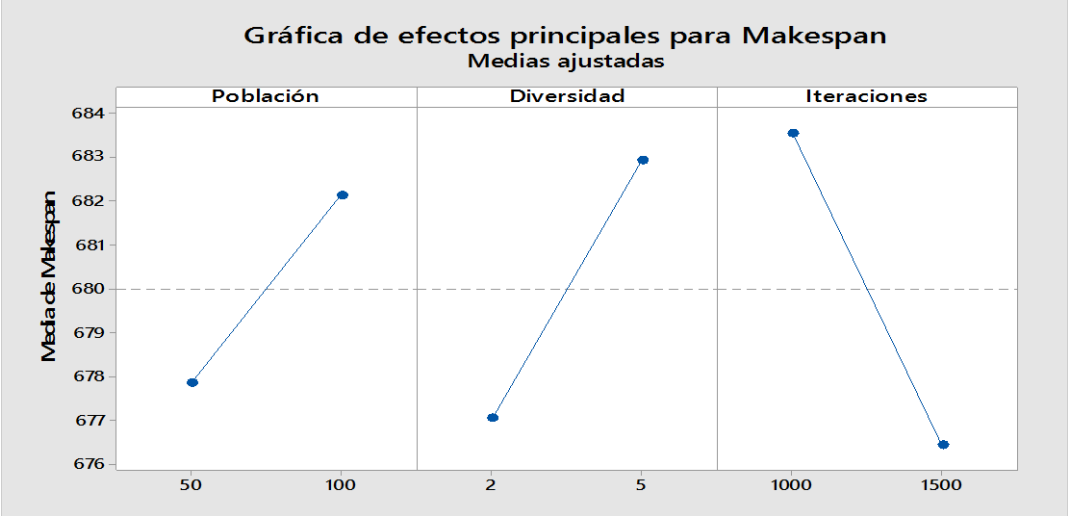
Los efectos son estadísticamente significativos sobre la variable makespan cuando sus valores p son menores que el nivel de significancia (α), que para el análisis de varianza considerado es del 5%. En el análisis de cada instancia se presentan el diagrama de Pareto de efectos estandarizados en el cual los efectos que se extienden más allá de la línea de referencia son significativos, de igual manera, se presenta las gráficas de efectos principales, la cual muestra en qué nivel la variable respuesta se minimiza. Las tablas ANOVA, se presentan como anexo debido a que estos resultados se resumen en las gráficas de cada instancia.

Tabla 16. Niveles de los factores para el diseño factorial $\mathbf{2^3}$ completo

Parámetros de las diferentes instancias			
Niveles	**Diversidad de genes**	**Tamaño de población**	**Iteraciones**
Bajo	2	50	1000
Alto	5	100	1500

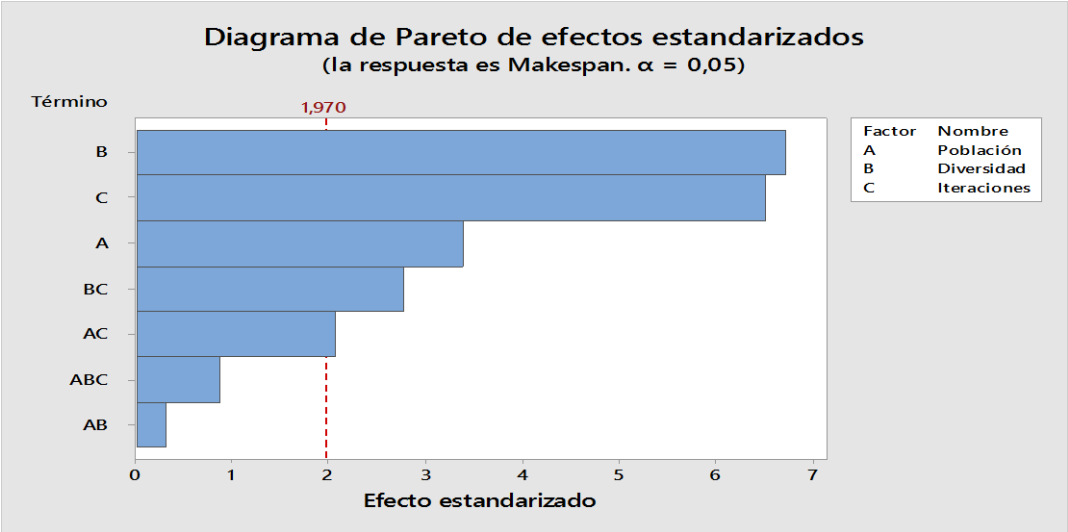
10.1 ANÁLISIS DE VARIANZA PARA LA INSTANCIA 20X2

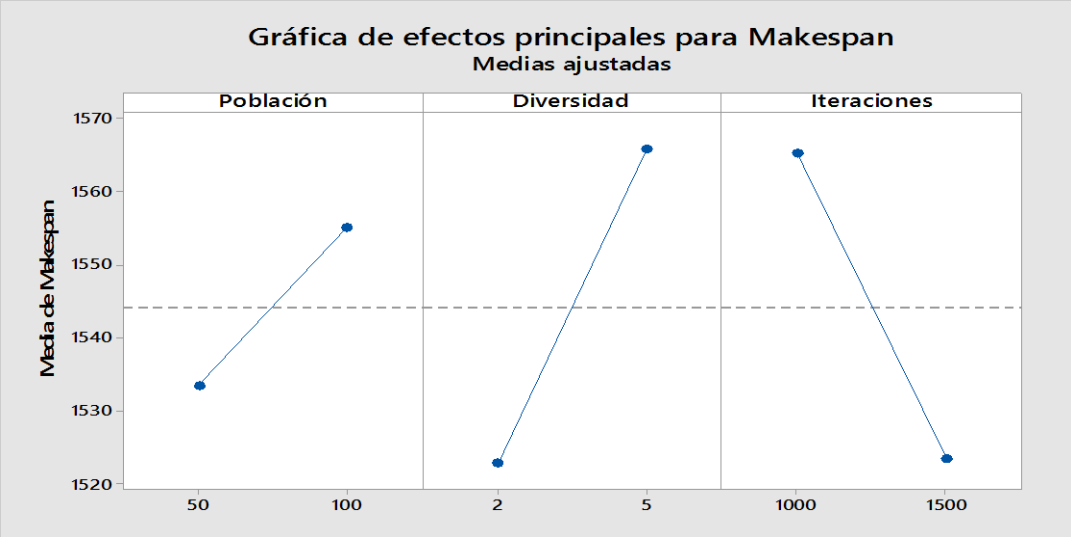




En esta instancia los factores con efectos significativos son el número de iteraciones, la diversidad y el tamaño de la población; las iteraciones en el nivel alto y los otros en un nivel bajo arrojaron respuestas de un makespan menor. Las demás interacciones de estos no evidencian un efecto significativo.

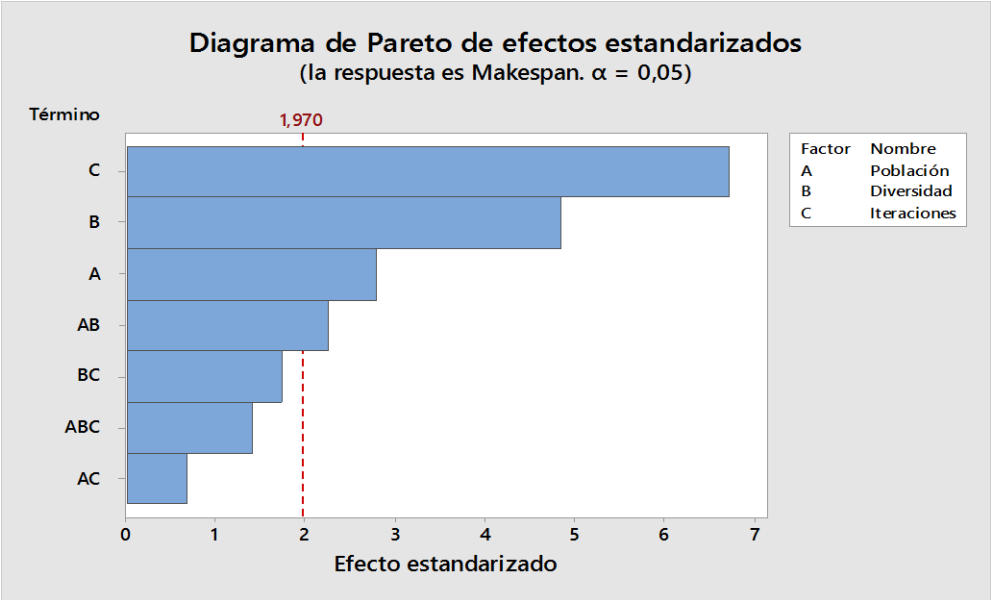
10.2 ANÁLISIS DE LA VARIANZA PARA LA INSTANCIA 20X4

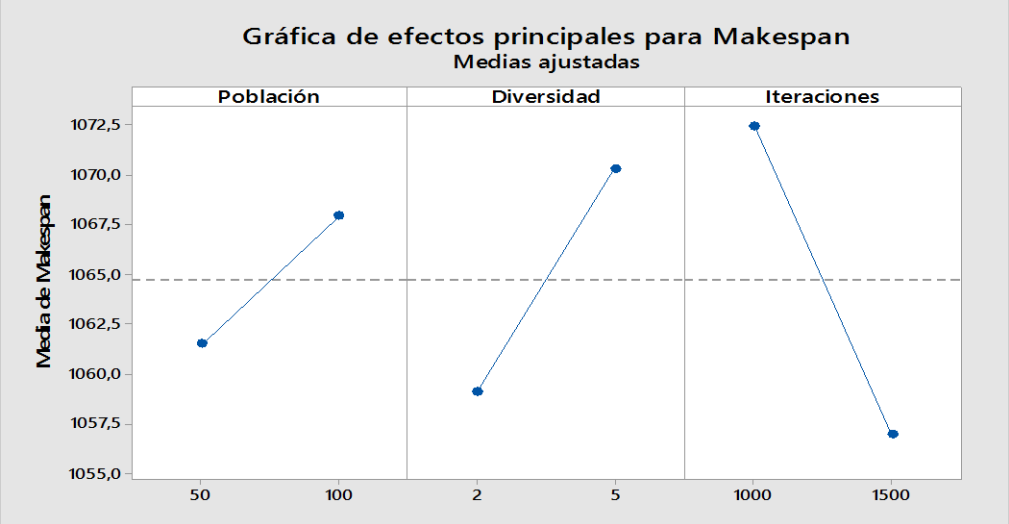




Para este caso los factores con efectos significativos son el tamaño de la población, la diversidad de genes y las iteraciones, los dos primeros en el nivel bajo arrojaron respuestas de un makespan menor a excepción del número de iteraciones. También la interacción entre diversidad e iteraciones y población e iteraciones arroja un efecto significativo. Los demás interacciones de estos no presentan un efecto significativo.

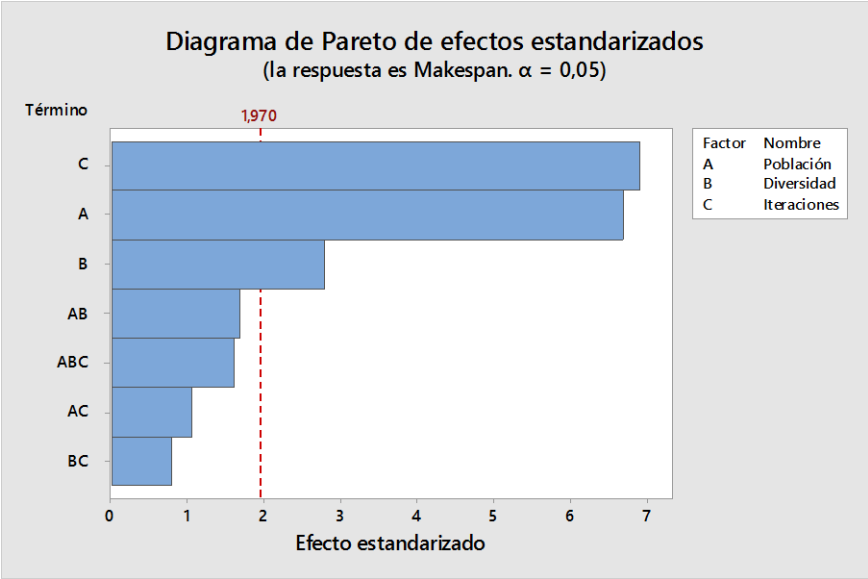
10.3 ANÁLISIS DE LA VARIANZA PARA LA INSTANCIA 20X8

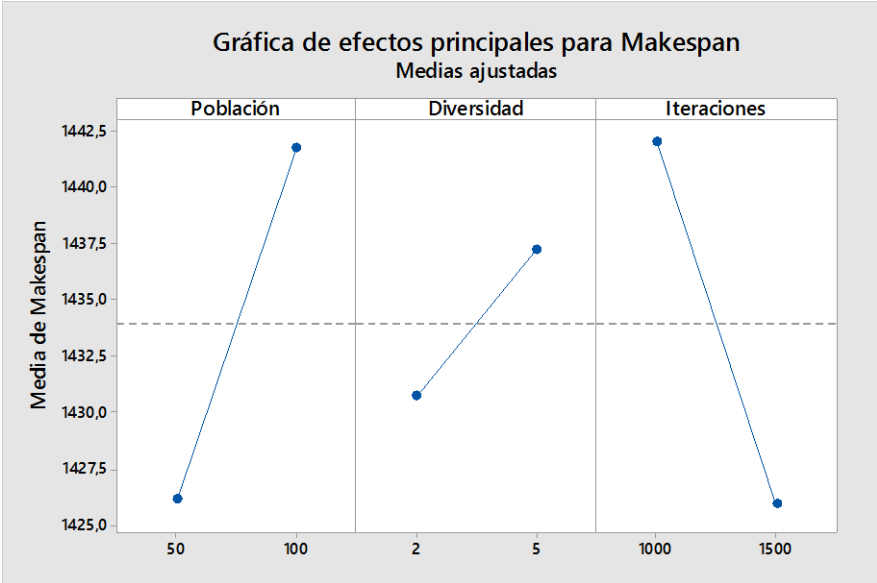




En esta instancia los factores con efectos significativos son el tamaño de la población, la diversidad de genes y el número de iteraciones; también la interacción entre población y diversidad arroja un efecto significativo; en el caso del tamaño de población y diversidad de genes los niveles bajos de cada factor son los que dieron como resultado un menor valor del makespan, mientras que en el número de iteraciones es el nivel bajo de este factor.

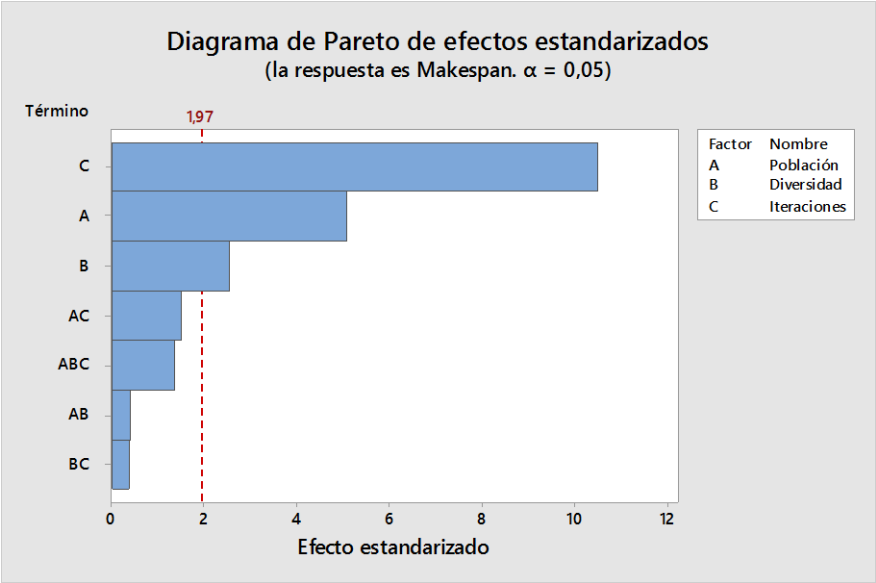
10.4 ANÁLISIS DE LA VARIANZA PARA LA INSTANCIA 50X2

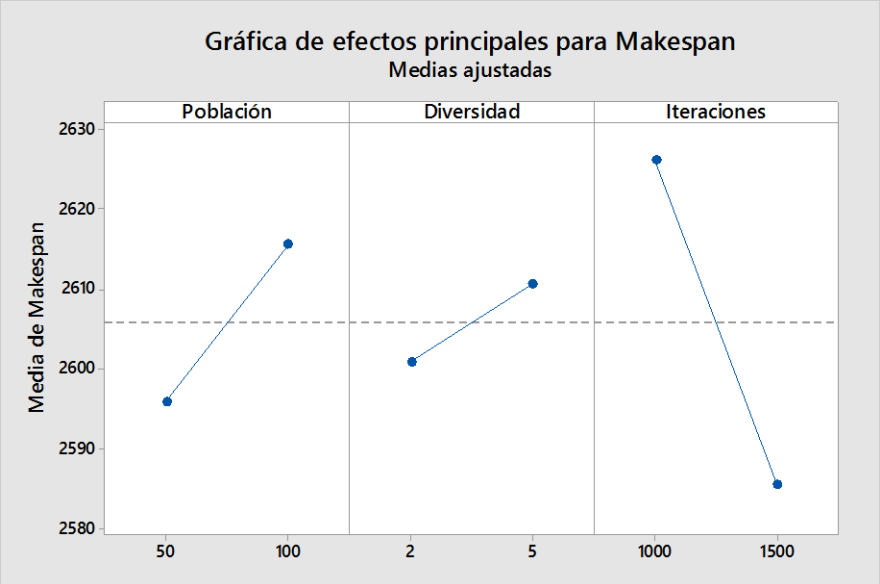




Los factores que evidencian un efecto significativo en la variable respuesta makespan son el tamaño de población, la diversidad de genes y el número de iteraciones. En la gráfica de efectos principales se observa que en los dos primeros factores el nivel bajo arroja un valor menor del makespan, mientras que en el número de iteraciones es el nivel alto de este factor. Los demás factores y las interacciones de estos no presentan un efecto significativo.

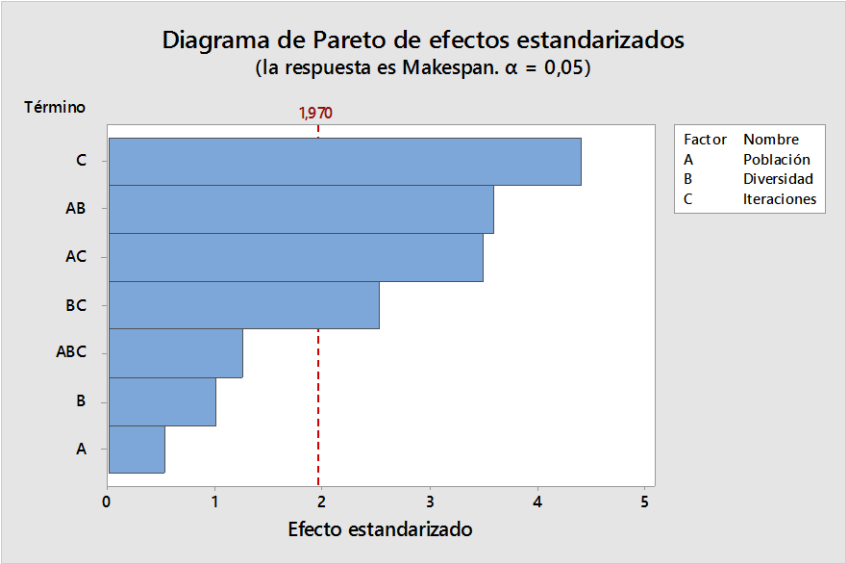
10.5 ANÁLISIS DE LA VARIANZA PARA LA INSTANCIA 50X4

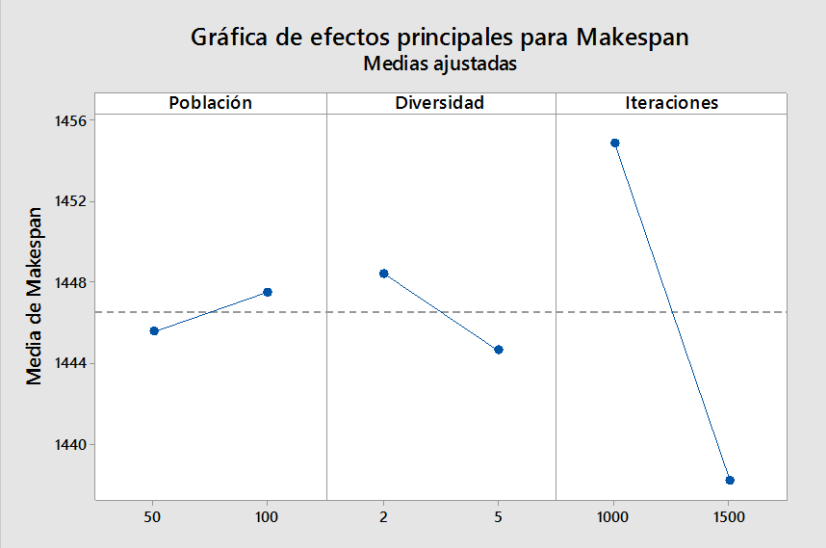




En esta instancia los factores, tamaño de población, diversidad de genes y el número de iteraciones evidencian efecto significativo sobre el makespan, en el nivel bajo los dos primeros factores y en el nivel alto del factor iteraciones se observaron las mejores respuestas.

