ADAPTACIÓN DE UN ALGORITMO DE SISTEMA BIOLÓGICO INMUNE PARA EL MEJORAMIENTO DE SISTEMAS DE PRODUCCIÓN TIPO FLOW SHOP CON MÁQUINAS EN PARALELO

Luis Alejandro Orellano Duran

Id. 000336662

Universidad Pontificia Bolivariana – Seccional Bucaramanga

Escuela de Ingeniería

Bucaramanga

2020

ADAPTACIÓN DE UN ALGORITMO DE SISTEMA BIOLÓGICO INMUNE PARA EL MEJORAMIENTO DE SISTEMAS DE PRODUCCIÓN TIPO FLOW SHOP CON MÁQUINAS EN PARALELO

Luis Alejandro Orellano Duran

Id. 000336662

Proyecto de grado presentado como requisito para optar al título de:

INGENIERO INDUSTRIAL

Director del Proyecto

M.Sc. Orlando Federico González Casallas

Universidad Pontificia Bolivariana – Seccional Bucaramanga

Escuela de Ingeniería

Bucaramanga

2020

Tabla de Contenido

Lista de tablas

Tabla de figuras

.

RESUMEN GENERAL DE TRABAJO DE GRADO

TITULO: ADAPTACIÓN DE UN ALGORITMO DE SISTEMA BIOLÓGICO INMUNE PARA EL MEJORAMIENTO DE SISTEMAS DE PRODUCCIÓN TIPO FLOW SHOP CON MÁQUINAS EN PARALELO

AUTOR(ES): Luis Alejandro Orellano Duran

PROGRAMA: Facultad de Ingeniería Industrial

DIRECTOR(A): Orlando Federico González Casallas

RESUMEN

La presente investigación aborda el problema de secuenciación y asignación de máquinas en paralelo con restricción de precedencia y flujo continuo entre trabajos. El problema es considerado de tipo NP-Hard por su complejidad computacional y de gran importancia en la industria de optimización de recursos. La técnica empleada para solucionarlo es una metaheurística llamada algoritmo de Sistema Inmune, que surge como adaptación de los algoritmos genéticos a partir de la evolución biológica. Este procedimiento presenta operadores de selección por criterio, cruce en un punto y mutación en dos posiciones, basado en la estrategia propuesta de autores previos en esta rama. Con base en esto, se diseñó un programa que sirviera de software gestor del problema y a su vez, representara la adecuada asignación de trabajos. Posteriormente, se realizó una validación a través de la comparación de resultados obtenidos con lo encontrado por diversos autores para determinadas instancias. En consecuencia, se confirmó la eficiencia y eficacia del programa propuesto. Lo anterior se logró gracias al análisis de resultados por medio de un diseño de experimentos 2^3 con 8 réplicas, en el que se concluyó principalmente sobre los efectos de los factores escogidos.

PALABRAS CLAVE:

Flow Shop Scheduling; Algoritmo de Sistema Inmune; Metaheurística

V° B° DIRECTOR DE TRABAJO DE GRADO

GENERAL SUMMARY OF WORK OF GRADE

TITLE: ADAPTATION OF AN IMMUNE SYSTEM BIOLOGICAL ALGORITHM FOR THE IMPROVEMENT OF FLOW SHOP PRODUCTION SYSTEMS WITH PARALLEL MACHINES

AUTHOR(S): Luis Alejandro Orellano Duran

FACULTY: Facultad de Ingeniería Industrial

DIRECTOR: Orlando Federico González Casallas

ABSTRACT

In this research, the sequencing flowshop with assignment of parallel machines including precedence constriction and continuous flow is presented. This kind of problem is considered as NP-Hard computational complexity solution with high importance in industrial optimization resource optimization. Immune system algorithm technique, an adaptation of the genetic algorithm from biological evolution, was employed to find the solution problem. This metaheuristic structure presents selection operators by criteria, crossing on one point and two position mutation, considering proposed strategy of previous authors in this field. According with the problem and metaheuristic structure, a program was designed to serve as a problem managing software, and so on, will represent the spot-on job assignment. Furthermore, a comparison between optimal solution, reported in research papers, and the results obtained by the proposed technique define the validation process in terms of efficiency and efficacy of the metaheuristic. This validation process including factorial experimental design considering 8 iterations, to fitting the parameters defined in the metaheuristic structure.

KEYWORDS: Flow Shop Scheduling; Immune System Algorithm; Metaheuristics

V° B° DIRECTOR OF GRADUATE WORK

.

1. Introducción

Actualmente, la gestión y control adecuado de los recursos juega un papel de alta relevancia en la planeación de la producción dentro de las empresas que se enfocan en la mejora continua para un mercado de alta competitividad. Esto requiere ser eficiente al máximo en cada una de las operaciones y desde distintos enfoques.

Por otra parte, el área de producción para organizaciones que cuentan con líneas de ensamble requieren constantemente de una adecuada programación de las máquinas en función de las distintas actividades, lo que ha llevado a que esta toma de decisiones tenga gran importancia debido a los beneficios que puede objetar como disminución de costos, mayor aprovechamiento de los recursos y mejor servicio al cliente.

Debido a esto, el investigador Wilson (1989) estudió el problema que enmarca este tipo de contexto llamado *Flow shop scheduling*. También, propuso un modelo matemático el cual presenta soluciones para esta situación, de acuerdo a la precedencia y flujo continuo de trabajos entre las máquinas. Siendo de gran utilidad para el desarrollo de la presente propuesta.

En el transcurso del proyecto se pretende analizar el problema de la programación, asignación y secuenciación de máquinas para la ejecución de múltiples trabajos. Para ello, es necesario emplear una técnica metaheurística como la del algoritmo genético de Sistema Inmune que sirve de motor de búsqueda de las condiciones óptimas. Seguido de esto, diseñar un programa que funcione como software gestor de dicho problema y que soporte la toma de decisiones para este nivel operativo. Además de validar lo propuesto con un

diseño de experimentos de naturaleza factorial, que ajusta los factores inmersos en el algoritmo y a la información encontrada por diferentes investigadores ampliamente conocidos en el tema.

Finalmente, se describen las conclusiones y recomendaciones obtenidas en el transcurso de la investigación como parte inicial de futuros avances en el tema.

2. Delimitación del problema

La propuesta de este proyecto de grado está orientada al proceso de análisis y construcción de un modelo de optimización para el problema de secuenciación de trabajos en un sistema de producción tipo flow-shop con máquinas en paralelo.

Este proyecto es de carácter teórico investigativo, ya que se desea abordar un problema propuesto en la literatura de sistemas de producción y, mediante técnicas analíticas exactas y heurísticas, poder plantear escenarios de solución. Cabe destacar que, según Wilson (1989), a diferencia de otros procedimientos, el modelado matemático que él propone cuenta con menos variables de decisión y más restricciones, ajustándose a un modelado de precedencia más natural que logra converger más rápido que las técnicas únicamente heurísticas. Por otro lado, acoge de manera genérica los distintos parámetros relacionados con este problema y se considera amplio para el abordaje y desarrollo de la secuenciación requerida.

Una vez que se obtenga el modelo de optimización propuesto para el problema de secuenciación de operaciones, se propone una adaptación de un algoritmo metaheurístico conocido como “Algoritmo de Sistema Biológico Inmune”, en forma de programación que, daría una mejor solución cuando se tiene mayor complejidad computacional frente al incremento del número de instancias -trabajos y máquinas- en el problema.

Adicionalmente, se realizarán comparaciones del modelo propuesto con instancias de la literatura, así como la propuesta de un diseño de experimentos para el ajuste

paramétrico asociado a la solución de un problema de secuenciación con el algoritmo metaheurístico propuesto.

3. Antecedentes

La distribución para la producción manufacturera con líneas de ensamble según Chase, Jacobs, & Aquilano (2009) es un lugar donde las tareas similares se agrupan por equipos y el proceso es consecutivo de acuerdo al orden predispuesto para la fabricación de las unidades requeridas. Por consiguiente, su gestión ha sido abordada por distintos procedimientos dada la complejidad de su desarrollo. En la anterior dinámica, Khalili & Alireza (2014) han propuesto a partir de líneas con dos máquinas por etapa, un modelo que integra dos estructuras de programación lineal de enteros mixtos y solucionado con un algoritmo metaheurístico de búsqueda de Caza, en el que se minimiza el tiempo de ciclo, equilibrando las velocidades y aumentando el rendimiento del taller. Por otra parte, Gupta & Tunc (1991) estudiaron talleres con dos etapas y donde la segunda cuenta con máquinas idénticas. El procedimiento que plantearon radica en la comparación constante de los tiempos de ciclo con sus respectivos límites inferiores globales, y desarrollándolo con modelos heurísticos polinómicos los cuales afirman ser bastante efectivos. Mientras que Rabbie & Jolai (2015) señalan que, los sistemas productivos de esta índole deben ser simulados con uniformidad en la configuración de las ramas paralelas, ya que así se eliminan o reducen las etapas de cuello de botella y mejoran la producción. El modelo que idearon contiene el tiempo de configuración de flujo flexible entre cada corrida y está orientado a minimizar el tiempo por cada lote o unidad en fabricación. Para la función objetivo, unieron una técnica heurística fusionada con métodos metaheurísticos como lo son el de Búsqueda invasiva de maleza, Búsqueda de vecindad variable y Recocido Simulado.

Por el contrario, Attar & Mohammadi (2013) sugiere estudiar varias limitaciones conjuntas como máquinas paralelas, pero no relacionadas, tiempos de espera limitados entre cada dos operaciones sucesivas y tiempo por corrida a cumplir. Estos parámetros los solucionó a través del algoritmo metaheurístico de Optimización basada en Biogeografía a partir de dos experimentos simulados. De igual forma, Figielska (2009) consideró una máquina en la segunda etapa y máquinas en paralelo sin relación en la primera etapa. La primera fase cuenta con recursos renovables que están disponibles en cantidades limitadas. El objetivo que se fijó fue el de maximizar la productividad con algoritmos heurísticos que primero resuelven los problemas presentados en la primera sección y después proponen el cronograma del resto de la línea de ensamble. Finalmente, se encuentra la propuesta de Zhou, Li & Chen (2016) que a través de dos máquinas paralelas de procesamiento por lotes, proyectaron minimizar el tiempo por corrida sin esperas entre cada estación y con volúmenes de trabajo y tiempos de preparación desiguales. Lo anterior lo lograron con un método evolutivo de Estimación de Distribución basado en secuencias de trabajo, un modelo probabilístico para generar nuevas iteraciones y una técnica heurística para aglomerar los trabajos por lotes.

4. Justificación

El control del tiempo de operación logística de la producción es de las decisiones operativas de mayor complejidad, ya que delimita la dinámica de la capacidad de cada compañía manufacturera. Es así como su gestión requiere de la investigación y estudio riguroso de cada componente inmerso en la línea de ensamble, priorizando el rendimiento de cada corrida a realizar. Dado esto, la industria necesita desarrollar herramientas cuya eficiencia sea alta, para así soportar con éxito la administración de sus operaciones. Luego es importante que los protocolos utilizados en el sector, estén a la vanguardia con ayuda de la ofimática, ya que estas aplicaciones agilizan la consecución de mejores resultados de acuerdo a las condiciones o recursos con los que cuenta cada empresa. Por consiguiente, se solucionaría otro factor relevante como lo es el tiempo inoperante de las máquinas. Un problema común, que tiene serias implicaciones en la ociosidad de la planta de personal, y está relacionado con la incorrecta programación de los trabajos. Cabe mencionar que, y contrario a lo anterior, la sobrecarga desmesurada de las máquinas también es un pilar importante para la determinación del orden productivo de las tareas, ya que de esta forma se agiliza la depreciación de estos activos y, en consecuencia, su obsolescencia dentro del cumplimiento de los trabajos. Por otra parte, para Süer (1998) es necesario mitigar las estaciones que no cumplan con las condiciones estándar del resto del sistema, porque desmejoran la eficiencia de la línea integrada, al igual que la prolongación del tiempo de ciclo, disminuyendo las unidades predispuestas para la jornada laboral y, por ende, la capacidad de satisfacer la demanda establecida. No obstante, para Globerson & Tamir (2007) la circunstancia que mejor engloba las adversidades previas, es la agravación del

tiempo de respuesta del sistema en general, ante fluctuaciones requeridas por el cliente interno de despacho. Del mismo modo que los retardos de lotes de producción que aquejan cuellos de botella y que espacialmente obstaculizan el flujo continuo característico de la línea de ensamble.

5. Objetivos

6.1. Objetivo General

Adaptar un procedimiento de solución al problema de secuenciación de trabajos en un sistema de producción tipo flow-shop con máquinas en paralelo, considerando el enfoque del algoritmo Sistema Biológico Inmune, con el propósito de minimizar el tiempo de terminación de todos los trabajos.

6.2. Objetivos Específicos

- Establecer un modelo de programación no lineal asociado al problema de secuenciación tipo flow-shop con máquinas en paralelo considerando la minimización del tiempo de terminación de todos los trabajos.
- Desarrollar un procedimiento de solución para el problema de secuenciación de trabajos en un sistema de producción flow-shop con máquinas en paralelo, mediante la adaptación del algoritmo Sistema Biológico Inmune en lenguaje de programación Visual Basic.
- Evaluar el tiempo computacional, y solución obtenida entre el modelo exacto y el algoritmo de Sistema Biológico Inmune, de tal forma que permita un ajuste paramétrico del modelo para las instancias de prueba propuestas para el problema de secuenciación de trabajos en un sistema de producción tipo flow-shop con máquinas en paralelo.

6. Marco Teórico

7.1. Optimización combinatoria

Para Martí (2003), es una rama de la matemática aplicada y la ciencia computacional, la cual se encarga del estudio del modelado y la solución algorítmica de problemas de optimización, en función de variables definidas sobre un conjunto discreto.

Estos tipos de problemas están enmarcados de la siguiente forma:

- Conjunto de variables
- Dominio de variables
- Restricción entre variables
- Función objetivo

En donde las soluciones se encuentran un área de posibles soluciones dentro del límite de las denominadas factibles. Bajo estos criterios, la función objetivo se caracteriza por minimizar o maximizar las condiciones de entrada según el problema de interés. Es por ello que cada respuesta hallada será un óptimo global del conjunto de factibilidad.

Los métodos de optimización generalmente mencionados son: no lineal, lineal, entero, lineal entero, estocástico y multiobjetivo.

Lo previamente mencionado se radica fundamentalmente en encontrar el valor que puede tomar la variable y que optimiza la función objetivo bajo una serie de restricciones establecidas. Donde la naturaleza de dichas variables depende del tipo de problema. Es así como surge la programación entera pura, la programación entera binaria y la programación entera mixta. No obstante, los problemas de optimización también pueden ser clasificados

según su complejidad computacional, es decir, el grado de dificultad que conlleva la solución mediante distintos algoritmos.

7.1.1. Tipos de complejidad computacional

La complejidad se radica en el estudio de los componentes necesarios durante el cálculo para resolver un problema en los que el tiempo y el espacio, están dados por el número de actividades que se efectúan. Según Cortés (2004), al igual que el tamaño de los datos de entrada al modelo. Usualmente la máquina de *Turing* es el punto de referencia para clasificar esta teoría. Dicha máquina se puede adaptar para simular cualquier algoritmo de computador y es útil para determinar los límites del cálculo que se realizará. Luego esta diversificación está dada por:

Clase NP: Según Garey & Graham (1972), es el conjunto de problemas que pueden ser resueltos por un algoritmo en una máquina *Turing* no determinística. Su principal característica es que puede ser verificada su solución por medio de un algoritmo relacionado a la temática de estudio. El tiempo de procesamiento de la solución es proporcional al tamaño del problema, lo que indica una dependencia funcional que no puede ser acotada por un polinomio.

Clase NP completos: Sea P' un problema de la clase NP. Entonces P' es NP- completo si cualquier problema en clase NP se puede reducir a P' en tiempo polinomial.

Clase P: Se define como la sub clase de problemas de la clase NP que pueden ser solucionados por la máquina de *Turing* determinística en tiempo polinomial.

Clase NP Hard: Correa & Rodriguez (2008) enuncia que es subconjunto de la clase NP para los que no se puede tener una solución en tiempo polinomial para todas sus instancias. Todos los problemas de esta clase pueden ser reducidos a NP.

Esta última clase es la que aplica para los problemas de tipo *Shop Scheduling* en las que el *Flow Shop* es de los principales temas abordados, ya que se incrementa exponencialmente el tamaño de las variables de acuerdo a los problemas que estudian.

7.2. Shop scheduling

Es un problema de optimización combinatoria que detalla una gran cantidad de sistemas productivos, caracterizado por múltiples problemas en la literatura según Baker (1974).

También se define como el sistema de *n* trabajos y *m* máquinas. Donde cada trabajo es una operación en la que se emplea un tiempo de procesamiento, y cada operación debe ser desarrollada en las máquinas disponibles. Todos los trabajos cuentan con la precedencia entre cada operación y el objetivo es hallar una programación de actividades factibles que optimicen lo requerido para la solución del problema.

No obstante, existen diferentes tipos de clasificación del problema de *Shop scheduling*: de acuerdo al instante en el que llegan los trabajos al proceso o según el flujo de los productos. En esta última se encuentran los siguientes esquemas:

- Flow shop scheduling problem
- Job shop scheduling problema
- Open shop scheduling problem

Finalmente, para los efectos de esta investigación, el esquema acorde al problema definido es el de Flow Shop y se explica a continuación.

7.2.1. Flow shop scheduling problem

Este tipo de problema se caracteriza por contar con trabajos que poseen la restricción de precedencia entre cada trabajo. Es decir, cada trabajo debe pasar en igual orden por todas las máquinas y en la misma secuencia de flujo. Lo que para este caso el objetivo es encontrar un orden de ejecución de cada trabajo para cada máquina. A diferencia del Open shop scheduling en el que el orden es irrelevante dentro de la asignación de trabajos según Dannenbring (1977).

Para lo anterior, existen algoritmos evolutivos los cuales suelen modelar problemas de la vida real. A estos se les sueles llamar también como inteligencia artificial o evolución biológica en los que se logran las soluciones satisfactorias en tiempos cortos. No obstante, no garantizan una solución óptima, pues depende de la complejidad computacional del problema.

Con el transcurrir del tiempo, el estudio ha ido pasando de métodos exactos hacia heurísticos y finalmente metaheurísticos. Estos son los tipos de métodos que se presentan a continuación.

Métodos exactos: Son aquellos que solucionan problemas que pertenecen a la Clase P. Usualmente se emplean algoritmos voraces, de ramificación y poda, *backtracking*, etc.

Se caracterizan por explorar el espacio completo de soluciones de un problema y por medio de funciones de acotamiento, se descartan las partes donde no se contiene la mejor solución.

Heurísticas: Surgen como respuesta a necesidades de tiempo, costo y menor flexibilidad que suelen tener los métodos exactos. Adicionalmente, se puede utilizar como parte de un

procedimiento global en el cual proporciona una buena solución inicial o de partida en un paso intermedio del mismo. También se define como una aproximación intuitiva de la mejor solución. En ellos se encuentran algoritmos de búsqueda local, constructivos, entre otros.

Metaheurísticos: Se consideran procedimientos iterativos que guían una heurística combinando de forma inteligente distintos para explorar y explotar en mejor forma, el espacio de búsqueda. Suelen ser utilizados para los problemas de mayor envergadura, ya que son más efectivos que los dos tipos previamente enunciados. En ellos se encuentra la búsqueda tabú, recocido simualdo, algoritmos genéticos, colonia de hormigas, entre otros.

Las líneas de ensamble suelen registrar rigurosos análisis para la toma de decisiones en la programación de sus actividades. Es así como su gestión se dinamiza de acuerdo a los problemas que pueda tener cada organización. Rossi, Pandolfi & Lanzetta (2014) proponen aplicar dos técnicas heurísticas enfocadas a un taller flexible en relación a las máquinas en cada etapa, con tiempos de preparación uniformes independientes y con lotes de producción compatibles para así, disminuir el tiempo de producción por cada corrida.

Las problemáticas anteriormente mencionadas que están ancladas al control de tiempos de logísticos de producción, han sido abordadas por distintos procedimientos que en consonancia buscan reducir los retrasos. Estos procedimientos ordenan los trabajos según cada entrega a realizar y procesan en paralelo múltiples máquinas idénticas que también mejoran la fecha de vencimiento de productos como baños químicos, pinturas y de fundición. Por otra parte, Cortés, García & Hernández (2012) desarrollaron una técnica

híbrida de heurísticas la cual tiene presente los cambios en las fechas de entrega solicitados por el cliente, las cantidades a despachar, los materiales requeridos al proveedor y la gestión de inventario después del cumplimiento de cada orden de pedido. El anterior modelo minimiza el tiempo de fabricación y como criterio auxiliar, optimiza los recursos. Además, Rashidi, Jahandar & Zandieh (2010) desarrolló de otra forma la programación de máquinas paralelas por fases. Ellos partieron de la no retroalimentación entre las máquinas y de tiempos de alistamiento dependientes de la secuencia de la línea, para lo cual utilizaron un algoritmo genético que segmenta las poblaciones en subpoblaciones de acuerdo a porcentajes con el fin de evaluar distintas direcciones. Sin embargo, dada la robustez de los datos con los que usualmente se enfrentan en estas situaciones, formularon una función de redireccionamiento en el caso en que la programación se centre en óptimos locales. En cambio, López & Arango (2015) presentaron un algoritmo genético simple en el que se busca minimizar el tiempo de producción en una línea de ensamble con máquinas no relacionadas y tiempos de alistamiento dependientes de la secuencia que se lleva a cabo.

