Solución de un Sistema Open Shop en un Taller de Mecánica Automotriz Mediante el Diseño de un Algoritmo Genético

Ing. Luis Fernando Delgado Moore

Nota de Aceptación

Certificamos que el presente Trabajo de Grado
Satisface, en alcances y calidad, todos los requisitos
Que demanda un Trabajo de Grado de Maestría.

Daniel Morillo Torres, Ph.D.
Director

Ing. Fabián Castaño
Jurado

Ing. Hernando Prado
Jurado

Aprobado en cumplimiento de los requisitos exigidos por la Pontificia Universidad Javeriana Cali, para optar el título de Magister en Ingeniería énfasis Industrial.

HERNÁN CAMILO ROCHA NIÑO Ph. D.
Decano Facultad de Ingeniería y Ciencias

JUAN CARLOS MARTÍNEZ ARIAS
Director Posgrados de Ingeniería y Ciencias

Santiago de Cali, 20 de enero de 2021.

Maestría en Ingeniería
Facultad de Ingeniería y Ciencias

Acta de Correcciones al Documento de Trabajo de Grado

Santiago de Cali, 20 de enero de 2021

Autor: Ing. Luis Fernando Delgado Moore

Título del Trabajo de Grado: "Solución de un Sistema Open Shop en un Taller de Mecánica Automotriz Mediante el Diseño de un Algoritmo Genético"

Director: Daniel Morillo Torres, Ph.D.

Como indica el artículo 2.13 de las Directrices para Trabajo de Grado de Maestría, he verificado que el estudiante indicado arriba ha implementado todas las correcciones que los Jurados del Proyecto de Trabajo de Grado definieron que se efectuaran, como consta en el Acta de Evaluación correspondiente.

Firma del Director del Trabajo de Grado

Solución de un Sistema Open Shop en un Taller de Mecánica Automotriz Mediante el Diseño de un Algoritmo Genético
Solution of an Open Shop System in an Automotive Mechanics Workshop by Designing of a Genetic Algorithm

Ing. Luis Fernando Delgado Moore

Trabajo de Grado para optar por el título de Magister en Ingeniería

Director:
Daniel Morillo Torres, Ph.D.

Pontificia Universidad Javeriana de Cali
Facultad de Ingeniería y Ciencias
Maestría en Ingeniería Industrial
CALI
2020

Solución de un Sistema Open Shop en un Taller de Mecánica Automotriz Mediante el Diseño de un Algoritmo Genético
Solution of an Open Shop System in an Automotive Mechanics Workshop by Designing of a Genetic Algorithm

Ing. Luis Fernando Delgado Moore

Trabajo de Grado para optar por el título de Magister en Ingeniería

Pontificia Universidad Javeriana de Cali
Facultad de Ingeniería Y Ciencias
Maestría en Ingeniería Industrial
CALI
2020

Tabla de Contenidos

RESUMEN 2
INTRODUCCIÓN 3
Capítulo 1 5
Planteamiento del Problema. 5
Capítulo 2. 9
Objetivos del Proyecto. 9
2.1. Objetivo General. 9
2.2. Objetivos Específicos. 9
Capítulo 3. 10
Marco Teórico. 10
3.1. Estado del Arte. 10
3.2 Marco Conceptual. 12
3.2.1. Descripción del Algoritmo Genético. 12
3.2.2. Operadores del Algoritmo Genético. 12
3.2.2.1. Criterio de Selección. 13
3.2.2.2. Operador de Cruce. 13
3.2.2.3. Operador de Mutación. 15
3.2.2.4. Criterio de Parada. 16
3.2.3. Parámetros del Algoritmo Genético. 16
Capítulo 4. 17
Desarrollo del Proyecto. 17
4.1. Diseño del Modelo Matemático. 17
Capítulo 5. 20
Nuevo Algoritmo Genético. 20
5.1. Diseño del Algoritmo Genético. 20
5.1.1. Población Inicial y Codificación. 20
5.1.2. Medida de Desempeño. 21
5.1.3. Selección de Padres. 21
5.1.4. Operador de Cruce. 21
5.1.5. Operador de Mutación. 21
5.1.6. Reemplazo de Población. 22
5.1.7. Condición de Terminación. 23
5.1.8. Parámetros de Ajuste. 23
5.1.9. Pseudocódigo del AG. 23
Capítulo 6. 25
Resultados. 25
6.1. Detalle de la Implementación. 25
6.2. Resultados Obtenidos. 25
6.2.1. Resultados del Modelo Matemático. 25
6.2.1. Resultados del Algoritmo Genético. 26
6.3. Caso Aplicado. 29
6.3.1 Beneficio Económico de la Propuesta. 30
Capítulo 7. 32