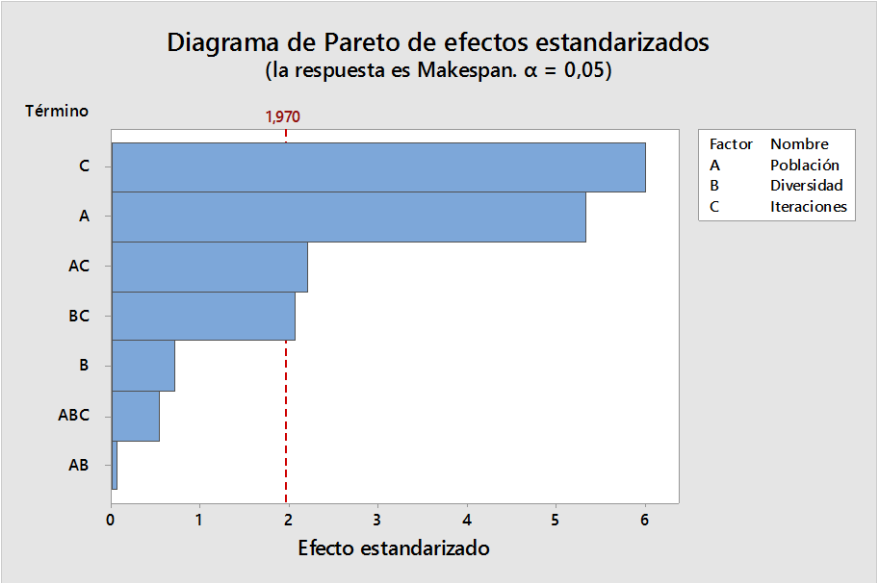
10.6 ANÁLISIS DE LA VARIANZA PARA LA INSTANCIA 50X8

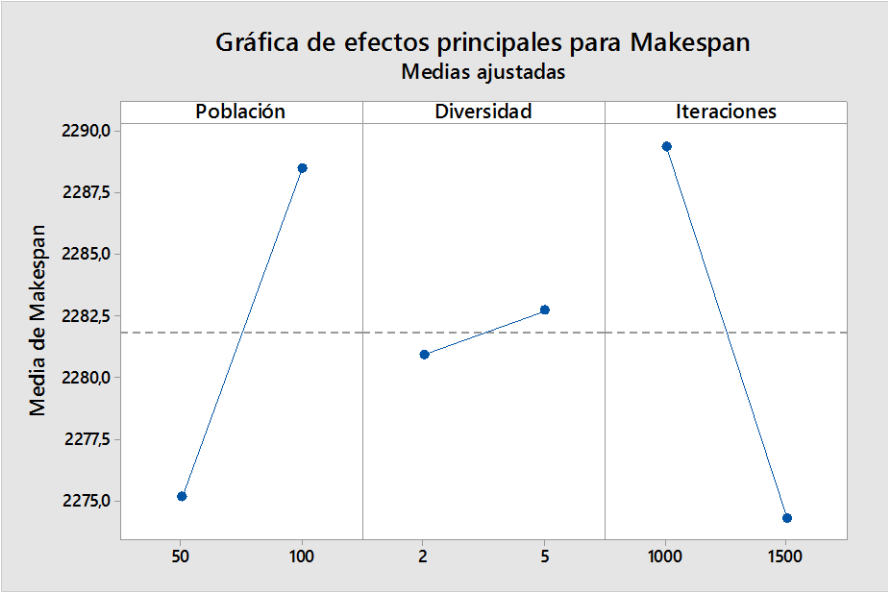




Para este caso los factores con efectos significativos son el número de iteraciones y las interacciones entre población-diversidad, población-iteraciones y diversidad-iteraciones. El factor número de iteraciones en el nivel alto arrojó respuestas de un makespan menor. Los demás factores no evidencian un efecto significativo.

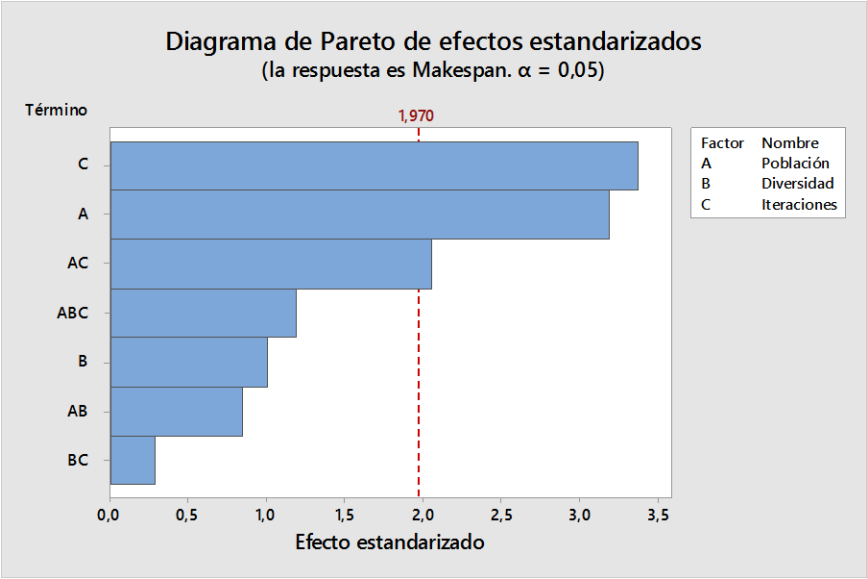
10.7 ANÁLISIS DE LA VARIANZA PARA LA INSTANCIA 80X2

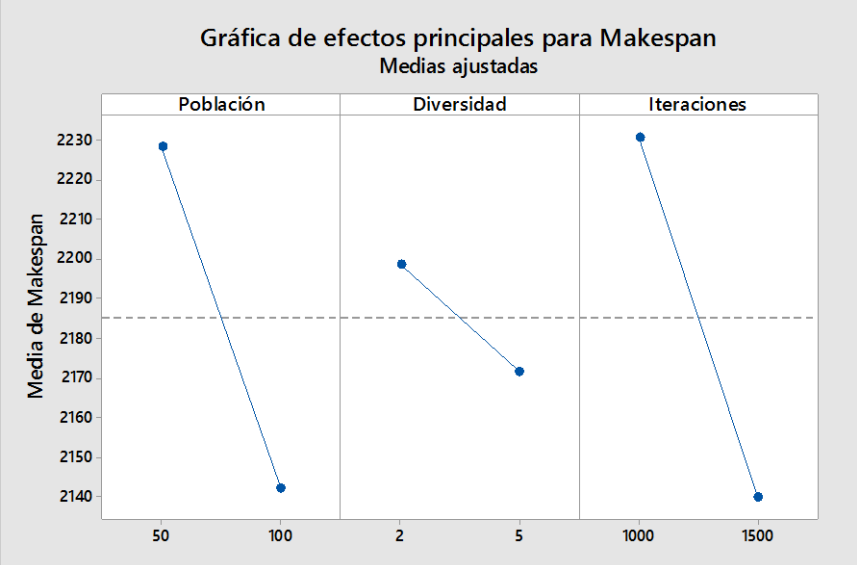




Los factores que evidencian un efecto significativo en la variable respuesta makespan son el número de iteraciones y el tamaño de población, adicionalmente las interacciones entre población e iteraciones y diversidad e iteraciones. En el caso del número de iteraciones el nivel alto disminuye la variable respuesta, mientras que en el tamaño de la población es el nivel bajo de este factor.

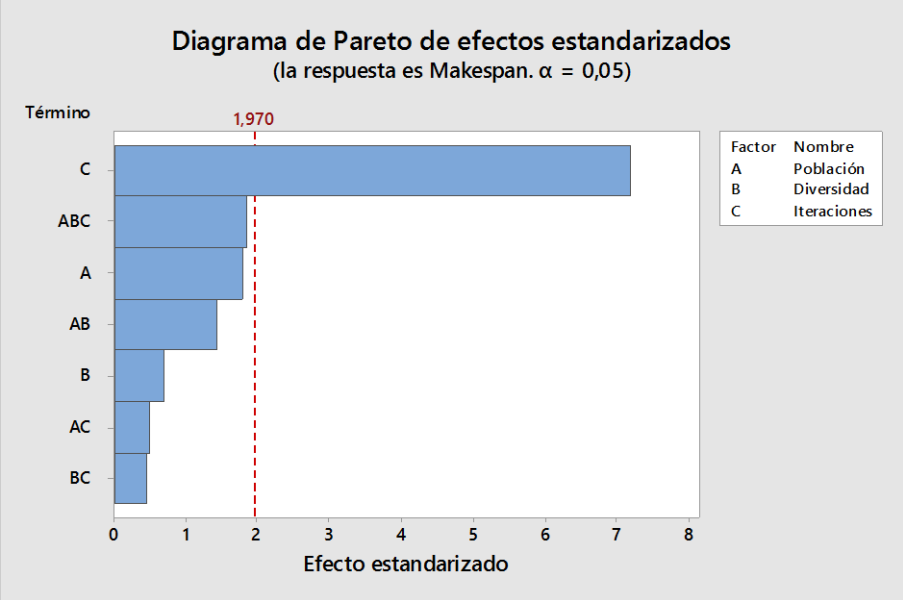
10.8 ANÁLISIS DE LA VARIANZA PARA LA INSTANCIA 80X4

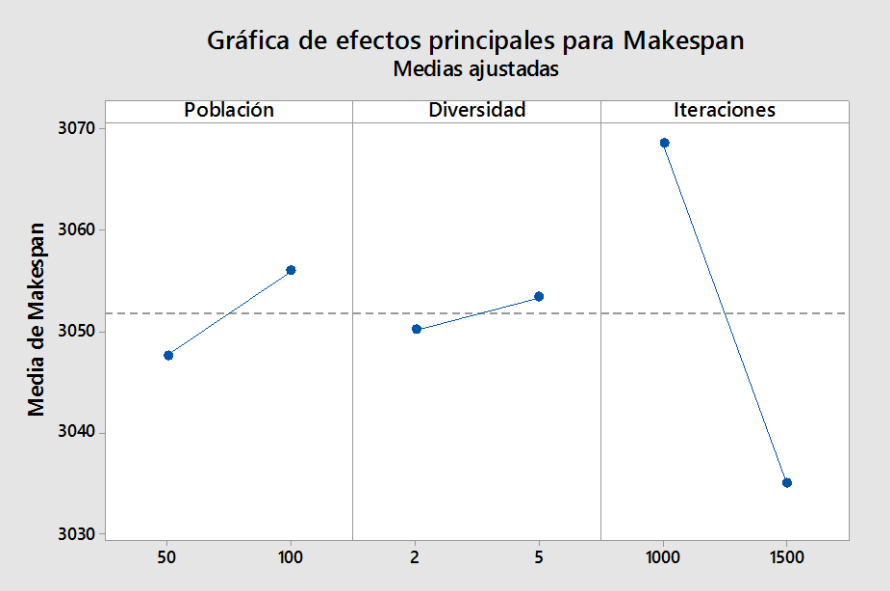




En esta instancia los factores con efectos significativos son el número de iteraciones, el tamaño de la población, también la interacción de los dos factores analizados arroja un efecto significativo; los niveles altos de cada factor son los que dieron como resultado un menor valor del makespan.

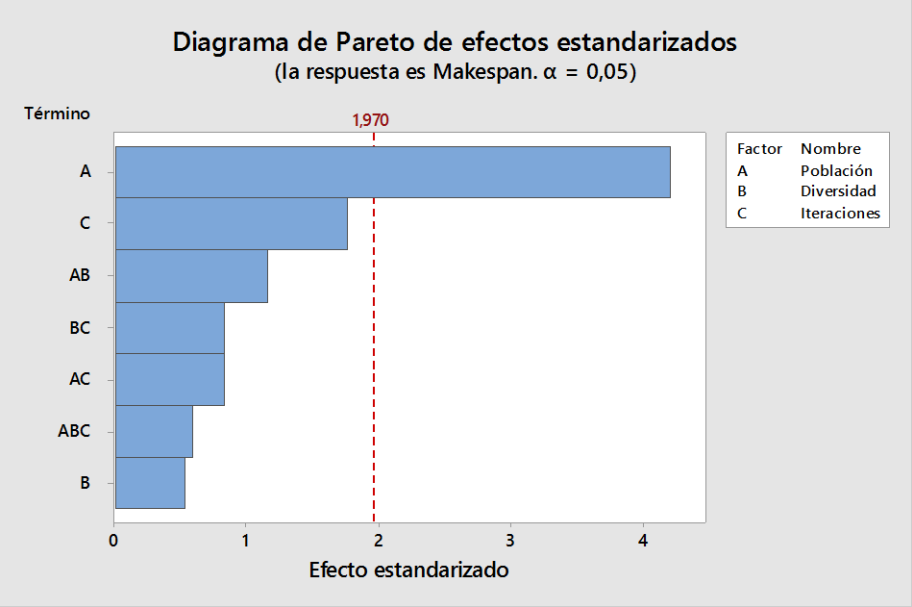
10.9 ANÁLISIS DE LA VARIANZA PARA LA INSTANCIA 80X8

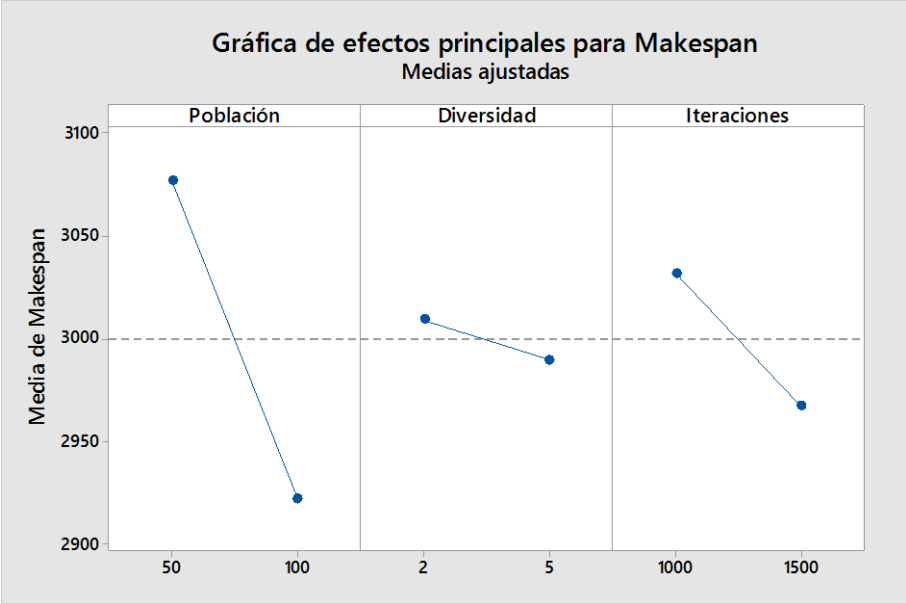




Para esta instancia el efecto significativo se evidencia en el factor número de iteraciones, estando este en un nivel alto arroja menores valores del makespan. Los demás factores y las interacciones de estos no evidencian un efecto significativo sobre la variable respuesta.

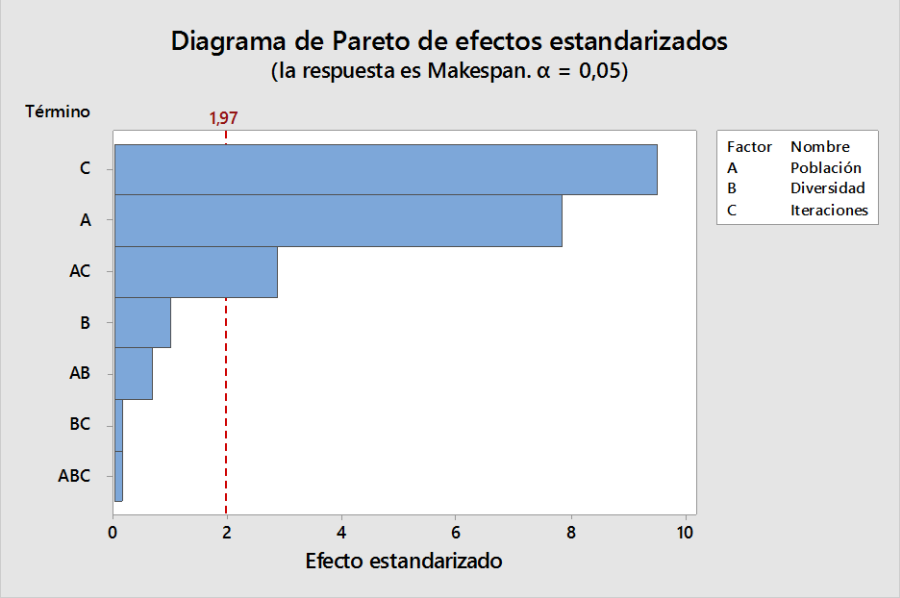
10.10 ANÁLISIS DE LA VARIANZA PARA LA INSTANCIA 120X2

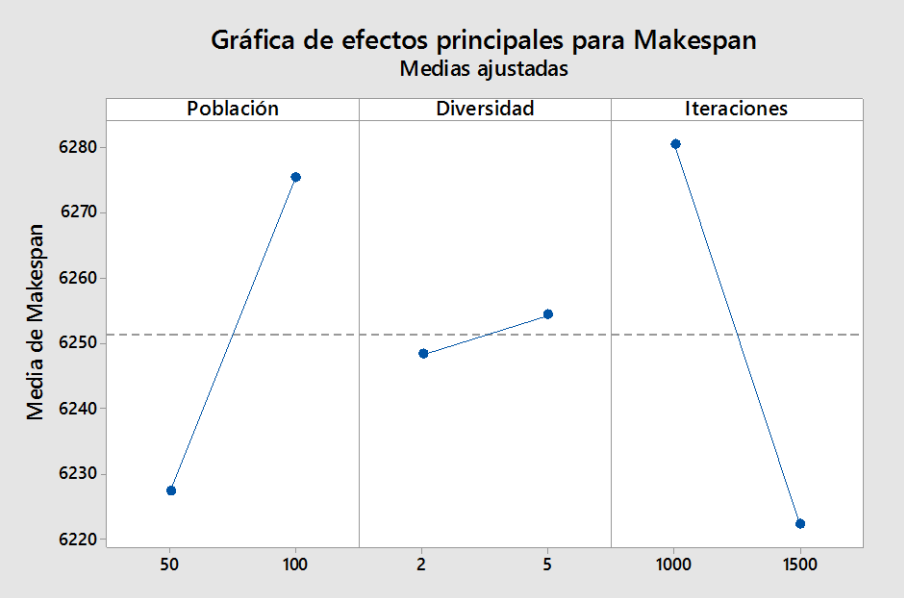




En este caso el efecto significativo se evidencia en el factor tamaño de población, en donde un nivel alto disminuye la variable respuesta. Los demás factores y las interacciones de estos no evidencian un efecto significativo.

10.11 ANÁLISIS DE LA VARIANZA PARA LA INSTANCIA 120X4

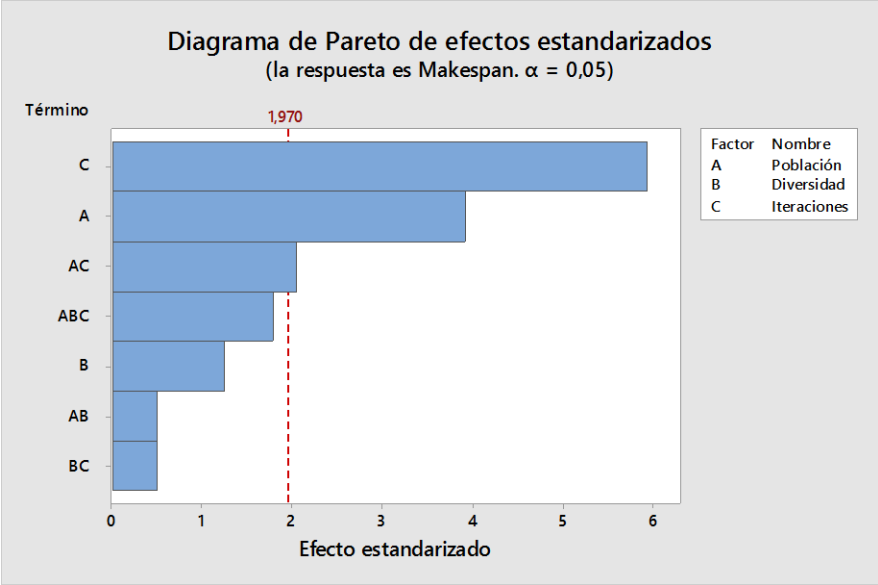


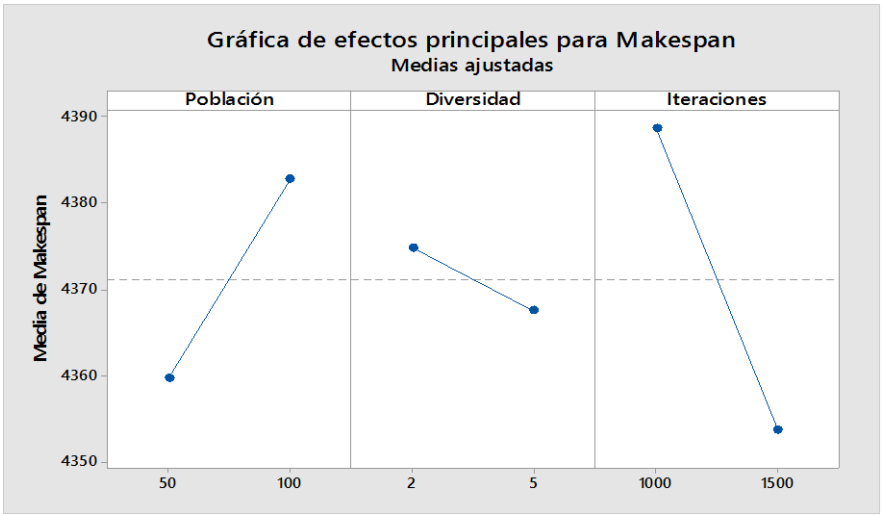


Los factores con efectos significativos son el número de iteraciones, el tamaño de la población, también la interacción entre estos dos factores analizados arroja un efecto significativo; en el caso del tamaño de la población el nivel bajo provee

como resultado un menor valor del makespan, por el contrario en el número de iteraciones es el nivel alto quien minimiza la variable respuesta.