Por otro lado, Wilson (1989) propuso un enfoque del problema en el que incluye menos variables que satisfacen más restricciones, con el objetivo de reducir los tiempos de fabricación y flujo entre cada trabajo para cada corrida. No obstante, fue validado y experimentado con el software de simulación SCICONIC/VM a través de programación entera del modelo construido, como procedimiento heurístico de la solución.

Finalmente, y seguido de la lectura previa y resumida en el presente apartado, se enuncia el tipo de solución que se empleará a lo largo del documento.

7.2.2. Optimización por Sistema Biológico Inmune

El algoritmo genético inmune para R.J, Y.H, Zulani E & F.C (2013) consta de tres operaciones, denominadas clon, cruce y mutación. El método de clonación es similar al método de la ruleta. Básicamente, un anticuerpo se selecciona al azar y su clon se registra de manera especial dentro del conjunto total, para así, evaluar sus características. Este algoritmo genético tiene como mejora la memoria del sistema inmune, la cual recibe, evoluciona y replica la información útil para la defensa del cuerpo humano en relación a agentes nocivos. Este también supera las desventajas del algoritmo genético y mejora la capacidad de búsqueda global y su eficiencia.

El desarrollo de la función objetivo corresponde al "antígeno" del mecanismo inmunológico, y la solución del modelo, corresponde al "anticuerpo" del algoritmo. El proceso del algoritmo se presenta de la siguiente manera: El primer paso es el reconocimiento del antígeno y la actualización de la célula de memoria; luego, el anticuerpo inicial se crea con el cálculo referente a la aptitud y diversidad del anticuerpo; Sobre la base de la aptitud y la diversidad, se realizan las operaciones de promoción e inhibición de anticuerpos, que son seguidas de la producción masiva en todo el sistema; si cumple con la condición de terminación en relación a la situación óptima, el algoritmo finaliza, de lo contrario, se repite el bucle.

7. Diseño metodológico

La investigación será llevada a cabo a través de las siguientes características:

Tabla 1
Conceptualización metodológica

Ítem	Descripción
Enfoque	El desarrollo de esta investigación se caracteriza por ser **Mixto**, ya que inicialmente se realizarán cálculos referentes a la optimización de los modelos y seguidamente, se analizarán y evaluarán, dichos resultados.
Tipo de investigación	La investigación realizada consta de distintas etapas dada su complejidad y sus diferentes enfoques al desarrollarla. Inicialmente se abarca de forma **Exploratoria** ya que se busca e identifica el estado de la literatura actual, relacionada con propuestas de modelos de líneas de ensamble con máquinas en paralelo, a partir del algoritmo de Sistema Inmune. También se revisan proposiciones semejantes a esta metaheurística y se denota la importancia de la investigación y aplicación de este. Seguidamente de una fase **Explicativa** en la cual se estudia y explica el algoritmo previamente mencionado, y la dificultad radicada en el desarrollo de su naturaleza no lineal con restricciones lineales. Esta etapa aglomera la relevancia de entender completa y correctamente el modelo para su posterior aplicación. Finalmente se concluiría en un estado **Experimental** de la investigación, ya que, a partir de la estructura y programación del algoritmo, se probaría para distintos escenarios la gestión de la línea de ensamble a favor de la mejor optimización de las condiciones iniciales del fenómeno que se evalúa.
Estructura	En relación al protocolo resolutivo para el problema planteado, se opta por emplear la metodología propuesta por Taha (2004) en relación a la implementación de la Investigación de Operaciones. A continuación se encuentra la forma en que será abordada la propuesta.
1. Definición del problema	En primer lugar, se definen las circunstancias del problema ya que son los puntos de vista que enmarcarán los caminos para solucionarlo. Estas condiciones principalmente apuntan hacia la secuenciación de máquinas en un proceso de manufactura, las demoras en entregas de pedidos y el aumento en el tiempo en la fabricación del producto. Dado esto, se determinaría el objetivo orientado a la optimización en el control de tiempos en la logística interna de las empresas, en aras de una mayor productividad, bajo las limitaciones que se pueden padecer en el fenómeno descrito.
2. Construcción del modelo	Por consiguiente, se traduce la anterior descripción a un procedimiento matemático que engrane las características señaladas. Para ello se escogió el algoritmo de Sistema Inmune que, según su dinámica, puede

	arrojar resultados afines con la mejora del control y gestión de los tiempos de la logística interna de una empresa manufacturera.
3. Solución del modelo	En tercera instancia, se desarrollará el algoritmo de forma iterativa, ya que estos serán debidamente programados en lenguaje Visual Basic, para así lograr una estructura genérica útil, que soporte el manejo de inventarios y agilice los procesos de toma de decisiones a partir del planteamiento del mejor escenario. No obstante, la cúspide de esta etapa será alcanzada a través del éxito del análisis de sensibilidad del modelo, ya que es lo que permite evaluar las discrepantes situaciones en las que se podría estar para los datos evaluados.
4. Validación del modelo	Por otro lado, y paralelo a lo anterior, es necesario confirmar que los resultados obtenidos son veraces y se adecúan a lo esperado en busca de tener mayor certeza de lo esperado en la realidad. Esta evaluación se daría desde el correcto funcionamiento matemático del algoritmo a través de la programación, y a su vez, desde las respuestas que arroje .Es por esto que la iteración es parte fundamental de la aplicación de la propuesta de solución, ya que al tener diferentes acercamientos a la respuesta, el gestor de secuenciación de la línea de ensamble encontraría la programación de la producción que mejor se adapte a los parámetros establecidos.
5. Implementación de la solución	Finalmente se busca concluir sobre la articulación del algoritmo "Sistema Inmune", especialmente en la calidad de los resultados, tiempos computacionales y adaptabilidad a otras características, bajo el mismo modelo de cadena de abastecimiento. Sobre todo las virtudes que brinde para la toma de decisiones gerenciales a mediano y largo plazo.

Fuente: Orellano, Luis (2020)

8. Cronograma de actividades

A continuación, se denota el plan de trabajo en función del tiempo acordado para su desarrollo.

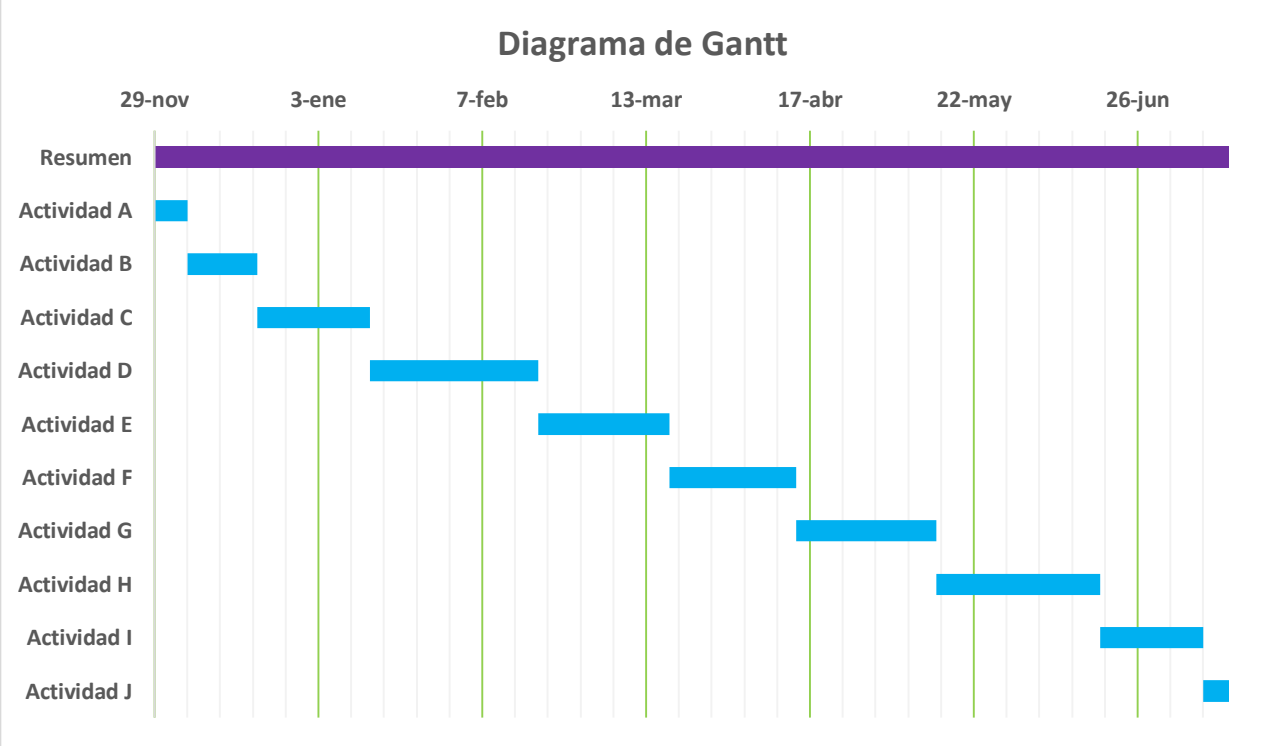


Figura 1. Cronograma de actividades.

Fuente:Orellano, Luis (2020)

Tabla 2
Información detallada de actividades a realizar

Ítem	Nombre de actividad	Fecha de Inicio	Duración en días	Fecha Fin
Resumen	**Trabajo de grado II**	29-nov	**239**	25-jul
Actividad A	**Revisión inicial de la literatura sobre modelos de línea de ensamble con máquinas paralelas**	29-nov	**7**	6-dic
Actividad B	**Presentación y estudio del anteproyecto**	6-dic	**15**	21-dic
Actividad C	**Estructurar la programación del algoritmo**	21-dic	**24**	14-ene
Actividad D	**Realizar pruebas de validación de la programación**	14-ene	**36**	19-feb
Actividad E	**Iteración del algoritmo**	19-feb	**28**	18-mar
Actividad F	**Registrar y procesar los resultados**	18-mar	**27**	14-abr
Actividad G	**Verificar la consistencia de los resultados**	14-abr	**30**	14-may
Actividad H	**Analizar y concluir sobre los datos obtenidos**	14-may	**35**	18-jun
Actividad I	**Organizar y presentar la información acorde al comportamiento del algoritmos y su utilidad**	18-jun	**22**	10-jul
Actividad J	**Entrega de informe final**	10-jul	**15**	25-jul

Fuente: Orellano, Luis (2020)

9. Presupuesto

Tabla 3
Balance general del presupuesto requerido

Presupuesto			
Ingresos		**Egresos**	
Ítem	**Monto**	**Ítem**	**Monto**
Recursos propios	$ 282.000	Papelería	$ 72.000
		Transporte	$ 123.000
		Otros gastos	$ 66.000
		Software y servicios técnicos	$ 21.000
Total ingresos	**$ 282.000**	**Total Egresos**	**$ 282.000**

Fuente: Orellano, Luis (2020)

Tabla 4

Detalle del registro de egresos presupuestado

Nombre del Proyecto	**Adaptación de un algoritmo de sistema biológico inmune para el mejoramiento de sistemas de producción tipo flow shop con máquinas en paralelo**						
Rubros	**Mes 1**	**Mes 2**	**Mes 3**	**Mes 4**	**Mes 5**	**Mes 6**	**Total**
Equipos							
Computador	$ 11.000	$ 11.000	$ 11.000	$ 11.000	$ 11.000	$ 11.000	$ 66.000
Internet	$ 7.500	$ 7.500	$ 7.500	$ 7.500	$ 7.500	$ 7.500	$ 45.000
Celular	$ 6.500	$ 6.500	$ 6.500	$ 6.500	$ 6.500	$ 6.500	$ 111.000
Materiales							
Fotocopias	$ 7.000	$ 7.000	$ 7.000	$ 7.000	$ 7.000	$ 7.000	$ 42.000
Documentos							
Informe final	$ 8.500	$ 8.500	$ 8.500	$ 8.500	$ 8.500	$ 8.500	$ 51.000
Contingencias	$ 6.500	$ 6.500	$ 6.500	$ 6.500	$ 6.500	$ 6.500	$ 39.000
Total proyecto	$ 47.000	$ 47.000	$ 47.000	$ 47.000	$ 47.000	$ 47.000	$ 282.000

Fuente: Orellano, Luis (2020)

10. Flow-shop scheduling

El problema abordado se relaciona con la apropiada programación de trabajos a realizar dentro de una línea de ensamble, cuyas estaciones o máquinas, se encuentran en paralelo desarrollo. Es necesario que el ciclo de tareas en proceso sea de flujo continuo con un mínimo de tiempo de inactividad y espera entre cada máquina. Adicional a la anterior consideración, es relevante tener presente que este tipo de entornos productivos son un caso especial del *Jobshop Scheduling* y que el orden de procesamiento se mantiene constante en el cumplimiento de cada estación. Por otra parte, es necesario conocer el siguiente concepto ya que será empleado en múltiples ocasiones.

- **Matriz de tiempos de procesamiento**: Es la batería de información que data los tiempos de procesamiento de cada trabajo en cada máquina. (Ver Tabla 5)

Tabla 5
Ejemplo Matriz de tiempos de procesamiento

	Máquinas			
Trabajos	**1**	**2**	**3**	**4**
1	3	1	3	6
2	4	2	2	2
3	3	1	4	4

Fuente: Orellano, Luis (2020)

Por consiguiente y en concordancia con lo propuesto por Wilson (1989), y lo estudiado en la literatura, el modelo resolutivo define los siguientes componentes:

11.1. Parámetros del modelo

$i =$ Subíndice para trabajos a realizar, $i = 1,2, \dots, N; N$ es el número total de trabajos;

$j =$ Subíndice para posiciones a programar, $j = 1,2, \dots, N; N$ es el número total de las posiciones posibles;

$k =$ Subíndice para máquinas a programar, $k = 1,2, \dots, M; M$ es el número total de máquinas;

$X_{ij} = 1$ si el trabajo i está programado en la posición j para procesar, 0 si no es así.

$\rho_{ik} =$ Es el tiempo de procesamiento del trabajo i en la máquina k;

$C_{jk} =$ Tiempo en el que el trabajo programado j comienza el procesamiento en la máquina k.

11.2. Restricciones del modelo

- **Asignación**: Estas restricciones son el eje central de la solución del problema, ya que determinan la programación afirmativa o nula de los trabajos en cada máquina. Es decir, cumple con incluir los tiempos de las tareas en el cálculo estacional del *Makespan*.

$$\begin{cases} \sum_{j=1}^{N} X_{ij} = 1 \quad ; \quad j = 1,2, \dots, N \\ \sum_{i=1}^{N} X_{ij} = 1 \quad ; \quad i = 1,2, \dots, N \end{cases}$$

- **Condiciones iniciales de secuenciación:** Para este tipo de restricciones, su principal función se fundamenta al comienzo de la programación, ya que inicializa el ciclo de trabajo sin tareas en ejecución. Además de que no hay conflicto por traslape de tiempos entre trabajos de distintas máquinas.

$$\begin{cases} C_{j,k+1} = C_{jk} + \sum_{i=1}^{N} \rho_{ik} X_{ij} \quad ; \; k = 1,2,\dots,M-1 \\ C_{11} = 0 \\ C_{j+1,k} = C_{jk} + \sum_{i=1}^{N} \rho_{ik} X_{ij} \quad ; \; j = 1,2,\dots,N-1 \end{cases}$$

- **Condiciones intermedias de secuenciación**: Por otro lado, las restricciones detalladas a continuación son útiles al momento de asignar un trabajo de una máquina a otra, ya que estas determinan si la tarea previa en la siguiente máquina, tiene mayor duración que la que está por establecerse. Lo anterior es relevante porque hace que la programación sea continua entre las máquinas dentro de la línea de ensamble.

$$\begin{cases} C_{j,k+1} \geq C_{jk} + \sum_{i=1}^{N} \rho_{ik} X_{ij} \quad ; \quad k = 1,2,\dots,M-1 \quad ; \; j = 2,3,\dots,N \\ C_{11} = 0 \\ C_{j+1,k} \geq C_{jk} + \sum_{i=1}^{N} \rho_{ik} X_{ij} \quad ; \quad j = 1,2,\dots,N-1 \quad ; \; k = 2,3,\dots,M \end{cases}$$

11.3. Función objetivo del modelo

Finalmente, el mínimo *Makespan* se halla encontrando el menor valor de tiempo de procesamiento de todos los trabajos entre las posibles combinaciones que se puedan dar.

$$\boldsymbol{min}(C_{max}) = C_{NM} + \sum_{i=1}^{N} \rho_{iM} X_{iN}$$

Para concluir se presenta un ejemplo que mejore el entendimiento del modelo aplicado.

Ejemplo base

A continuación, se presenta el orden de trabajos a desarrollar (Ver Figura 2), la matriz de tiempos (Ver Tabla 6) y las formulas en detalle para el ejercicio del mismo.

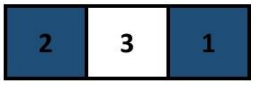


Figura 2. Ejemplo de ordenamiento de trabajos.
Fuente:Orellano, Luis (2020)

Tabla 6
Matriz de tiempos para ejercicio

	Máquinas		
Trabajos	**1**	**2**	**3**
1	3	1	3
2	4	2	2
3	3	1	4

Fuente: Orellano, Luis (2020)

Para empezar y como se indicó anteriormente, se realiza el cálculo de tiempos de procesamiento con las restricciones de las condiciones iniciales.

$$C_{11} = 0$$

$$C_{21} = C_{11} + \rho_{21} = 0 + 4 = 4$$

$$C_{22} = C_{21} + \rho_{22} = 4 + 2 = 6$$

$$C_{23} = C_{22} + \rho_{23} = 6 + 2 = 8$$

Seguido de esto, se sugiere llevar el acumulado de manera gráfica para apoyarse visualmente de lo que se realiza. (Ver Figura 3)

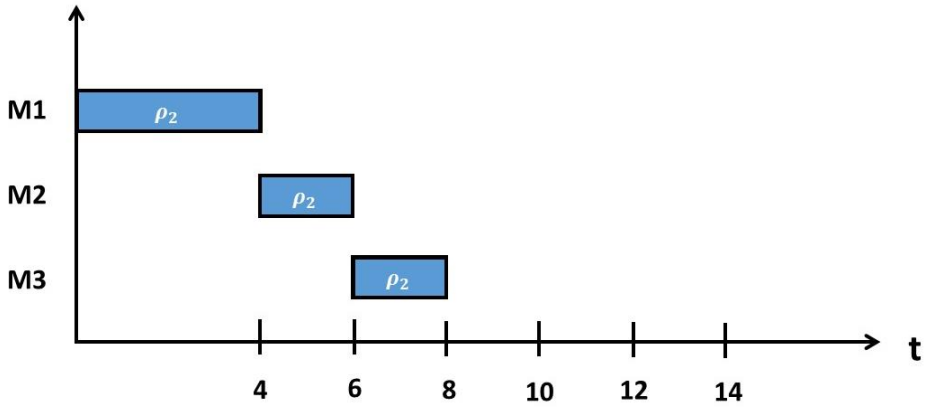


Figura 3. Diagrama de Gantt 1 – Ejercicio.
Fuente: Orellano, Luis (2020)

Después de asignar el trabajo ubicado en la primera posición, se procede a aplicar las restricciones de carácter intermedio que impiden que se pierda el flujo continuo de la línea de ensamble. Para ello se calcula la duración del trabajo a programar junto con la tarea previa, y se compara con el tiempo total que requiere la tarea en ser ejecutada teniendo en cuenta la anterior que ya fue establecida. (Ver Figura 4)

$$C_{31} = C_{21} + \rho_{31} = 4 + 3 = 7$$

$$\begin{cases} C_{22} + \rho_{32} \\ C_{31} + \rho_{32} \end{cases} = \begin{cases} 6 + 1 = 7 \\ 7 + 1 = \mathbf{8} \end{cases}$$

$$C_{32} = C_{31} + \rho_{32} = 7 + 1 = 8$$

$$\begin{cases} C_{32} + \rho_{33} \\ C_{23} + \rho_{33} \end{cases} = \begin{cases} 8 + 4 = \mathbf{12} \\ 8 + 4 = \mathbf{12} \end{cases}$$

$$C_{33} = C_{32} + \rho_{33} = 12$$

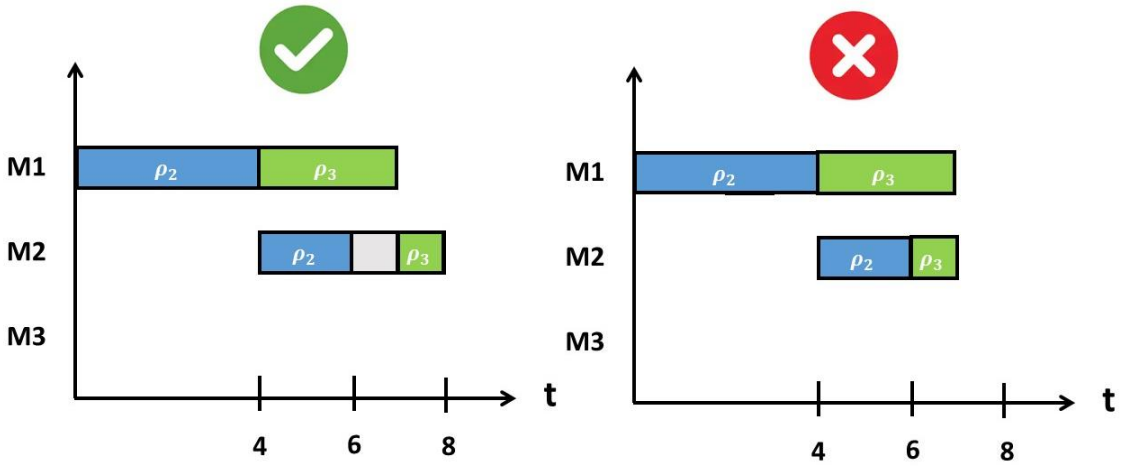


Figura 4. Diagrama de Gantt 2 – Ejercicio.