Conclusiones. 32vi
Capítulo 8. 33
Referencias Bibliográficas. 33

Lista de Tablas

Tabla 1. Tipo de Servicio y Duración Media. 5
Tabla 2. Tipos de Mantenimiento. 6
Tabla 3. Estaciones de Servicio. 6
Tabla 4. Lista de Operaciones. 6
Tabla 5. Codificación de Operaciones Caso Aplicado 7
Tabla 6. Requerimientos de Servicio Alta Demanda 8
Tabla 7. Resultados de Iteraciones Modelo Matemático. 25
Tabla 8. Resultados Modelo Matemático Instancias 5 x 5. 25
Tabla 9. Comparativo Instancias 5x5 AG vs. Modelo Matemático. 26
Tabla 10. Resultados AG vs. Grupo de Instancias 26
Tabla 11. Cambios en Operador de Torneo 27
Tabla 12. Cambios en Probabilidad de Mutación y Probabilidad de cruce 0,8. 27
Tabla 13. Cambios en Probabilidad de Cruce sin Mutación. 28
Tabla 14. Cambios en Probabilidad de Mutación sin Cruce. 28
Tabla 15. Resultados Operadores de Mutación M.A y M.S. 28
Tabla 16. Codificación de Actividades y Asignación de Máquinas 29
Tabla 17. Nombre de Máquinas. 29
Tabla 18. Resultados AG en Caso Aplicado 30
Tabla 19. Asignación de Máquinas de AG en Caso Aplicado 31

Lista de Figuras

Figura 1. Desempeño de Diferentes Algoritmos en instancias 7x7. 11
Figura 2. Pseudocódigo del Algoritmo Genético Simple. 13
Figura 3. Operador de Cruce (C.S.). 14
Figura 4. Operador de Cruce (C.R.)..... 14
Figura 5. Operador de Cruce (C.O). 15
Figura 6. Operador de Cruce (C.D) 15
Figura 7. Operador de Mutación (M.A.)..... 16
Figura 8. Operador de Mutación (M.S.) 16
Figura 9. Representación de Cromosomas..... 20
Figura 10. Operador de Cruce..... 22
Figura 11. Operadores de Cruce y Operador de Mutación M.A. (7x7)..... 22
Figura 12. Operadores de Mutación y Operadores de Cruce (15 x 15)..... 23
Figura 13. Pseudocódigo del Algoritmo Genético..... 24

ABSTRACT

The task scheduling area (also known as scheduling) is responsible for planning the best allocation of resources within a productive system that allows the achievement of a set of tasks; achieving efficiency and effectiveness in such assignment. The incentive to study problems related to scheduling arises from the need to obtain high quality feasible solutions from a group of tasks (jobs) and resources (such as machines) that need to be sequenced in such a way as to optimize productive resources and thus obtain an economic benefit. The present study seeks to solve one of the most relevant problems in this area: the Open Shop, through the development of a genetic algorithm.

An Open Shop problem is defined as a set of *m* machines which must process a set of *n* jobs that have a specific processing time; each job must be processed by all machines (or a subset of them). Additionally, only one job can be run at a time, and each machine can only process one job simultaneously. Initially, a Mathematical Model was developed that generated feasible solutions obtained through the AMPL optimization software, which was efficient only for instances of less than 5 machines and 5 jobs. With the aim of improving the breadth of results and applications, a Genetic Algorithm was developed which found solutions for instances of up to 20 machines and 20 jobs, which in turn were evaluated with the most used databases in literature: instances of Taillard (1993), Brucker (2007) and Guéret & Prins (1999) finding average deviations of less than 9% in sizes of up to 20 machines x 20 jobs.