10.12 ANÁLISIS DE LA VARIANZA PARA LA INSTANCIA 120X8





Para esta instancia los efectos significativos se evidencian en los factores número de iteraciones y tamaño de población, también la interacción entre estos dos factores arroja un efecto significativo; en el caso del tamaño de la población el nivel bajo evidencia una disminución en la variable respuesta, mientras que en el

número de iteraciones es el nivel alto de este factor quien influye en la variable. Los demás factores y las interacciones de estos no presentan un efecto significativo.

En la tabla 17 se resume los factores con efectos significativos para cada instancia

Tabla 17. Instancias con sus factores significativos

INSTANCIA	FACTORES CON EFECTO SIGNIFICATIVO
20x2	Diversidad - Población - Iteraciones
20x4	Diversidad - Población – Iteraciones Diversidad*Iteraciones Población*Iteraciones
20x8	Iteraciones - Población – Diversidad Población*Diversidad
50x2	Iteraciones - Población – Diversidad
50x4	Iteraciones - Población – Diversidad
50x8	Iteraciones Población*Diversidad Población*Iteraciones Diversidad*Iteraciones
80x2	Iteraciones – Población Población*Iteraciones Diversidad*Iteraciones
80x4	Iteraciones – Población Diversidad*Iteraciones
80x8	Iteraciones
120x2	Población
120x4	Iteraciones – Población Población*Iteraciones
120x8	Iteraciones – Población Población*Iteraciones

Es de resaltar que la interacción entre los factores propuestos no tienen un efecto significativo en la mayoría de las instancias y que el factor más influyente es el número de iteraciones seguido del tamaño de la población y finalmente de la diversidad aceptada del cromosoma.

11. RESULTADOS

Los resultados de la ejecución del algoritmo teniendo en cuenta los parámetros establecidos como de influencia hallados en el capítulo anterior fueron comparados con los resultados publicados por Naderi, Ruiz y Zandieh[111] en el 2010 en donde se presentan tres variaciones al algoritmo ILS y por Branz Moreira[112] en el 2013 en donde se aplica un algoritmo GRASP-ILS. En la tabla 18 se presenta el resumen de esta comparación con sus respectivos porcentajes de diferencia.

Tabla 18. Comparación de resultados

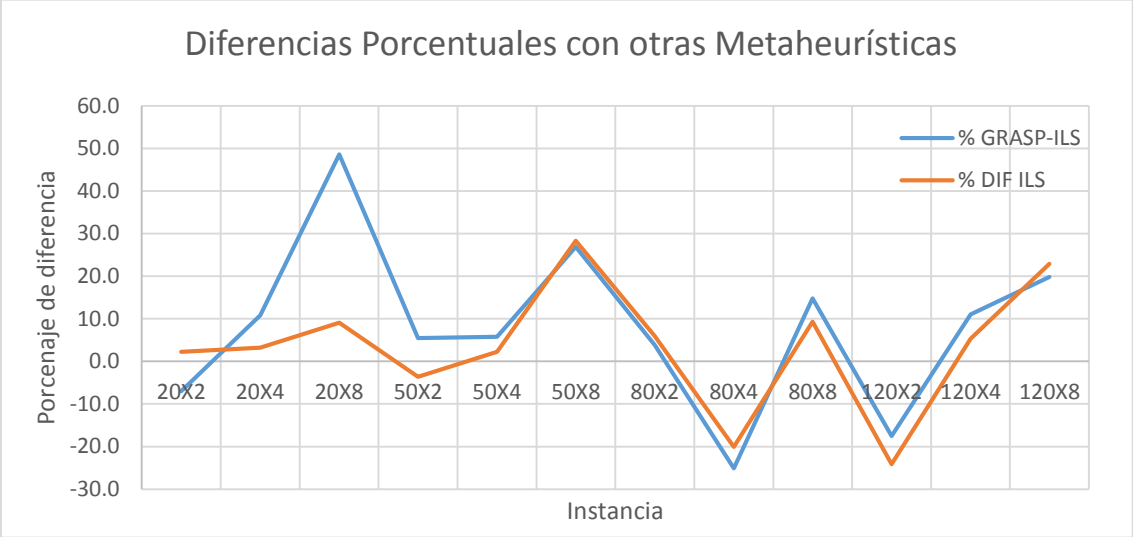
INSTANCIA	AGCB				GRASP-ILS BRANZ MOREIRA	ILS NADERI	% DIF GRASP-ILS	% DIF ILS
	POBLACION	DIVERSIDAD	ITERACIONES	MEJOR SOLUCION				
20X2	50	5	1000	648	697	634	-7,0	2,2
20X4	50	2	1500	1365	1232	1322	10,8	3,3
20X8	50	2	1000	1018	685	933	48,6	9,1
50X2	50	2	1500	1385	1313	1437	5,5	-3,6
50X4	50	5	1500	2502	2366	2447	5,7	2,2
50X8	50	5	1500	1351	1065	1053	26,9	28,3
80X2	50	5	1500	2210	2128	2086	3,9	5,9
80X4	50	5	1000	1819	2430	2276	-25,1	-20,1
80X8	100	5	1500	2939	2561	2689	14,8	9,3
120X2	50	5	1500	2545	3086	3356	-17,5	-24,2
120X4	50	5	1500	6078	5475	5771	11,0	5,3
120X8	50	5	1500	4231	3531	3442	19,8	22,9

Se encuentra que en al comprar las respuestas obtenidas con el Algoritmo Genético Modificado de Chu-Beasley con el algoritmo GRASP-ILS existe una mejora en tan solo 3 instancias: 20X2, 80X4 y 120X2 y los porcentajes de diferencia más altos se encuentran en las instancias que contemplan 8 etapas. Al comparar con los resultados del algoritmo ILS se encuentra que se logran mejorar

[111] NADERI, B; RUIZ, Rubén y ZANDIEH M. Op. Cit. p.244
[112] BRAZ MOREIRA, Neuma Eufrazio. Op. Cit. p. 61

los tiempos de finalización en 3 instancias 50X2, 80X4 y 120X2 y el porcentaje de diferencia más alto se encuentra en la instancia 120x8.

Figura 13. Diferencias porcentuales de la mejor respuesta con otras Metaheurísticas



11.1 TIEMPO COMPUTACIONAL

Se compararon los tiempos de ejecución del algoritmo para cada una de las 30 repeticiones que se realizaron y se presenta en la siguiente tabla 19 el tiempo medio de ejecución para cada instancia en cada familia analizada y se compara con los tiempos de ejecución proporcionados por Branz Moreira[113].

Se puede evidenciar que en las instancias más pequeñas el algoritmo GRASP-ILS es realmente muy rápido en contraparte el AGCB para estas instancias es más lento, en la instancia 20x2 mientras el algoritmo GRASP-ILS demora 0,178

[113] BRAZ MOREIRA, Neuma Eufrazio. Op. Cit. p. 61

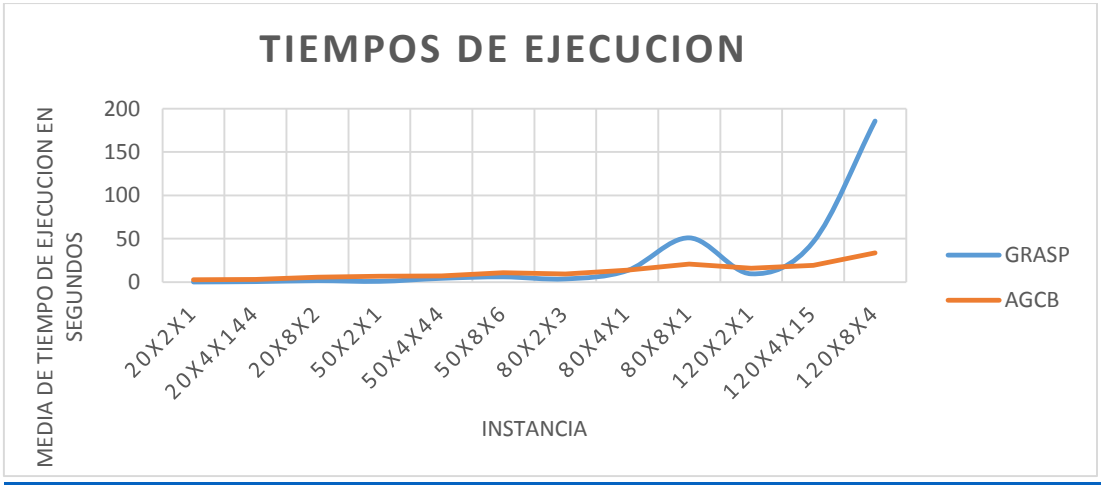
segundos el AGCB demora su ejecución 2,65 segundos es decir un 1389% más lento que el GRASP-ILS.

Tabla 19. Tiempos computacionales en segundos

INSTANCIA	AGCB [sg]	GRASP [sg]	% DIF
20X2	2,65	0,178	1389
20X4	3,19	0,620	414
20X8	5,67	1,660	242
50X2	6,71	0,930	621
50X4	7,35	4,400	67
50X8	10,74	6,100	76
80X2	9,33	3,710	151
80X4	13,75	13,290	3
80X8	20,71	50,990	-59
120X2	15,85	9,600	65
120X4	19,44	46,060	-58
120X8	33,87	185,550	-82

En las instancia grandes el algoritmo GRASP-ILS aumenta considerablemente su tiempo de ejecución mientras tanto el aumento en tiempo computacional del AGCB aumenta de una manera más moderada, obteniendo finalmente en la instancia 120x8 una ejecución más rápida en un 82% que el GRASP-ILS. En la gráfica se puede observar el comportamiento de los tiempos de ejecución de los algoritmos comparados en cada una de las instancias analizadas.

Figura 14. Tiempos de ejecución del algoritmo



12. CONCLUSIONES

La revisión de la literatura, dio las bases necesarias para definir el problema de la programación de producción en ambientes tipo Flow Shop Híbrido Flexible, además se logró establecer una visión de la situación teórica de las diferentes metaheurísticas para solucionar este tipo de problemas. Se desarrolló una metodología basada en la técnica Metaheurística Algoritmo Genético de Chu-Beasley como método de solución al problema.

Los resultados obtenidos con el algoritmo genético de Chu-Beasley fueron satisfactorios en las instancias con las etapas 2 y 4, ya que en el 75% de estas, la diferencia entre los valores de las instancias y los mejores valores reportados no sobrepasa el 5%, y con un tiempo computacional de máximo 20 segundos para instancia más grande de estas etapas. Además, en las instancias restantes, es decir, en el 25% el porcentaje de diferencia es inferior al 6%.

Para las instancias de etapa igual a 8, no se logra alcanzar el mínimo Makespan reportado y la diferencia entre los mejores valores conocidos y los valores de las instancias es bastante considerable. En cuanto al tiempo computacional encontrado en la literatura y el de las instancias se tiene una disminución de la diferencia de tiempos computacionales a medida que aumenta el número de trabajos para esta cantidad de etapas.

En el análisis del diseño experimental de las instancias se concluye que el efecto del factor número de iteraciones es significativo, ya que está presente en 11 de las 12 instancias analizadas, como el factor más influyente y al modificar este es posible llegar a mejores resultados; sin embargo, al realizar nuevamente corridas

aumentando este valor, se incrementaba el tiempo computacional sin tener una mejora importante en el makespan, por ello los resultados finales se manejan con el nivel alto de 1500 tomado para los diseños factoriales.

El algoritmo ofrece mejores soluciones cuando el factor tamaño de población se establece en 50 individuos en lugar de una población de 100 individuos, esta situación nos indica como postulan López, Vargas y Arango[114] "La variedad y diversidad que presenta una población mayor implica mayores esfuerzos para mejorar la calidad de la población, en cambio una población de tamaño pequeño tiende a facilitar mejoras en la calidad de sus individuos.", debido a que su investigación arrojo resultados similares a la presentada en este documento.

[114] LÓPEZ, Juan C; GIRALDO, Jaime A. y ARANGO, Jaime A. Algoritmo genético para reducir el makespan en un Flow Shop Hibrido Flexible con máquinas paralelas no relacionadas y tiempos de alistamiento dependientes de la secuencia. [online]. 2015,Vol. 11, No.1,p.250-262 [Citado 2016-10-06]

13. RECOMENDACIONES

Implementar el modelo y la metodología desarrollada en sistemas productivos de la industria que presenten configuraciones productivas tipo Flow Shop Híbrido Flexible, con el fin de disminuir la brecha entre la teoría y la práctica y de seguir contribuyendo a la relación empresa-academia.

Implementar una heurística en la generación de la población inicial, principalmente para los casos de alta complejidad donde puede representar grandes ventajas en la reducción del costo computacional en el AGCB al generar una secuencia muy cercana a la solución.

Aplicar el Algoritmo genético de Chu-Beasley a otros problemas de optimización, con el fin de validar la eficiencia de modelo mostrada en este proyecto.

Se recomienda para investigaciones futuras, la revisión y análisis de otros parámetros que puede ser clave en la calidad de la solución final.

Realizar el estudio de Flow Shop Hibrido Flexible con tiempos dependientes de la secuencia utilizando otra técnica Metaheurística de optimización con el fin de validar y comparar la eficiencia computacional de distintas técnicas.

BIBLIOGRAFÍA

ÁNGEL-BELLO, Francisco, *et al* A heuristic approach for a scheduling problem with periodic maintenance and sequence-dependent setup times. En: *Computers & Mathematics with Applications*. [Online]. 2011, Vol. 61, No. 4, pp.797-808. [Citado:12-05-2016].

ARRANZ DE LA PEÑA, Jorge y PARRA TRUYOL, Antonio. Algoritmos Genéticos. Universidad Carlos III, 2007.

BLUM, Christian y ROLI, Andrea. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. En: *ACM Computing Surveys*. [Online], 2003, Vol. 35, No.3, p.268-308. [Citado: 2016-05-16].

BRASSAR Gilles y BRATLEY Paul. Fundamental of algorithmics. Editorial Prentice Hall, New Jersey - USA, 1996.

BRAZ MOREIRA, Neuma Eufrazio. Heurísticas para Minimização do Makespan na Classe de Problemas de Sequenciamento Flowshop Híbrido e Flexível com Tempo de Setup Dependente da Sequência: Com e Sem Eventos de Quebra de Máquinas. Belo Horizonte, 2013. Trabajo de Grado (Maestria en Modelamiento Matemático y Computacional). Centro federal de educação Tecnológica de minas gerais. [Citado: 2016-18-08].

BROOKSHEAR, J. Glenn. Introducción a la computación. Décimo primera edición. Pearson Educación, S.A., Madrid, 2012, p.594-595

CEVIKCAN, E.; DURMUSLOGLU, M. y BASKAK, M. Integrating parts design characteristics and scheduling on parallel machines. En: *Expert Systems with Applications*. [Online]. 2011, Vol.38, No. 10, p.13232-13253. [Citado: 2016-05-12].

CERVANTES POSADA, Mariamar. Nuevos Métodos Meta Heurísticos para la Asignación Eficiente, Optimizada y Robusta de Recursos Limitados. Valencia, 2010. Tesis (Doctor en Informática). Universidad Politécnica de Valencia. Departamento de Sistemas Informáticos y Computación.

CHASE, Richard; JACOBS, Robert y AQUILANO, Nicholas. Administración de la producción y operaciones para una ventaja competitiva, Décima edición ed. Mc Graw Hill, México D.F., México, 2005, p. 518

CHEN, Huaping *et al.* A Hybrid Differential Evolution Algorithm for a Two-Stage Flow Shop on Batch Processing Machines with Arbitrary Release Times and Blocking. En *International Journal of Production Research*, [online]. 2014, Vol.52, No.19, [Citado 2016-04-18], p. 5714-5734.

CHU, P.C. y BEASLEY, J.E..A Genetic Algorithm for the Generalised Assignment Problem. En: *Computers Ops. Res.* [Online],1997. Vol. 24, No. 1, p.17-23. [Citado: 2016-17-05].

CONWAY, R. W.; MAXWELL, W. L., y MILLER, L. W. Theory of scheduling. Dover Publicationes, Inc. New York, 1967.

COOK, Stephen. The Complexity of Theorem Proving Procedures, Proceedings 3rd ACM Symposium Theory of Computing [online], 1971, p.151-158. [Citado 2016-05-05]. Disponible en: http://www.cs.utoronto.ca/~sacook/homepage/1971.pdf

CORTÉZ, Augusto. Teoría de la complejidad computacional y teoría de la computabilidad. En *Rev. investig. sist. inform*., [En línea], 2004, Vol. 1, No. 1, p.102-105. [Citado 2016-05-04].

CORTÉZ VÁSQUEZ, Augusto. *et al.* Sistema de apoyo a la generación de horarios basado en algoritmos genéticos. En: *Revista de Investigación de Sistemas e Informática.* [En línea], 2010. Vol. 7, No.1, p.37-55. [Citado: 2016-05-15]. Disponible en: http://revistasinvestigacion.unmsm.edu.pe/index.php/sistem/article/view/3264/2729

CUATRECASAS, Luis. Organización de la producción y dirección de operaciones: sistemas actuales de gestión eficiente y competitiva. Ediciones Díaz De Santos, 2011.

DOMINGUEZ MACHUCA, José Antonio, *et al.* Dirección de Operaciones: Aspectos estratégicos. Madrid, España: Mc Graw-Hill, 1995

DUARTE MUÑOZ, Abraham; PANTIGRO FERNÁNDEZ, Juan José y GALLEGO CARRILLO, Micael; Metaheurísticas. Universidad Rey Juan Carlos. Editorial DYKINSON.S.L., 2007, Madrid. p.1.

DUPUY, Germán, *et al.* Métodos Metaheurísticos aplicados a procesos de gestión en una empresa. Laboratorio de Investigación en Sistemas Inteligentes. XVII Workshop de Investigadores en Ciencias de la Computación, Salta, 2015.

EBRAHIMI, M.; FATEMI GHOMI. S.M.T y KARIMI, B. Hybrid Flow Shop Scheduling with Sequence Dependt Family Setup Time and Uncertain Due Dates. En *Applied Mathematical Modelling Research* [online]. 2014, Vol.38, No.9-10 [Citado 2016-04-18], p.2490-2504.

FARAHMAND-MEHR, Mohammad *et al.* An efficient genetic algorithm for a hybrid Flow shop scheduling problem with time lags and sequence-dependent setup time.

En *The International Journal of Advanced Manufacturing Technology*.[online].2014, Vol.63, [Citado 2016-04-19], p. 337-348 .

FONSECA-REYNA, Yunior César, *et al*.. Influencia de los parámetros principales de un Algoritmo Genético para el Flow Shop Scheduling. *Rev cuba cienc informat* [online]. 2014, Vol.8, No.1 [Citado 2016-04-18], p. 53-59.

GALLEGO, Ramón; TORO, Eliana y ESCOBAR, Antonio. Técnicas Heurísticas y Metaheurísticas. Universidad Tecnológica de Pereira, Pereira, 2015. p 158

GAREY M; JONSON D. Local Search in Combinatorial Optimization. Editorial John Wiley & Sons, Inglaterra, 1997.

GOLDBERG, David. Genetics Algorithms in Search, Optimization and Machine Learning. Addison Wesley Logman Publishing Co., Boston, 1989.

GÓMEZ GASQUET, Pedro. Programación de la producción en un taller de flujo hibrido sujeto a incertidumbre: arquitectura y algoritmos. Aplicación a la industria cerámica. Valencia, 2010. Tesis (Doctorado en Gestión de la Cadena de Suministro e Integración Empresarial). Universidad Politécnica de Valencia.

GRANADA, Mauricio y TORO, Eliana. Mauricio. Método Hibrido entre el algoritmo genético de Chu-Beasley y Simulated Annealing para la solución del problema de asignación generalizada. En: *Scientia et Technica.* [En línea], 2005, Vol.2, No.28.[Citado:2016-04-05].

HOLLAND, John Henry. Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan Press; Pittsburgh, 1975.

JABBARIZADEH, F.; ZANDIEH, M. y TALEBI, D. Hybrid Flexible Flowshops with sequence-dependent setup times and machine availability constraints. En *Computers & Industrial Engineering.* [Online]. 2009, Vol.57, [Citado 2016-04-25], p. 949-957

LEE, Wen-Chiung; WANG, Jen-Ya, y LEE, Lin-Yo. A Hybrid Genetic Algorithm for an Identical Parallel-Machine Problem with Maintenance Activity. *The Journal of the Operational Research Society* [Online]. *2015. Vol. 66, No.*11, [Citado 2016-04-18], p. 1906-1918.

LONDOÑO POSSO, Julián; HINCAPIÉ ISAZA, Ricardo y GALLEGO RENDÓN, Ramón. Planeamiento de Redes de Baja Tensión, utilizando un Modelo Trifásico. En: *Ciencia e Ingeniería Neogranadina.* [En línea], 2011, Vol. 21, No. 2, p. 41-56. [Citado:2016-05-17].

LÓPEZ, Juan C; GIRALDO, Jaime A. y ARANGO, Jaime A. Algoritmo genético para reducir el makespan en un Flow Shop Hibrido Flexible con máquinas paralelas no relacionadas y tiempos de alistamiento dependientes de la secuencia. [En línea]. 2015, Vol. 11, No.1, p.250-262 [Citado 2016-10-06].