Fuente: Orellano, Luis (2020)

Para finalizar, se continúa llevando el control riguroso de que las tareas sean consecutivas al ejecutar un trabajo de una máquina a otra. Por consiguiente, el último valor de C_{jk} será el tiempo total de *Makespan* para la secuenciación propuesta al inicio. (Ver Figura 5)

$$C_{11} = C_{31} + \rho_{11} = 7 + 3 = 10$$

$$\begin{cases} C_{11} + \rho_{12} \\ C_{32} + \rho_{12} \end{cases} = \begin{cases} 10 + 1 = \mathbf{11} \\ 8 + 1 = 9 \end{cases}$$

$$C_{12} = C_{11} + \rho_{12} = 10 + 1 = 11$$

$$\begin{cases} C_{12} + \rho_{13} \\ C_{33} + \rho_{13} \end{cases} = \begin{cases} 11 + 3 = 14 \\ 12 + 3 = \mathbf{15} \end{cases}$$

$$C_{13} = C_{33} + \rho_{13} = 15$$

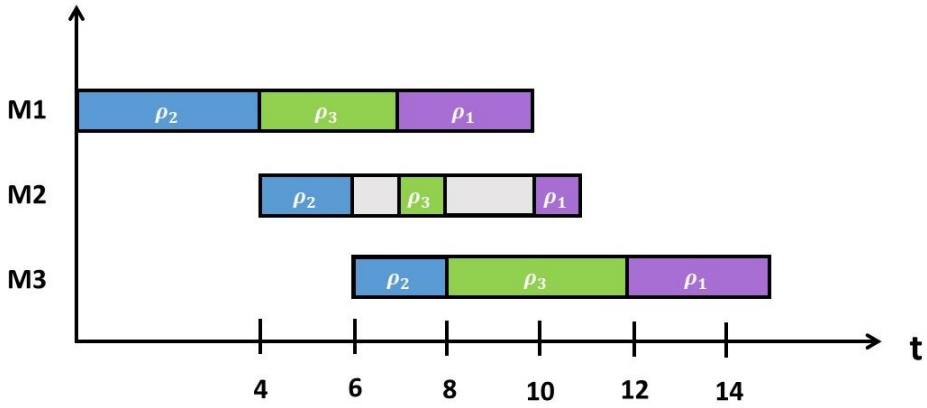


Figura 5. Diagrama de Gantt 3 – Ejercicio.

Fuente: Orellano, Luis (2020)

11. Algoritmo de Sistema Biológico Inmune aplicado al flow shop scheduling

Teniendo en cuenta la complejidad del problema, y conociendo casos donde existe una gran cantidad de trabajos a asignar en diversas máquinas, el algoritmo de Sistema Inmune como técnica metaheurística es de alta utilidad como método de solución a la hora de tener una aproximación a lo deseado como óptimo en relación al *Makespan*. Partiendo de lo expuesto por Yingchen, Geng, Zhang & Junhu (2018) para dar solución al problema de planeación y gestión de inventarios multiproducto, se propone un seudocódigo programado para solucionar problemas de secuenciación de máquinas en paralelo tipo flow shop ejecutado sobre el modelo de Wilson (1989).

Con base en lo anterior, es necesario ampliar algunos fundamentos que serán empleados en la simulación, tales como:

- **Número de Generaciones**: Indica el número de ocasiones en que se itera el algoritmo en su totalidad (selección, cruce y mutación).
- **Número de individuos:** Es la cantidad de cromosomas o soluciones que integran cada generación.
- **Porcentaje de mutación**: Es la probabilidad que se le asigna por igual a cada generación, para determinar la cantidad de individuos que serán permutados, respecto a su organización de trabajos como solución presentada del individuo. Esta modificación a la información genética se realiza solo en una pareja de trabajos dentro del individuo seleccionado. Es importante señalar que los cromosomas son escogidos al azar para cada generación.

- **Gen(es):** Es el valor que indica el trabajo que debe ser procesado. La confabulación de todos los genes también es llamada información genética. (Ver Figura 6)
- **Cromosomas:** Son la representación de la información genética en esta investigación, y se basa en la secuencia de operaciones que integran la programación completa de las actividades. De esta forma facilita el desarrollo del programa a la hora de simular las funciones de cruce y mutación. (Ver Figura 6)

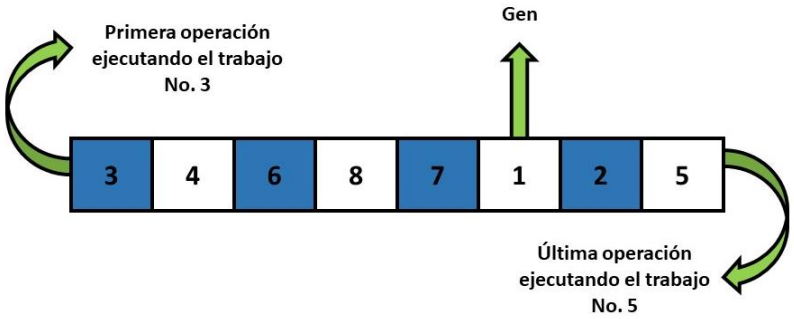


Figura 6. Cromosoma y Gen.
Fuente: Orellano, Luis (2020)

Por lo tanto, se entiende que la secuenciación de izquierda a derecha es una posible solución a la programación de trabajos que este individuo plantea. Por consiguiente, internamente cada trabajo contiene el tiempo de procesamiento que desempeña en cada máquina y que sirve para determinar el *Makespan* de todo el ciclo.

- **Antígeno:** Es el cromosoma que cuenta con mejor rendimiento dentro de la generación estudiada. Sobre él se realiza el cálculo de la afinidad de las parejas restantes y con ello se distinguen los predecesores de la siguiente generación. Cabe resaltar que, en cada generación se compara el antígeno escogido con el de la

inmediatamente anterior, para así fijar siempre el comportamiento de la simulación hacia las mejores respuestas.

Por consiguiente, se procede a describir la propuesta de manera general con el fin de tener una idea sobre el próximo análisis en detalle.

12. Codificación propuesta

El programa realizado comienza generando una población completamente aleatorizada de individuos llamados cromosomas. Estos cromosomas son diseñados también aleatoriamente siguiendo la información de la matriz de tiempos. En tercer lugar, se calcula el *Makespan* de cada individuo para así poder organizar de menor a mayor los integrantes de esta generación. Seguido de lo anterior, se denomina como antígeno al cromosoma que menor tiempo registra en comparación con los demás. En consecuencia, se emparejan de todas las formas posibles los individuos restantes y se relacionan según el criterio de afinidad con el antígeno. Al obtener los valores de este criterio, se escoge la pareja de cuyo porcentaje sea mayor y se les nombra como padres o predecesores de la siguiente generación. Luego, dichos cromosomas a partir de la operación de cruce y mutación, repueblan la primera descendencia de acuerdo al número de individuos definidos. Por lo tanto, el procedimiento inicia de nuevo a partir de la organización de acuerdo al *Makespan*, y posterior escogencia del antígeno. Sin embargo, al seleccionar este individuo, se debe comparar con el cromosoma previo de su misma naturaleza y determinar si el nuevo es mejor o si de lo contrario el anterior prevalece con mejor información genética. Definido esto, el programa sigue las mismas instrucciones señaladas hasta cumplir con el número de generaciones registradas. (Ver Figura 7)

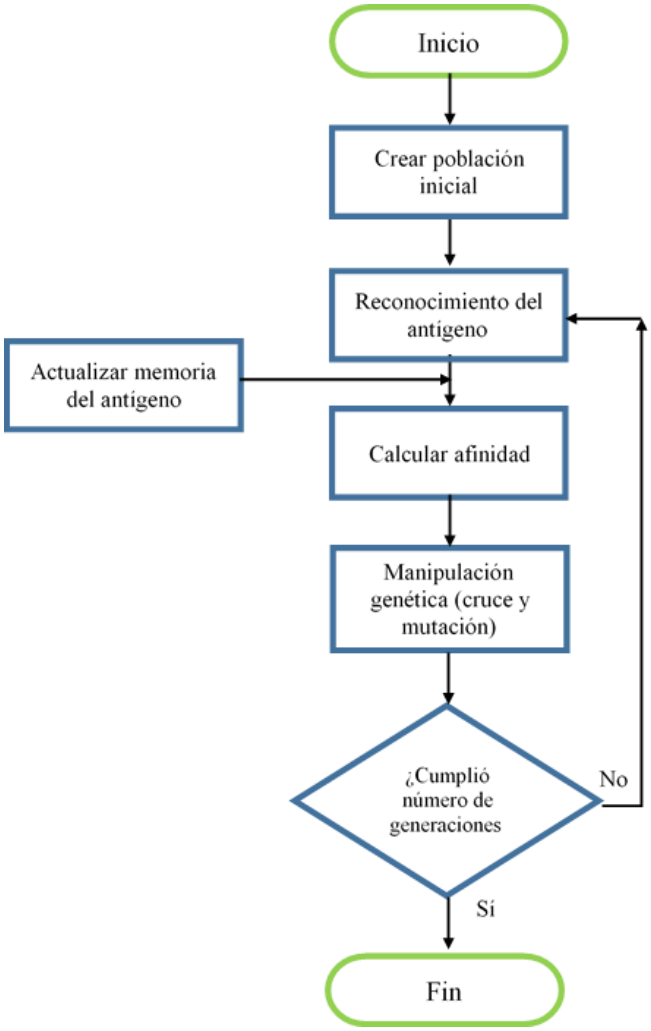


Figura 7. Diagrama de flujo.
Fuente: Orellano, Luis (2020)

No obstante, en seguida se pormenoriza cada procedimiento mencionado.

13.1. Generación de la población inicial

Es el primer pilar sobre el que se desarrolla el programa y juega un papel importante a la hora de evaluar múltiples regiones dentro de la simulación. En esta fase inicial se crean los individuos que serán pioneros en evaluar las posibles soluciones del problema. (Ver Figura 8)

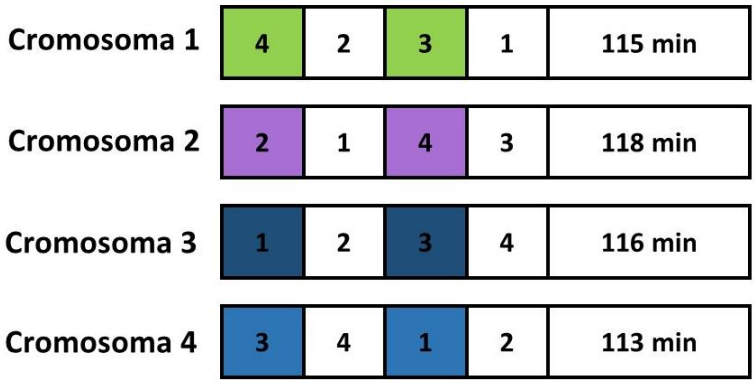


Figura 8. Población de ejercicio.
Fuente: Orellano, Luis (2020)

Esto se da gracias a que su formulación es de carácter aleatorio por lo que varía el escenario inicial para una misma matriz de tiempos. Para ello, Visual Basic emplea la popular técnica de generación de números aleatorios de Mersenne Twister MT19937, creada por Makoto Matsumoto y Takuji Nishimura en 1997. Este proceso utiliza datos de tipo Mersenne los cuales son de una unidad menor que una potencia del 2, decir $M_n = 2^n - 1$. Seguidamente su sucesión es de tipo $2^{2^{n-1}}$ y con período $2^{19937} - 1 = 10^{600}$ números según Harase (2019). No obstante, este número es adaptado al intervalo de valores que integran todos los trabajos definidos para el modelo.

13.2. Selección

Este procedimiento cumple con organizar los cromosomas en función de su *Makespan*, para así, determinar los individuos que serán los padres que originarán la siguiente generación. El orden se establece de menor a mayor tiempo como lo indica la figura 9.

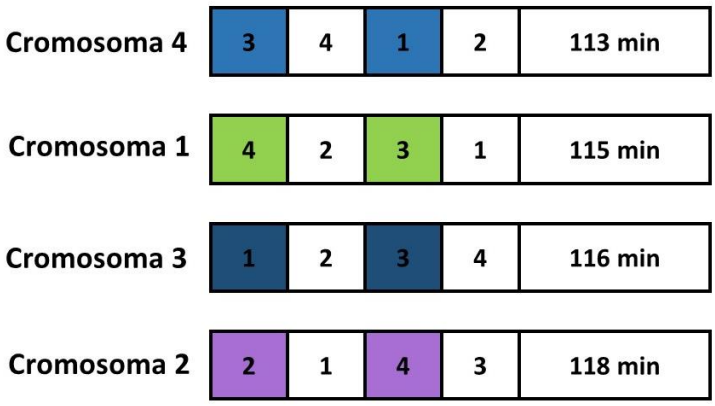


Figura 9. Selección 1.
Fuente: Orellano, Luis (2020)

Sin embargo, la principal selección de cromosomas se da a partir del cálculo de la afinidad. Esta característica parte de denominar al mejor individuo como antígeno mientras los demás individuos se emparejan bajo todas las combinaciones posibles como se observa en la figura 10.

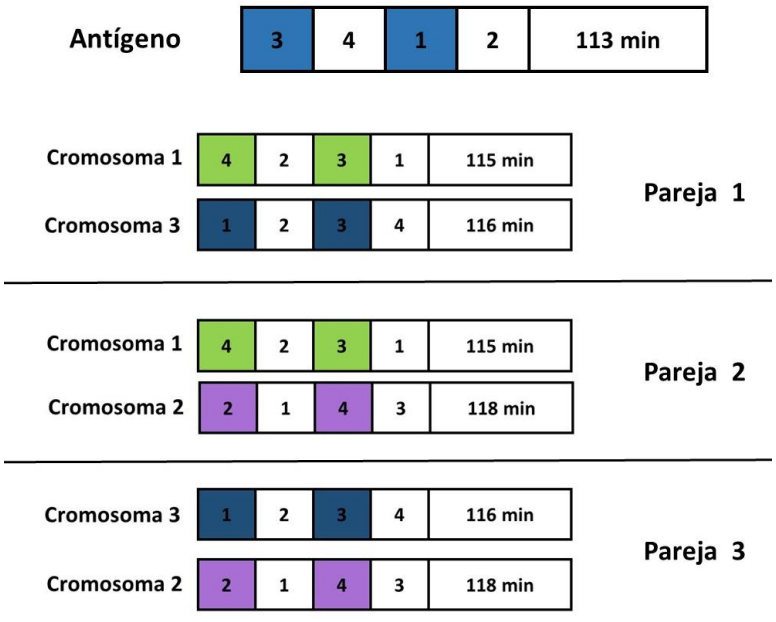


Figura 10. Selección 2.

Fuente: Orellano, Luis (2020)

Por otra parte, se procede a emplear la técnica propuesta por Yingchen, Geng, Zhang & Junhu (2018) donde se obtienen los valores de semejanza entre el antígeno seleccionado y las parejas de individuos que se conformaron. Inicialmente para la primera pareja, se compara cada gen entre el antígeno y cada cromosoma, donde de ser iguales, se asigna el número 1 a la variable auxiliar λ_{ij}, de lo contrario 0,05. Este último término está indexado en *j* con motivo de la posición que se evalúa, al igual que *i*, para el cromosoma en estudio. (Ver figura 11)

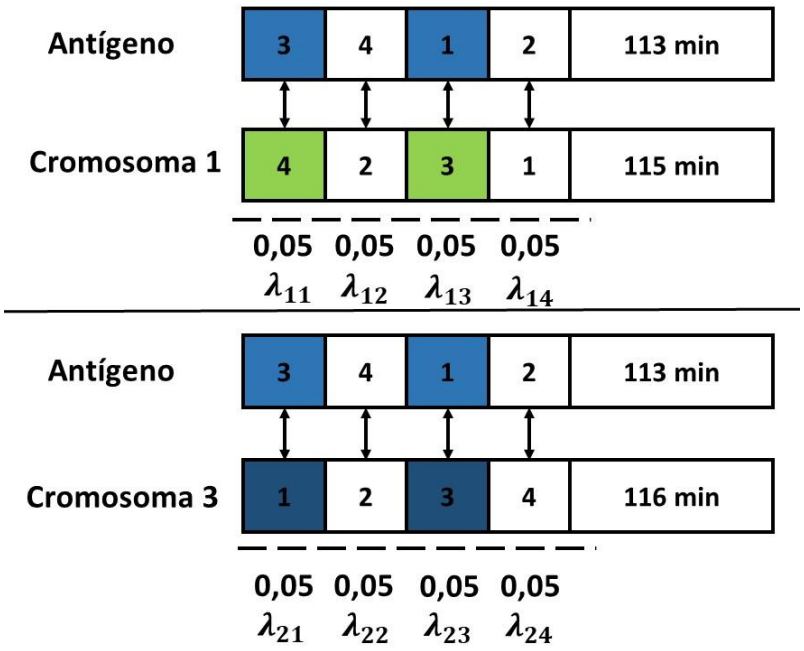


Figura 11.Selección 3.

Fuente: Orellano, Luis (2020)

En consecuencia, se desarrolla el cálculo del nivel de entropía $H_j(C_i, C_i)$ que se enuncia a continuación.

$$H_1(C_1, C_3) = -\{\lambda_{11}\log_2(\lambda_{11}) + \lambda_{21}\log_2(\lambda_{21})\}$$

$$H_1(C_1, C_3) = -\{0{,}05 \log_2(0{,}05) + 0{,}05 \log_2(0{,}05)\}$$

$$H_1(C_1, C_3) = -\{-0{,}22 - 0{,}22\}$$

$$H_1(C_1, C_3) = 0{,}44$$

$$H_2(C_1, C_3) = -\{\lambda_{12} \log_2(\lambda_{12}) + \lambda_{22} \log_2(\lambda_{22})\}$$

$$H_2(C_1, C_3) = -\{0{,}05 \log_2(0{,}05) + 0{,}05 \log_2(0{,}05)\}$$

$$H_2(C_1, C_3) = -\{-0{,}22 - 0{,}22\}$$

$$H_2(C_1, C_3) = 0{,}44$$

$$H_3(C_1, C_3) = -\{\lambda_{13} \log_2(\lambda_{13}) + \lambda_{23} \log_2(\lambda_{23})\}$$

$$H_3(C_1, C_3) = -\{0{,}05 \log_2(0{,}05) + 0{,}05 \log_2(0{,}05)\}$$

$$H_3(C_1, C_3) = -\{-0{,}22 - 0{,}22\}$$

$$H_3(C_1, C_3) = 0{,}44$$

$$H_4(C_1, C_3) = -\{\lambda_{14} \log_2(\lambda_{14}) + \lambda_{24} \log_2(\lambda_{24})\}$$

$$H_4(C_1, C_3) = -\{0{,}05 \log_2(0{,}05) + 0{,}05 \log_2(0{,}05)\}$$

$$H_4(C_1, C_3) = -\{-0{,}22 - 0{,}22\}$$

$$H_4(C_1, C_3) = 0{,}44$$

Entre tanto, se promedia el nivel de entropía para cada gen comparado. Donde M indica el número de genes que alberga cada cromosoma y se obtiene mediante:

$$H(C_i, C_i) = \left(\frac{1}{M}\right) \sum_{j=1}^{M} H_j(C_i, C_i)$$

$$H(C_1, C_3) = \left(\frac{1}{4}\right) \sum_{j=1}^{4} H_j(C_1, C_3)$$

$$H(C_1, C_3) = \left(\frac{1}{4}\right)\sum_{j=1}^{4}(0{,}44 + 0{,}44 + 0{,}44 + 0{,}44)$$

$$H(C_1, C_3) = \left(\frac{1}{4}\right)\sum_{j=1}^{4}(0{,}44 + 0{,}44 + 0{,}44 + 0{,}44)$$

$$H(C_1, C_3) = 0{,}44$$

Finalmente, la afinidad está dada por:

$$A_{13} = \frac{1}{1 + H(C_i, C_i)} = \frac{1}{1 + H(C_1, C_3)} = \frac{1}{1 + 0{,}44} = 0{,}69$$

El anterior proceso se escala a todas las parejas previamente detalladas y sobre las cuales se obtuvieron los siguientes valores:

$$A_{12} = 0{,}69 \; ; A_{32} = 0{,}69$$

Posterior a ello, la pareja con mayor porcentaje encontrado serán los padres que con su información genética crearán la nueva generación de cromosomas. Para esta ocasión que las afinidades fueron iguales, se escoge la primera pareja que registra menos *Makespan* para ambos individuos. En conclusión, los cromosomas 1 y 3 aportarán su información genética relacionada con el orden de los trabajos para repoblar una posterior generación.