Finally, the aforementioned algorithm was applied to a real case of an auto mechanic workshop in which it was shown that by assigning the sequences of operations and machines in a better way, the execution time (*makespan*) would improve substantially, allowing to increase the efficiency of the operation quantified with the following indicators: Total execution of daily work orders (100%), 14% of available time and 86% of occupancy.

Keywords: Open Shop, Scheduling, Makespan, Metaheuristics, Genetic Algorithm.

RESUMEN

El área de programación de tareas (también conocida como *scheduling*) se encarga de planificar la mejor asignación de los recursos dentro de un sistema productivo que permita la consecución de un conjunto de tareas; logrando eficiencia y efectividad en tal asignación. El incentivo de estudiar problemas relacionados con el *scheduling* nace de la necesidad de obtener soluciones factibles de alta calidad de un grupo de tareas y recursos (como máquinas) que requieren ser secuenciados de forma que se optimicen los recursos productivos y obtener así un beneficio económico. El presente estudio buscar solucionar uno de los problemas más relevantes de esta área: el *Open Shop*, mediante el desarrollo de un algoritmo genético.

Un problema *Open Shop* se define como un conjunto de *m* máquinas las cuales deben procesar un conjunto de *n* tareas que tienen un tiempo de procesamiento específico; cada tarea debe ser procesado por todas las máquinas (o un subconjunto de ellas). Adicionalmente, solo se puede ejecutar un solo tarea a la vez, y cada máquina solo puede procesar una tarea simultáneamente. Inicialmente, se desarrolló un Modelo Matemático que generó soluciones factibles obtenidas mediante el software de optimización AMPL, quien resultó eficiente solo para instancias inferiores a 5 máquinas y 5 tareas. Con el ánimo de mejorar amplitud en resultados y aplicaciones, se desarrolló un Algoritmo Genético el cual encontró soluciones para instancias de hasta 20 máquinas y 20 tareas, las que a su vez fueron evaluadas con las bases de datos más usadas en literatura: las instancias de Taillard (1993), Brucker (2007) y Guéret & Prins (1999) encontrando desviaciones promedio inferiores al 9% en tamaños de hasta 20 máquinas x 20 tareas.

Finalmente, el algoritmo mencionado fue aplicado a un caso real de un taller de mecánica automotriz en el que se evidenció que asignando de una mejor manera las secuencias de operaciones y las máquinas, el tiempo de ejecución (*makespan*) mejoraría sustancialmente permitiendo incrementar la eficiencia de la operación cuantificada con los siguientes indicadores: Ejecución total de órdenes de trabajo diarias (100%), 14% de tiempo disponible y 86% de ocupación.

Palabras Clave: *Open Shop*, *Scheduling*, *makespan*, Metaheurísticas, Algoritmo Genético.

INTRODUCCIÓN

En la actualidad, la optimización de los recursos productivos juega un papel crucial en la planeación de la producción dentro de las organizaciones. Una adecuada programación dentro del área productiva que involucre la asignación de máquinas y la secuenciación de operaciones dentro de las mismas, es un proceso valioso dentro de los sistemas de producción (Castillo, Díaz B., Gómez, Orduz, & Niño, 2018). En los últimos años se ha incrementado el interés de la ciencia en estas temáticas, es por esto que varios investigadores han unido esfuerzos para estudiar un problema que describe este tipo de situaciones, el *Open Shop*. La programación del *Open Shop* es un caso especial del área de programación de tareas conocida en la literatura como *scheduling* en la que cada tarea o trabajo consta de *n* operaciones que se pueden ejecutar en cualquier orden, siempre y cuando no se presente simultaneidad de tareas (Schuurman & Woeginger, 1999). Este problema de programación es tipo *NP-Hard* (Liaw, 2003) es decir, no admite soluciones exactas eficientes, por tanto, el uso de metaheurísticas ha sido inevitable para encontrar la mejor asignación de máquinas y recursos en un tiempo razonable dentro de un sistema productivo de tamaño realista.