LÓPEZ, Juan C; GIRALDO, Jaime A. y ARANGO, Jaime A. Reducción del Tiempo de Terminación en la Programación de la Producción de una Línea de Flujo Hibrida Flexible (HFS). En *Información Tecnológica* [En línea]. 2015, Vol.26, No.3 [Citado 2016-04-18], p. 157-172.

MATÍA DEPRIT, Javier. Optimización de la secuenciación de tareas en taller mediante algoritmos genéticos. Madrid, 2007. Tesis (Ingeniería Industrial). Universidad Pontificia Comillas. Disponible en: http://www.iit.comillas.edu/pfc/resumenes/466d95863d807.pdf

MÁRQUEZ DELGADO, José *et al.* Algoritmo Genético aplicado al Problema de programación en procesos tecnológicos de maquinado con ambiente Flow Shop. En *Revista Ciencias Técnicas Agropecuarias* [En línea]. 2012, Vol.21, No. 2, [Citado 2016-04-19], p. 70-75.

MARTÍ, Rafael. Procedimientos Metaheurísticos en optimización combinatoria. Departamento de estadística e investigación de operaciones. Facultad de Matemáticas. Universidad de Valencia. 2003

MARTÍ, Rafael y GERHARD, Reinelt. The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization. Applied Mathematical Sciences Springer, Berlin, 2011

NADERI, B; RUIZ, Rubén y ZANDIEH M. Algorithms for a realistic variant of flow shop scheduling. En: *Computers Ops. Res.* [online],2010. Vol. 36, [Citado 2016-17-08], p.236-246.

NAGANO, Marcelo; MIYATA, Hugo y CASTRO, Daniella. A Constructive Heuristic for Total Flow time Minimization in a no –wait Flow shop with Sequence Dependent Setup Times. En *Journal of Manufacturing Systems* [online]. 2015, Vol.36, [Citado 2016-04-19], p.224-230.

PAN, Quan-Ke; RUIZ, Rubén. An estimation of distribution algorithm for lot-streaming flow shop problems with setup times. En *Omega.* [Online]. 2012, Vol. 40, [Citado 2016-04-25], p.166-180.

PAN, Quan-Ke y RUIZ, Rubén. An effective iterated greedy algorithm for the mixed no-idle permutation flow shop scheduling problem. En: *Omega*. [online]. 2014, Vol.44, [Citado: 2016-05-12], p. 41–50.

PAN, Quan-Ke y RUIZ, Rubén. An effective iterated greedy algorithm for the mixed no-idle permutation flow shop scheduling problem. En: *Omega*. [Online]. 2014, Vol.44, [Citado: 2016-05-12], p. 41–50.

PAPADIMITRIOU, Christos y KLEINGBER, Jon. Computability and Complexity. En: *Cornell University. Department of Computer Science*. [online], 2004, p.8. [Citado: 2016-05-05].

PINEDO, Michael L.. Planning and Scheduling in Manufacturing and Services. Springer Science + Business Media, LLC, New York, 2005.

RUIZ, Rubén y VASQUÉZ RODRÍGUEZ, José Antonio. The Hybrid Flow Shop Scheduling Problem. En: *European Journal of Operational Research* [online]. 2010, Vol.205, No.1 [Citado 2016-04-18], pp. 1-18.

SALAZAR HORNIG, Eduardo y SARZURI GUARACHI, René A. Improved Genetic Algorithm for Total Tardiness Minimization in a Flexible Flow Shop with Sequence-Dependent Setup Times. En *Ingeniare. Rev. chil. ing.* [online]. 2015, Vol.23, No.1 [Citado 2016-04-18], pp. 118-127.

SANHUEZA, Raúl, *et al.* Aplicación de algoritmos genéticos al problema de la planificación de sistemas eléctricos de distribución. En: *Revista Facultad de Ingeniería.* [En línea], 1999, Vol.6. p.55-63, [Citado: 201-05-17].

SCHROEDER, Roger; Meyer, Susan y RUNGTUSANATHAM, Johnny. *Administración de Operaciones: Conceptos y casos contemporáneos,* quinta edición. México D.F., Editorial McGraw-Hill, 2008.

SOLARTE MARTÍNEZ, Guillermo; CASTILLO SANZ, Andrés y RODRIGUEZ GAHONA, Guillermo. Optimization of a vehicular routing using simple genetic Chu-

Beasley algorithm. En: Revista Tecnura. [online]. 2015, Vol.19, No. (44), [Citado: 2016-05-05], p 93-108.

TORO OCAMPO, Eliana; RESTREPO GRISALES, Yov Steven y GRANADA ECHEVERRI, Mauricio. Algoritmo genético modificado aplicado al problema de secuenciamiento de tareas en sistemas de producción lineal - Flow Shop. En: *Scientia Et Technica*. [En línea]. 2006, Vol. XII, No.30, p. 285-290. [Citado: 2016-05-12].

TUPIA ANTICONA, Manuel Francisco. Un algoritmo GRASP para resolver el problema de la programación de tareas dependientes en máquinas diferentes (task scheduling).Lima, 2005. Tesis (Magister en ingeniería de sistemas). Universidad Nacional Mayor de San Marcos

WANG, Sheng-yao *et al.* An enhanced estimation of distribution algorithm for solving hybrid flow-shop scheduling problem with identical parallel machines. En *The International Journal of Advanced Manufacturing Technology.* [online]. 2013, Vol.68, No. 9-12, [Citado 2016-04-19], p. 2043-2056.

ZANDIEH,M. y KARIMI, N. An adaptive multi-population genetic algorithm to solve the multi-objective group Scheduling problem in Hybrid Flexible Flow Shop with sequence-dependent setup times. En: *Journal of Intelligent Manufacturing*, [online].2011, Vol.22, No. 6 [Citado 2016-04-22], p 979-989

ZHANG, Songyan. Large-scale flow shop scheduling based on genetic algorithm. En: *2nd International Conference on Education Technology and Computer.* [online], 2010. p.308-310. [Citado: 2016-05-17].

ANEXOS

ANEXO A. CÓDIGO EN MATLAB

Función: Instancias

Esta función permite seleccionar la instancia que se quiere ejecutar dentro de las opciones preestablecidas.

Salidas

- MTA: Matriz de tiempo de preparación o alistamiento si es el primer trabajo que se realiza en cada máquina.
- MTAS: Matriz de tiempo de alistamiento cuando los trabajos son dependientes de la secuencia
- mtas: Matriz que determina la posición de las filas de la matriz MTAS teniendo en cuenta el trabajo actual y el anterior.
- MTP: Matriz de tiempo de preparación de los trabajos en cada etapa.
- maquinas: Vector que contiene la cantidad de máquinas disponibles en cada etapa.
- etapas: Cantidad de etapas de la instancia
- trabajos: Cantidad de trabajos

Código:

```
function [MTA,MTP,MTAS,mtas,maquina,etapas,trabajos]=instancias()
instancias=menu('Escoja la instancia a ejecutar:','20x2','20x4',
'20x8','50x2','50x4','50x8','80x2','80x4','80x8','120x2','120x4','120x8')
if isempty(instancias)
    instancias='Debe seleccionar una instancia de prueba';
end
switch instancias
    case 1
        MTA=xlsread('20X2X1','MTA','A1:B20');
        MTP=xlsread('20X2X1','MTP','A1:B20');
```

```
    MTAS=xlsread('20X2X1','MTAS','A1:B380');
    mtas=xlsread('20X2X1','mtas1','A1:T20');
    maquina=[2,2];
    etapas=2;
    trabajos=20;
case 2
    MTA=xlsread('20X4X144','MTA','A1:D20');
    MTP=xlsread('20X4X144','MTP','A1:D20');
    MTAS=xlsread('20X4X144','MTAS','A1:D380');
    mtas=xlsread('20X4X144','mtas1','A1:T20');
    maquina=[1,3,4,2];
    etapas=4;
    trabajos=20;
case 3
    MTA=xlsread('20X8X2','MTA','A1:H20');
    MTP=xlsread('20X8X2','MTP','A1:H20');
    MTAS=xlsread('20X8X2','MTAS','A1:H380');
    mtas=xlsread('20X8X2','mtas1','A1:T20');
    maquina=[2,2,2,2,2,2,2,2];
    etapas=8;
    trabajos=20;
case 4,
    MTA=xlsread('50X2X1','MTA','A1:B50');
    MTP=xlsread('50X2X1','MTP','A1:B50');
    MTAS=xlsread('50X2X1','MTAS','A1:B2450');
    mtas=xlsread('50X2X1','mtas1','A1:AX50');
    maquina=[2,2];
    etapas =2;
    trabajos=50;
case 5
    MTA=xlsread('50X4X44','MTA','A1:D50');
    MTP=xlsread('50X4X44','MTP','A1:D50');
    MTAS=xlsread('50X4X44','MTAS','A1:D2450');
    mtas=xlsread('50X4X44','mtas1','A1:AX50');
    maquina=[2,1,4,3];
    etapas=4;
    trabajos=50;
case 6
```

```
    MTA=xlsread('50X8X6','MTA','A1:H50');
    MTP=xlsread('50X8X6','MTP','A1:H50');
    MTAS=xlsread('50X8X6','MTAS','A1:H2450');
    mtas=xlsread('50X8X6','mtas1','A1:AX50');
    maquina=[2,2,2,2,2,2,2,2];
    etapas=8;
    trabajos=50;
case 7
    MTA=xlsread('80X2X6','MTA','A1:B80');
    MTP=xlsread('80X2X6','MTP','A1:B80');
    MTAS=xlsread('80X2X6','MTAS','A1:B6320');
    mtas=xlsread('80X2X6','mtas1','A1:CB80');
    maquina=[2,2];
    etapas=2;
    trabajos=80;
case 8
    MTA=xlsread('80X4X1','MTA','A1:D80');
    MTP=xlsread('80X4X1','MTP','A1:D80');
    MTAS=xlsread('80X4X1','MTAS','A1:D6320');
    mtas=xlsread('80X4X1','mtas1','A1:CB80');
    maquina=[2,2,2,2];
    etapas=4;
    trabajos=80;
case 9
    MTA=xlsread('80X8X1','MTA','A1:H80');
    MTP=xlsread('80X8X1','MTP','A1:H80');
    MTAS=xlsread('80X8X1','MTAS','A1:H6320');
    mtas=xlsread('80X8X1','mtas1','A1:CB80');
    maquina=[2,2,2,2,2,2,2,2];
    etapas=8;
    trabajos=80;
case 10
    MTA=xlsread('120X2X1','MTA','A1:B120');
    MTP=xlsread('120X2X1','MTP','A1:B120');
    MTAS=xlsread('120X2X1','MTAS','A1:B14280');
    mtas=xlsread('120X2X1','mtas1','A1:DP120');
    maquina=[2,2];
    etapas=2;
```

```
        trabajos=120;
    case 11
        MTA=xlsread('120X4X15','MTA','A1:D120');
        MTP=xlsread('120X4X15','MTP','A1:D120');
        MTAS=xlsread('120X4X15','MTAS','A1:D14280');
        mtas=xlsread('120X4X15','mtas1','A1:DP120');
        maquina=[3,1,4,2];
        etapas=4;
        trabajos=120;
    case 12
        MTA=xlsread('120X8X4','MTA','A1:H120');
        MTP=xlsread('120X8X4','MTP','A1:H120');
        MTAS=xlsread('120X8X4','MTAS','A1:H14280');
        mtas=xlsread('120X8X4','mtas1','A1:DP120');
        maquina=[2,2,2,2,2,2,2,2];
        etapas=8;
        trabajos=120;
    otherwise
        warning('Debe Seleccionar una instancia valida:')
end
```

Función: Parámetros

Esta función permite escoger la combinación de parámetros: tamaño de poblacio, diversidad e iteraciones con los cuales se ejecutara el algoritmo.

Salidas

- tampob: Tamaño de la población
- poblacion: Tamaño de la población
- d: Diversidad aceptada
- niter número de iteraciones a ejecutarse

Código:

```
function[tampob,poblacion,d,niter]=parametros()
parametros=menu('Escoja la combinacion de parametros donde m es tamaño de
la población y d es la diversidad aceptada: ',
'm=100,d=2,iteraciones=1000 ','m=100,d=5,iteraciones=1000',
'm=100,d=2,iteraciones=1500','m=100,d=5,
iteraciones=1500','m=50,d=2,iteraciones=1000
','m=50,d=5,iteraciones=1000', 'm=50,d=2,iteraciones=1500','m=50,d=5,
iteraciones=1500');
if isempty(parametros)
    parametros='Debe seleccionar una combinacion de parametros';
end
switch parametros
    case 1
        tampob=100;
        poblacion=100;
        d=2;
        niter=1000;
    case 2
        tampob=100;
        poblacion=100;
        d=5;
        niter=1000;
    case 3
        tampob=100;
        poblacion=100;
        d=2;
        niter=1500;
    case 4
        tampob=100;
        poblacion=100;
        d=5;
        niter=1500;
    case 5
        tampob=50;
        poblacion=50;
        d=2;
        niter=1000;
```

```
    case 6
        tampob=50;
        poblacion=50;
        d=5;
        niter=1000;
    case 7
        tampob=50;
        poblacion=50;
        d=2;
        niter=1500;
    case 8
        tampob=50;
        poblacion=50;
        d=5;
        niter=1500;
    otherwise
        warning('Debe seleccionar una combinacion valida')
end
```

Función: Población inicial

Esta función genera la población inicial del problema estudiado a través de una permutación de los trabajos a realizarse; se crea una matriz denominada A de tamaño m x n, donde m es el tamaño de la población y n el número de trabajos a ser procesados. También se evalúa que no existan soluciones repetidas en la matriz A.

Entradas:

- tampob: tamaño de la población determinada por los parámetros del problema.
- trabajos: número de trabajos correspondientes a la instancia validada.

Salidas:

- matriz A: matriz de soluciones del problema

Código:

```
function[A]=pinicial(tampob,trabajos)
A=zeros(tampob,trabajos);
for ii=1:tampob,
    A(ii,:)=randperm(trabajos);
end
A;
%%% Aplicación Criterio de unicidad
posicion=1;
while posicion<=tampob,
    comp=A(posicion,:);
    z=0;
    for mm=1:tampob,
        if mm~=posicion,
            if comp==A(mm,:),
                z=1;
                break
            end
        end
    end
    if z==1,
        A(posicion,:)=randperm(trabajos);
    else posicion=posicion+1;
    end
end
A
```

Función: MAKESPAN O FITNESS

Esta función calcula el Makespan de todos los cromosomas que conforman la matriz A, para esto primero se realiza la asignación de las maquinas disponibles en cada etapa, posteriormente se calculan los tiempos de alistamiento para cada trabajo en cada etapa teniendo en cuenta si es el primer trabajo que se realiza en cada máquina o si es un trabajo antecedido por otro, igualmente se calcula el tiempo de procesamiento de cada trabajo en cada etapa para finalmente calcular

el momento de terminación de los trabajos para así poder generar una matriz denominada MAKESPAN de tamaño mx1 donde m es el tamaño de la población, es decir, esta matriz tiene el makespan para cada una de las soluciones de la matriz A.

Entradas:

- etapas: Cantidad de etapas en cada instancia
- trabajos: Cantidad de trabajos en cada instancia
- maquina: Vector de cantidad de máquinas disponibles en cada etapa de la instancia.
- MTA: Matriz de tiempos de alistamiento si es el primer trabajo que es procesado en la máquina.
- MTAS: Matriz de tiempos de alistamiento si los trabajos son dependientes de la secuencia.
- mtas: Matriz que determina la posición de las filas de la matriz MTAS teniendo en cuenta el trabajo actual y el anterior.
- MTP: Matriz de tiempos de procesamiento de los trabajos en cada etapa
- poblacion: Tamaño de la poblacion
- A: Matriz de soluciones del problema

Salida

- MAKESPAN: Vector columna que contiene el momento de terminación de la secuencia de los trabajos de la matriz A
- asignacion: Hipermatriz de tamaño etapas x trabajos x poblacion que representa la asignación de las máquinas disponibles en cada etapa para cada trabajo para cada individuo de la matriz A

Código:

```
function[MAKESPAN,asignacion]=fitness(etapas,trabajos,maquina,MTA,MTAS,mt
as,MTP,poblacion,A)
%%% Asignacion de maquinas disponibles por etapa
asignacion=zeros(etapas,trabajos,poblacion);
for k=1:poblacion;
    for i=1:etapas;
        for j=1:trabajos;
            w=0;
            if j==1;
                if MTP(A(k,j),i)==0;
%% Si el tiempo de procesamiento del trabajo en esa etapa es 0, es decir,
no es procesado en esa etapa por consiguiente se asigna la maquina 0
                    asignacion(i,j,k)=0;
                else asignacion(i,j,k)=ceil(maquina(i)*rand(1,1));
                end
            end
            while j~=1 && w==0;
                if MTP(A(k,j),i)==0;
%% Si el tiempo de procesamiento del trabajo en esa etapa es 0, es decir,
no es procesado en esa etapa por consiguiente se asigna la maquina 0
                    asignacion(i,j,k)=0;
                else  asignacion(i,j,k)=ceil(maquina(i)*rand(1,1));
                end
                if asignacion(i,j,k)==asignacion(i,j-1,k);
%%% Si la maquina asignada es igual a la maquina asignada para el trabajo
anterior, se debe volver a realizar la asignacion
                    if asignacion(i,j,k)==0
                        w=1;
                    else
                        if maquina(1,i)==1
                            w=1;
                       else
asignacion(i,j,k)=ceil(maquina(i)*rand(1,1));
                            w=0;
                        end
                    end
                else
```

```
                    if asignacion(i,j-1,k)==0
                        if j==2
                            w=1;
                        else
                            if maquina(1,i)==1
                                w=1;
                            else
                                for s=j-2:-1:1
                                    if asignacion(i,s,k)~=0
                                        maq=asignacion(i,s,k);
                                        break
                                    else w=1;
                                        maq=0;
                                    end
                                    break
                                end
                                if asignacion(i,j,k)==maq
asignacion(i,j,k)=ceil(maquina(i)*rand(1,1));
                                    w=0;
                                else w=1;
                                end
                            end
                        else w=1;
                        end
                    end
                end
            end
        end
    end
end
%%% Se determina cuales trabajos son los primeros en realizarse en cada
maquina disponible en cada etapa
asignacion1=zeros(etapas,trabajos,poblacion);
for z=1:poblacion,
    for k=1:etapas,
        for i=1:maquina(k),
            w=0;
            while w == 0;
```

```
                for j=1:trabajos,
                    if asignacion(k,j,z)==i,
                        w=1;
                        break
                    end
                end
                if w==1,
                    asignacion1(k,j,z)=1;%%% Si es 1 es el primer trabajo
en la maquina si es 0 tiene una secuencia antes de ser procesado
                end
                break
            end
        end
    end
end
 %%% Construccion de la matriz mtiempo que guardara los tiempos de
preparacion de los trabajos dependiendo las maquinas asignadas para ellos
en cada etapa, ademas de los tiempos de procesamiento de cada trabajo en
cada etapa
mtiempo=zeros(trabajos,etapas,poblacion);
for z=1:poblacion,
    cromosoma=A(z,:);
%%% Suma de los tiempos de alistamientos de cada trabajo si es el primero
en realizarse en cada maquina en cada etapa
    for k=1:etapas
        for j=1:trabajos
            if asignacion1(k,j,z)==1
                mtiempo(j,k,z)=MTA(cromosoma(1,j),k);
            end
        end
    end
%% Suma de los tiempos de alistamiento dependientes de la secuencia de
produccion
    for k=1:etapas
        for j=1:trabajos
            if asignacion1(k,j,z)==0
                if asignacion(k,j,z)~=0
                    for i=j:-1:1
```

```
                        if asignacion(k,j,z)==asignacion(k,i-1,z)
                            anterior=cromosoma(1,i-1);
                            actual=cromosoma(1,j);
                            break
                        end
                    end
                    psecuencia=mtas(anterior,actual);
                    mtiempo(j,k,z)=MTAS(psecuencia,k);
                else mtiempo(j,k,z)=0;
                end
            end
        end
    end
%%%suma en la matriz mtiempo el valor de los tiempos de procesamiento de
los trabajos en cada etapa
    for k=1:etapas;
        for j=1:trabajos;
            mtiempo(j,k,z)=mtiempo(j,k,z)+MTP(cromosoma(1,j),k);
        end
    end
    mtiempo;
end
%%% Matriz makespan determina el tiempo de finalizacion de cada trabajo
en cada maquina en cada etapa; para todas las etapas excepto la primera
se tiene en cuenta el tiempo de finalizacion de el mismo trabajo en la
etapa anterior y el tiempo de finalizacion del trabajo anterior en la
maquina que sera utilizada, eligiendose el tiempo mayor de estos dos.

makespan=zeros(trabajos,etapas,poblacion);
MAKESPAN=zeros(poblacion,1);
for z=1:poblacion,
    kk=1;
%%% Para la primera etapa de procesamiento
    for ii=1:trabajos,
        if ii==1,
            makespan(1,kk,z)=mtiempo(1,kk,z);
        end
        maquina1=asignacion(kk,ii,z);
```

```
        w=0;
        for j=ii-1:-1:1;
            if maquina1==asignacion(kk,j,z),
                w=1;
                break
            end
        end
        if w==1,
            makespan(ii,kk,z)=mtiempo(ii,kk,z)+makespan(j,kk,z);
        else
            makespan(ii,kk,z)=mtiempo(ii,kk,z);
        end
    end

    for k=2:etapas,
        i=1;
        makespan(1,k,z)=makespan(i,k-1,z)+mtiempo(i,k,z);
        for i=2:trabajos,
            maquina1=asignacion(k,i,z);
            w=0;
            for j=i-1:-1:1;
                if maquina1==asignacion(k,j,z);
                    w=1;
                    break
                end
            end
            if w==1;
                mk=makespan(i,k-1,z);
%%% Punto del tiempo donde termino de ser procesado el trabajo en la
etapa anterior
                mk1=makespan(j,k,z);
%%% Punto del tiempo donde se desocupa la maquina a utilizar, es decir,
punto en el cual la tarea anterior ejecutada en esa maquina se termina de
realizar
                if mk1>mk,
%%% Si el tiempo de desocupe de la maquina es mayor al tiempo donde se
termina el mismo trabajo en la etapa anterior
                    makespan(i,k,z)=mtiempo(i,k,z)+mk1;
```

```
                else
                    makespan(i,k,z)=mtiempo(i,k,z)+mk;
                end
            else
                makespan(i,k,z)=mtiempo(i,k,z)+makespan(i,k-1,z);
            end
        end
    end
    MAKESPAN(z,1)=max(makespan(:,etapas,z),[],1);
end
```

Función: Torneos

Esta función realiza dos torneos, en los cuales se escogen de forma aleatoria dos individuos de la población a los cuales se les compara el MAKESPAN, de tal manera que el individuo con menor MAKESPAN es seleccionado como ganador.