13.3. Cruce

Esta fase del algoritmo se encarga de fusionar las características que trae consigo cada padre escogido en la etapa anterior. Para ello se determinan posiciones aleatorias dentro del cromosoma y se seccionan de manera fraccionada, para luego unir como lo muestra la figura 12.

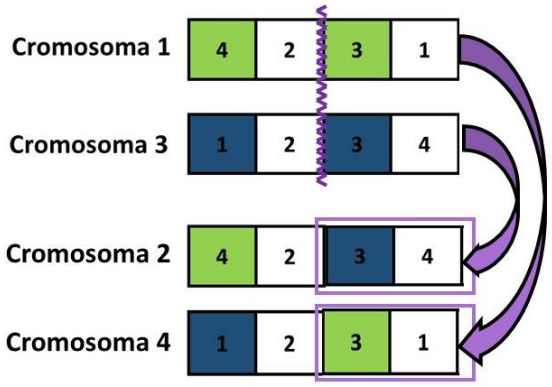


Figura 12. Cruce 1.
Fuente: Orellano, Luis (2020)

Retomando los valores calculados en la sección de selección, se continua con la ejecución del algoritmo. Inicialmente se realizarían los cruces de genes entre la pareja conformada por el cromosoma 1 y 3. Para lo cual se generarían dos individuos de igual información genética ya que al tener pocos trabajos, las combinaciones posibles son pocas. Por otro lado, usualmente ocurrirá que resulten traslapes de trabajos como se aprecia en la figura 13. No obstante, este posible error es solucionado y explicado la sección de funcionamiento del programa.

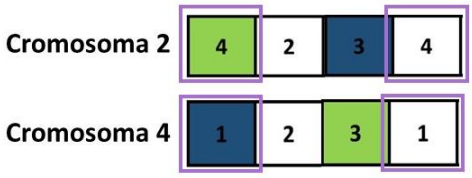


Figura 13. Cruce 2.
Fuente: Orellano, Luis (2020)

De igual forma al solucionarse dicho inconveniente, la población quedaría de la siguiente forma. (Ver Figura 14)

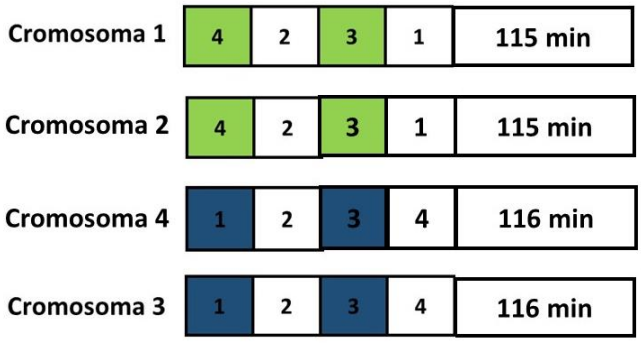


Figura 14.Cruce 3.
Fuente: Orellano, Luis (2020)

13.4. Mutación

Esta actividad cumple la función de explorar distintos escenarios dentro de la población, permutando los genes de algunos individuos en cada generación. Es importante resaltar que, gracias a esta operación, el algoritmo mejora su rendimiento ya que anula parcialmente la rápida terminación de la búsqueda en óptimos locales.

Siguiendo el ejemplo anterior, se selecciona aleatoriamente el individuo No. 4 y se procede a mutar un par de sus posiciones. (Ver Figura 15)

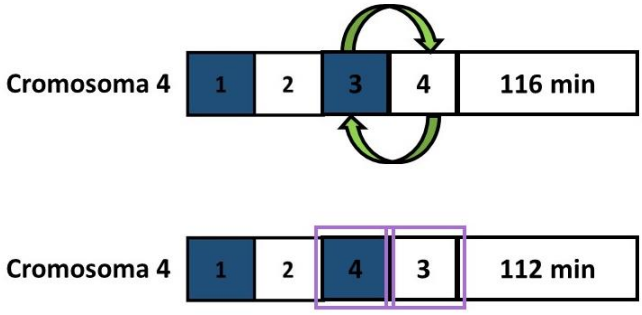


Figura 15. Mutación 1.
Fuente: Orellano, Luis (2020)

En conclusión, se esboza la población resultante o segunda generación, de todo el proceso descrito respecto al funcionamiento del algoritmo. (Ver Figura 16)

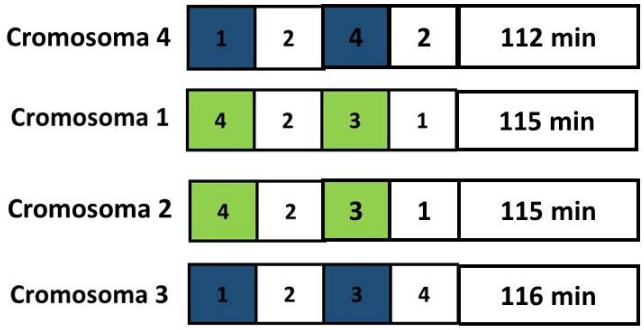


Figura 16. Mutación 2.
Fuente: Orellano, Luis (2020)

Todo lo anterior se itera hasta alcanzar el número de generaciones determinado por el usuario del programa.

13. Funcionamiento del programa

En representación de todo lo explicado anteriormente, a continuación, se explica la puesta en marcha del algoritmo a partir de lo que será visto por el usuario cuando lo ejecute.

Sin embargo, antes de afrontar todo el programa se debe definir la función interna que actúa como controlador y corrector de la información genética. En ella recae la vigilancia de que no se repitan los trabajos, y para mayor comprensión se utilizará el ejemplo de la figura 13.

Volviendo al problema suscrito a ese caso, se escoge el cromosoma 2 para ser estudiado. (Ver Figura 17)

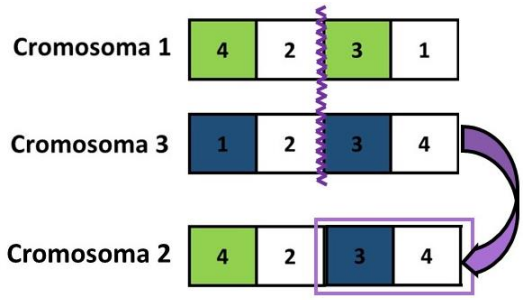


Figura 17. Corrección de error 1.
Fuente: Orellano, Luis (2020)

En términos generales esta función interviene con 4 variables caracterizadas de la siguiente forma:

$x =$ Variable Booleana ; $x =$ "*True*" or "*False*"

$j =$ Indica la posición del trabajo donde se realizará el corte ; $j = 1,2,3,..,N$

$k =$ Recorre los números de las posiciones que serán los genes del individuo a partir del segundo padre $k = 1,2,3,..,N$

$m =$ Sirve para revisar posición a posición la asignación de los genes del primer hijo

$m = 1,2,3,..,j-1$

Teniendo claro lo anterior, a continuación, se demuestra su funcionamiento.

Recapitulando el corte planteado en la sección de cruce, se inicializan las variables desde su mínimo valor. Dado esto, el programa entiende el número de trabajos y por ende de posiciones que va asignar. Cumplida dicha lectura, el algoritmo guarda los genes del primer padre hasta el punto de corte y lo copia como información del hijo. Paralelo a esto, *m* recorre un bucle interno dentro de las posiciones del cromosoma 1, para verificar que el trabajo no haya sido asignado hasta el momento. (Ver Figura 18)

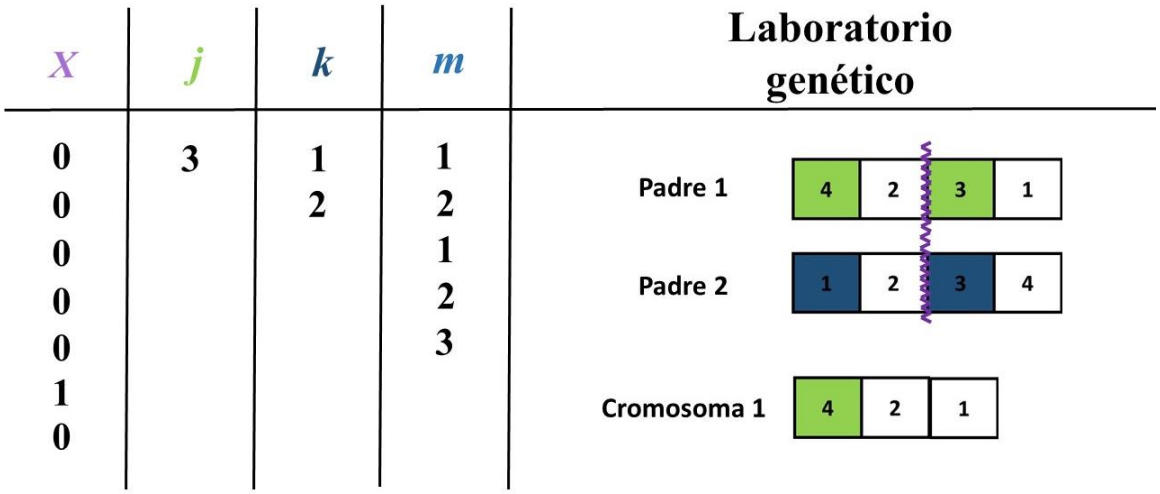


Figura 18. Corrección del error 2.
Fuente: Orellano, Luis (2020)

Finalizado el ciclo anterior, el booleano se actualiza para poder empezar con la segunda posición del segundo padre a partir del cumplimiento del condicional. Siendo así, se guarda el valor en esa posición. Además, siguiendo el mismo curso del inciso preliminar. (Ver Figura 19)

X	*j*	*k*	*m*
0	3	1	1
0		2	2
0			1
0			2
0			3
1			
0			

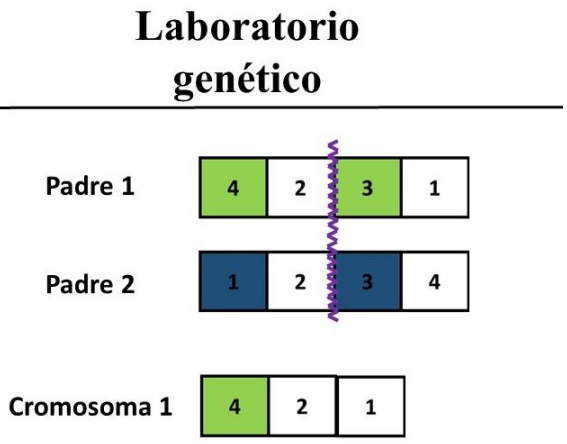


Figura 19. Corrección del error 3.
Fuente: Orellano, Luis (2020)

Culminada la asignación de los genes del segundo padre (antes del punto de corte), se procede a dar lectura de la información genética del primer padre con un ciclo de iguales características. No obstante, dicho bucle prioriza la correcta asignación de las tareas y es por ello que, de repetirse el trabajo, fija los trabajos faltantes de acuerdo a los ya programados. Esto indica que, si hay pocas actividades a realizar, habrá menos posibilidades de que el cruce sea totalmente completo ya que es difícil que no se repitan en la secuenciación. (Ver Figura 20)

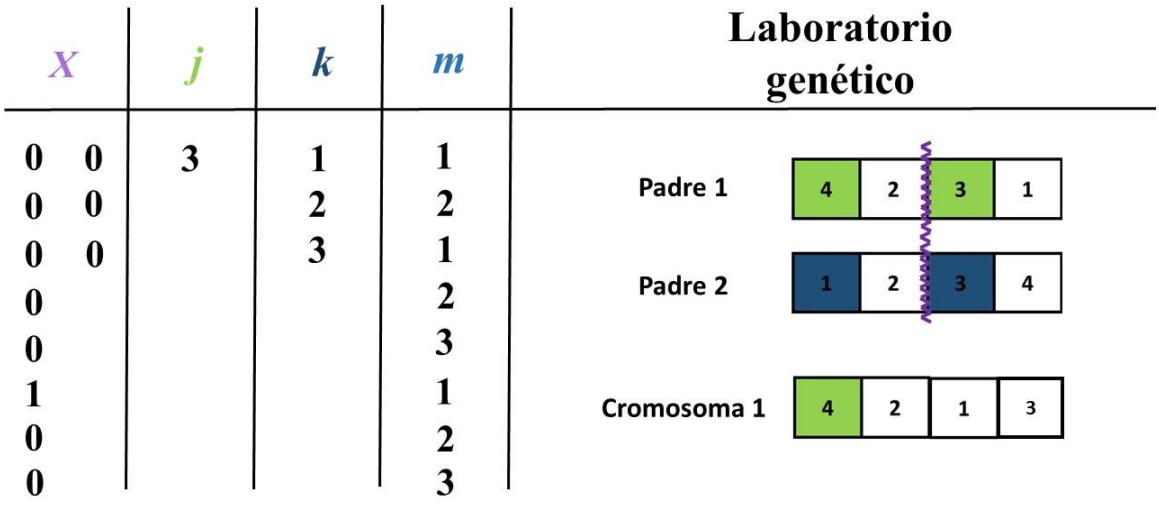


Figura 20. Corrección del error 4.

Fuente: Orellano, Luis (2020)

Explicada la función vigilante de la secuenciación de trabajos, en seguida se detalla paso a paso la ejecución del programa.

Para comenzar se despliega un mensaje de saludo cordial como se muestra en la figura 21, el cual indica en primera instancia que el archivo del programa está cargando correctamente el seudocódigo.

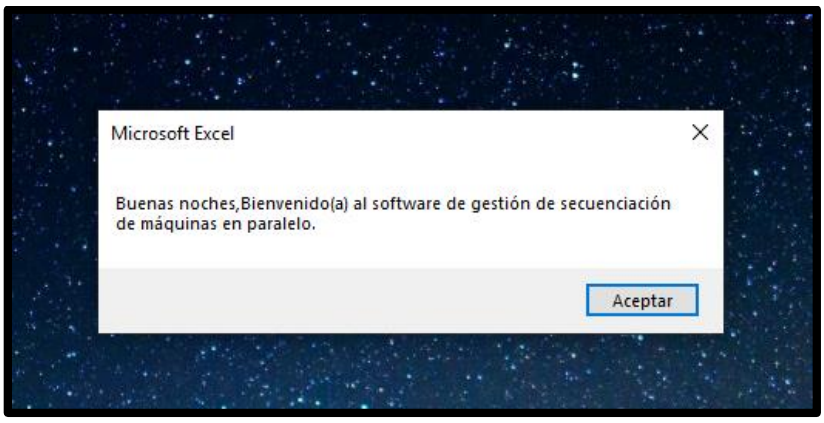


Figura 21. Mensaje de Bienvenida.

Fuente: Orellano, Luis (2020)

En segundo lugar, se presenta la información general del programa de acuerdo a su autor e institución a la que pertenece. (Ver Figura 22)

Figura 22. Información general.
Fuente: Orellano, Luis (2020)

Seguido de lo anterior, se develan los parámetros que previamente fueron explicados y que cumplen con el dimensionamiento del problema. En este caso se completa la información en las casillas donde a manera de ejemplo está el número 5. (Ver Figura 23)

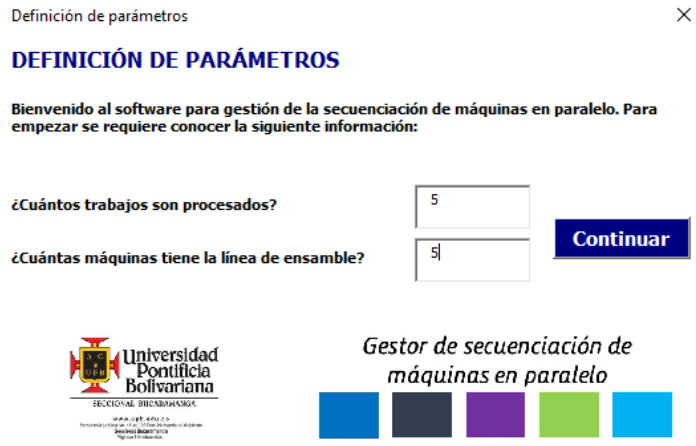


Figura 23.Definición de parámetros.
Fuente: Orellano, Luis (2020)

Al momento de oprimir el botón de continuar, se activa el mensaje que informa si los parámetros se registraron correctamente. Esto se da en aras del correcto funcionamiento

del programa y para evitar problemas al usuario en caso de cometer errores al digitar la información. (Ver Figura 24)

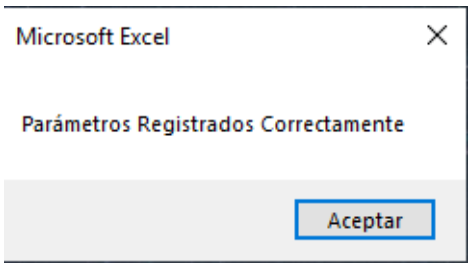


Figura 24. Verificación de datos.
Fuente: Orellano, Luis (2020)

Posteriormente, el programa automáticamente crea la matriz de tiempos según los parámetros registrados. (Ver Figura 25)

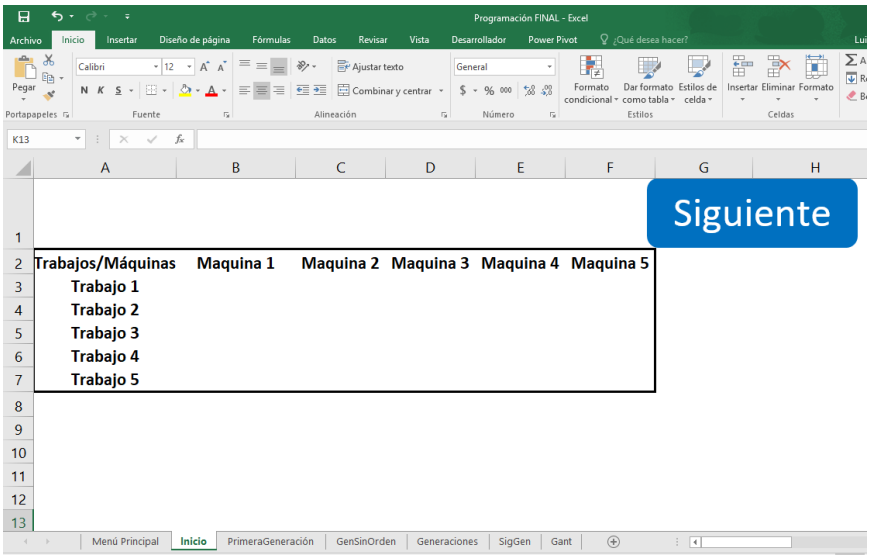


Figura 25. Generación de matriz de tiempos.
Fuente: Orellano, Luis (2020)

Es importante advertir que los tiempos utilizados para ejecutar el algoritmo, deben ir dentro del delineado de color negro que en la hoja de Excel se observa. (Ver Figura 26)

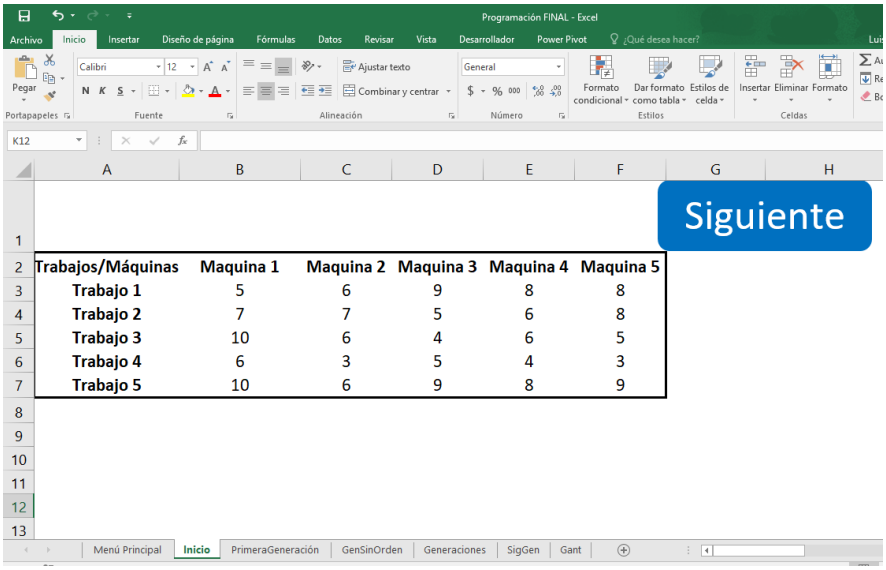


Figura 26. Llenado de matriz de tiempos.