Durante las últimas décadas han surgido nuevos métodos de optimización, siendo uno de estos las llamadas Metaheurísticas, definidas como técnicas que guiadas por una heurísticas y combinadas con conceptos de estadística e inteligencia artificial, logran en la mayoría de los casos, una búsqueda eficiente de una solución factible al problema aunque no garantizan que la solución represente el óptimo global (Gendreau & Potvin, 2010). Dentro las anteriores se encuentra presente el llamado Algoritmo Genético (GA), usado en esta investigación, y se define como un procedimiento adaptativo que encuentra soluciones a problemas mediante un proceso evolutivo basado en selección natural y genética (Holland, 1975). Este algoritmo funciona con una población de soluciones y mecanismos para orientar la búsqueda hacia la mejor, utilizando el principio de supervivencia del más apto. La búsqueda del algoritmo genético utiliza la elección aleatoria como herramienta para guiar hacia una búsqueda altamente explosiva en la que se puedan encontrar mejores soluciones a través de técnicas de codificación que consiguen acercarse a puntos óptimos globales.

El *OSSP* (*Open Shop Scheduling Problem*) es un problema de secuenciamiento de actividades pertenecientes a tareas las cuales deben ejecutarse bajo una asignación de máquinas con la condición de que una misma tarea no puede ser ejecutada simultáneamente en otra máquina. Este tipo de problema se introdujo por primera vez en los principales talleres de reparación de automóviles (Gonzalez & Sahni, 1976). Con el paso del tiempo se ha demostrado que muchas otras aplicaciones, entre las que destacan los centros de control de calidad, los centros de fabricación de semiconductores, las asignaciones de profesores en las diferentes aulas, los programas de inspección y las comunicaciones por satélite (Kubiak, Sriskandarajah, & Zaras, 1991; Liu & Bulfin, 1987; Prins, 1994)

Los problemas de programación se plantean como un conjunto de máquinas y tareas, donde las máquinas representan recursos productivos y los tareas representan actividades que deben realizarse utilizando estos recursos (Kolen, Lenstra, Papadimitriou, & Spieksma, 2007). Cada *tarea* incluye una serie de operaciones y cada operación debe procesarse en una máquina predeterminada. Sin embargo, el punto importante que lo distingue del resto de problemas similares como el Tarea Shop o el *Flow Shop* es que el orden de las operaciones de cada trabajo no está predeterminado. Por lo tanto, se puede considerar cualquier orden arbitrario para las operaciones de una tarea. En el *OSSP*, cada máquina puede procesar como máximo un tarea a la vez, y cada tarea también puede procesarse como máximo en una sola máquina a la vez (Rahmani Hosseinabadi, Vahidi, Saemi, Sangaiah, & Elhoseny, 2019). Este contexto de problema se puede aplicar en distintas labores productivas, incluso las máquinas pueden ser reemplazadas por otro recurso que genere actividad y que tengan como objetivo una buena asignación de secuencias de operaciones.

Complementado los conceptos anteriormente mencionado, un Modelo Heurístico se define como aquel que es elaborado utilizando diversos métodos empíricos tales como estrategias basadas en la experiencia, práctica, lógica y observación de los hechos con el fin de llegar a la solución eficaz de un problema determinado (Bishop & Polya, 1991).

Este estudio se focalizará en encontrar una programación (Schedule) mediante el desarrollo de un modelo matemático y de un modelo heurístico que minimice el tiempo necesario para completar todas las operaciones, denominado *makespan.*

Capítulo 1

Planteamiento del Problema.