Entradas:

- tampob: Cantidad de individuos que conforman la matriz A
- A: Matriz de soluciones
- MAKESPAN: Vector de tiempos de terminación de los trabajos en la instancia

Salidas:

- ganador1: Cromosoma ganador del primer torneo
- ganador2: Cromosoma ganador del segundo torneo

Código:

```
function[ganador1,ganador2]=torneos(tampob,A,MAKESPAN)
%%% Primer torneo
m=tampob;
%%% Seleccion de individuos de la poblacion, asegurando que sean
diferentes
```

```
cromosoma1=ceil(m*rand(1,1));
cromosoma2=ceil(m*rand(1,1));
while cromosoma1==cromosoma2,
    cromosoma2=ceil(m*rand(1,1));
end
%%% Despues de  seleccionado el individuo se saca de la matriz poblacion
ind1=A(cromosoma1,:);
ind2=A(cromosoma2,:);
%%% Makespan del individuo seleccionado
fobj1=MAKESPAN(cromosoma1);
fobj2=MAKESPAN(cromosoma2);
%%% Si el makespan del individuo 1 es menor que el makespan del individuo
2 el cromosoma ganador es el 1
if fobj1<fobj2,
   ganador1=ind1;
else
    ganador1=ind2;
end
%%% Segundo toneo
%%% Seleccion de individuos de la poblacion, asegurando que sean
diferentes
cromosoma3=ceil(m*rand(1,1));
cromosoma4=ceil(m*rand(1,1));
while cromosoma3==cromosoma4,
    cromosoma4=ceil(m*rand(1,1));
end
%%% Despues de  seleccionado el individuo se saca de la matriz poblacion
ind3=A(cromosoma3,:);
ind4=A(cromosoma4,:);

%%% Makespan del individuo seleccionado
fobj3=MAKESPAN(cromosoma3);
fobj4=MAKESPAN(cromosoma4);
%%% Si el makespan del individuo 3 es menor que el makespan del individuo
4 el cromosoma ganador es el 3
if fobj3<fobj4,
ganador2=ind3;
else
```

```
    ganador2=ind4;
end
```

Función: Cruzamiento

Esta función realiza el cruzamiento o reproducción, en el cual se cruza la información de los dos individuos ganadores para la creación de los hijos.

Entrada:

- trabajos: Cantidad de trabajos de la instancia
- ganador1: Cromosoma que será catalogado como Padre1
- ganador2: Cromosoma que será catalogado como Padre2

Salidas:

- hijo1: Cromosoma hijo
- hijo2: Cromosoma hijo

Código:

```
function [hijo1, hijo2]=cruzamiento(trabajos,ganador1,ganador2)
n=trabajos;
padre1=ganador1;
padre2=ganador2;
%%% Establecimiento del punto de cruce, exeptuando la primera y ultima
posicion para asegurarse de que exista el cruce de cromosomas
puntocruce=ceil(n*rand(1,1));
if puntocruce==n,
    puntocruce=ceil(n*rand(1,1));
end
%%% Procedimiento de cruce
vcruce=zeros(1,n);
for i=1:n,
%%% Establece un vector de 0 y 1 donde 0 significa que el valor debe
cambiar por el del otro padre y 1 debe permanecer el mismo
```

```
    if i<=puntocruce,
        pc=1;
    else pc=0;
    end
    vcruce(1,i)=pc;
end
disp(vcruce);
%%% Generacion de los hijos
hijo1=zeros(1,n);
hijo2=zeros(1,n);

for i=1:n,
    if vcruce(1,i)==1,
        hijo1(1,i)=padre1(1,i);
        hijo2(1,i)=padre2(1,i);
    else
        hijo1(1,i)=padre2(1,i);
        hijo2(1,i)=padre1(1,i);
    end
end
%%%% Al generarse los hijos en el paso anterior y como producto del cruce
de informacion algunos trabajos pueden quedarse repetidos u olvidados;
por lo tanto se  debe detectar cuales genes de los cromosomas hijos
corresponden a trabajos ya secuenciados. Los trabajos repetidos o ya
secuenciados son reemplazados por 0 para posteriormente ser suplidos con
los trabajos olvidados en el cruzamiento

for i=2:n,
    w=1;
    for j=1:i,
        while j~=i,
            if hijo1(1,i)==hijo1(1,j),
                w=0;
                break
            end
            break
        end
    end
```

```
    if w==0,
        hijo1(1,i)=0;
    end
end
for i=2:n,
    w=1;
    for j=1:i,
        while j~=i,
            if hijo2(1,i)==hijo2(1,j),
                w=0;
                break
            end
            break
        end
    end
    if w==0,
        hijo2(1,i)=0;
    end
end
Se crea un vector con los trabajos olvidados y aleatoriamente llenan las
posiciones que estan en 0 en los cromosomas hijos
faltante=zeros(1,n);
for i=1:n,
    trabajofaltante=1:n;
    tf=zeros(1,n);
    for ij=1:n,
        if trabajofaltante(1,i)==hijo1(1,ij),
            tf(1,i)=1;
            break
        end
    end
    suma=sum(tf);
    if suma==0,
        faltante(1,i)=i;
    end
end
falta=0;
for i=1:n,
```

```
    if faltante(1,i)~=0,
        falta=1+falta;
    end
end
tfaltante=zeros(1,falta);

for i=1:falta,
    for j=1:n,
            if faltante(1,j)~=0,
                tfaltante(1,i)=faltante(1,j);
                faltante(1,j)=0;
                break
            end
    end
end

tp=randperm(falta);
tordenado=zeros(1,falta);
for i=1:falta,
    tordenado(1,i)=tfaltante(1,tp(i));
end
w=0;
for i=1:n,
    if hijo1(1,i)==0,
        w=1+w;
       for j=1:falta,
           if w==j;
               hijo1(1,i)=tordenado(1,j);
               break
           end
       end
    end
end
%%%% LLenado de las posiciones faltantes del hijo2
faltante=zeros(1,n);
for i=1:n,
    trabajofaltante=1:n;
    tf=zeros(1,n);
```

```
    for ij=1:n,
        if trabajofaltante(1,i)==hijo2(1,ij),
            tf(1,i)=1;
            break
        end
    end
    suma=sum(tf);
    if suma==0,
        faltante(1,i)=i;
    end
end
falta=0;
for i=1:n,
    if faltante(1,i)~=0,
        falta=1+falta;
    end
end
tfaltante=zeros(1,falta);

for i=1:falta,
    for j=1:n,
        if faltante(1,j)~=0,
            tfaltante(1,i)=faltante(1,j);
            faltante(1,j)=0;
            break
        end
    end
end
tp=randperm(falta);
tordenado=zeros(1,falta);
for i=1:falta,
    tordenado(1,i)=tfaltante(1,tp(i));
end
w=0;
for i=1:n,
    if hijo2(1,i)==0,
        w=1+w;
       for j=1:falta,
```

```
            if w==j,
                hijo2(1,i)=tordenado(1,j);
             break
            end
        end
    end
end
```

Función: Mutación

Esta función intercambia el valor de dos posiciones en el cromosoma seleccionado, siempre que se supere el valor establecido como tasa de mutación.

Entrada:

- trabajos: Cantidad de trabajos de la instancia
- hijo1: Cromosoma hijo
- hijo2: Cromosoma hijo

Salida:

- hijo1: Cromosoma hijo afectado por la mutación
- hijo2: Cromosoma hijo afectado por la mutación

Código:

```
function [hijo1,hijo2]=mutacion(trabajos,hijo1,hijo2)
TM=0.01;    %%% Tasa de mutacion
mtc=rand;
n=trabajos;
if mtc<TM,
    gen1=ceil(n*rand(1,1));
    gen2=ceil(n*rand(1,1));
%%% Se asegura que los genes a intercambiar no sean los mismos
    while gen1==gen2,
```

```
        gen2=ceil(n*rand(1,1));
    end
    mut1=hijo1(gen1);
    mut2=hijo1(gen2);
    hijo1(gen1)=mut2;
    hijo1(gen2)=mut1;
end
if mtc<TM,
    gen1=ceil(n*rand(1,1));
    gen2=ceil(n*rand(1,1));
%%% Se asegura que los genes a intercambiar no sean los mismos
    while gen1==gen2,
        gen2=ceil(n*rand(1,1));
    end
    mut1=hijo2(gen1);
    mut2=hijo2(gen2);
    hijo2(gen1)=mut2;
    hijo2(gen2)=mut1;
end
```

Función: MAKESPAN1 O FITNESS1

Esta función determina el MAKESPAN de los hijos.

Entrada:

- CROMOSOMA: Cromosoma hijo al cual se le calculara el makespan
- etapas: Cantidad de etapas de la instancia.
- trabajos: Cantidad de trabajos de la instancia
- maquina: Vector de cantidad de máquinas disponibles en cada etapa de la instancia.
- MTA: Matriz de tiempos de alistamiento si es el primer trabajo que es procesado en la máquina.
- MTAS: Matriz de tiempos de alistamiento si los trabajos son dependientes de la secuencia.

- mtas: Matriz que determina la posición de las filas de la matriz MTAS teniendo en cuenta el trabajo actual y el anterior.
- MTP: Matriz de tiempos de procesamiento de los trabajos en cada etapa población: Tamaño de la población.

Salida:

- MAKESPAN1: Momento de terminación de la secuencia de trabajos.
- ASIG: Asignación de las máquinas disponibles en cada etapa para cada trabajo

Código:

```
function[MAKESPAN1,ASIG]=fitness1(CROMOSOMA,etapas,trabajos,maquina,MTA,M
TAS,mtas,MTP)
%%% Asignacion de maquinas disponibles por etapa
ASIG=zeros(etapas,trabajos);
for i=1:etapas
    for j=1:trabajos
        w=0;
           if j==1
               if MTP(CROMOSOMA(1,j),i)==0
%% Si el tiempo de procesamiento del trabajo en esa etapa es 0, es decir,
no es procesado en esa etapa por consiguiente se asigna la maquina 0
                   ASIG(i,j)=0;
               else ASIG(i,j)=ceil(maquina(i)*rand(1,1));
               end
           end
           while j~=1 && w==0
               if MTP(CROMOSOMA(1,j),i)==0
%% Si el tiempo de procesamiento del trabajo en esa etapa es 0, es decir,
no es procesado en esa etapa por consiguiente se asigna la maquina 0
                   ASIG(i,j)=0 ;
                   w=1;
               else  ASIG(i,j)=ceil(maquina(i)*rand(1,1));
               end
```

```
            if ASIG(i,j)==ASIG(i,j-1)
%% Si la maquina asignada es igual a la maquina asignada para el trabajo
anterior, se debe volver a realizar la asignacion
                if ASIG(i,j)==0
                    w=1;
                else
                    if maquina(1,i)==1
                        w=1;
                    else ASIG(i,j)=ceil(maquina(i)*rand(1,1));
                        w=0;
                    end
                end
            else
                if ASIG(i,j-1)==0
                    if j==2
                        w=1;
                    else
                        if maquina(1,i)==1
                            w=1;
                        else
                            for s=j-2:-1:1
                                if ASIG(i,s)~=0
                                    maq=ASIG(i,s);
                                    break
                                else w=1;
                                    maq=0;
                                end
                                break
                            end
                            if ASIG(i,j)==maq
                                ASIG(i,j)=ceil(maquina(i)*rand(1,1));
                                w=0;
                            else w=1;
                            end
                        end
                    end
                else w=1;
                end
```

```
            end
        end
    end
end
%% Se determina cuales trabajos son los primeros en realizarse en cada
maquina disponible en cada etapa
asignacion1=zeros(etapas,trabajos);
for k=1:etapas,
    for i=1:maquina(k),
        w=0;
        while w == 0;
            for j=1:trabajos,
                if ASIG(k,j)==i,
                    w=1;
                    break
                end
            end
            if w==1,
                asignacion1(k,j)=1;
%% Si es 1 es el primer trabajo en la maquina si es 0 tiene una secuencia
antes de ser procesado
            end
            break
        end
    end
end
%%% Construccion de la matriz mtiempo que guardara los tiempos de
preparacion de los trabajos dependiendo las maquinas asignadas para ellos
en cada etapa, ademas de los tiempos de procesamiento de cada trabajo en
cada etapa
mtiempo=zeros(trabajos,etapas);
%%% Suma de los tiempos de alistamientos de cada trabajo si es el primero
en realizarse en cada maquina en cada etapa
for k=1:etapas;
    for j=1:trabajos;
        if asignacion1(k,j)==1;
            mtiempo(j,k)=MTA(CROMOSOMA(1,j),k);
        end
```

```
    end
end
%%% Suma de los tiempos de alistamiento dependientes de la secuencia de
produccion
for k=1:etapas;
    for j=1:trabajos;
        if asignacion1(k,j)==0;
            if ASIG(k,j)~=0
            for i=j:-1:1;
                if ASIG(k,j)==ASIG(k,i-1);
                    anterior=CROMOSOMA(1,i-1);
                    actual=CROMOSOMA(1,j);
                    break
                end
            end
            psecuencia=mtas(anterior,actual);
            mtiempo(j,k)=MTAS(psecuencia,k);
            else mtiempo(j,k)=0;
            end
        end
    end
end
%%%%suma en la matriz mtiempo el valor de los tiempos de procesamiento de
los trabajos en cada etapa
for k=1:etapas;
    for j=1:trabajos;
        mtiempo(j,k)=mtiempo(j,k)+MTP(CROMOSOMA(1,j),k);
    end
end
%%% Matriz makespan determina el tiempo de finalizacion de cada trabajo
en cada maquina en cada etapa; para todas las etapas exepto la primera se
tiene en cuenta el tiempo de finalizacion de el mismo trabajo en la etapa
anterior y el tiempo de finalizacion del trabajo anterior en la maquina
que sera utilizada, eligiendose el tiempo mayor de estos dos.
makespan=zeros(trabajos,etapas);
kk=1;
for ii=1:trabajos,
    if ii==1,
```

```
        makespan(1,kk)=mtiempo(1,kk);
    end
    maquina1=ASIG(kk,ii);
    w=0;
    for j=ii-1:-1:1;
        if maquina1==ASIG(kk,j),
            w=1;
            break
        end
    end
    if w==1,
        makespan(ii,kk)=mtiempo(ii,kk)+makespan(j,kk);
    else
        makespan(ii,kk)=mtiempo(ii,kk);
    end
end

for k=2:etapas,
    i=1;
    makespan(1,k)=makespan(i,k-1)+mtiempo(i,k);
    for i=2:trabajos,
        maquina1=ASIG(k,i);
        w=0;
        for j=i-1:-1:1;
            if maquina1==ASIG(k,j);
                w=1;
                break
            end
        end
        if w==1;
%% Punto del tiempo donde termino la tarea en la etapa anterior
            mk=makespan(i,k-1);
%% Punto del tiempo donde se desocupa la maquina a utilizar
            mk1=makespan(j,k);
%% Si el tiempo de desocupe de la maquina es mayor al tiempo donde se
termina el mismo trabajo en la actividad anterior
            if mk1>mk,
                makespan(i,k)=mtiempo(i,k)+mk1;
```

```
            else
                makespan(i,k)=mtiempo(i,k)+mk;
            end
        else
            makespan(i,k)=mtiempo(i,k)+makespan(i,k-1);
        end
    end
end
MAKESPAN1(1,1)=max(makespan(:,etapas),[],1);
end
```

Función: Descendiente

Esta función determina a partir de las dos soluciones hijas cuál de ellas tendrá la posibilidad de ingresar a la población o matriz A. El procedimiento para la selección de una de las soluciones hijas se realiza de forma aleatoria pero donde el hijo con mejor Makespan tiene más posibilidades de ser elegido.

Entradas:

- hijo1: Cromosoma hijo
- hijo2: Cromosoma hijo
- hijo1MAKESPAN1: Makespan del hijo1
- hijo2MAKESPAN2: Makespan del hijo2
- hijo1asig: Asignación de máquinas disponibles en cada etapa para cada trabajo del hijo1.
- hijo2asig: Asignación de máquinas disponibles en cada etapa para cada trabajo del hijo2.

Salidas:

- descendiente: Cromosoma seleccionado para ingresar a la matriz A
- fobjdescendiente: Makespan del descendiente
- FOBASIG: Asignación de máquinas en cada etapa para el descendiente.

Código:

```
function[descendiente,fobjdescendiente,FOBASIG]=descendiente1(hijo1,hijo2
,hijo1MAKESPAN1,hijo2MAKESPAN1,hijo1asig,hijo2asig)
total=hijo1MAKESPAN1+hijo2MAKESPAN1;
prob1=1-(hijo1MAKESPAN1/total);
prob2=1-(hijo2MAKESPAN1/total);
entrar=rand;
if entrar<=prob1
    descendiente=hijo1;
    fobjdescendiente=hijo1MAKESPAN1;
    FOBASIG=hijo1asig;
else
    descendiente=hijo2;
    fobjdescendiente=hijo2MAKESPAN1;
    FOBASIG=hijo2asig;
end
```

FUNCION: UNICO

Esta función determina si la secuencia de trabajos que representa el descendiente no se encuentra repetida en la matriz A o matriz de soluciones.

Entradas:

- descendiente: Cromosoma seleccionado para ingresar a la matriz A
- tampob: Tamaño de la poblacion o matriz A
- A: Matriz de soluciones o poblacion

Salidas:

- descendienteunico: Escalar que representa si el cromosoma descendiente es único (1) o repetido (0).

Código:

```
function [descendienteunico]=unico(descendiente,tampob,A)
z=0;
 for mm=1:tampob
     if descendiente==A(mm,:)
         z=1;
         break
     else
     end
 end
 if z==0
     disp('descendiente unico')
     descendienteunico=1;
 else
     disp('descendiente repetido')
    descendienteunico=0;
 end
```

Función: Diversidad

Esta función determina si el descendiente cumple con el criterio de diversidad respecto a los individuos de la matriz A.

Entradas:

- d:diversidad aceptada
- tampob: Tamaño de la población
- descendiente: Cromosoma seleccionado para ingresar a la matriz A
- trabajos: Cantidad de trabajos de la instancia
- A: Matriz de soluciones

Salidas:

- criteriodiversidad: Escalar que representa si el cromosoma descendiente cumple con el criterio de diversidad. Si cumple el valor es 1, si no cumple es valor es 0.

Código:

```
function[criteriodiversidad]=diversidad(d,tampob,descendiente,trabajos,A)
%%% Diversidad aceptada
dv=trabajos-d;
Z=zeros(tampob,trabajos);
for j=1:tampob
    Z(j,:)=(descendiente==A(j,:));
end
diversidad1=zeros(j,1);
for j=1:tampob
    diversidad1(j,1)=sum(Z(j,:));
end
%% Criterio de diversidad 1 aceptado 0 rechazado
if max(diversidad1(:,1))<dv
    criteriodiversidad=1;
else criteriodiversidad=0;
end
```

Función: Peor

Esta función determina cual es el individuo en la matriz A con la peor solución para el problema ejecutado.

Entradas:

- Tampob: Tamaño de la población de la matriz A.
- MAKESPAN: Vector de tiempos de terminación de los trabajos en la instancia.

Salidas:

- posicionpeor: Posición de la peor solución en la matriz A
- fpeor: Makespan de la peor solución de la matriz A

Código:

```
function [posicionpeor,fpeor]=peor(tampob,MAKESPAN)
fpeor=max(MAKESPAN,[],1)
for z=1:tampob
    if fpeor==MAKESPAN(z,1)
        posicionpeor=z;
        break
    end
end
```

Función: Reemplazo

Esta función determina si el descendiente puede ingresar a la matriz A de acuerdo a las condiciones de entrada, como lo son:

1. Si el descendiente cumple con los criterios de unicidad y diversidad y el makespan del descendiente es menor al makespan de la peor solución.
2. Si el descendiente no cumple con el criterio de diversidad pero el makespan del descendiente es mejor que el makespan de la incumbente.

Entrada:

- descendienteunico: Escalar que representa si el cromosoma descendiente es único (1) o repetido (0).
- criteriodiversidad: Escalar que representa si el cromosoma descendiente cumple con el criterio de diversidad. Si cumple el valor es 1, si no cumple es valor es 0.
- MAKESPAN: Vector de tiempos de terminación de los trabajos en la instancia.
- fobjdescendiente: Makespan del descendiente

- A: Matriz de soluciones
- posicionpeor: posición de la peor solución de la matriz A
- fpeor: Makespan de la peor solución de la matriz A
- descendiente: Cromosoma hijo
- asignación: Hipermatriz de asignación de las máquinas disponibles en cada etapa para cada trabajo para cada individuo de la matriz A
- FOBASIG: Asignación de las máquinas en cada etapa para cada trabajo del descendiente
- parada: Vector que almacena la cantidad de iteraciones consecutivas donde no se registra un ingreso de descendiente a la matriz A
- iteración: Número de veces que se ha ejecutado el algoritmo en la repetición actual

Salidas:

- iteración: Número de veces que se ha ejecutado el algoritmo en la repetición actual más uno
- parada: Vector que almacena la cantidad de iteraciones consecutivas donde no se registra un ingreso de descendiente a la matriz A
- A: Matriz de soluciones
- Asignación: Hipermatriz de asignación de las máquinas disponibles en cada etapa para cada trabajo para la población actualizada
- MAKESPAN: Vector de tiempos de terminación de los trabajos en la secuencia de la matriz A actualizada.