Fuente:Orellano, Luis (2020)

Terminado lo anterior y al oprimir el botón de “Siguiente”, inmediatamente aparecerá un formulario el cual solicita los valores de número de individuos y generaciones, junto con el porcentaje a mutar en cada población. (Ver Figura 27)

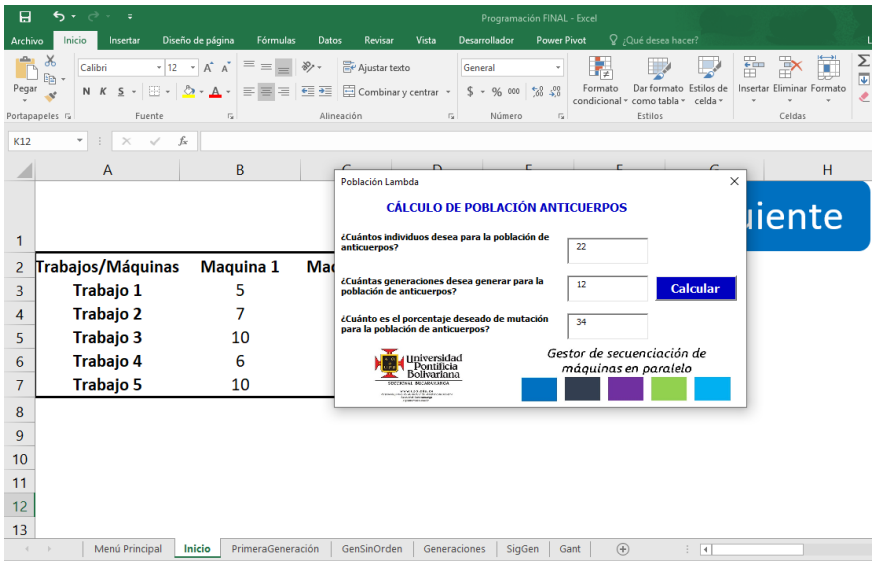


Figura 27. Definición de parámetros del algoritmo.

Fuente: Orellano, Luis (2020)

Finalmente, Excel informará el tiempo que tardó el seudocódigo propuesto en encontrar la solución al problema. (Ver Figura 28)

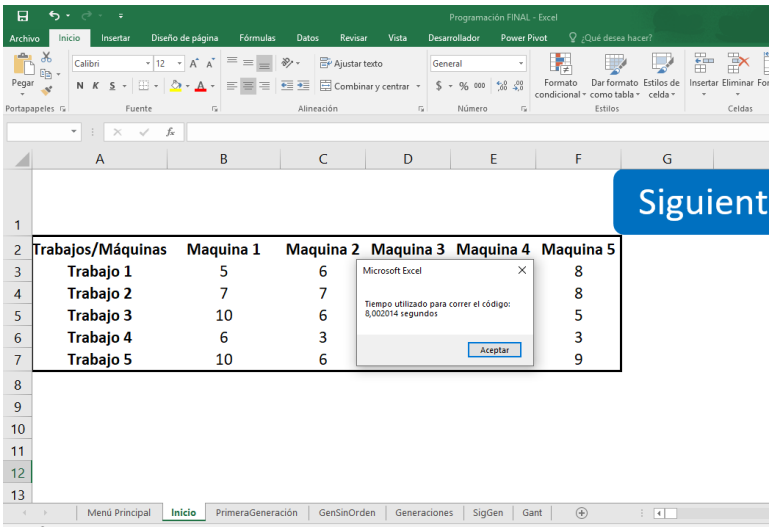


Figura 28. Finalizado del proceso.
Fuente: Orellano, Luis (2020)

En consonancia con lo anterior, y después de haber seleccionado la opción de “Aceptar”, se desplegará en otra hoja de cálculo un diagrama de Gantt que le indica al usuario el orden de secuenciación de los trabajos en las máquinas. Además, en la parte superior se detallará el tiempo unitario que consume cada actividad hasta concluir el total del ciclo programado. Sin embargo, existen unas pestañas intermedias antes de llegar al diagrama, las cuales mostrarán el desarrollo del modelo. Por consiguiente, a continuación, se aborda la caracterización de cada parte del procedimiento.

- **Primera generación:** En esta pestaña se encuentra la primera población de individuos completamente aleatorizados y organizados según su respectivo *Makespan*, como se observa en la figura 29.

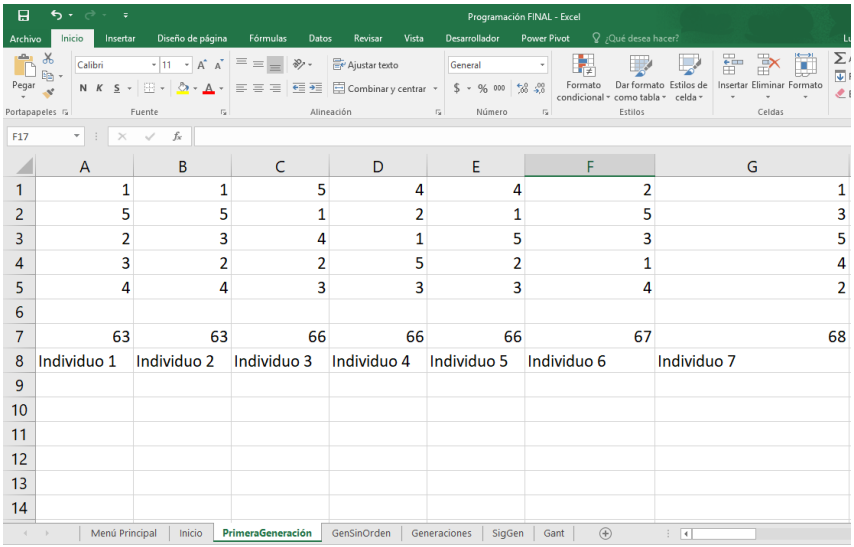


Figura 29.Primera generación.

Fuente: Orellano, Luis (2020)

Por otro lado, al final de la información anterior se ubican las parejas conformadas con su respectivo valor de afinidad con el antígeno seleccionado. De allí se escoge la pareja mejor relacionada con el antígeno. (Ver Figura 30)

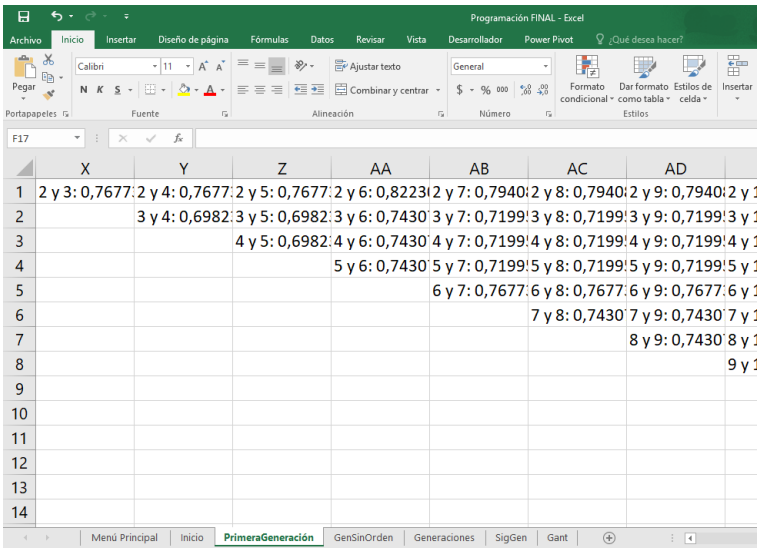
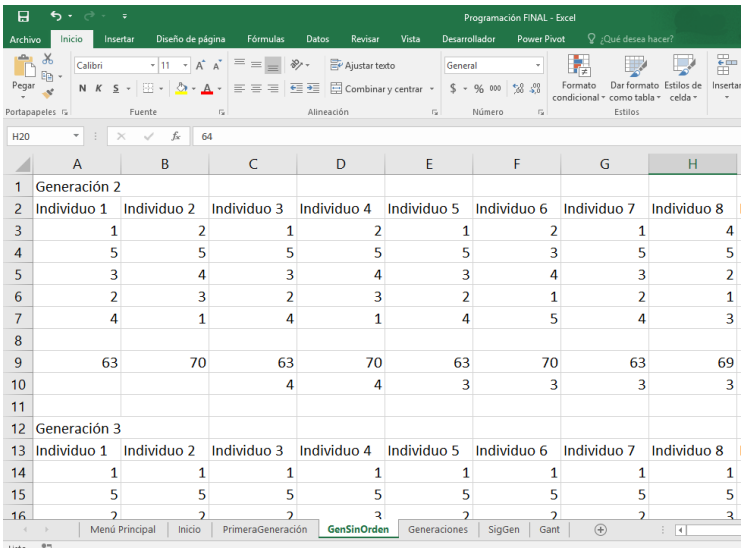


Figura 30. Primera generación 2.

Fuente: Orellano, Luis (2020)

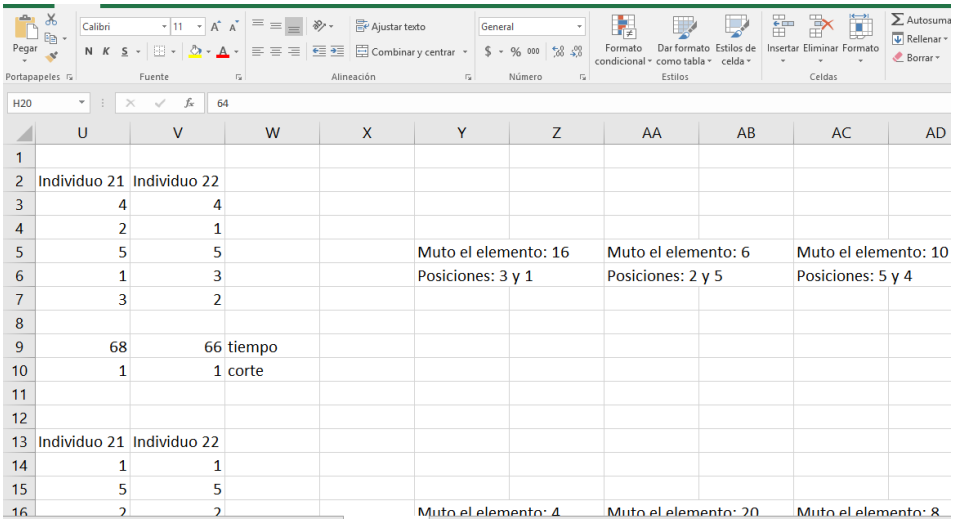
- **GenSinOrden:** En esta pestaña se encuentran todas las generaciones creadas sin la jerarquización de acuerdo al tiempo total de procesamiento. (Ver Figura 31)



	A	B	C	D	E	F	G	H
1	Generación 2							
2	Individuo 1	Individuo 2	Individuo 3	Individuo 4	Individuo 5	Individuo 6	Individuo 7	Individuo 8
3	1	2	1	2	1	2	1	4
4	5	5	5	5	5	3	5	5
5	3	4	3	4	3	4	3	2
6	2	3	2	3	2	1	2	1
7	4	1	4	1	4	5	4	3
8								
9	63	70	63	70	63	70	63	69
10			4	4	3	3	3	3
11								
12	Generación 3							
13	Individuo 1	Individuo 2	Individuo 3	Individuo 4	Individuo 5	Individuo 6	Individuo 7	Individuo 8
14	1	1	1	1	1	1	1	1
15	5	5	5	5	5	5	5	5
16	2	2	2	3	2	2	2	3

Figura 31. GenSinOrden.
Fuente: Orellano, Luis (2020)

También cuenta con los puntos de cruce que se tomaron para originar a cada individiduo, al igual que el registro de los cromosomas mutados con las posiciones intercambiadas. (Ver Figura 32)

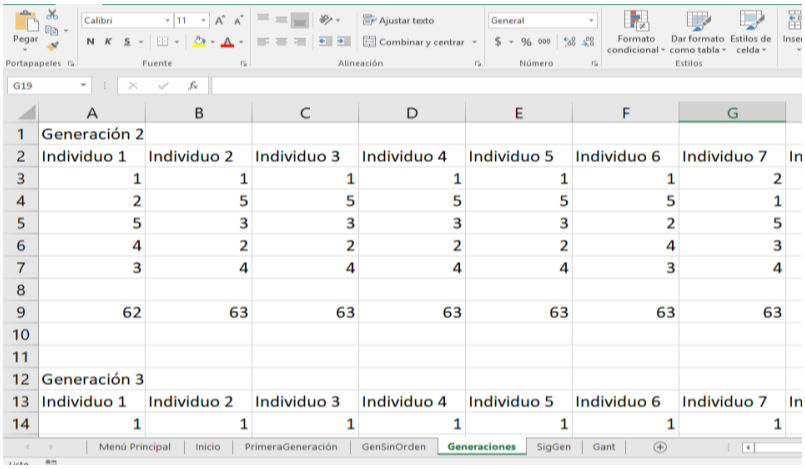


	U	V	W	X	Y	Z	AA	AB	AC	AD
1										
2	Individuo 21	Individuo 22								
3	4	4								
4	2	1								
5	5	5			Muto el elemento: 16		Muto el elemento: 6		Muto el elemento: 10	
6	1	3			Posiciones: 3 y 1		Posiciones: 2 y 5		Posiciones: 5 y 4	
7	3	2								
8										
9	68	66	tiempo							
10	1	1	corte							
11										
12										
13	Individuo 21	Individuo 22								
14	1	1								
15	5	5								
16	2	2			Muto el elemento: 4		Muto el elemento: 20		Muto el elemento: 8	

Figura 32. GenSinOrden 2.

Fuente: Orellano, Luis (2020)

- **Generaciones:** Para esta etapa se organizan los individuos de acuerdo al *Makespan*, ya que con ello se puede escoger el antígeno y calcular la afinidad que determinará los nuevos padres de la siguiente generación. (Ver Figura 33)



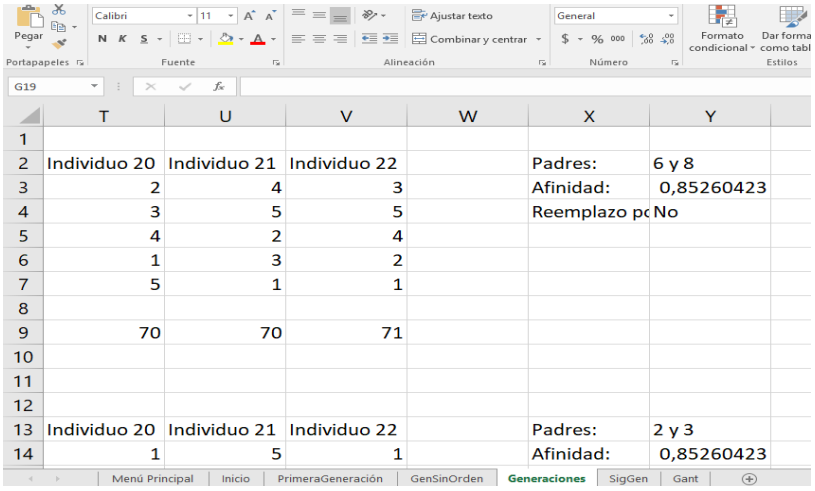
	A	B	C	D	E	F	G
1	Generación 2						
2	Individuo 1	Individuo 2	Individuo 3	Individuo 4	Individuo 5	Individuo 6	Individuo 7
3	1	1	1	1	1	1	2
4	2	5	5	5	5	5	1
5	5	3	3	3	3	2	5
6	4	2	2	2	2	4	3
7	3	4	4	4	4	3	4
8							
9	62	63	63	63	63	63	63
10							
11							
12	Generación 3						
13	Individuo 1	Individuo 2	Individuo 3	Individuo 4	Individuo 5	Individuo 6	Individuo 7
14	1	1	1	1	1	1	1

Figura 33. Generaciones.

Fuente: Orellano, Luis (2020)

Las casillas denominadas como “Reemplazo por antígeno anterior”, indican en la casilla adyacente si se mantuvo el antígeno de la generación inmediatamente

anterior, o si de lo contrario, el de la nueva es mejor y por ende será el pivote para el cálculo de la afinidad. Además, señala los individuos que serán los padres de la siguiente descendencia. (Ver Figura 34)



	T	U	V	W	X	Y
1						
2	Individuo 20	Individuo 21	Individuo 22		Padres:	6 y 8
3	2	4	3		Afinidad:	0,85260423
4	3	5	5		Reemplazo pc	No
5	4	2	4			
6	1	3	2			
7	5	1	1			
8						
9	70	70	71			
10						
11						
12						
13	Individuo 20	Individuo 21	Individuo 22		Padres:	2 y 3
14	1	5	1		Afinidad:	0,85260423

Figura 34. Generaciones 2.
Fuente: Orellano, Luis (2020)

- **SigGen:** En esta hoja de cálculo se encuentra la última generación organizada en función del *Makespan* de los individuos y seguido de esto, las parejas que se conformarían para una ocasional siguiente generación. Cabe señalar que, el primer individuo será el mejor cromosoma de todas las generaciones, ya que se mantuvo al compararse con los distintos antígenos de cada población. (Ver Figura 35)

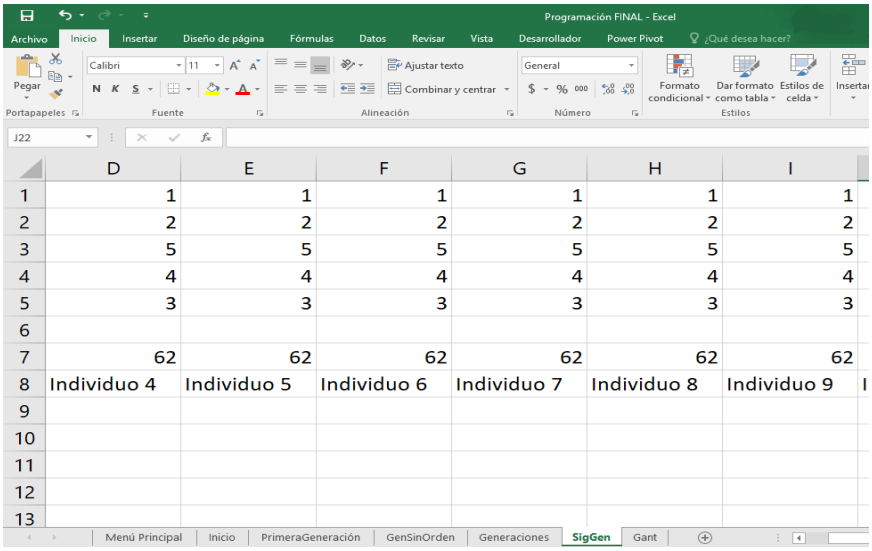


Figura 35. SigGen.

Fuente: Orellano, Luis (2020)

En conclusión, se grafica el Diagrama de Gantt para la asignación de los trabajos de acuerdo a la mejor secuenciación encontrada. (Ver Figura 36)

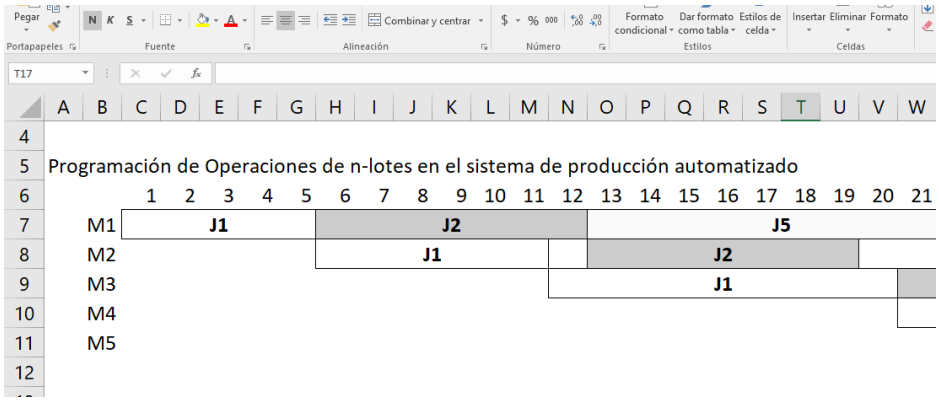


Figura 36. Diagrama de Gantt Final.

Fuente: Orellano, Luis (2020)

Por último, se aclara que para efectos de implementación por parte de un usuario que no conozca del tema, habrá un instructivo junto al programa el cual tendrá la información básica para entender su funcionamiento. (Ver Anexo H)

14. Resultados y discusión

Para validar el desempeño de la codificación, se utilizaron datos planteados por distintos autores y se compararon los resultados conseguidos a través de la codificación formulada. La información numérica se obtuvo por medio de un seudocódigo hecho en el lenguaje de programación Visual Basic ®, y ejecutado en un computador con procesador Intel Core i5-7200, 2.50 GHz y 8 Gb de memoria RAM.

Las instancias a implementar para validar el algoritmo genético de Sistema Inmune fueron enunciadas por Carlier (1978), Heller (1969) y Reeves (1995) para problemas de secuenciación de tipo flow shop.

Inicialmente, por medio de la batería de datos desarrollados en los trabajos previamente mencionados, se procede a confrontar los resultados obtenidos, para así, verificar la calidad de respuesta arrojada y el tiempo computacional de la programación propuesta en la presente investigación.

En consecuencia, se ejecutó primeramente la modificación de los parámetros fijos de entrada, con base en lo requerido por el modelo abordado en la sección de Flow Shop Scheduling. Seguidamente se completó la matriz de tiempos y se procedió a indicar los niveles por cada factor que se explica seguido al apartado de variación del algoritmo.