El *OSSP* es un problema de secuencias de elementos que tienen una serie de actividades las cuales deben ser ejecutadas en una máquina asignada sin existir simultaneidad de tareas de la misma referencia. El *OSSP* tiene un grupo de aplicaciones industriales, como lo son, moldeo de plástico, procesos químicos, industria petrolera y producción de alimentos; mientras que en el sector de servicios se utiliza para modelar servicios de atención médica, mantenimiento de vehículos y telecomunicaciones entre otros (Gonzalez & Sahni, 1976; Lin, Lee, & Pan, 2008). Para algunos autores, el diseño del *Open Shop* se asume implícitamente como la disciplina de programación operativa en los Sistemas De Fabricación Flexible (*FMS – Flexible Manufacturing System*) (Ben-Arieh & Dror, 1991; Kusiak, 1985), que a su vez puede utilizarse dentro del sistema de fabricación inteligente previsto por la Industria 4.0 (Yin, Stecke, & Li, 2018). Por tanto, no es de extrañar que, hoy en día, el *OSSP* sea un tema de investigación activo y prometedor en el área de programación de la producción (Anand & Panneerselvam, 2015). En esta investigación el *OSSP* fue aplicado a un caso real de servicios de mecánica automotriz descrito a continuación.

En la ciudad de Palmira – Valle del Cauca, Colombia, se encuentra localizado un Centro de Investigaciones de Agricultura el cual cuenta con una flota autos propios conformada por 170 vehículos de los cuales son 15 buses, 3 busetas, 60 pickups, 32 sedan y 60 SUV. Dada la magnitud del equipo y con el fin de ser más eficientes en el servicio de mecánica preventiva y correctiva, se tiene establecida una unidad estratégica de servicio destinada a prestar estas labores. Actualmente se prestan los servicios ilustrados en la Tabla 1. Adicionalmente, se pueden visualizar los tipos de mantenimiento en la Tabla 2, las estaciones de servicio en la Tabla 3 y la lista de operaciones en la Tabla 4.

Tabla 1. Tipo de Servicio y Duración Media.

CLASE DE MANTENIMIENTO	DURACION MEDIA (MINUTOS)
TIPO A	90
TIPO B	105
TIPO C	180
TIPO D	180
LAVADO LIVIANO	30
LAVADO 4X4	45
Revisión Viaje	30
Vulcanizada llanta	30
Cambio pastillas	60
Cambio llantas 2 unid.	90

Tabla 2. Tipos de Mantenimiento.

ITEM	DESCRIPCIÓN
TIPO A	Múltiplo de 5,000km
TIPO B	Múltiplo de 10,000km
TIPO C	Múltiplo de 20,000km
TIPO D	Para buses y busetas

Tabla 3. Estaciones de Servicio.

ESTACIÓN DE SERVICIO	MÁQUINA ASIGNADA	NÚMERO DE OPERADORES
LAVADO	LAVADORA Y ASPIRADORA	1 PERSONA
LUBRICACIÓN	Equipo Neumático	1 PERSONA
ALINEACIÓN Y BALANCEO	ALINEADOR	1 PERSONA
INSPECCIÓN	ELEVADOR	1 PERSONA

Tabla 4. Lista de Operaciones.

	LAVADO	LUBRICACIÓN	ALINEACIÓN	BALANCEO	INSPECCIÓN
TIPO A	X				X
TIPO B	X	X	X	X	X
TIPO C	X	X	X	X	X
TIPO D		X			X
LAVADO	X				
Revisión Viaje					X
Vulcanizada llanta					X
Cambio pastillas					X
Cambio llantas 2 unid.			X	X	

La demanda actual es de 170 vehículos de los cuales 152 son potenciales de operación de lavado diario, pero por condiciones operativas y de capacidad nunca se podrán atender todo el mismo día. Por lo anterior, los 152 vehículos mencionados pueden ser programados con anterioridad dependiendo la capacidad disponible para los servicios de Mantenimiento tipo A, B, C o D según lo requieran, adicionalmente, los operadores trabajan turnos diarios de 8 horas. El problema actual radica en que no se tiene una asignación óptima de tareas en las distintas estaciones de servicio lo que trae como resultado ineficiencias en la operación tales como incumplimiento en entrega de autos y sobre costos financieros. Esto se debe a que la programación de tareas se hace de manera manual, es decir, por orden de llegada de vehículos, estos son dirigidos a las estaciones correspondientes.