Código:

```
function
[iteracion,parada,A,asignacion,MAKESPAN]=reemplazo(descendienteunico,crit
eriodiversidad,MAKESPAN,fobjdescendiente,A,posicionpeor,fpeor,descendient
e,asignacion,FOBASIG,parada,iteracion)
cp=100; % Cantidad de iteraciones sin mejora con las cuales el criterio
de parada cambia a 1
```

```
if descendienteunico==1 && criteriodiversidad==1
%%% Si el descendiente cumple los criterios de unicidad y diversidad
    if fobjdescendiente<fpeor
%%% Si el descendiente tiene mejor makespan que la peor solucion de la
poblacion
        %%% Se realiza el remplazo en la poblacion
        A(posicionpeor,:)=descendiente;
        asignacion(:,:,posicionpeor)=FOBASIG;
        MAKESPAN(posicionpeor,:)=fobjdescendiente;
        parada(1,iteracion)=0;
        iteracion=iteracion+1;
    else
        if iteracion==1
            parada(1,iteracion)=1;
        else
            parada(1,iteracion)=1+parada(1,iteracion-1);
        end
        if parada(1,iteracion)==cp
            criterioparada=1;
        end
        iteracion=iteracion+1;
    end
else
    if criteriodiversidad==0
%%% El descendiente no cumple con el criterio de diversidad
        fmejor=min(MAKESPAN);
        if fobjdescendiente<fmejor
%%% Si el descendiente es mejor que la incumbente
            %%% Como el descendiente es mejor que la incumbente ingresa a
la poblacion a reemplazar a la peor solucion
            A(posicionpeor,:)=descendiente;
            MAKESPAN(posicionpeor,:)=fobjdescendiente;
            asignacion(:,:,posicionpeor)=FOBASIG  ;
            parada(1,iteracion)=0 ;
            iteracion=iteracion+1;
        else %%% El descendiente no es mejor que la incumbente, por lo
tanto no se reemplaza en la poblacion
            if iteracion==1
```

```
                parada(1,iteracion)=1;
            else
                if parada(1,iteracion-1)~=0
                    parada(1,iteracion)=parada(1,iteracion-1)+1;
                    if parada(1,iteracion)==cp
                        criterioparada=1;
                    end
                else parada(1,iteracion)=1;
                end
            end
            iteracion=iteracion+1;
        end
    else%%% El descendiente esta repetido, no se reemplaza en la
poblacion
        if iteracion==1
            parada(1,iteracion)=1;
        else
            if parada(1,iteracion-1)~=0
                parada(1,iteracion)=parada(1,iteracion-1)+1;
                if parada(1,iteracion)==cp;
                    criterioparada=1 ;
%%% Se cumple criterio de parada
                end
            else parada(1,iteracion)=1;
            end
        end
        iteracion=iteracion+1;
    end
end
```

PROGRAMA PRINCIPAL

```
%%%SOLUCION DEL PROBLEMA FLOW SHOP HIBRIDO FLEXIBLE EN MAQUINAS PARALELAS
%%%IDENTICAS MEDIANTE EL ALGORITMO GENETICO MODIFICADO DE CHU-BEASLEY
%%%%***Tiempos de alistamiento dependientes de la secuencia
```

```
clear all
clc

%%% SELECCION DE INSTANCIA DE PRUEBA
[MTA,MTP,MTAS,mtas,maquina,etapas,trabajos]=instancias();

%%% SELECCION DE LOS PARAMETROS INICIALES
[tampob,poblacion,d,niter]=parametros();

%%% Cantidad de veces que se ejecutara el algoritmo
f=30;
%%% Matriz donde se guardara la mejor solución encontrada y su tiempo de
ejecución correspondiente  para cada corrida del algoritmo
solu=zeros(f,3);
%%% Matriz donde se guardara la secuencia de la mejor solución encontrada
en cada corrida del algoritmo
PFINAL=zeros(f,trabajos);
%%% Hipermatriz donde se guardara la asignación de las máquinas para cada
mejor solución encontrada en cada corrida del algoritmo
ASIGFINAL=zeros(etapas,trabajos,f);

%%% EJECUCION DEL ALGORITMO
for veces=1:f
    tic   %%% Contador de tiempo de ejecución

%%%%GENERACION DE LA POBLACION INICIAL
[A]=pinicial(tampob,trabajos)

%%%CALCULO DEL MAKESPAN DE LA POBLACION INICIAL
[MAKESPAN,asignacion]=fitness(etapas,trabajos,maquina,MTA,MTAS,mtas,MTP,p
oblacion,A)

%%% Parametros iniciales tenidos en cuenta para el criterio de parada
criterioparada=0;
iteracion=1;
parada=zeros(1,niter);
```

```
while  criterioparada==0 & iteracion<=niter

    %%%%SELECCION POR TORNEO
    [ganador1,ganador2]=torneos(tampob,A,MAKESPAN)

    %%%%CRUZAMIENTO
    [hijo1, hijo2]=cruzamiento(trabajos,ganador1,ganador2)

    %%%MUTACION
    [hijo1,hijo2]=mutacion(trabajos,hijo1,hijo2)

    %%%CALCULO FUNCION OBJETIVO A LOS HIJOS
    %%%hijo1
    CROMOSOMA=hijo1
[MAKESPAN1,ASIG]=fitness1(CROMOSOMA,etapas,trabajos,maquina,MTA,MTAS,mtas
,MTP)
    hijo1MAKESPAN1=MAKESPAN1
    hijo1asig=ASIG;
    %%%hijo2
    CROMOSOMA=hijo2

[MAKESPAN1,ASIG]=fitness1(CROMOSOMA,etapas,trabajos,maquina,MTA,MTAS,mtas
,MTP)
    hijo2MAKESPAN1=MAKESPAN1
    hijo2asig=ASIG;

    %%%SELECCION DESCENDIENTE
[descendiente,fobjdescendiente,FOBASIG]=descendiente1(hijo1,hijo2,hijo1MA
KESPAN1,hijo2MAKESPAN1,hijo1asig,hijo2asig)

    %CRITERIO DE UNICIDAD PARA EVITAR DUPLICIDAD DE INDIVIDUOS
    [descendienteunico]=unico(descendiente,tampob,A)

    %CRITERIO DE DIVERSIDAD
    [criteriodiversidad]=diversidad(d,tampob,descendiente,trabajos,A)
```

```
    %%% REEMPLAZO EN LA POBLACION
    [posicionpeor,fpeor]=peor(tampob,MAKESPAN)
[iteracion,parada,A,asignacion,MAKESPAN]=reemplazo(descendienteunico,crit
eriodiversidad,MAKESPAN,fobjdescendiente,A,posicionpeor,fpeor,descendient
e,asignacion,FOBASIG,parada,iteracion);
end

A;
asignacion;

%%% Seleccion de la mejor solucion de la repeticion
solucion=min(MAKESPAN);
for z=1:tampob
    if solucion==MAKESPAN(z,1)
        posicionsolucion=z;
        break
    end
 end
Mejorsecuencia=A(posicionsolucion,:);
ASIGNACIONSOLUCION=asignacion(:,:,posicionsolucion);
toc
%%% Finaliza el contador de tiempo de la ejecucion de la repeticion

%%% ACTULIZACION DE LA MATRIZ RESPUESTA
solu(veces,1)=solucion;
solu(veces,2)=toc;
solu(veces,3)=iteracion;
PFINAL(veces,:)=Mejorsecuencia;
ASIGFINAL(:,:,veces)=ASIGNACIONSOLUCION;
end

%%% MEJOR SOLUCION ENCONTRADA EN TODA LA EJECUCION
solu
%%% Mejor solucion
mej=min(solu(:,1),[],1)
%%% Posicion en la matriz de soluciones
```

```
[pos]=mejor(mej,solu)
%%% Secuencia de trabajos de la mejor solucion
fin=PFINAL(pos,:)
%%% Asignacion de maquinas de la mejor solucion
asimej=ASIGFINAL(:,:,pos)
```

ANEXO B. ANOVAS

En este anexo se encuentran el análisis de varianza para cada una de las instancias.

ANOVA 20X2

Análisis de Varianza	20X2				
Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Modelo	7	6477,5	925,36	7,53	0
Lineal	3	6173,2	2057,75	16,74	0
Población	1	1096,5	1096,54	8,92	0,003
Diversidad	1	2059,2	2059,2	16,75	0
Iteraciones	1	3017,5	3017,5	24,55	0
Interacciones de 2 términos	3	280,8	93,62	0,76	0,517
Población*Diversidad	1	178,5	178,5	1,45	0,229
Población*Iteraciones	1	2,2	2,2	0,02	0,894
Diversidad*Iteraciones	1	100,1	100,1	0,81	0,386
Interacciones de 3 términos	1	23,4	23,4	0,19	0,663
Población*Diversidad*Iteraciones	1	23,4	23,4	0,19	0,663
Error	232	28519,4	122,93		
Total	239	34997			

ANOVA 20X4

Análisis de Varianza	20X4				
Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Modelo	7	275285	39326	15,95	0
Lineal	3	243744	81248	32,96	0
Población	1	28210	28210	11,44	0,001
Diversidad	1	111284	111284	45,14	0
Iteraciones	1	104250	104250	42,29	0
Interacciones de 2 términos	3	29681	9894	4,01	0,008
Población*Diversidad	1	252	252	0,1	0,749
Población*Iteraciones	1	10454	10454	4,24	0,041
Diversidad*Iteraciones	1	18975	18975	7,7	0,006
Interacciones de 3 términos	1	1859	1859	0,75	0,386
Población*Diversidad*Iteraciones	1	1859	1859	0,75	0,386
Error	232	571902	2465		
Total	239	847187			

ANOVA 20X8

Análisis de Varianza	20X8				
Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Modelo	7	27763	3966,2	12,42	0
Lineal	3	24402	8134,1	25,48	0
Población	1	2483	2483,33	7,78	0,006
Diversidad	1	7504	7504	23,5	0
Iteraciones	1	14415	14415	45,15	0
Interacciones de 2 términos	3	2727	909,1	2,85	0,038
Población*Diversidad	1	1612	1612	5,05	0,026
Población*Iteraciones	1	147	147,3	0,46	0,498
Diversidad*Iteraciones	1	968	968	3,03	0,083
Interacciones de 3 términos	1	634	633,8	1,98	0,16
Población*Diversidad*Iteraciones	1	634	633,8	1,98	0,16
Error	232	74076	319,3		
Total	239	101839			

ANOVA 50X2

Análisis de Varianza	**50x2**				
Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Modelo	7	34798	4971,2	15,35	0
Lineal	3	32480	10826,8	33,42	0
Población	1	14493	14492,6	44,74	0
Diversidad	1	2516	2515,5	7,77	0
Iteraciones	1	15472	15472,2	47,76	0
Interacciones de 2 términos	3	1486	495,2	1,53	0,208
Población*Diversidad	1	924	924,3	2,85	0,0903
Población*Iteraciones	1	358	357,7	1,1	0,294
Diversidad*Iteraciones	1	204	203,5	0,63	0,429
Interacciones de 3 términos	1	833	832,5	2,57	0,11
Población*Diversidad*Iteraciones	1	833	832,5	2,57	0,11
Error	232	75156	323,9		
Total	239	109955			

ANOVA 50X4

Análisis de Varianza	**50x4**				
Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Modelo	7	132288	18898,3	20,94	0
Lineal	3	128314	42771,2	47,4	0
Población	1	23285	23285,4	25,8	0
Diversidad	1	5802	5801,7	6,43	0,012
Iteraciones	1	99227	99226,7	109,95	0
Interacciones de 2 términos	3	2289	763	0,85	0,47
Población*Diversidad	1	147	147,3	0,16	0,687
Población*Iteraciones	1	2018	2018,4	2,24	0,136
Diversidad*Iteraciones	1	123	123,3	0,14	0,712
Interacciones de 3 términos	1	1685	1685,4	1,87	0,173
Población*Diversidad*Iteraciones	1	1685	1685,4	1,87	0,173
Error	232	209366	902,4		
Total	239	341654			

ANOVA 50X8

Análisis de Varianza	**50x8**				
Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Modelo	7	46013	6573,3	7,68	0
Lineal	3	17699	5899,5	6,89	0
Población	1	232	232,1	0,27	0,603
Diversidad	1	866	866,4	1,01	0,315
Iteraciones	1	16600	16600,1	19,4	0
Interacciones de 2 términos	3	26971	8990,2	10,51	0
Población*Diversidad	1	11070	11070	12,94	0
Población*Iteraciones	1	10428	10428	12,19	0,001
Diversidad*Iteraciones	1	5472	5472,2	6,4	0,012
Interacciones de 3 términos	1	1344	1344,3	1,57	0,211
Población*Diversidad*Iteraciones	1	1344	1344,3	1,57	0,211
Error	232	198510	855,6		
Total	239	244524			

ANOVA 80X2

Análisis de Varianza	80x2				
Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Modelo	7	28086	4012,2	10,66	0
Lineal	3	24528	8175,9	21,73	0
Población	1	10733	10733,4	28,53	0
Diversidad	1	189	189	0,5	0,479
Iteraciones	1	13605	13605,2	36,16	0
Interacciones de 2 términos	3	3450	1150	3,06	0,029
Población*Diversidad	1	1	1,2	0	0,955
Población*Iteraciones	1	1832	1831,5	4,87	0,028
Diversidad*Iteraciones	1	1617	1617,2	4,3	0,039
Interacciones de 3 términos	1	108	108	0,29	0,593
Población*Diversidad*Iteraciones	1	108	108	0,29	0,593
Error	232	87287	376,2		
Total	239	115373			

ANOVA 80X4

Análisis de Varianza	80x4				
Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Modelo	7	1263670	180524	4,15	0
Lineal	3	983630	327877	7,54	0
Población	1	442986	442986	10,18	0,002
Diversidad	1	43875	43875	1,01	0,316
Iteraciones	1	496769	496769	11,42	0,001
Interacciones de 2 términos	3	218183	72728	1,67	0,174
Población*Diversidad	1	31031	31031	0,71	0,399
Población*Iteraciones	1	183541	183541	4,22	0,041
Diversidad*Iteraciones	1	3612	3612	0,08	0,774
Interacciones de 3 términos	1	61857	61857	1,42	0,234
Población*Diversidad*Iteraciones	1	61857	61857	1,42	0,234
Error	232	10094235	43510		
Total	239	11357905			

ANOVA 80X8

Análisis de Varianza	80x8				
Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Modelo	7	79884	11412	8,77	0
Lineal	3	72193	24064,5	18,49	0
Población	1	4183	4183,4	3,21	0,074
Diversidad	1	608	608	0,47	0,495
Iteraciones	1	67402	67402	51,8	0
Interacciones de 2 términos	3	3236	1078,6	0,83	0,479
Población*Diversidad	1	2653	2653,4	2,04	0,155
Población*Iteraciones	1	322	322	0,25	0,619
Diversidad*Iteraciones	1	260	260,4	0,2	0,655
Interacciones de 3 términos	1	4454	4454,8	3,42	0,066
Población*Diversidad*Iteraciones	1	4454	4454,8	3,42	0,066
Error	232	301882	1301,2		
Total	239	381766			

ANOVA 120X2

Análisis de Varianza	120x2				
Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Modelo	7	1965155	280736	3,46	0,002
Lineal	3	1714956	571652	7,05	0
Población	1	1439641	1439641	17,75	0
Diversidad	1	22854	22854	0,28	0,596
Iteraciones	1	252461	252461	3,11	0,079
Interacciones de 2 términos	3	222075	74025	0,91	0,435
Población*Diversidad	1	108545	108545	1,34	0,249
Población*Iteraciones	1	56365	56365	0,69	0,405
Diversidad*Iteraciones	1	57165	57165	0,7	0,402
Interacciones de 3 términos	1	28123	28123	0,35	0,557
Población*Diversidad*Iteraciones	1	28123	28123	0,35	0,557
Error	232	18816135	81104		
Total	239	20781290			

ANOVA 120X4

Análisis de Varianza	**120x4**				
Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Modelo	7	364867	52124	23,15	0
Lineal	3	345179	115060	51,1	0
Población	1	138817	138817	61,65	0
Diversidad	1	2196	2196	0,98	0,324
Iteraciones	1	204167	204167	90,67	0
Interacciones de 2 términos	3	19641	6547	2,91	0,035
Población*Diversidad	1	1042	1042	0,46	0,497
Población*Iteraciones	1	18550	18550	8,24	0,004
Diversidad*Iteraciones	1	49	49	0,02	0,883
Interacciones de 3 términos	1	47	47	0,02	0,885
Población*Diversidad*Iteraciones	1	47	47	0,02	0,885
Error	232	522425	2252		
Total	239	887292			

ANOVA 120X8

Análisis de Varianza	**120x8**				
Fuente	GL	SC Ajust	MC Ajust	Valor F	Valor p
Modelo	7	123833	17690,5	8,59	0
Lineal	3	107691	35896,9	17,42	0
Población	1	31763	31763	15,42	0
Diversidad	1	3161	3161	1,53	0,217
Iteraciones	1	72767	72766,8	35,32	0
Interacciones de 2 términos	3	9559	3186,3	1,55	0,203
Población*Diversidad	1	513	513,3	0,25	0,618
Población*Iteraciones	1	8532	8532,3	4,14	0,043
Diversidad*Iteraciones	1	513	513,3	0,25	0,618
Interacciones de 3 términos	1	6584	6583,5	3,2	0,075
Población*Diversidad*Iteraciones	1	6584	6583,5	3,2	0,075
Error	232	478021	2060,4		
Total	239	601855			

ANEXO C. DIAGRAMA DE GANTT

Se desarrolló un ejercicio para representar gráficamente una solución al problema SDST/HFFS. A continuación los parámetros iniciales y las matrices de tiempos de alistamiento y de procesamiento utilizadas.

Parámetros iniciales:

- Trabajos: 5
- Etapas: 3
- Máquinas por etapa: 2-3-2

Matriz de tiempos de procesamiento

	ETAPA		
TRABAJOS	1	2	3
1	4	3	3
2	1	3	1
3	2	5	0
4	0	3	2
5	2	1	1

Matriz de tiempos de preparación si es el primer trabajo en la máquina

	ETAPA		
TRABAJOS	1	2	3
1	1	2	1
2	3	1	1
3	2	1	0
4	0	2	2
5	1	1	1

Matriz de tiempos de preparación dependientes de la secuencia

	ETAPA		
SECUENCIA	1	2	3
1	2	1	0
2	1	2	1
3	1	1	1
4	2	1	2
5	0	2	1
6	1	1	1
7	2	2	2
8	2	0	1
9	3	0	2
10	2	2	1
11	1	1	2
12	2	3	0
13	1	1	1
14	2	2	2
15	1	2	1
16	1	1	2
17	2	1	0
18	2	0	1
19	0	1	1
20	1	1	1

Matriz de secuencia

	TRABAJO ACTUAL				
TRABAJO ANTERIOR	1	2	3	4	5
1	0	1	2	3	4
2	5	0	6	7	8
3	9	10	0	11	12
4	13	14	15	0	16
5	17	18	19	20	0

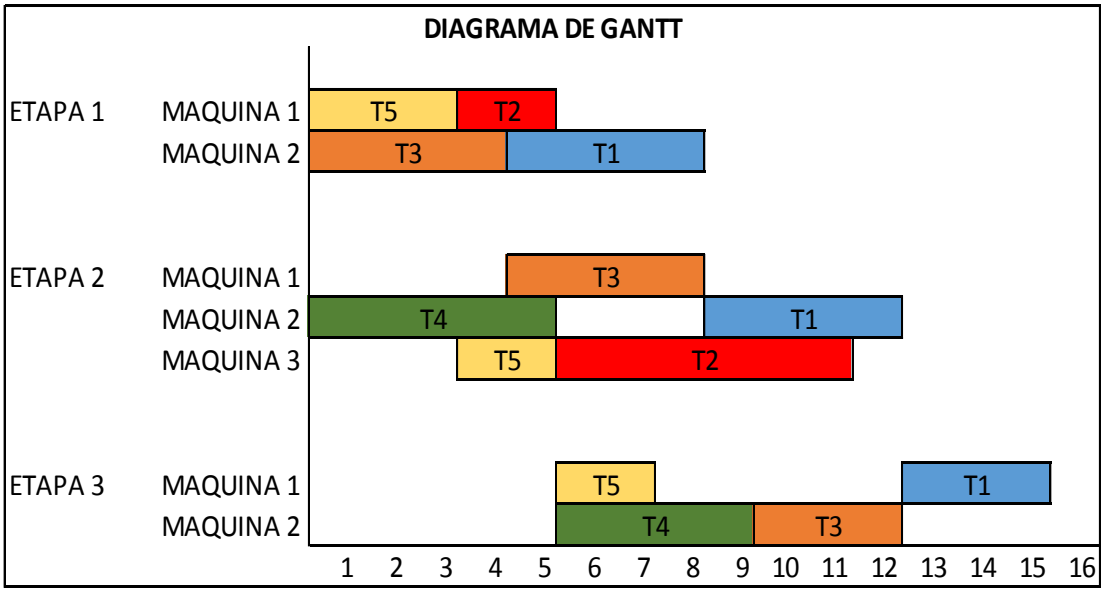
A continuación, se presenta la secuencia o cromosoma a representar gráficamente y la asignación de máquinas en cada etapa para cada trabajo