15.1. Validación del algoritmo de Sistema Inmune

La evaluación del algoritmo se origina a partir de un diseño de experimentos de carácter factorial 2^3 , ya que con este procedimiento se pueden analizar los tres factores (Número de individuos por generación, número de generaciones y porcentaje de los individuos a mutar) dados a ajustar en la ejecución del programa, con dos niveles de prueba. Cabe

señalar que, a diferencia del diseño por bloques, el implementado aborda los factores con igual relevancia al desarrollarlo. De hecho, según Gutiérrez & De la Vara (2012) el diseño factorial de cualquier índole es el más indicado a la hora buscar que la variable de salida deseada sea la mejor para los parámetros indicados. No obstante, podría haberse trabajado el diseño 3^3 pero este aumentaría el tamaño de la matriz factorial y por ende, su complejidad. Por otro lado, no se decidió fraccionar por grupos de factores ya que no son suficientes y se perderían datos que aporten información valiosa a la investigación. Concluido lo anterior, a continuación, se describen los factores inmersos en la programación junto con sus respectivos niveles evaluados.

15.1.1. Factores

Los factores sobre los que se evaluó la programación, son los principales parámetros a modificar dentro del laboratorio teórico de la secuenciación de máquinas en paralelo, solucionado en el marco del algoritmo genético de Sistema Inmune. Estos son: Número de generaciones, número de individuos y porcentaje de mutación. No obstante, para cada ítem señalado, se definieron niveles máximos y mínimos adecuados para la presente prueba. Cabe aclarar que, a la hora de determinar los valores, se fijaban dos de los tres factores y con ello se iteraba la programación, evaluando su comportamiento hacia el mejor estado de las variables de respuesta. A continuación, se presentan los valores escogidos y el detalle del diseño factorial 2^3, en el cual, se realizaron 8 réplicas para cada tratamiento y en todas las instancias evaluadas. (Ver tabla 7 y 8)

Tabla 7
Diseño factorial 2^3

Factores	Nombre	Niveles	
		Bajo	Alto
A	Número de individuos	12	22
B	Número de generaciones	12	26
C	Porcentaje a mutar	10	34

Fuente: Orellano, Luis (2020)

Tabla 8
Diseño factorial 2^3*con tratamientos*

	A	B	C
Tratamiento 1	12	12	10
Tratamiento 2	26	12	10
Tratamiento 3	12	22	10
Tratamiento 4	26	22	10
Tratamiento 5	12	12	34
Tratamiento 6	26	12	34
Tratamiento 7	12	22	34
Tratamiento 8	26	22	34

Fuente: Orellano, Luis (2020)

15.2. Análisis de resultados

En consecuencia, se procedió a estudiar el comportamiento de cada instancia dentro de un marco estadístico. Para ello se utilizó el software Minitab 18 ® en el que se calcularon todos los indicadores pertinentes teniendo como resultado lo siguiente:

15.2.1. Instancia No.1 de Carlier (1978)

En primer lugar, y tal como lo indica la figura 37 y la tabla 9, se observa que ninguno de los factores o interacciones entre los mismos, son significativos para el modelo.

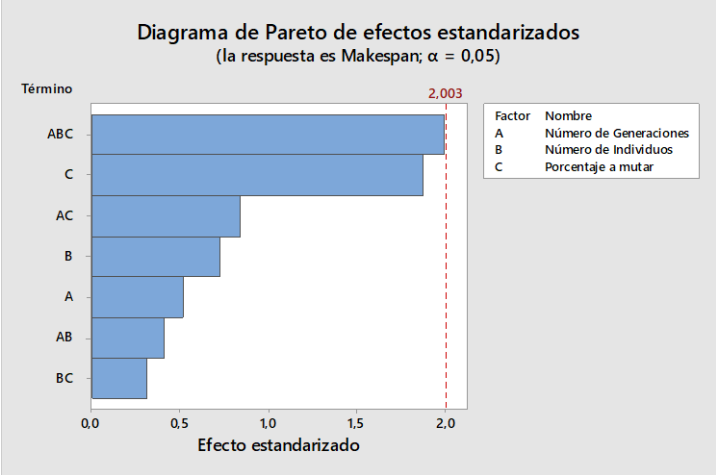


Figura 37. Pareto Car01.

Fuente: Orellano, Luis (2020)

Tabla 9
ANOVA Car01

Análisis de varianza					
Fuente	**GL**	**SC Ajust.**	**MC Ajust.**	**Valor F**	**Valor p**
Modelo	7	637251	91036	1,32	0,258
Lineal	3	296619	98873	1,43	0,243
Número de Generaciones	1	18428	18428	0,27	0,607
Número de Individuos	1	36864	36864	0,53	0,468
Porcentaje a mutar	1	241327	241327	3,50	0,067
Interacciones de 2 términos	3	67103	22368	0,32	0,808
Número de Generaciones*Número de Individuos	1	11718	11718	0,17	0,682
Número de Generaciones*Porcentaje a mutar	1	48620	48620	0,70	0,405
Número de Individuos*Porcentaje a mutar	1	6765	6765	0,10	0,755
Interacciones de 3 términos	1	273529	273529	3,97	0,051
Número de Generaciones*Número de Individuos*Porcentaje a mutar	1	273529	273529	3,97	0,051
Error	56	3862799	68979		
Total	63	4500050			

Fuente: Orellano, Luis (2020)

Además, según la figura 38, se podría afirmar que el *Makespan* se minimiza cuando el porcentaje a mutar se encuentra en su mayor nivel.

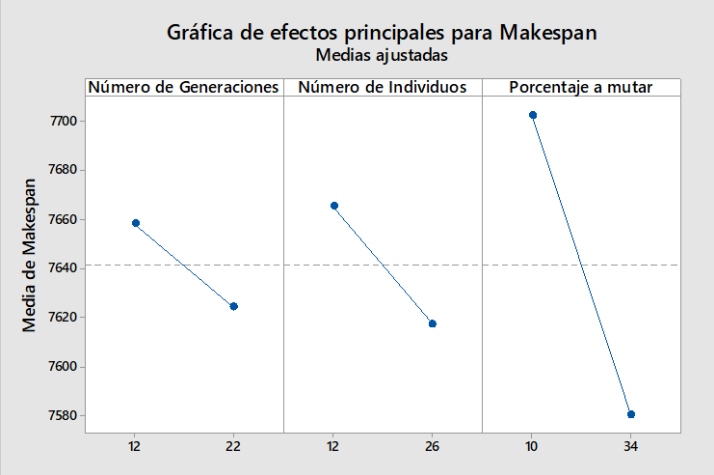


Figura 38. Efectos Car01.

Fuente: Orellano, Luis (2020)

También, es importante mencionar que de acuerdo a la figura 39, la mejor combinación para lograr lo anterior se registra de la siguiente forma: Número de Generaciones (22), Número de Individuos (12) y Porcentaje de mutar (34).

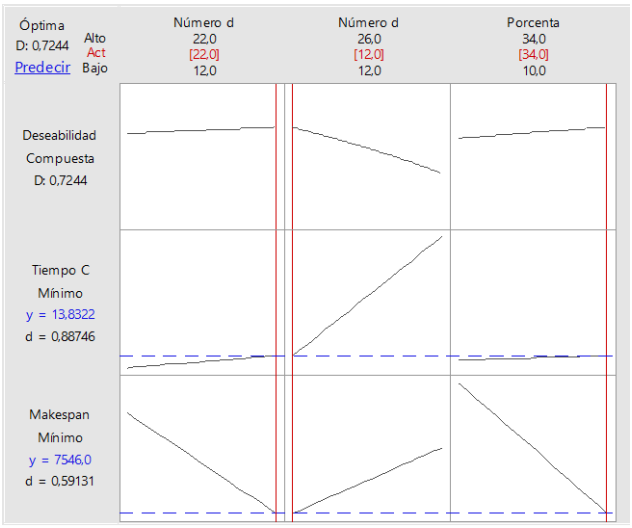


Figura 39. Optimización Car 01.

Fuente: Orellano, Luis (2020)

15.2.2. Instancia No.5 de Carlier (1978)

Para esta instancia, se detalla que el número de individuos y el porcentaje a mutar, son los factores estadísticamente significativos que más pueden influenciar a la consecución de un mejor valor de *Makespan*. (*Ver* tabla 10 y figura 40)

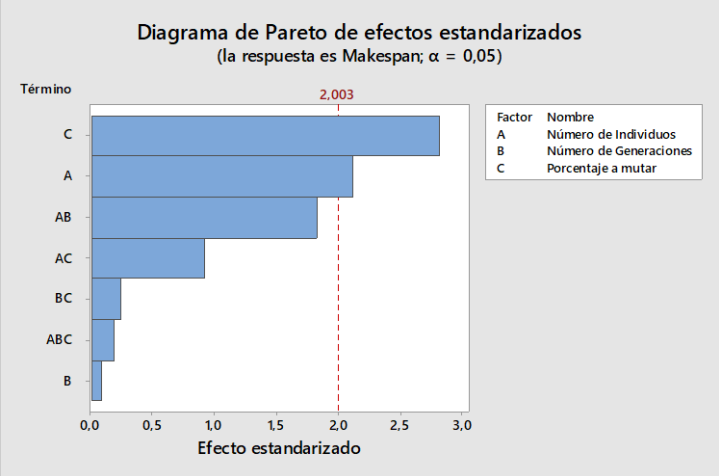


Figura 40. Pareto Car 05.

Fuente: Orellano, Luis (2020)

Tabla 10
ANOVA Car 05

Análisis de varianza					
Fuente	**GL**	**SC Ajust.**	**MC Ajust.**	**Valor F**	**Valor p**
Modelo	7	939655	134236	2,38	0,033
Lineal	3	700008	233336	4,14	0,010
Número de Individuos	1	253890	253890	4,51	0,038
Número de Generaciones	1	395	395	0,01	0,934
Porcentaje a mutar	1	445723	445723	7,92	0,007
Interacciones de 2 términos	3	237700	79233	1,41	0,250
Número de Individuos*Número de Generaciones	1	187164	187164	3,32	0,074
Número de Individuos*Porcentaje a mutar	1	47470	47470	0,84	0,362

Número de Generaciones*Porcentaje a mutar	1	3066	3066	0,05	0,816
Interacciones de 3 términos	1	1947	1947	0,03	0,853
Número de Individuos*Número de Generaciones*Porcentaje a mutar	1	1947	1947	0,03	0,853
Error	56	3152481	56294		
Total	63	4092136			

Fuente: Orellano, Luis (2020)

No obstante, se podría afirmar que, en concordancia con mayores niveles de estos factores, la tendencia será a minimizar la variable de respuesta. Por otro lado, las interacciones señaladas entre los mismos no son de mayor relevancia para el comportamiento de la simulación. (Ver Figura 41)

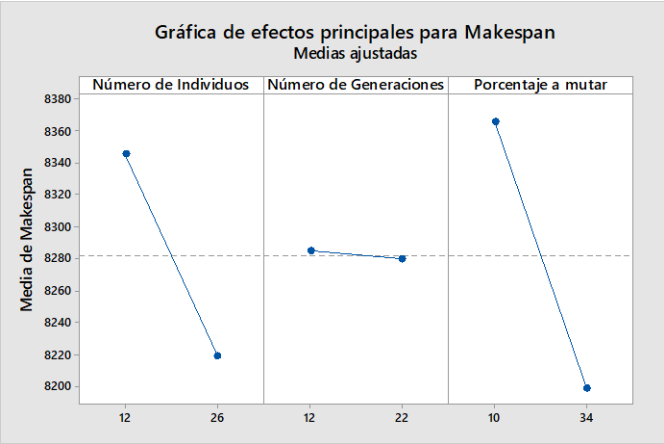


Figura 41. Efectos Car 05.

Fuente: Orellano, Luis (2020)

Finalmente, y según la figura 42, la mejor combinación que minimiza el *Makespan* y tiempo computacional es: Número de individuos (12), Número de Generaciones (22) y Porcentaje a mutar (34).

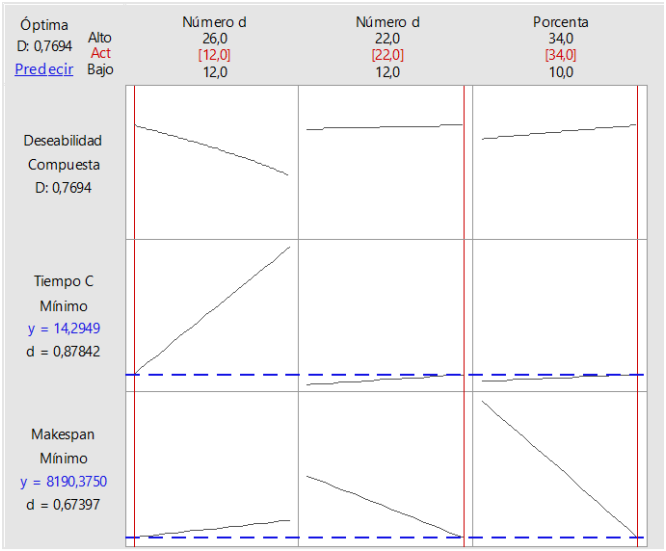


Figura 42. Optimización Car 05.

Fuente: Orellano, Luis (2020)

15.2.3. Instancia No.6 de Carlier (1978)

En este caso se analizó que ninguno de los factores o interacciones logra ser estadísticamente significativos para el modelo. De hecho, los valores no son cercanos a ser determinantes para el *Makespan*. (Ver figura 43 y tabla 11)

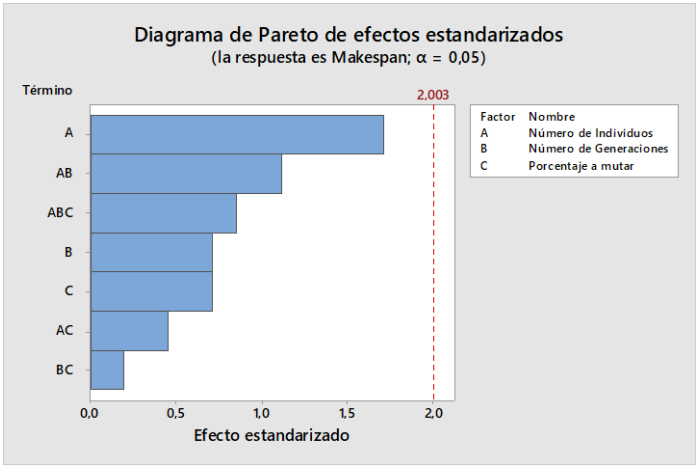


Figura 43. Pareto Car 06.

Fuente: Orellano, Luis (2020)

Tabla 11
ANOVA Car 06

Análisis de varianza					
Fuente	**GL**	**SC Ajust.**	**MC Ajust.**	**Valor F**	**Valor p**
Modelo	7	459706	65672	0,88	0,527
Lineal	3	293731	97910	1,31	0,279
Número de Individuos	1	217972	217972	2,93	0,093
Número de Generaciones	1	38074	38074	0,51	0,478
Porcentaje a mutar	1	37685	37685	0,51	0,480
Interacciones de 2 términos	3	111278	37093	0,50	0,685
Número de Individuos*Número de Generaciones	1	93101	93101	1,25	0,268
Número de Individuos*Porcentaje a mutar	1	15407	15407	0,21	0,651
Número de Generaciones*Porcentaje a mutar	1	2769	2769	0,04	0,848
Interacciones de 3 términos	1	54698	54698	0,73	0,395
Número de Individuos*Número de Generaciones*Porcentaje a mutar	1	54698	54698	0,73	0,395
Error	56	4172488	74509		
Total	63	4632194			

Fuente: Orellano, Luis (2020)

Sin embargo, y en consonancia con la figura 44, se observa que el comportamiento del *Makespan* cambia considerablemente de acuerdo al cambio de los niveles mínimos y máximos de cada factor, por ende, al aumentar cada valor se logra una mejor respuesta.

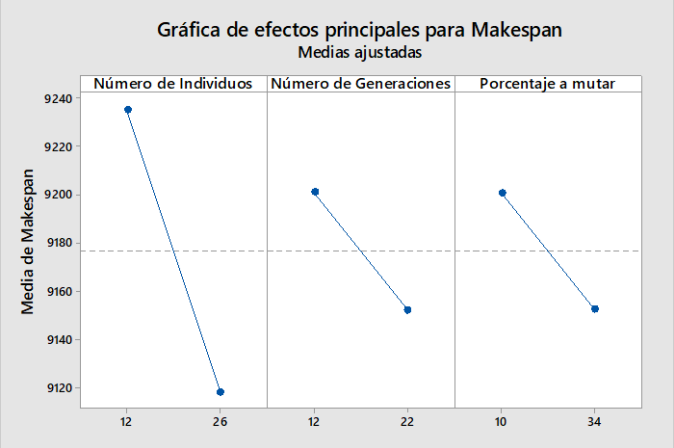


Figura 44. Efectos Car 06.

Fuente: Orellano, Luis (2020)

Por último, la mejor combinación para alcanzar el mínimo tiempo computacional y *Makespan* es: Número de Individuos (26), Número de Generaciones (12) y Porcentaje a mutar (10). (Ver figura 45)

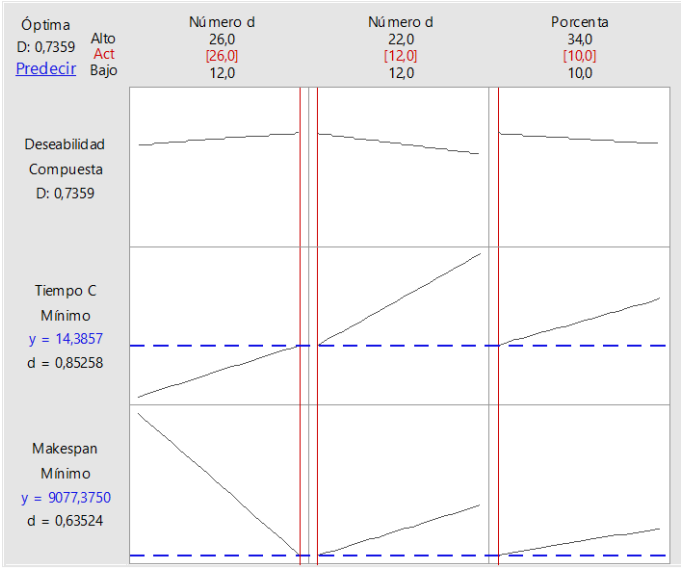


Figura 45. Optimización Car 06.

Fuente: Orellano, Luis (2020)

15.2.4. Instancia No.2 de Heller (1960)

En esta instancia se detalla que el número de individuos es el único factor que es estadísticamente significativo o que tiene efecto sobre el modelo. (Ver figura 46 y tabla 12)

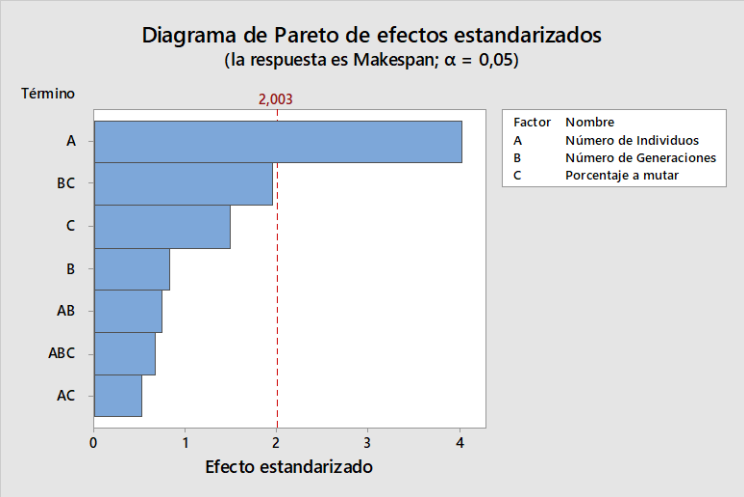


Figura 46. Pareto Hel 02.

Fuente: Orellano, Luis (2020)

Tabla 12
ANOVA Hel 02

Análisis de varianza					
Fuente	**GL**	**SC Ajust.**	**MC Ajust.**	**Valor F**	**Valor p**
Modelo	7	271,437	38,777	3,47	0,004
Lineal	3	214,812	71,604	6,40	0,001
Número de Individuos	1	182,250	182,250	16,29	0,000
Número de Generaciones	1	7,563	7,563	0,68	0,414
Porcentaje a mutar	1	25,000	25,000	2,23	0,141
Interacciones de 2 términos	3	51,562	17,187	1,54	0,215
Número de Individuos*Número de Generaciones	1	6,250	6,250	0,56	0,458
Número de Individuos*Porcentaje a mutar	1	3,062	3,062	0,27	0,603
Número de Generaciones*Porcentaje a mutar	1	42,250	42,250	3,78	0,057
Interacciones de 3 términos	1	5,062	5,062	0,45	0,504
Número de Individuos*Número de Generaciones*Porcentaje a mutar	1	5,062	5,062	0,45	0,504
Error	56	626,500	11,188		
Total	63	897,938			

Fuente: Orellano, Luis (2020)

Luego al alcanzar el mayor valor del mismo factor, se tiene gran relevancia sobre la variable de respuesta analizada. (Ver figura 47)

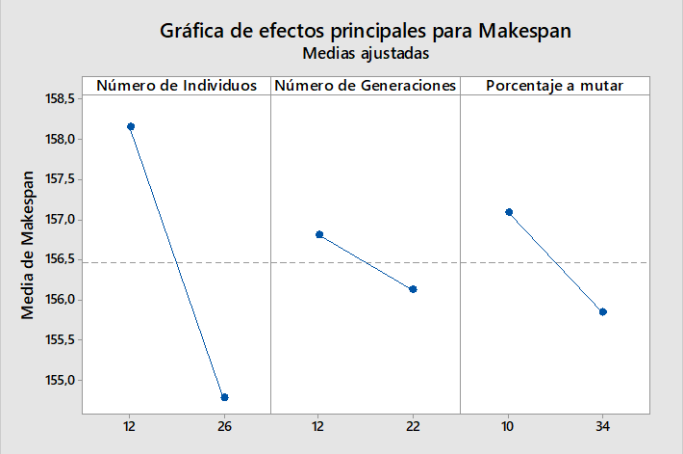


Figura 47. Efectos Hel 02.