Con el fin de disminuir demoras en los procesos y de realizar efectivamente las tareas de mantenimiento diarias, en este estudio se diseñó un algoritmo metaheurístico para obtener asignaciones eficientes de los recursos disponibles (soluciones cercanas a las óptimas) logrando así mejorar la eficiencia en la prestación del servicio de mecánica automotriz.

Teniendo en cuenta la información anteriormente descrita y las características de la operación, estas fueron modeladas bajo un sistema *Open Shop* el cual se identifica de la siguiente manera. Cada Tarea hace referencia a los vehículos que requieren las solicitudes de servicio tales como Mantenimientos tipo A, B, C, D, Revisión de Viaje, Lavado, Vulcanizada de llanta, Cambio de pastillas y Cambio de 2 unidades de llantas. En cada estación de servicio se encuentra asignada una máquina *m* con la que se ejecutará la operación correspondiente, las distintas tareas enumeradas tienen un conjunto de operaciones que deber ser realizadas, estas son ejecutadas en las estaciones de servicio (Tabla 3). En la Tabla 5 se muestra la codificación y simbología que representa el problema. En la Tabla 6 se muestran los requerimientos de un día de alta demanda de servicios el cual tuvo como resultado final de tres vehículos sin entregar de un total de 12 que ingresaron en ese mismo día.

Tabla 5. Codificación de Operaciones Caso Aplicado

OPERACIÓN	NOMBRE	MAQUINAS	TIEMPO (min)
1	LAVADO	M1	30
2	LAVADO PICKUP	M1	45
3	VULCANIZADO	M2	30
4	CAMBIO DE LLANTAS	M3	90
5	REVISIÓN DE VIAJE	M2	30
6	REVISIÓN 5000KM	M1, M2	M1(30) + M2(60) = 90
7	REVISIÓN 5000KM	M1, M2	M1(45) + M2(60) = 90
8	REVISIÓN 10.000 KM	M1, M3	M1(30) + M3(75) = 105
9	REVISIÓN 10.000 KM	M1, M3	M1(45) + M3(75) = 105
10	REVISIÓN 20.000 KM	M1, M3	M1(30) + M3(150) = 180
11	REVISIÓN 20.000 KM	M1, M3	M1(45) + M3(150) = 180
12	REVISIÓN BUSES	M2	180

Tabla 6. Requerimientos de Servicio Alta Demanda

TAREA	OPERACIÓN	HORA DE LLEGADA	HORA DE SALIDA
1	1	8:00:00	8:40:00
2	7	8:00:00	10:00:00
3	6	11:00:00	13:00:00
4	6	9:00:00	16:00:00
5	12	8:30:00	14:00:00
6	1	8:30:00	SIN FINALIZAR
7	1	9:00:00	SIN FINALIZAR
8	1	9:00:00	14:00:00
9	1	9:30:00	13:00:00
10	1	9:00:00	14:00:00
11	6	8:00:00	15:00:00
12	11	8:00:00	SIN FINALIZAR

Capítulo 2.

Objetivos del Proyecto.

2.1. Objetivo General.

Minimizar costos en la cadena de servicio de la unidad de mecánica de un Centro de Investigación mediante la generación de secuencias de trabajo eficientes.

2.2. Objetivos Específicos.

- Caracterizar el sistema actual de asignación de servicios en el taller de mecánica.
- Realizar una revisión del estado del arte e identificar la metodología más apropiada para el problema a resolver.
- Diseñar e implementar un algoritmo metaheurístico que permita encontrar secuencias eficientes a las actividades de mecánica requeridas.
- Validar los resultados obtenidos realizando una evaluación financiera de la implementación del algoritmo en el caso práctico y en los casos de prueba más relevantes de la literatura.

Capítulo 3.

Marco Teórico.

3.1.Estado del Arte.