CROMOSOMA	4	5	3	2	1

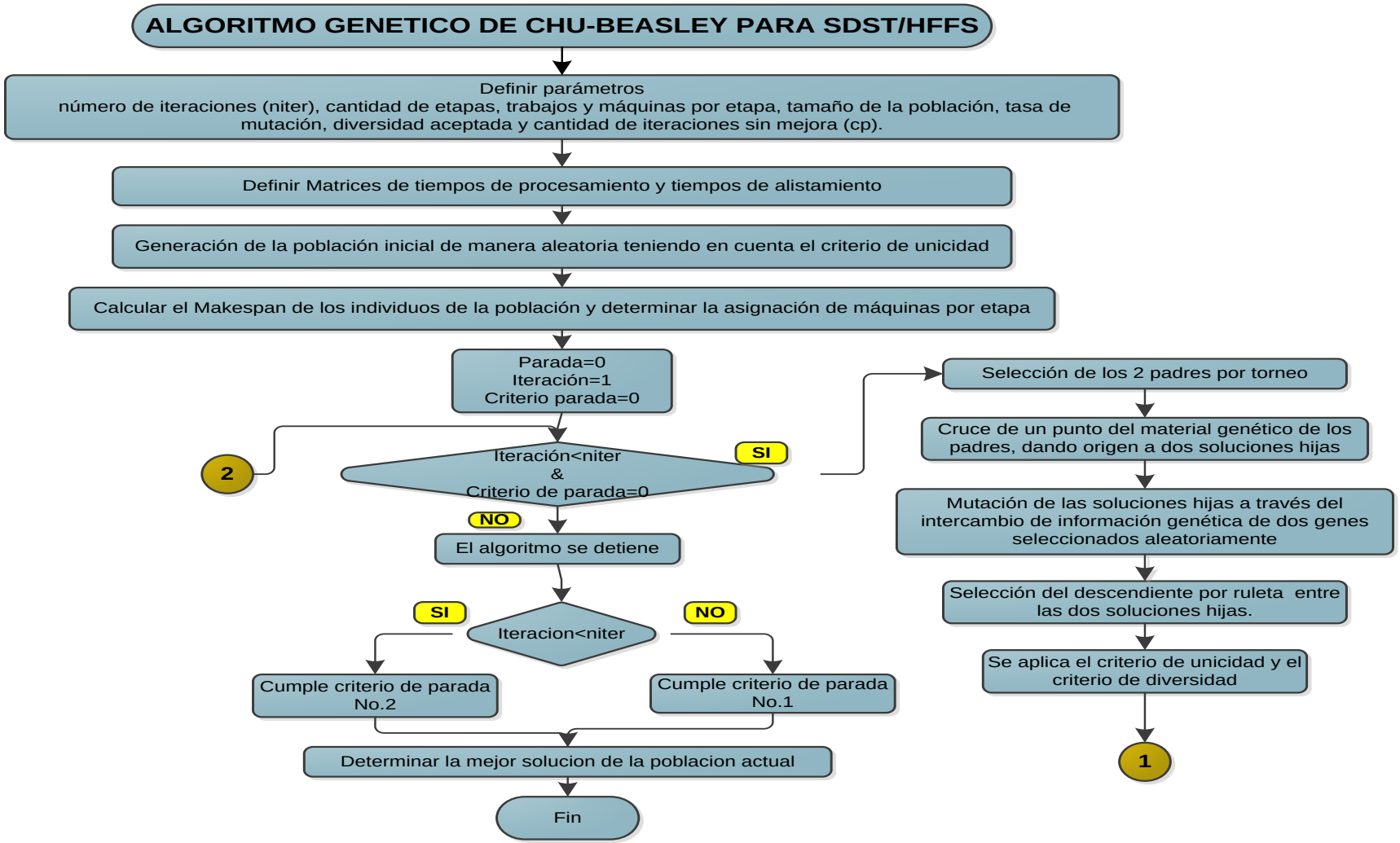
ASIGNACION DE MAQUINARIA					
	TRABAJOS				
ETAPAS	**4**	**5**	**3**	**2**	**1**
1	0	1	2	1	2
2	2	3	1	3	2
3	2	1	2	0	1

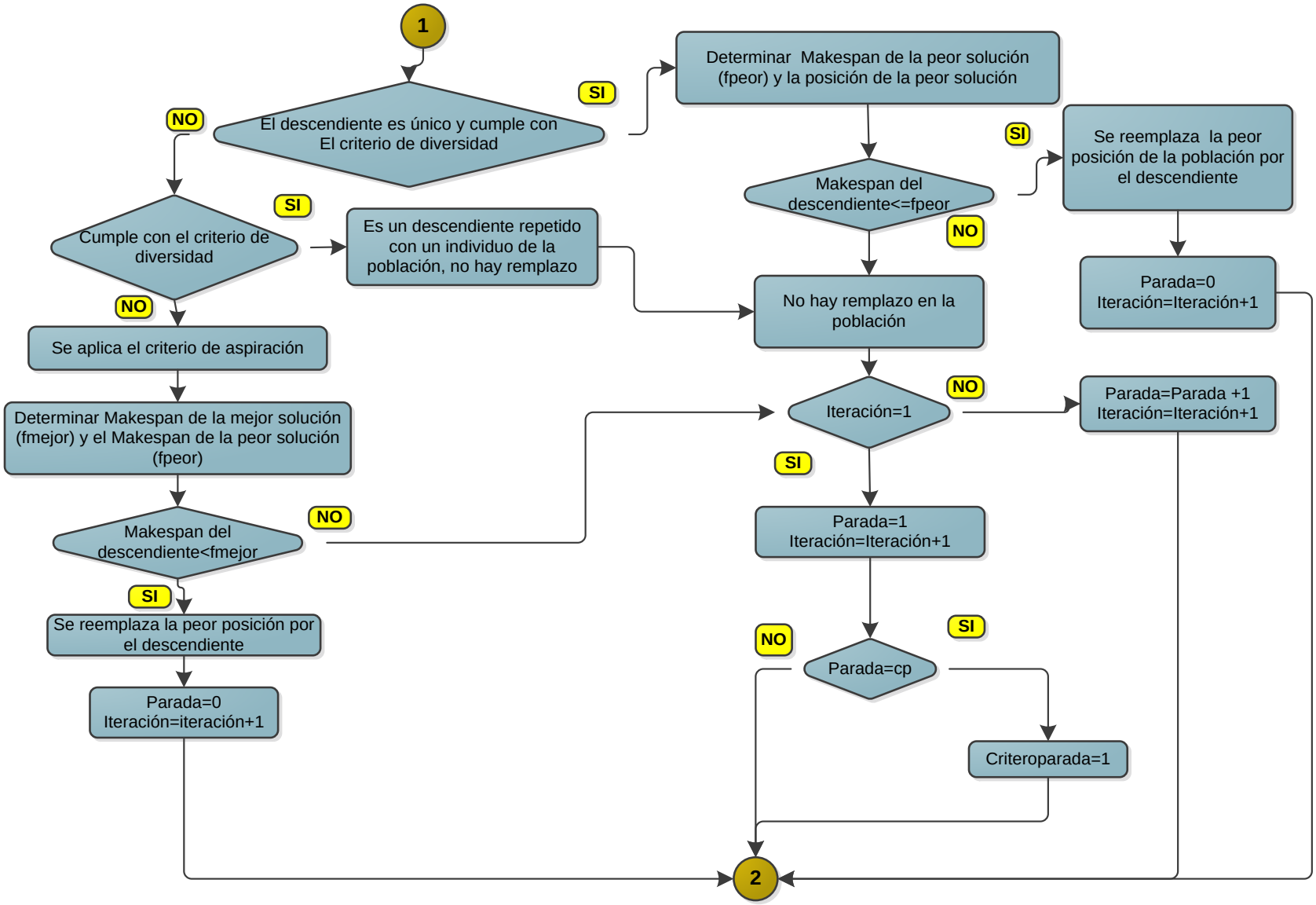
Se debe tener en cuenta que en la asignación de máquinas, cuando el trabajo no pasa por esa etapa se asigna la máquina 0.

A continuación se presenta el diagrama de Gantt que representa el Makespan del cromosoma ya mencionado; en el cual se encuentra la línea de tiempo que siguen todos los trabajos los cuales terminan de ser procesados en 15 unidades de tiempo.



ANEXO D. DIAGRAMA DE FLUJO DEL ALGORITMO GENETICO DE CHU-BEASLEY




1
El descendiente es único y cumple con El criterio de diversidad
SI
NO
Determinar Makespan de la peor solución (fpeor) y la posición de la peor solución
Makespan del descendiente<=fpeor
SI
NO
Se reemplaza la peor posición de la población por el descendiente
Parada=0
Iteración=Iteración+1
Cumple con el criterio de diversidad
SI
NO
Es un descendiente repetido con un individuo de la población, no hay remplazo
No hay remplazo en la población
Se aplica el criterio de aspiración
Determinar Makespan de la mejor solución (fmejor) y el Makespan de la peor solución (fpeor)
Iteración=1
NO
SI
Parada=Parada +1
Iteración=Iteración+1
Makespan del descendiente<fmejor
NO
SI
Parada=1
Iteración=Iteración+1
Se reemplaza la peor posición por el descendiente
Parada=0
Iteración=iteración+1
NO
Parada=cp
SI
Criteroparada=1
2

ANEXO E. ARTÍCULO

SOLUCIÓN DEL PROBLEMA FLOW SHOP HÍBRIDO FLEXIBLE EN MÁQUINAS IDÉNTICAS EMPLEANDO EL ALGORITMO GENÉTICO DE CHU-BEASLEY

Yesica Paola Murillo Coronel, Staci Marcela Méndez Calderón
Escuela de Estudios Industriales y Empresariales, Universidad Industrial de Santander

Bucaramanga, Colombia.
ysik92@hotmail.com
stacimendez18@hotmail.com

***Abstrac* -En el trabajo se aborda el problema Flow Shop Hibrido Flexible con el objetivo de minimizar los tiempos de producción se emplea una metaheurística denominada algoritmo genético con una pequeña modificación introducida por Chu-Beasley. Para validar el algoritmo se hacen necesarias unas instancias las cuales fueron encontradas en la literatura.**
Palabras Claves: Flow Shop Hibrido, algoritmo genético, flexible, Chu-Beasley, metaheurísticas, makespan**.**

I. INTRODUCCIÓN

La competitividad es, para una organización productiva, la capacidad inmediata y futura de diseñar, producir y vender bienes y servicios cuyos precios y otras cualidades formen un conjunto más atractivo que el de sus competidores[1], creando así una ventaja competitiva para la organización en términos de: precios, diferenciación del producto, innovación, servicio post-venta, etc. Para enfrentar los retos y suplir las necesidades del mercado, las empresas que consideran necesario crear o mejorar la ventaja competitiva con el fin de asegurar su permanencia en el mercado deben perfeccionar sus sistemas de producción y disponer de herramientas que les permitan obtener mejoras en términos de costo y calidad, que permitan dar respuesta pronta, expedita, flexible y eficiente a las necesidades que la industria impone con gran dinamismo y las cuales definen el rumbo de la industria.

En las empresas industriales o manufactureras uno de los recursos que se deben gestionar acertadamente son las máquinas que se emplean para la transformación de materia prima y que hacen parte del proceso productivo; en este sentido gana importancia la gestión de la producción que implica todo el proceso de planeación desde el largo plazo (planeación estratégica), pasando por la programación táctica o de mediano plazo, y

llegando a la programación operativa o de corto plazo, en lo que se conoce como el enfoque jerárquico de la producción [2], la finalidad de la programación de la producción es la asignación de los recursos que se encuentran disponibles para satisfacer un conjunto de requerimientos y determinar el orden adecuado que genere más beneficios dependiendo de los objetivos que se planteen previamente.

En este trabajo se abordará uno de los sistemas que comúnmente se encuentran en las empresas textiles y ensambladoras el Flow Shop Hibrido Flexible con tiempos de configuración dependientes de la secuencia (SDST/HFFS) en máquinas paralelas idénticas con el objetivo de minimizar el makespan o tiempo de terminación de todos los trabajos; la secuenciación de trabajos en un entorno como el propuesto es muy común en la industria, por lo cual se hace importante su análisis con el fin de desarrollar herramientas que permitan a las empresas ofrecer soluciones oportunas a sus problemas de secuenciación y toma de decisiones.

II. REVISION LITERARIA

El problema SDST/HFFS ha sido abordado por diversos autores, a continuación se presentan algunas de las investigaciones realizadas en temáticas relacionadas entre los años 2010 -2015.

En el 2010, Naderi, Ruiz y Zandieh [3] consideran un sistema SDST/HFFS en máquinas paralelas idénticas y lo solucionan mediante la metaheurística Búsqueda Local (ILS); en este mismo año Ruiz y Vásquez Rodríguez [4] ofrecen una definición del problema Flow Shop Hibrido (HFS) además realizan una extensa revisión en la literatura de este problema y los diversos métodos de solución.

En el 2012, Márquez Delgado et al [5] estudian el problema Flow Shop con la finalidad de minimizar el tiempo de finalización de todos los trabajos a través de la aplicación de un algoritmo genético; posteriormente Wang et al [6] en el 2013 abarcan en su investigación el problema Flow Shop Hibrido con máquinas paralelas idénticas con tiempos de procesamiento idénticos en cada máquina en cualquier etapa, implementan el algoritmo de estimación de distribución (EDA)

En el 2014, Fonseca-Reyna et al [7], realizan una investigación que busca resolver el problema de secuenciación de tareas Flow Shop Scheduling (FSSP) para minimizar el tiempo de finalización de todos los trabajos a través de un algoritmo genético, además analizan cuales son los parámetros de mayor influencia en la variable respuesta, encontrando que la combinación dos puntos de cruce-Ranking con elitismo es la que arroja los mejores resultados al problema estudiado.

En el 2014, Chen et al [8], analizan un sistema Flow Shop de dos etapas con máquinas de procesamiento por lotes (BPMs), el problema fue resuelto utilizando el algoritmo HDDE (Hybrid Discrete Differential Evolution) con un método de búsqueda local para aumentar la capacidad de explotación.

En el 2014, Faranhmand-Mehr et al [9] presentan el problema de programación Flow Shop Híbrido con un nuevo enfoque considerando los lapsos de tiempo y los tiempos de configuración dependientes de la secuencia para llevarlos a situaciones reales que se encuentran en las industrias como la de procesamiento de alimentos, química, textil, metalúrgica y fabricación de automóviles. Este problema es solucionado a través de un algoritmo genético.

Salazar Horning y Sarzuri Guarachi [10] en el 2014, analizan el problema Flow Shop Flexible (FFS) con tiempos de preparación dependientes de la secuencia a través de un algoritmo genético mejorado en el cual se realiza la generación de la población inicial utilizando vecindades de las heurísticas EDD (Earliest Due Date) y Slack (Holgura).

Nagano, Miyata y Castro [11] en el 2014, abordan el problema de la programación de trabajos Flow Shop sin esperas con secuencia de tiempos de preparación dependientes con el objetivo de minimizar el tiempo de flujo total. Utilizan las heurísticas constructivas BAH, BIH, TRIPS y QUARTS.

En el 2014, Ebrahimi, Fatemi y Karimi [12] estudian el problema de la programación de Flow Shop Hibrido, con máquinas paralelas idénticas con tiempos de configuración dependientes de la secuencia, se desarrollan dos algoritmos metaheurísticos basados en el algoritmo genético: Non-dominated Sorting Genetic Algorithm (NSGAII) y Multi Objective Genetic (MOGA).

López, Giraldo y Arango [13] en el 2014 estudian el problema Flow Shop Hibrido Flexible en máquinas paralelas no relacionadas en donde los tiempos de configuración son dependientes de la secuencia, tienen en cuenta un factor de eficiencia para cada máquina que está relacionado con problemas de calidad, errores humanos y condiciones de la máquina, este factor se calcula como la diferencia entre el porcentaje de utilización de la máquina y el factor de mantenimiento de esta. Aplican un algoritmo genético simple (AGS) a un problema real de la industria textil.

Lee et al [14] en el 2015 estudian el problema de programación de las actividades de mantenimiento en máquinas paralelas idénticas, utilizaron un Algoritmo Branch and Bound (B&B) para un tamaño pequeño del problema y un Algoritmo Genético Simple (GA) y un Algoritmo Genético con Búsqueda Local (GA+LS) para un tamaño grande del problema.

También en este mismo lapso de tiempo se encontraron algunas investigaciones relacionadas con el método de solución propuesto.

En el 2013 Jiménez, Muñoz y Toro [15] realizan una investigación para solucionar el problema de secuenciación de tareas en un ambiente Flow Shop Flexible (FFS) para el cual implementaron el algoritmo genético modificado de Chu-Beasley además utilizaron una heurística NEH para la generación de la población inicial.

Martínez, Sanz y Gahona [16] en el 2015 aplican el algoritmo genético de Chu-Beasley para solucionar el problema de ruteo vehicular aplicado al caso estudio de Súper Almacenes Olímpica en la ciudad de Bogotá

III. PLANTEAMIENTO DEL PROBLEMA

El problema Flow Shop hibrido Flexible (HFFS) es un caso particular de la configuración Flow Shop, el cual enuncia que un conjunto de *N* trabajos deben ser procesados en *M* etapas, donde cada etapa está compuesta por varias máquinas paralelas idénticas y cada trabajo puede omitir una o más etapas en su ruta de procesamiento [17]; otras características del problema de programación de tareas Flow Shop en general son [18]:

- ✓ Cada máquina está disponible continuamente y sin interrupciones.
- ✓ Cada máquina puede procesar un trabajo por vez.
- ✓ Cada trabajo sólo puede ser procesado por una máquina cada vez.
- ✓ Los tiempos de procesamiento de las tareas en las diferentes máquinas son determinados y fijos.
- ✓ Las tareas tienen la misma opción de ser programadas.
- ✓ Las operaciones en las máquinas, una vez iniciadas no deben ser interrumpidas.

El problema Flow Shop con más de dos etapas está clasificado como un problema NP-Hard, debido a que su tiempo de ejecución aumentará exponencialmente con el tamaño del problema [19].

Existen tiempos entre el procesamiento de los trabajos, es decir, existe un periodo de tiempo destinado a la preparación o alistamiento de las máquinas entre la continuación de dos trabajos[20], este tiempo de configuración en cada máquina puede depender del trabajo inmediatamente anterior y es denominado como tiempo de alistamiento dependiente de la secuencia [17]; la unión del problema Flow Shop Híbrido Flexible con una configuración con tiempos dependientes de la secuencia da origen al problema SDST/HFFS [21] que considera líneas de flujo en el que concurren varias etapas con múltiples máquinas donde existe la posibilidad de que no todos los trabajos deban ser procesados en todas las etapas,

considerando que existe un tiempo de configuración o *setup* de las máquinas entre dos trabajos diferentes y que este *setup* depende de la secuencia de trabajos a procesar.

Como finalidad se pretende encontrar la programación o secuencia de trabajo adecuada para obtener el mínimo makespan, teniendo en cuenta que el makespan o Cmax hace referencia al tiempo que transcurre entre el inicio del primer trabajo en la primera máquina y la finalización del último trabajo en la última máquina [22].

Mientras que en el Flow Shop básico al solo existir un recurso por etapa únicamente es necesario tomar la decisión del secuenciamiento de tareas, por el contrario en el Flow Shop Híbrido se deben domar dos decisiones: la asignación de los trabajos a las máquinas de cada etapa y el secuenciamiento de los trabajos en las diferentes máquinas [23].

IV. MODELO MATEMATICO

A continuación se presentan cada uno de los conjuntos, parámetros y variables consideradas en el modelo matemático teniendo como base el modelo presentado por Gómez Gasquet [23] en el 2010.

- **Conjuntos**

i,j: Subíndices para los trabajos $i=1, 2, 3....N+1$ $j=0, 1, 2...N$

k: Subíndice para las etapas $k=1,2,3....K$

m: Subíndice para las máquinas, depende de la etapa. $m=1, 2, 3...M$

- **Parámetros**

$S(i,m,k)$: Tiempo de alistamiento del trabajo i en la máquina m, si es el primero que se programa en la máquina m en la etapa k.

$P(i,k)$: Tiempo de procesamiento o de ejecución del trabajo i en la etapa k.

$ST(i,j,m,k)$: Tiempo de preparación de la máquina m en la etapa k para pasar de realizar el trabajo j al trabajo i.

$M(i,k)$: Matriz que determina si el trabajo i es procesado en la etapa k.

- **Variables**

Continuas

$TI(i,m,k)$: Momento de inicio del trabajo i en la maquina m en la etapa k.

$TF(i,m,k)$: Momento de finalización del trabajo i en la maquina m en la etapa k.

$TP(i,m,k)$: Suma de los tiempos de preparación y de procesamiento del trabajo i en la máquina m en la etapa k.

$TPr(i,m,k)$: Tiempo de preparación del trabajo i en la máquina m en la etapa k.