Fuente: Orellano, Luis (2020)

Finalmente, la mejor composición de los parámetros de entrada se puede tener de la siguiente forma según la figura 48: Número de Individuos (26), Número de Generaciones (12) y Porcentaje de mutar (10).

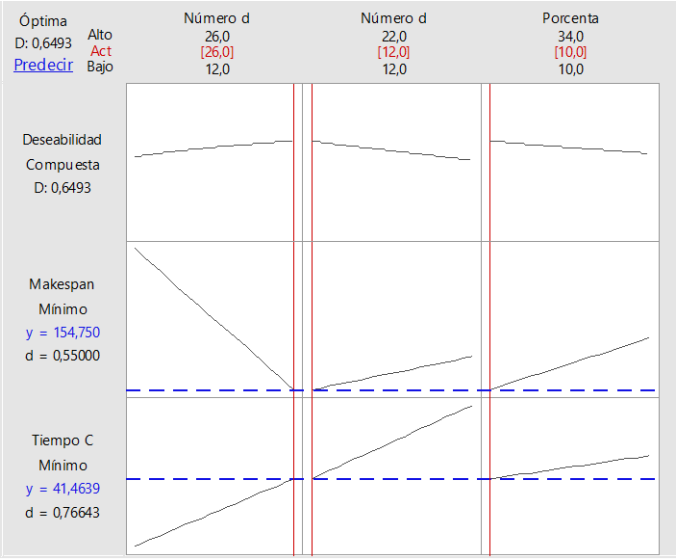


Figura 48. Optimización Hel 02.

Fuente: Orellano, Luis (2020)

15.2.5. Instancia No.7 de Reeves (1995)

En este caso se encontró que ninguno de los factores modificables es de relevancia estadística o cumplen con un efecto importante sobre el modelo desarrollado. De hecho, contrario al resto de instancias, los valores de p son los más alejados a indicar que según su cambio haya relación con el comportamiento del *Makespan.* (Ver figura 49 y tabla 13)

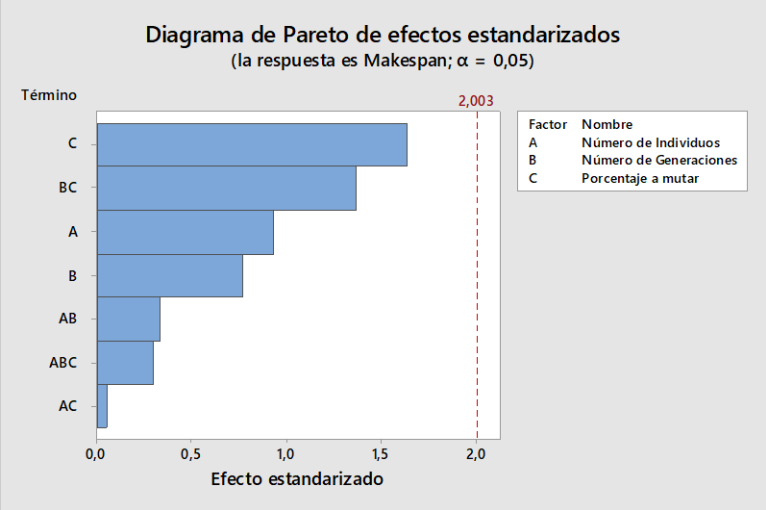


Figura 49. Pareto Rec 07.

Fuente: Orellano, Luis (2020)

Tabla 13
ANOVA Rec07

Análisis de varianza					
Fuente	**GL**	**SC Ajust.**	**MC Ajust.**	**Valor F**	**Valor p**
Modelo	7	17957	2565,36	0,89	0,523
Lineal	3	11966	3988,75	1,38	0,259
Número de Individuos	1	2500	2500,00	0,86	0,357
Número de Generaciones	1	1722	1722,25	0,60	0,444
Porcentaje a mutar	1	7744	7744,00	2,68	0,107
Interacciones de 2 términos	3	5735	1911,75	0,66	0,580

Número de Individuos*Número de Generaciones	1	324	324,00	0,11	0,739
Número de Individuos*Porcentaje a mutar	1	9	9,00	0,00	0,956
Número de Generaciones*Porcentaje a mutar	1	5402	5402,25	1,87	0,177
Interacciones de 3 términos	1	256	256,00	0,09	0,767
Número de Individuos*Número de Generaciones*Porcentaje a mutar	1	256	256,00	0,09	0,767
Error	56	161989	2892,67		
Total	63	179947			

Fuente: Orellano, Luis (2020)

No obstante, el porcentaje a mutar es el factor que más puede incidir sobre las condiciones óptimas requeridas según lo indica la figura 50.

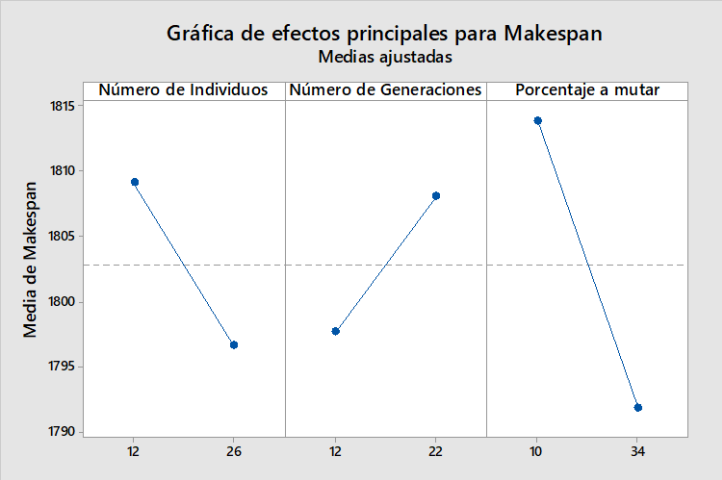


Figura 50. Efectos Rec 07.

Fuente: Orellano, Luis (2020)

Para lo cual, se determinan los niveles que minimizan el *Makespan* y tiempo computacional de acuerdo a los siguientes valores obtenidos en la figura 51: Número de Individuos (12), Número de Generaciones (12) y Porcentaje a mutar (10).

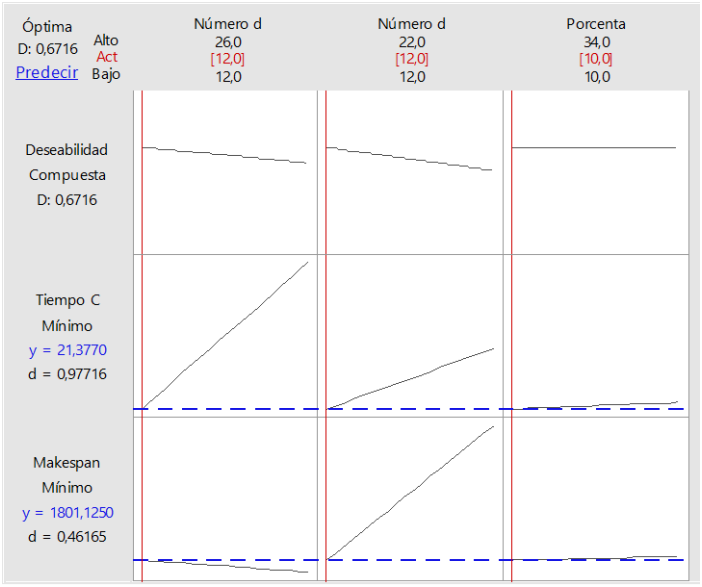


Figura 51. Optimización Rec 07.

Fuente: Orellano, Luis (2020)

15.2.6. Instancia No.13 de Reeves (1995)

Para esta instancia se observa que el número de individuos y la interacción del número de generaciones y el porcentaje a mutar, son los factores de mayor efecto sobre la variable de respuesta. (Ver tabla 14 y figura 52)

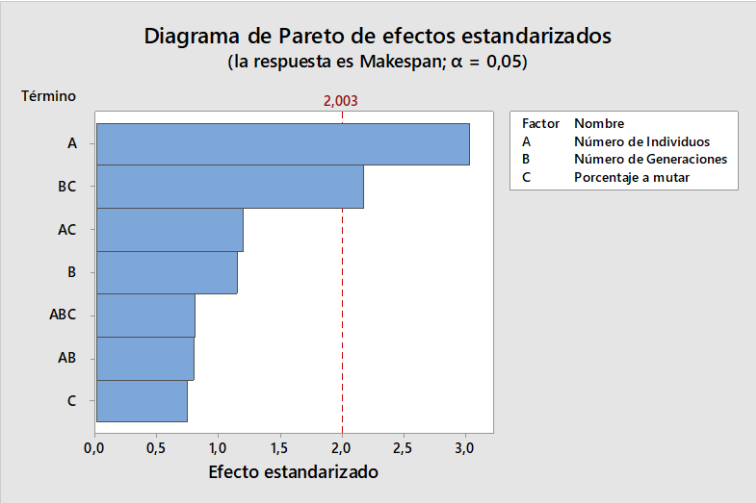


Figura 52. Pareto Rec 13.

Fuente: Orellano, Luis (2020)

Tabla 14
ANOVA Rec 13

Análisis de varianza					
Fuente	**GL**	**SC Ajust.**	**MC Ajust.**	**Valor F**	**Valor p**
Modelo	7	31262	4466,0	2,66	0,019
Lineal	3	18725	6241,8	3,72	0,017
Número de Individuos	1	15563	15562,6	9,26	0,004
Número de Generaciones	1	2233	2232,6	1,33	0,254
Porcentaje a mutar	1	930	930,3	0,55	0,460
Interacciones de 2 términos	3	11447	3815,8	2,27	0,090
Número de Individuos*Número de Generaciones	1	1056	1056,3	0,63	0,431
Número de Individuos*Porcentaje a mutar	1	2426	2425,6	1,44	0,235
Número de Generaciones*Porcentaje a mutar	1	7966	7965,6	4,74	0,034
Interacciones de 3 términos	1	1089	1089,0	0,65	0,424
Número de Individuos*Número de Generaciones*Porcentaje a mutar	1	1089	1089,0	0,65	0,424
Error	56	94079	1680,0		
Total	63	125341			

Fuente: Orellano, Luis (2020)

Además, según la figura 53, el primer ítem mencionado es capaz de acercar la simulación al mínimo valor de *Makespan*, estando en su mayor nivel.

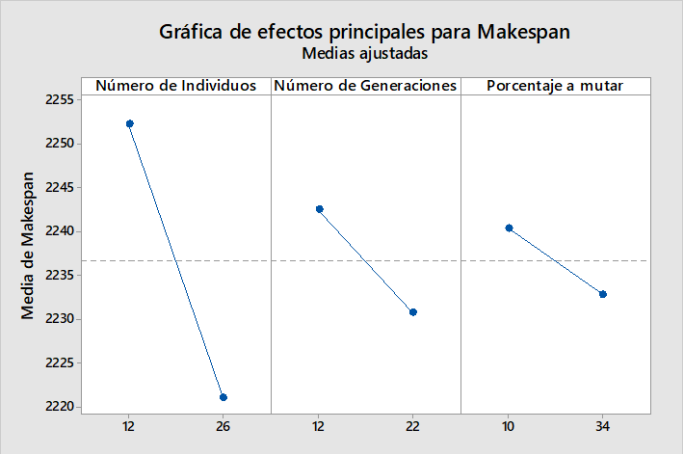


Figura 53. Efectos Rec 13.

Fuente: Orellano, Luis (2020)

Por otro lado, la mejor combinación de niveles que minimiza el *Makespan* y tiempo computacional se radica en los siguientes valores: Número de individuos (16), Número de Generaciones (12) y Porcentaje a mutar (34). (Ver figura 54)

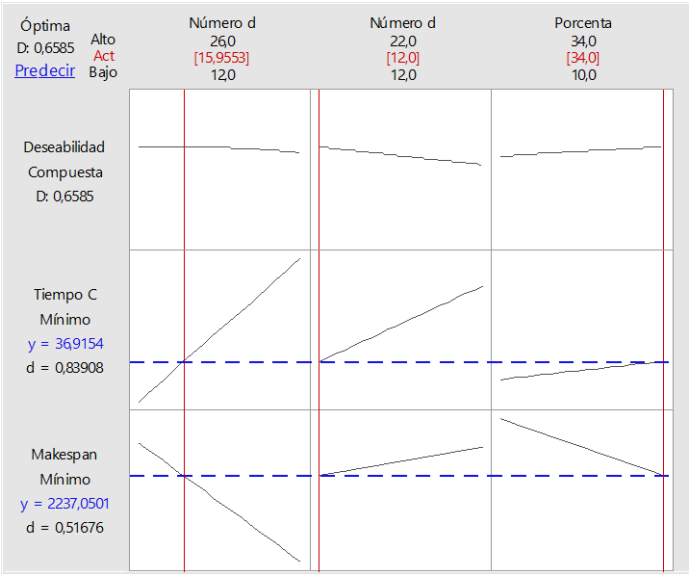


Figura 54. Optimización Rec 13.

Fuente: Orellano, Luis (2020)

15.2.7. Instancia No.21 de Reeves (1995)

En este diseño se obtuvo que el número de individuos es el único factor que incide ampliamente en el desarrollo de la simulación. Lo anterior se considera como una tendencia marcada en relación al resto de instancias. (Ver figura 55 y tabla 15)

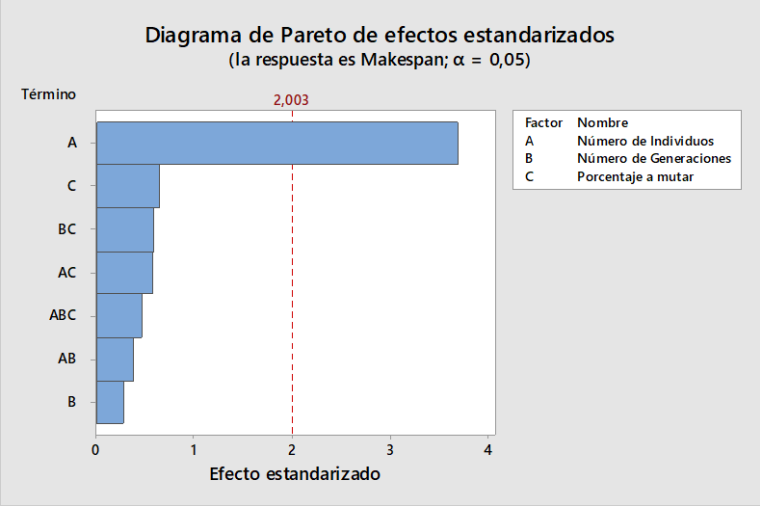


Figura 55. Parero Rec 21.

Fuente: Orellano, Luis (2020)

Tabla 15
ANOVA Rec 21

Análisis de varianza					
Fuente	**GL**	**SC Ajust.**	**MC Ajust.**	**Valor F**	**Valor p**
Modelo	7	32821	4688,7	2,17	0,051
Lineal	3	30578	10192,7	4,71	0,005
Número de Individuos	1	29541	29541,0	13,65	0,001
Número de Generaciones	1	159	159,4	0,07	0,787
Porcentaje a mutar	1	878	877,6	0,41	0,527
Interacciones de 2 términos	3	1765	588,2	0,27	0,846
Número de Individuos*Número de Generaciones	1	320	319,5	0,15	0,702

Número de Individuos*Porcentaje a mutar	1	696	695,6	0,32	0,573
Número de Generaciones*Porcentaje a mutar	1	749	749,4	0,35	0,559
Interacciones de 3 términos	1	479	478,5	0,22	0,640
Número de Individuos*Número de Generaciones*Porcentaje a mutar	1	479	478,5	0,22	0,640
Error	56	121225	2164,7		
Total	63	154046			

Fuente: Orellano, Luis (2020)

También, es fácilmente perceptible que gracias a su máximo nivel se puede registrar el mínimo *Makespan.* Por otra parte, los otros dos factores principales no aportan influencia en la disminución de la variable de respuesta, es casi nula. (Ver Figura 56)

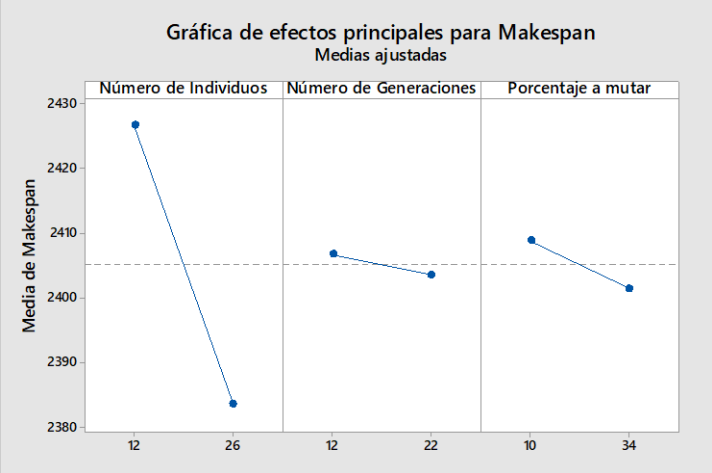


Figura 56. Efectos Rec 21.

Fuente: Orellano, Luis (2020)

Finalmente, las condiciones óptimas para ejecutar la simulación con base en esta instancia son: Número de Individuos (26), Número de Generaciones (12) y Porcentaje a Mutar (34). (Ver figura 57)

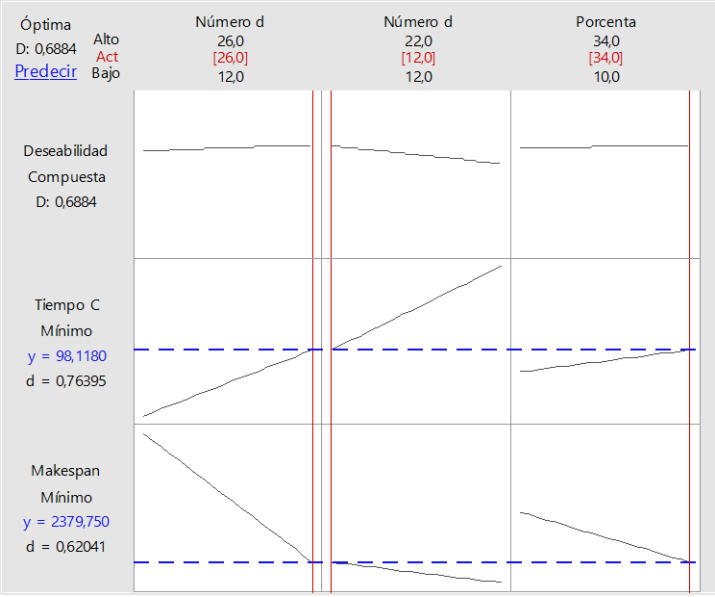


Figura 57. Optimización Rec 21.