El problema de programación en *Open Shop* (*OSSP*) es un caso de optimización combinatoria ampliamente estudiado en las últimas décadas desde el artículo de González y Sahni (1976). Los problemas de optimización combinatoria implican la búsqueda de una solución óptima entre una colección discreta de posibles soluciones. Normalmente, aumenta el número de posibles soluciones exponencialmente con la cantidad de elementos en el problema (MacGregor & Chu, 2011).

En el *OSSP*, un conjunto de *n* trabajos deben procesarse en un conjunto de *m* máquinas, por lo que cada trabajo debe visitar todas las máquinas. Sin embargo, en el diseño de *Open Shop* no hay rutas predefinidas para los trabajos en las máquinas, por lo que la ruta de procesamiento para cada trabajo se determinará en el proceso de programación (Framinan, Leisten, & García, 2014). Por lo tanto, el número de soluciones factibles para este problema es mayor que para otros diseños, como, por ejemplo, máquinas paralelas, *Flow Shop* o entornos de producción de *Job Shop*.

En el estado del arte se encontraron las siguientes ventajas del uso de un GA en problemas de *scheduling*, cuyos argumentos apoyan su utilización en la investigación presente (Dorigo et al., 2009; Glover & Kochenberger, 2003).

- Los AG exploran un espacio de búsqueda sin perder por completo las soluciones parciales que ya han sido encontradas.
- La mutación aleatoria garantiza hasta cierto punto que se encuentren una amplia gama de soluciones.
- Los AG gestionan un adecuado procesamiento en paralelo.
- En los AG es menos factible que converjan a un máximo o mínimo local dado que operan realizando búsquedas en una población de puntos y adicionalmente están basados en reglas de transición probabilísticas las cuales permiten escapar de óptimos locales.
- Los AG son fáciles de implementar.
- Los AG no hacen suposiciones sobre el espacio del problema.

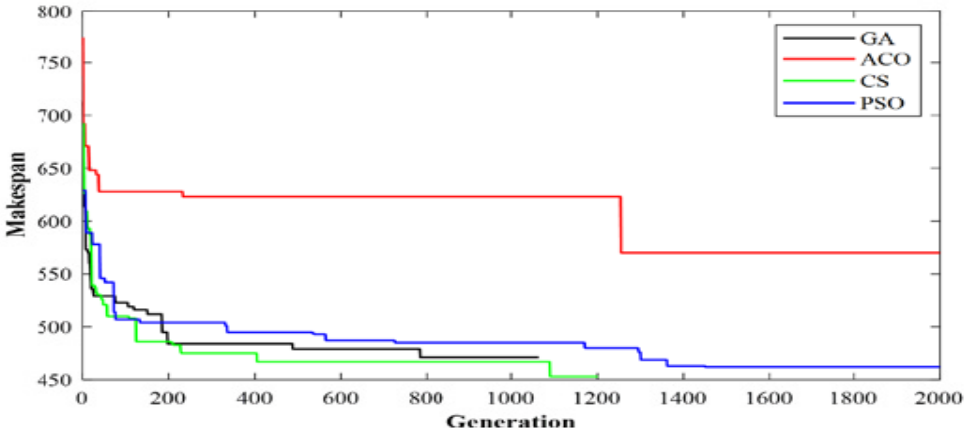
Castrillón (2011) realizó una investigación titulada Aplicación de un Algoritmo Evolutivo en la Solución de Problemas *Job Shop-Open Shop***,** cuyo objetivo se centró en disminuir el *makespan* y aumentar la utilización de las máquinas. Para cumplirlo, empleó el Algoritmo Genético como instrumento y fue aplicado en una empresa del sector metalmecánico, teniendo como resultado reducciones del 33% del tiempo total en el proceso productivo y del 51% de tiempo muerto de recursos, y a su vez, una aproximación del 99% respecto a la solución óptima estimada con una cota inferior. Adicionalmente, la confiabilidad del resultado fue complementada con un análisis de varianza de los distintos

valores obtenidos del *makespan* obteniendo un 99.5% de confianza, lo cual otorga peso a la utilización del algoritmo mencionado en la presente investigación