Binarias

$$Y(i,m,k): \left\{\begin{matrix}1 \text{ si el trabajo i es asignado a la máquina m e n la etapa k}\\ 0 \text{ En caso contrario}\end{matrix}\right\}$$

$$X(j,i,m,k): \left\{\begin{matrix}1 \text{ si el trabajo j es procesado antes}\\ \text{del trabajo i en la máquina m en la etapa k}\\ 0 \text{ En caso contrario}\end{matrix}\right\}$$

$$X(0,i,m,k): \left\{\begin{matrix}1 \text{ si el trabajo i es el primero en ser procesado}\\ \text{en la máquina m en la etapa k}\\ 0 \text{ En caso contrario}\end{matrix}\right\}$$

$$X(j,+1,m,k): \left\{\begin{matrix}1 \text{ si el trabajo j es el ultimo en ser procesado}\\ \text{en la maquina m en la etapa k}\\ 0 \text{ En caso contrario}\end{matrix}\right\}$$

- **Modelo Matemático**

$$Minimizar\ Z = max\ TF_{(i,m,k)} \qquad (1)$$

Sujeto a:

$$\sum_{m=1} Y_{(i,m,k)} \leq 1 \qquad \forall\, i,k; \qquad (2)$$

$$\sum_{j=0} X_{(j,i,m,k)} - Y_{(i,m,k)} = 0 \quad \forall\, i,m,k;\ j \neq i; \ (3)$$

$$\sum_{i=1} X_{(i,j,m,k)} - Y_{(i,m,k)} = 0 \quad \forall\, j,m,k;\ j \neq i; \ (4)$$

$$\sum_{i=1} X_{(0,i,m,k)} = 1 \qquad \forall\, m,k;\ j \neq i; \quad (5)$$

$$\sum_{j=1} X_{(j,N+1,m,k)} = 1 \qquad \forall\, m,k;\ j \neq i; \quad (6)$$

$$\sum_{m=1} Y_{(i,m,k)} = M_{(i,k)} \quad \forall\, i,k; \qquad (7)$$

$$TF_{(i,m,k)} \geq TI_{(i,m,k)} + TP_{(i,m,k)} \quad \forall\, i,m,k; \text{ (8)}$$

$$TPr_{(i,m,k)} \geq S_{(i,m,k)} \times X_{(0,i,m,k)} \;\; \forall\, i,m,k; \quad \text{(9)}$$

$$TPr_{(i,m,k)} \geq ST_{(i,m,k)} \times X_{(j,i,m,k)} \;\; \forall\, j,i,m,k;\; jz0\,; j \neq i \quad \text{(10)}$$

$$TP_{(i,m,k)} \geq (TPr_{(i,m,k)} + P_{(i,k)}) \times Y_{(i,m,k)} \;\; \forall\, i,m,k; \text{ (11)}$$

$$TI_{(i,m,k)} \geq \sum_m TF_{(i,m,k-1)} \quad \forall\, i,m,k;\;\; k \neq 1 \quad \text{(12)}$$

$$TI_{(i,m,k)} \geq TF_{(j,m,k)} \times X_{(j,i,m,k)} \; \forall\, i,m,k;\;\; j \neq i \;\; \text{(13)}$$

V. METODOLOGIA DE SOLUCION

A. REPRESENTACION DEL CROMOSOMA

Se define un vector de permutaciones, en el cual la longitud del mismo corresponde a la cantidad de tareas o trabajos, garantizando que ninguna tarea se repita y que todas sean terminadas. La posición que cada trabajo ocupa en el cromosoma indica la secuencia en que pasaran a las máquinas en cada etapa.

B. GENERACION DE LA POBLACION INICIAL

La población inicial del algoritmo se genera a través de una matriz de tamaño $m \times n$ donde m es el tamaño de la población y n el número de trabajos, cada solución creada en la población inicial es una secuencia permutacional generada de manera aleatoria. El algoritmo no permite que se creen dos soluciones iguales, es por esto que cada solución de la población inicial es comparada con sus antecesoras para de esta manera determinar que no se repita y obtener mayor diversidad de soluciones; si la solución generada se encuentra repetida esta es sustituida por una nueva solución que también debe ser evaluada para determinar que sea única.

C. CALCULO DE LA FUNCION OBJETIVO O MAKESPAN

Como primer paso se realiza la asignación aleatoria de las máquinas disponibles en cada etapa teniendo en cuenta que no es posible asignar la misma máquina dos veces seguidas. Posteriormente se calcula el tiempo de preparación de cada trabajo en cada etapa si este es el primero en procesarse en cada máquina o si es dependiente de la secuencia.

Finalmente se calcula el tiempo de terminación del procesamiento en cada etapa atendiendo que este es la suma del tiempo de preparación y el tiempo de procesamiento; es importante resaltar que el makespan o tiempo de terminación se calcula para cada trabajo en cada etapa a partir del valor máximo entre el tiempo de finalización del mismo trabajo en la etapa anterior y el tiempo de terminación del trabajo anterior en la misma máquina.

D. SELECCIÓN

El operador de selección utilizado es torneo en donde se seleccionan de manera aleatoria dos cromosomas de la población, entre los cuales se compara el valor de makespan correspondiente a cada uno de ellos, y se selecciona aquel que posea un makespan inferior. Este proceso se realiza dos veces para dar origen a las dos padres.

E. PROCESO DE CRUCE

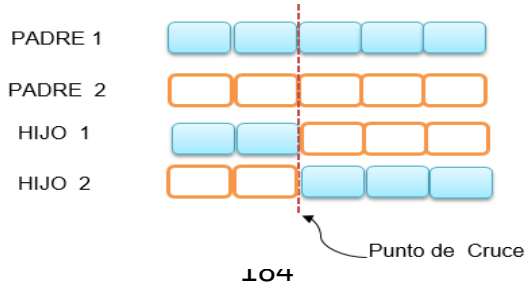
Este proceso inicia con la selección del punto de cruce de manera aleatoria; es decir, el punto de cruce es un número entero entre 1 y n-1, donde n es el número de trabajos. Seleccionado este punto los descendientes se crean de la siguiente manera:

Hijo 1: desde la posición 1 del padre 1 hasta la posición determinada por el punto de cruce en el padre 1 y desde la posición determinada por el punto de cruce en el padre 2 más uno hasta la posición final del padre 2.

Hijo 2: desde la posición 1 del padre 2 hasta la posición determinada por el punto de cruce en el padre 2 y desde la posición determinada por el punto de cruce en el padre 1 más uno hasta la posición final del padre 1.

Al realizarse el proceso de cruzamiento se debe tener en cuenta que la naturaleza del problema no permite que se repita algún número en la secuencia y es muy posible que al momento de generarse los hijos existan trabajos repetidos y por consecuencia trabajos olvidados; para solucionar esta anomalía de los cromosomas hijos se diseñó en el algoritmo un proceso de identificación de trabajos ya secuenciados y trabajos olvidados; los trabajos olvidados o faltantes ingresan al cromosoma a reemplazar a los trabajos que se repiten, esta reubicación de trabajos se realiza de manera aleatoria.

Figura 1. Proceso de Cruce

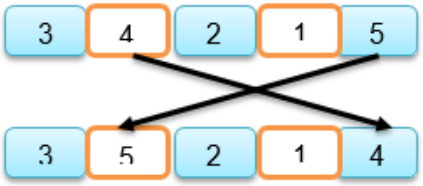


F. MUTACION

Obtenidos los hijos se genera un numero aleatorio entre 0 y 1, si este número es menor a la tasa de mutación establecida previamente en los parámetros ajustables del algoritmo se ejecuta el proceso de mutación de lo contrario los hijos pasan este proceso sin modificaciones.

Se aplica la mutación por intercambio, es decir, se seleccionan aleatoriamente dos posiciones del cromosoma y se intercambia la información genética en esos dos puntos.

Figura 2. Proceso de Mutación



G. SELECCIÓN DEL DESCENDIENTE

Con las dos soluciones hijas ya establecidas se realiza un proceso de selección de una sola de ellas para ingresar a la población; el proceso de selección del descendiente se realiza por ruleta donde el hijo con mejor makespan tiene mayor probabilidad de ser escogido mediante la generación de un número aleatorio entre 0 y 1.

H. PROCESO DE ACEPTACION

Para determinar si el descendiente es apto para ingresar a la población se aplican los siguientes criterios:

- **CRITERIO DE UNICIDAD**

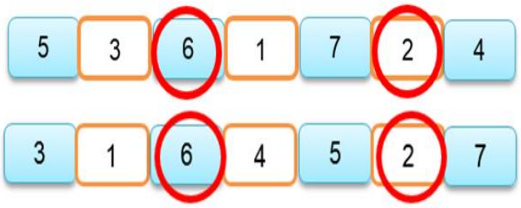
Se verifica que no exista una solución dentro de la población idéntica al descendiente, de ser así el descendiente se descarta y no puede ingresar a la población.

- **CRITERIO DE DIVERSIDAD**

Se verifica que la distancia mínima de separación del descendiente con cada una de las soluciones de la población cumpla con la diversidad aceptada establecida en los parámetros iniciales.

A continuación la representación de este criterio entre dos cromosomas con diversidad de 2 genes.

Figura 3. Diversidad de 2 genes entre cromosomas



Entre los dos cromosomas representados existen 2 genes donde el trabajo secuenciado es el mismo, es decir tiene una diversidad de 2 genes. Entre mayor sea el numero denominado diversidad más parecidas serán las soluciones propuestas. Si este número es menor se asegura una mejor exploración del espacio de soluciones.

Si el descendiente cumple con los dos criterios ya mencionados se acepta su ingreso a la población y el descendiente reemplaza al cromosoma con mayor makespan en la población.

- **CRITERIO DE ASPIRACION**

Se aplica este criterio cuando el descendiente no cumple con el criterio de diversidad pero su makespan es mejor que el makespan de la incumbente, es decir, de la solución con mejor makespan; por lo tanto se da la posibilidad para que este haga parte de la población lo cual conlleva a la eliminación de la solución con peor makespan.

I. *CRITERIOS DE PARADA*

Se aplican dos criterios de parada para finalizar la ejecución del algoritmo:

1. Si la cantidad de iteraciones alcanza el número máximo de iteraciones definido en los parámetros iniciales.

2. Si después de 100 iteraciones consecutivas no existe mejora.

VI. RESULTADOS

El algoritmo genético modificado de Chu-Beasley fue programado en Matlab y se probó en un computador Intel ® Core ™ i5- 3632 QM CPU 3.2 GHz, Memoria RAM de GB, con un sistema operacional Windows 7 de 64 bits.

En la validación se utilizaron las instancias disponibles en http://soa.iti.es/instancias-problemas, utilizadas por Naderi et al en el 2010 [24] en donde los autores presentan tres variaciones del algoritmo Búsqueda Local (ILS). En estas instancias propuestas existen 12 familias de instancias que están agrupadas por cantidad de trabajos y número de etapas, en cada instancia propuesta existen 80 problemas para su análisis, de los cuales para esta investigación fue escogido aleatoriamente un problema de cada una de las familias. La cantidad de máquinas disponibles en cada familia para cada estación varía de acuerdo a cada problema.

Para la ejecución se determinó variar el número de iteraciones, el tamaño de la población y la diversidad aceptada de la siguiente manera:

Parámetros	Valores
Número de iteraciones	1000, 1500
Tamaño de la población	50,100
Diversidad aceptada	2,5

Tabla 1. Parámetros iniciales

Los resultados de la ejecución del algoritmo fueron comparados con los resultados publicados por Naderi et al [3] en donde se presentan tres variaciones al algoritmo ILS y por Branz Moreira en el 2013[25], en donde se aplica un algoritmo GRASP-ILS. En la tabla se presenta el resumen de esta comparación con sus respectivos porcentajes de diferencia.

Se deben tener en cuenta las siguientes abreviaciones para la Tabla 2.

*INST: instancia ejecutada
*P=tamaño de población
*D: Diversidad aceptada
*I: Número de iteraciones
*MS: Mejor solución AGCB
*GRASP-ILS: Mejor solución GRASP-ILS
*ILS: Mejor solución ILS

INST	AGCB				GRASP-ILS	ILS	%GRASP-ILS	%ILS
	P	D	I	MS				
20X2	50	5	1000	648	697	634	-7.0	2.2
20X4	50	2	1500	1365	1232	1322	10.8	3.3
20X8	50	2	1000	1018	685	933	48.6	9.1
50X2	50	2	1500	1385	1313	1437	5.5	-3.6
50X4	50	5	1500	2502	2366	2447	5.7	2.2
50X8	50	5	1500	1351	1065	1053	26.9	28.3
80X2	50	5	1500	2210	2128	2086	3.9	5.9
80X4	50	5	1000	1819	2430	2276	-25.1	-20.1
80X8	100	5	1500	2939	2561	2689	14.8	9.3
120X2	50	5	1500	2545	3086	3356	-17.5	-24.2
120X4	50	5	1500	6078	5475	5771	11.0	5.3
120X8	50	5	1500	4231	3531	3442	19.8	22.9

Tabla 2. Diferencia porcentual de soluciones

Los porcentajes de diferencia respecto a las metaheurísticas comparadas reflejan que el AGCB ofrece buenas soluciones en las instancias más pequeñas y por el contrario en las instancias más grandes en las cuales existen 8 etapas sus soluciones tienen un elevado porcentaje de diferencia y por lo tanto no son recomendables.

TIEMPOS COMPUTACIONALES

Figura 4. Comparación de los tiempos de ejecución

Se realiza una comparación entre los tiempos de ejecución del AGCB y del algoritmo GRASP-ILS propuesto por Branz Moreira [25]. En la figura 4 se ilustra esa comparación.

En las instancias más pequeñas el algoritmo GRASP-ILS es realmente muy rápido en contraparte el AGCB para estas instancias es más lento y en las instancia grandes el algoritmo GRASP-ILS aumenta considerablemente su tiempo de ejecución mientras tanto

el aumento en tiempo computacional del AGCB aumenta de una manera más moderada, obteniendo finalmente en la instancia 120x8 una ejecución más rápida en un 82% que el GRASP-ILS.

VII. CONCLUSIONES

La revisión de la literatura, dio las bases necesarias para definir el problema de la programación de producción en ambientes tipo Flow Shop Híbrido Flexible, además se logró establecer una visión de la situación teórica de las diferentes metaheurísticas para solucionar este tipo de problemas. Se desarrolló una metodología basada en la técnica Metaheurística Algoritmo Genético de Chu-Beasley como método de solución al problema.

Los resultados obtenidos con el algoritmo genético de Chu-Beasley fueron satisfactorios en las instancias con las etapas 2 y 4, ya que en el 75% de estas, la diferencia entre los valores de las instancias y los mejores valores reportados no sobrepasa el 5%, y con un tiempo computacional de máximo 20 segundos para instancia más grande de estas etapas. Además, en las instancias restantes, es decir, en el 25% el porcentaje de diferencia es inferior al 6%.

Para las instancias de etapa igual a 8, no se logra alcanzar el mínimo Makespan reportado y la diferencia entre los mejores valores conocidos y los valores de las instancias es bastante considerable. En cuanto al tiempo computacional encontrado en la literatura y el de las instancias se tiene una disminución de la diferencia de tiempos computacionales a medida que aumenta el número de trabajos para esta cantidad de etapas.

En el análisis del diseño experimental de las instancias se concluye que el efecto del factor número de iteraciones es significativo, ya que está presente en 11 de las 12 instancias analizadas, como el factor más influyente y al modificar este es posible llegar a mejores resultados; sin embargo, al realizar nuevamente corridas aumentando este valor, se incrementaba el tiempo computacional sin tener una mejora importante en el makespan, por ello los resultados finales se manejan con el nivel alto de 1500 tomado para los diseños factoriales.

El algoritmo ofrece mejores soluciones cuando el factor tamaño de población se establece en 50 individuos en lugar de una población de 100 individuos, esta situación nos indica como postulan López, Vargas y Arango[26] “La variedad y diversidad que presenta una población mayor implica mayores esfuerzos para mejorar la calidad de la población, en cambio una población de tamaño pequeño tiende a facilitar mejoras en la calidad de sus individuos.”, debido a que su investigación arrojo resultados similares a la presentada en este documento.

REFERENCIAS

[1] Ariel Sarache, W; Ramos, R y Cespón, R. (2012). Aplicación de Indicadores para el Diagnóstico de Sistemas de Producción. Revista Universidad EAFIT,38(126), 56-66.

[2] Domínguez Machuca, J, *et al.* (1995). Dirección de Operaciones: Aspectos estratégicos. Madrid, España: Mc Graw-Hill.

[3] Naderi, B; Ruiz, R y Zandieh M. (2010). Algorithms for a realistic variant of flow shop scheduling. Computers Ops. Res. 36, 236-246.

[4] Ruiz, R y Vásquez Rodríguez, J. (2010). The Hybrid Flow Shop Scheduling Problem. European Journal of Operational Research. 205(1), 1-18.

[5] Marquez Delgado, J. *et al.* (2012). Algoritmo Genético aplicado al Problema de programación en procesos tecnológicos de maquinado con ambiente Flow Shop. Revista Ciencias Técnicas Agropecuarias 21(2), 70-75.

[6] Wang, S *et al.* (2013). An enhanced estimation of distribution algorithm for solving hybrid flow-shop scheduling problem with identical parallel machines. The International Journal of Advanced Manufacturing Technology. 68 (9-12), 2043-2056.

[7] Fonseca-Reyna, Y. *et al.* (2014). Influencia de los parámetros principales de un Algoritmo Genético para el Flow Shop Scheduling. Rev. cuba cienc. Informat. 8(1), 53-59.

[8] Chen, H *et al.* (2014) A Hybrid Differential Evolution Algorithm for a Two-Stage Flow Shop on Batch Processing Machines with Arbitrary Release Times and Blocking. International Journal of Production Research 52(19), 5714-5734.

[9] Farahmand-Mehr, M *et al.* An efficient genetic algorithm for a hybrid Flow shop scheduling problem with time lags and sequence-dependent setup time. The International Journal of Advanced.

[10] Salazar Hornig, E y Sarzuri Guarachi, R. (2015). An Improved Genetic Algorithm for Total Tardiness Minimization in a Flexible Flow Shop with Sequence-Dependent Setup Times. Rev. Chil. Ing. 23(1), 118-127.

[11] Nagano, M; Miyata, H y Castro, D. (2015). A Constructive Heuristic for Total Flow time Minimization in a no –wait Flow shop with Sequence Dependent Setup Times. Journal of Manufacturing Systems. 36, 224-230.

[12] Ebrahimi, M.; Fatemi Ghomi. S.M.T y Karimi, B. (2014). Hybrid Flow Shop Scheduling with Sequence Dependt Family Setup Time and Uncertain Due Dates. Applied Mathematical Modelling Research. 38(9-10), 2490-2504.

[13] López, J; Giraldo, J. y Arango, J. (2015). Reducción del Tiempo de Terminación en la Programación de la Producción de una Línea de Flujo Hibrida Flexible (HFS). Información Tecnológica. 26 (3), 157-172.

[14] Lee, W; Wang, J, y Lee, L. (2015). A Hybrid Genetic Algorithm for an Identical Parallel-Machine Problem with Maintenance Activity. The Journal of the Operational Research Society. 66 (11), 1906-1918.

[15] Jiménez, A; Muñoz, C y Toro, E. (2013). Solución del problema de Flow Shop Flexible aplicando el Algoritmo Genético de Chu-Beasley. Ciencia e Ingenieria.7 (13). 34-40.

[16] Solarte Martínez, G; Castillo Sanz, A y Rodríguez Gahona, G. (2015). Optimization of a vehicular routing using simple genetic Chu-Beasley algorithm. Revista Tecnura. 19(44), 93-108.

[17] Jabbarizadeh, F.; Zandieh, M. y Talebi, D. (2009). Hybrid Flexible Flow shops with sequence-dependent setup times and machine availability constraints. Computers & Industrial Engineering. 57, 949-957.

[18] Toro Ocampo, E; Restrepo Grisales, Y. y Granada Echeverri, M. (2006). Algoritmo genético modificado aplicado al problema de secuenciamiento de tareas en sistemas de producción lineal - Flow Shop. Scientia Et Technica. XII (30), 285-290.

[19] Álvarez Martínez, D; Toro Ocampo, E y Gallego Rendón, R. (2009). Estudio computacional con técnicas heurísticas basadas en recocidos para resolver el problema de secuenciación de tareas. Ingeniería & Desarrollo. Universidad del Norte.

[20]Ángel-Bello, F, *et al.* (2011).A heuristic approach for a scheduling problem with periodic maintenance and sequence-dependent setup times. Computers & Mathematics with Applications. 61(4), 797-808.

[21] Dupuy, G, *et al.* (2015). Métodos Metaheurísticos aplicados a procesos de gestión en una empresa. Laboratorio de Investigación en Sistemas Inteligentes. XVII Workshop de Investigadores en Ciencias de la Computación, Salta, 2015.

[22] Hojjati, S. y Sahraeyan, A. (2009). Minimizing makespan subject to budget limitation in hybrid flow shop. International Conference on Computers & Industrial Engineerin,g. 18-22.

[23] Gómez Gasquet, P. (2010). Programación de la producción en un taller de flujo hibrido sujeto a incertidumbre: arquitectura y algoritmos. Aplicación a la industria cerámica. Tesis (Doctorado en Gestión de la Cadena de Suministro e Integración Empresarial) Universidad Politécnica de Valencia., Valencia, España.

[24] Naderi, B; Ruiz, R. y Zandieh M. (2010), Algorithms for a realistic variant of flow shop scheduling. Computers Ops. Res. 2010.36, 236-246.

[25] Branz Moreira, N. (2013). Heurísticas para Minimização do Makespan na Classe de Problemas de Sequenciamento Flowshop Híbrido e Flexível com Tempo de Setup Dependente da Sequência: Com e Sem Eventos de Quebra de Máquinas. Belo Horizonte. Trabajo de Grado (Maestría en Modelamiento Matemático y Computacional). Centro Federal de Educação Tecnológica de Minas Gerais.

[26] López, J; Giraldo, J. y Arango, J. (2015). Algoritmo genético para reducir el makespan en un Flow Shop Hibrido Flexible con máquinas paralelas no relacionadas y tiempos de alistamiento dependientes de la secuencia. 11(1), 250-262.

ANEXO F. PROGRAMA EJECUTABLE

Carpeta Adjunta: Código Matlab