Fuente: Orellano, Luis (2020)

15.3. Resultados de la validación para las instancias de Carlier (1978) en secuenciación de tipo flow shop

Tabla 16
Resumen de instancias de Carlier

Instancia	Trabajos	Máquinas	C_{max} Previo	C_{max} Obtenido	C_{max} Promedio	%Error con el obtenido	%Error con el promedio	Tiempo computacional Obtenido [s]	Tiempo computacional promedio[s]
Car1	11	5	7038	7038	7641,43	0%	8,57%	35,43	22,61
Car5	10	6	7720	7835	8282,20	1,49%	5,71%	26,21	23,51
Car6	8	9	8505	8570	9176,54	0,76%	7,90%	27,67	16,75

Fuente: Orellano, Luis (2020)

En este caso el rendimiento del algoritmo fue acertado, ya que tuvo un leve margen de exactitud frente al valor óptimo. Por otro lado, el tiempo computacional presentó similar

comportamiento entre la solución para el valor esperado y el promedio de todas las soluciones en cada instancia. (Ver tabla 16)

15.4. Resultados de la validación para la instancia de Heller (1960) en secuenciación de tipo flow shop

Tabla 17
Resumen de instancia de Heller

Instancia	Trabajos	Máquinas	C_{max} Previo	C_{max} Obtenido	C_{max} Promedio	%Error con el obtenido	%Error con el promedio	Tiempo computacional Obtenido [s]	Tiempo computacional promedio[s]
Hel2	20	10	154	148	156,47	0%	0%	65,18	43,30

Fuente: Orellano, Luis (2020)

Para esta instancia, se encontró un mejor valor que el encontrado por el autor y con una significativa diferencia respecto al mismo. Es relevante dicha distancia ya que el valor definido por Heller (1960) se presenta como moda dentro de las simulaciones realizadas, y donde las restantes son cercanas a este valor. Por otra parte, el tiempo computacional incrementó ampliamente debido a la cantidad de información suministrada. (Ver tabla 17)

15.5. Resultados de la validación para las instancias de Reeves (1995) en secuenciación de tipo flow shop

Tabla 18
Resumen de instancias de Reeves

Instancia	Trabajos	Máquinas	C_{max} Previo	C_{max} Obtenido	C_{max} Promedio	%Error con el obtenido	%Error con el promedio	Tiempo computacional Obtenido [s]	Tiempo computacional promedio[s]
Rec7	20	10	1626	1666	1802,88	7,49%	11,72%	28,01	67,03
Rec13	20	15	2002	2152	2236,63	11,50%	15,89%	134,78	59,59
Rec21	30	10	2167	2297	2405,11	6,00%	10,99%	73,12	84,48

Fuente: Orellano, Luis (2020)

Para estas pruebas el margen de exactitud en el resultado aumentó proporcionalmente al procesamiento de más datos. No obstante, la diferencia podría estar en un límite tolerado dentro de la región factible de respuestas. Respecto a los tiempos computacionales, el promedio en general se mantuvo exceptuando las mejores respuestas en la instancia Rec13, donde para estos valores se consumió un mayor tiempo hallando la respuesta. Cabe aclarar que, la instancia Rec7 y Rec13 fue comparada con la técnica metaheurística propuesta por Ancau (2012). En el caso de Rec21 se utilizaron los valores hallados por Kumar (2011). Lo anterior fue necesario porque en el documento de Reeves (1995) no se mostraron los resultados para esas instancias.

15. Conclusiones

La programación de trabajos con máquinas en paralelo, para empresas productivas tipo flow shop, se estudió desde la aplicación de la técnica metaheurística del algoritmo de Sistema Inmune. Este a su vez, restringido por la precedencia y flujo continuo de tareas para obtener soluciones factibles dentro del entorno de la optimización. Para lo anterior, se propuso un programa cuya estructura está cimentada sobre el algoritmo mencionado, esto con el fin de cumplir en menos tiempo la minimización de *Makespan* en ciclos productivos. Por otro parte, la validación corroboró la eficiencia de la propuesta, a partir de que en promedio el programa concluye con óptimas respuestas, en un margen cercano al 5% del punto óptimo de todas las instancias probadas. Paralelo a lo anterior, el tiempo computacional no excede en promedio más de 45 segundos mientras presenta los resultados. Este valor es lógicamente ligado al tamaño de los datos, aunque no presenta mayor fluctuación entre cada corrida para la misma información. También es apropiado concluir que, el factor de número de individuos es el único y más relevante sobre los demás ya que aproximadamente el 60% de las instancias evaluadas, lo señalan como estadísticamente significativo para el modelo. Seguido de este, el porcentaje a mutar cuenta con mediana trascendencia sobre el rendimiento de la variable objetivo, el cual sobresalió en 2 instancias con alto nivel de repercusión. Además, se pudo deducir que, para pocos datos, y de cuantía cercana a 1000 unidades de tiempo, los niveles de los factores que aumentan la calidad de respuesta, son de 22 generaciones con 12 individuos y permutando en un 34% la población. Contrario a esto, cuando se presentan problemas de mayor tamaño es mejor emplear 12 generaciones con 26 individuos y mutando en un 34% la población.

Finalmente, durante la evaluación de los factores se observó que, con un porcentaje elevado de mutación, se ocasiona que se exploren otras regiones factibles, cercanas y lejanas al valor óptimo. Esto que mejora ampliamente la condición de converger en óptimos locales.

16. Recomendaciones

Debido a la discrecionalidad del usuario cuando ingresa la información de su problema en particular, se fijó una matriz interna de gran tamaño a la hora de almacenar el cálculo del *Makespan* para cada individuo. Esto se diseñó para evitar inconvenientes a la hora de ingresar una matriz de tiempos de procesamiento de alta complejidad. Lo anterior podría ser modificado con el fin de reducir el espacio de memoria empleada y mejorar el tiempo computacional.

Además, se recomienda para posteriores investigaciones, incluir los tiempos de ocio en detalle dentro de la programación del Diagrama de Gantt, y así, tener una visión más cercana a la realidad del problema. También, se propone para futuras investigaciones, considerar más máquinas por etapa, aplicando el modelo de Wilson (1989) y el algoritmo de Sistema Biológico Inmune. No obstante, se sugiere articular este simulador a las empresas productivas que están integradas a la universidad, a través del proyecto de AL-INVEST 5.0. Logrando mejorar el desempeño de las líneas de ensamble dentro de las PYME, con un trabajo más riguroso y colaborativo entre la academia y la empresa privada. Adicionalmente, se aconseja ahondar en el estudio del algoritmo y modelo para elevar la calidad del programa, en aras de obtener menores tiempos de procesamiento.

17. Referencias

Ancau, M. (2012). ON SOLVING FLOWSHOP SCHEDULING PROBLEMS. *PROCEEDINGS OF THE ROMANIAN ACADEMY*, 71-79.

Attar, S., & Mohammadi, M. (2013). Hybrid flexible flowshop scheduling problem with unrelated parallel machines and limited waiting times. *International Journal of Advanced Manufacturing Technology*, 1583-1599.

Baker, K. (1974). Introduction to sequencing and scheduling. *New York Wiley*.

Carlier, J. (1978). Ordonnancements a contraintes disjonctives. *R.A.I.R.O. Recherche operationelle/Operations Research 12*, 333-351.

Chase, R., Jacobs, R., & Aquilano, N. (2009). *Administración de operaciones: Producción y cadena de suministro.* México: Mc Graw Hill.

Correa, A., & Rodriguez, E. (2008). Secuenciación de operaciones para configuraciones de planta tipo flexible job shop. *Avances en Sistemas e informática*, 151-161.

Cortés, A. (2004). Toería de complejidad computacional y computabilidad . *Investigaciones de sistemas ingormaticos* , 103-104.

Cortés, B. M., García, J. C., & Hernández, F. (2012). Multi-objective flow-shop scheduling with parallel machines. *International Journal of Production Research*, 2796-2808.

Dannenbring, D. G. (1977). An Evaluation of Flow Shop Sequencing Heuristics. *Managment science*, 1156-1160.

Figielska, E. (2009). Heuristic algorithms for preemptive scheduling in a two-stage flowshop with unrelated parallel machines and 0-1 resource requirements. *Control & Cybernetics*, 723-743.

Garey, M., & Graham, R. (1972). Worst-case analysis of memory allocation algortihms. *Proceedings of the UK Workshp on Computational Intelligence*, 143-150.

Globerson, S., & Tamir, A. (2007). The relationship between job design, human behaviour and system response. *International Journal of Production Research*, 391-400.

Gupta, J., & Tunc, E. (1991). Schedules for a two-stage hybrid flowshop with parallel machines at the second stage. *International Journal of Production Research*, 1489.

Gutiérrez, H., & De la Vara, R. (2012). *Análisis y diseño de experimentos* . México: Mc Graw Hill.

Harase, S. (2019). Conversion of Mersenne Twister to double-precision floating-point. *Mathematics and computers in simulation*, 76-83.

Heller, J. (1960). Some numerical experiments for an MxJ flow shop and its decision - theoretical aspects. *Operations Research 8*, 5-13.

Khalili, M., & Alireza, A. (2014). Mathematical models and a hunting search algorithm for the no-wait flowshop scheduling with parallel machines. *International Journal of Production Research*, 2667-2681.

Kumar, R. (2011). Some novel methods for flow shop scheduling. *International Journal of Engineering Science and Technology*, 8395-8402.

Kyparisis, G., & Koulumas, C. (2006). Flexible flow shop scheduling with uniform parallel machines. *European Journal of Operational Research*, 985-997.

López, J. C., & Arango, J. (2015). Algoritmo genético para reducir el makespan en un flow shop híbrido flexible con máquinas paralelas no relacionadas y tiempos de alistamiento dependientes de la secuencia. *Entramado*, 250-262.

Martí, R. (2003). Procedimientos metaherísticos en optimización combinatoria. *Matemátiques*.

Nowicki, E., & Smutnicki, C. (1998). The flow shop with parallel machines: A tabu search approach. *European Journal of Operational Research* , 226-253.

Peña, E., Garavito, E., Pérez, L., & Moratto, E. (2016). Revisión de la Literatura Sobre el Problema de. *Ingeniería*, 9-22.

R.J, K., Y.H, L., Zulani E., S., & F.C, T. (2013). Solving bi-level linear programming problem through hybrid of immune genetic algorithm and particle swarm optimization algorithm. *Applied Mathematics and Computation*, 1013-1026.

Rabbiee, M., & Jolai, F. (2015). No-wait flexible flowshop with uniform parallel machines and sequence-dependent setup time: a hybrid meta-heuristic approach. *Journal of Intelligent Manufacturing*, 731-744.

Rashidi, E., Jahandar, M., & Zandieh, M. (2010). An improved hybrid multi-objective parallel genetic algorithm for hybrid flow shop scheduling with unrelated parallel machines. *International Journal of Advanced Manufacturing*, 1129-1139.

Reeves, C. (1995). A genetic algorithm for flowshop sequencing. *Computer Ops Res 22*, 5-13.

Rossi, A., Pandolfi, A., & Lanzetta, M. (2014). Dynamic set-up rules for hybrid flow shop scheduling with parallel batching machines. *International Journal of Production Research*, 3842-3857.

Süer, G. (1998). Designing parallel assembly lines. *Computers & Industrial*, 467-470.

Taha, H. (2004). *Investigación de Operaciones* . México: Pearson.

Wilson, J. (1989). Alternative Formulations of a Flow-shop Scheduling Problem. *Operational Research Society*, 395-399.

Yingchen, W., Geng, X., Zhang, F., & Junhu, R. (2018). An Immune Genetic Algorithm for Multi-Echelon.

Zhou, S., Li, X., & Chen, H. (2016). Minimizing makespan in a no-wait flowshop with two batch processing machines using estimation of distribution algorithm. *International Journal of Production Research*, 4919-4937.

18. Anexos

- **Anexo A:** Instancia Car1

	Máquina 1	Máquina 2	Máquina 3	Máquina 4	Máquina 5
Trabajo 1	375	12	142	245	412
Trabajo 2	632	452	758	278	398
Trabajo 3	12	876	124	534	765
Trabajo 4	460	542	523	120	499
Trabajo 5	528	101	789	124	999
Trabajo 6	796	245	632	375	123
Trabajo 7	532	230	543	896	452
Trabajo 8	14	124	214	543	785
Trabajo 9	257	527	753	210	463
Trabajo 10	896	896	214	258	259
Trabajo 11	532	302	501	765	988

- **Anexo B:** Instancia Car5

	Máquina 1	Máquina 2	Máquina 3	Máquina 4	Máquina 5	Máquina 6
Trabajo 1	333	991	996	123	145	234
Trabajo 2	333	111	663	456	785	532
Trabajo 3	252	222	222	789	214	586
Trabajo 4	222	204	114	876	752	532
Trabajo 5	255	477	123	543	143	142
Trabajo 6	555	566	456	210	698	573
Trabajo 7	558	899	789	124	532	12

Trabajo 8	888	965	876	537	145	14
Trabajo 9	889	588	543	854	247	527
Trabajo 10	999	889	210	632	451	856

- **Anexo C:** Instancia Car 6

	Máquina 1	**Máquina 2**	**Máquina 3**	**Máquina 4**	**Máquina 5**	**Máquina 6**	**Máquina 7**	**Máquina 8**	**Máquina 9**
Trabajo 1	887	447	234	159	201	555	463	456	753
Trabajo 2	799	779	567	267	478	444	123	789	21
Trabajo 3	999	999	852	483	520	120	456	630	427
Trabajo 4	666	666	140	753	145	142	789	258	520
Trabajo 5	663	25	222	420	699	578	876	741	142
Trabajo 6	333	558	558	159	875	965	543	36	534
Trabajo 7	222	886	965	25	633	412	210	985	157
Trabajo 8	114	541	412	863	222	25	123	214	896

- **Anexo D:** Instancia Hel2

	Máquina 1	**Máquina 2**	**Máquina 3**	**Máquina 4**	**Máquina 5**	**Máquina 6**	**Máquina 7**	**Máquina 8**	**Máquina 9**	**Máquina 10**
Trabajo 1	1	1	1	4	3	5	5	7	6	4
Trabajo 2	2	5	4	3	1	9	5	4	7	0
Trabajo 3	5	6	8	4	4	2	5	6	7	5
Trabajo 4	4	1	5	6	5	7	9	2	6	2
Trabajo 5	4	4	2	7	3	6	5	2	4	1
Trabajo 6	7	6	2	5	4	1	4	7	5	5
Trabajo 7	8	5	8	7	9	5	3	5	1	5
Trabajo 8	4	2	5	8	9	9	4	7	5	8
Trabajo 9	2	7	4	2	5	4	5	8	4	3
Trabajo 10	6	5	1	9	4	4	7	6	5	1
Trabajo 11	5	4	7	3	9	1	4	7	3	2

Trabajo 12	2	4	9	2	4	5	2	1	4	2
Trabajo 13	4	0	1	2	2	3	1	4	2	8
Trabajo 14	1	2	5	7	8	6	2	1	4	8
Trabajo 15	6	4	5	1	2	4	5	6	2	9
Trabajo 16	4	5	3	1	8	7	0	1	4	6
Trabajo 17	7	3	1	4	7	0	4	1	5	6
Trabajo 18	5	2	4	1	2	7	5	3	2	3
Trabajo 19	8	6	8	5	7	4	2	5	9	5
Trabajo 20	4	5	3	5	7	9	2	4	5	8

- **Anexo E:** Instancia Rec7

	Máquina 1	**Máquina 2**	**Máquina 3**	**Máquina 4**	**Máquina 5**	**Máquina 6**	**Máquina 7**	**Máquina 8**	**Máquina 9**	**Máquina 10**
Trabajo 1	28	18	38	11	97	23	90	52	79	63
Trabajo 2	50	30	75	82	38	39	28	84	48	57
Trabajo 3	75	50	33	58	56	41	51	29	75	97
Trabajo 4	65	42	66	29	36	29	10	84	14	67
Trabajo 5	84	68	42	41	86	23	95	30	73	97
Trabajo 6	33	72	79	85	81	51	72	19	48	48
Trabajo 7	91	66	87	88	97	36	21	59	61	4
Trabajo 8	51	23	100	93	48	84	74	7	98	55
Trabajo 9	58	61	17	54	25	71	52	47	49	86
Trabajo 10	44	27	40	19	34	33	3	89	39	66
Trabajo 11	70	94	7	19	31	48	38	48	73	34
Trabajo 12	60	38	34	55	63	28	70	35	68	88
Trabajo 13	39	33	53	87	2	6	51	42	93	67
Trabajo 14	72	35	45	20	84	23	10	34	8	48
Trabajo 15	100	71	80	89	47	15	90	33	97	26

Trabajo 16	79	23	57	54	70	99	85	5	9	4
Trabajo 17	14	23	36	79	4	65	78	51	95	79
Trabajo 18	3	32	81	26	19	59	80	90	44	33
Trabajo 19	68	33	94	37	33	74	64	50	22	17
Trabajo 20	94	17	54	27	55	34	7	56	10	41

- **Anexo F:** Instancia Rec13

	Máquina 1	**Máquina 2**	**Máquina 3**	**Máquina 4**	**Máquina 5**	**Máquina 6**	**Máquina 7**	**Máquina 8**	**Máquina 9**	**Máquina 10**	**Máquina 11**	**Máquina 12**	**Máquina 13**	**Máquina 14**	**Máquina 15**
Trabajo 1	78	37	79	98	100	21	82	27	7	22	9	57	84	91	5
Trabajo 2	13	81	100	77	45	39	60	87	38	91	17	85	43	81	33
Trabajo 3	47	9	31	40	86	27	69	50	87	34	13	15	95	96	72
Trabajo 4	26	61	5	22	26	52	57	97	10	68	8	49	41	16	35
Trabajo 5	68	41	46	58	37	59	22	43	49	21	42	70	13	2	76
Trabajo 6	47	60	23	23	57	60	35	56	54	73	81	61	15	70	51
Trabajo 7	63	21	10	50	12	84	91	7	44	75	60	72	28	83	52
Trabajo 8	52	100	38	79	90	52	81	54	51	11	76	50	24	12	23
Trabajo 9	91	21	59	55	39	75	3	11	14	24	36	72	48	69	55
Trabajo 10	100	54	77	53	90	91	53	68	18	86	28	61	86	36	15
Trabajo 11	63	58	93	24	18	78	74	57	100	92	51	73	6	12	83
Trabajo 12	96	60	79	84	86	31	41	79	63	17	31	65	41	77	74
Trabajo 13	54	55	89	77	40	71	21	43	60	58	13	3	73	44	85

Trabajo 14	50	33	51	100	46	27	30	80	56	50	97	70	9	22	92
Trabajo 15	17	7	12	66	98	25	76	25	76	25	69	69	73	45	74
Trabajo 16	46	76	32	32	28	48	63	92	85	69	24	38	57	21	66
Trabajo 17	91	42	30	39	97	72	96	78	71	75	17	26	94	3	39
Trabajo 18	99	93	57	23	52	89	4	74	21	10	53	94	59	100	68
Trabajo 19	20	84	25	1	72	63	79	63	17	38	21	36	1	73	58
Trabajo 20	18	4	21	43	55	86	99	38	81	74	34	40	61	76	100

- **Anexo G:** Instancia Rec21

	Máquina 1	**Máquina 2**	**Máquina 3**	**Máquina 4**	**Máquina 5**	**Máquina 6**	**Máquina 7**	**Máquina 8**	**Máquina 9**	**Máquina 10**
Trabajo 1	12	89	69	94	22	31	22	95	36	38
Trabajo 2	55	2	94	19	43	74	72	10	99	16
Trabajo 3	34	78	27	40	21	17	15	62	96	63
Trabajo 4	100	14	23	42	52	18	29	70	21	47
Trabajo 5	41	92	88	52	99	41	68	22	57	66
Trabajo 6	24	62	19	35	24	49	100	65	13	41
Trabajo 7	29	34	38	72	29	83	44	91	65	100
Trabajo 8	89	62	32	54	93	59	8	24	86	66
Trabajo 9	23	34	89	66	10	48	27	11	94	45
Trabajo 10	81	57	35	78	23	66	1	3	77	14
Trabajo 11	72	47	75	27	66	64	30	49	42	7
Trabajo 12	10	15	26	12	98	12	53	81	46	3
Trabajo 13	47	54	58	73	44	87	87	98	34	15
Trabajo 14	13	27	29	85	29	64	62	62	79	74
Trabajo 15	49	57	24	44	18	97	59	75	17	22

Trabajo 16	50	11	93	53	52	13	51	76	87	95
Trabajo 17	99	87	85	9	87	98	34	22	66	11
Trabajo 18	7	8	90	95	29	79	70	79	6	58
Trabajo 19	48	100	74	60	74	19	21	6	77	84
Trabajo 20	96	86	19	15	45	1	90	49	98	80
Trabajo 21	68	73	55	13	28	16	57	20	76	71
Trabajo 22	53	38	4	43	11	49	12	91	47	3
Trabajo 23	57	13	12	12	21	68	2	80	9	28
Trabajo 24	37	79	92	35	63	13	58	36	65	94
Trabajo 25	39	49	57	23	53	80	42	29	52	33
Trabajo 26	36	54	59	69	62	12	77	37	87	47
Trabajo 27	63	35	26	38	47	82	89	34	1	93
Trabajo 28	60	28	1	51	94	86	42	75	76	77
Trabajo 29	2	51	79	74	51	28	78	87	81	35
Trabajo 30	45	94	42	9	70	4	52	54	16	27

- **Anexo H:** Instructivo de uso del programa

Universidad Pontificia Bolivariana

SECCIONAL BUCARAMANGA

www.upb.edu.co
Personería jurídica No. 48 de 1937 del Ministerio de Gobierno
Seccional Bucaramanga
Vigilada Mineducación

Gestor de secuenciación de máquinas en paralelo en líneas de ensamble

Luis Alejandro Orellano Durán

1. Para comenzar, digite el número de trabajos y máquinas con las que cuenta la línea de ensamble.

2. Complete la matriz con los tiempos de procesamiento de todos los trabajos en cada maquina.

Trabajos/Máquina:	Maquina 1	Maquina 2	Maquina 3	Maquina 4	Maquina 5
Trabajo 1	6	9	4	7	6
Trabajo 2	8	3	6	3	4
Trabajo 3	10	5	5	3	8
Trabajo 4	6	8	10	3	9
Trabajo 5	7	6	8	6	3

3. Oprima el botón de "Siguiente"

4. Digite el número de iteraciones que desea en la casilla que se pregunta por la cantidad de generaciones.

Si cuenta con una gran cantidad de datos, ingrese 12 en el número de individuos y 34 en el porcentaje a mutar.
De lo contrario, digite 26 y 10 respectivamente.

5. Al finalizar, se encontrará con el diagrama de Gantt para asignar los trabajos de acuerdo a la mejor secuencia.