Otro de los motivos por los cuales se tomó el Algoritmo Genético en la solución del problema *Open Shop* tiene que ver con los resultados obtenidos en la investigación realizada por los autores (Huang et al., 2019) en el cual enfocó su estudio en encontrar un algoritmo inteligente efectivo que pueda usarse en el problema de programación de la *Open Shop*. El procedimiento fue utilizar algoritmos inteligentes, a saber, algoritmo genético (GA), optimización de enjambre de partículas (PSO), algoritmo de búsqueda de Cuco (CS) y optimización de colonias de hormigas (ACO). Los resultados mostraron que estos algoritmos inteligentes tienen un rendimiento altamente eficiente y pueden usarse para problemas a gran escala los cuales no pueden ser resueltos por métodos exactos. Entre estos algoritmos, el algoritmo de Búsqueda de Cuco (CS) es el mejor y puede resolver problemas de manera efectiva, el siguiente en la posición es el Algoritmo Genético (GA) el cual presenta un buen desempeño cuando las escalas van en aumento, las cuales iniciaron en 4 máquinas – 4 tareas y terminaron en 15 máquinas y 15 tareas. En la Figura 1 se puede evidenciar el comportamiento de cada uno de los métodos de solución observando la manera de cómo el *makespan* disminuye conforme el número de generaciones va en aumento. Pero definitivamente la escogencia del diseño de un AG obedece a que en escalas superiores a 5x5 se obtienen soluciones de mayor calidad, las cuales resultan relevantes al caso aplicado.

Para complementar la justificación de la heurística empleada para éste investigación, se encontró en la literatura, el estudio realizado por (Benziani, Kacem, Laroche, & Nagih, 2014), en el que se utilizó el algoritmo genético para resolver el problema *Open Shop* obteniendo soluciones óptimas en todas las instancias con 4, 5, 7 y 10 máquinas – Tareas.

Figura 1. Desempeño de Diferentes Algoritmos en instancias 7x7.



Los argumentos anteriores fueron suficientes para decidir que el Algoritmo Genético (AG) sería una buena heurística para resolver el problema del *Open Shop*.

3.2 Marco Conceptual.

En contexto con lo anterior, existen distintos métodos de solución relacionados al problema de *Open Shop*, tales como la Búsqueda Tabú, Recocido Simulado, Cómputo Evolutivo, Algoritmos Genéticos (AG), Lógica Difusa, Redes Neuronales y las Estrategias Evolutivas. Dentro de la industria han sido usados frecuentemente los llamados Algoritmos Genéticos los cuales encuentran soluciones utilizando el principio de supervivencia más apto basado en la genética en la que se generan un grupo de soluciones obtenidas por el uso de operadores tales como cruce y mutación (Dréo, Siarry, Pétrowski, & Taillard, 2006). Esta técnica ha sido utilizada por varios investigadores en diferentes tipos de problemas, entre los que sobresalen la solución de secuencias del Agente Viajero (Anaya Fuentes, Hernández Gress, Seck Tuoh Mora, & Medina Marín, 2016), programación de máquinas, enrutamiento vehicular y empaquetado de contenedores entre otros (Sivanandam & Deepa, 2008) . En el estudio del Agente Viajero los autores propusieron un AG para resolver la mejor ruta que debe tomar un vendedor de tal manera que visite todas las ciudades minimizando el tiempo total empleado en esta tarea. Los resultados fueron satisfactorios al ser aplicados al AG obteniendo variaciones promedio de 19% con respecto a los óptimos encontrados en la literatura (Beasley, 1990) . Las conclusiones mencionadas fueron un soporte más para decidir la implementación del AG para el presente estudio.

3.2.1. Descripción del Algoritmo Genético.

Los Algoritmos Genéticos (AG) pertenecen al grupo de los métodos metaheurísticos los cuales son producto de la combinación de métodos heurísticos con el fin de lograr una mejor exploración en el espacio de búsqueda de forma eficiente y efectiva obteniendo soluciones casi óptimas (Jiménez-Builes, Arango-Sanchez, & Jiménez-Pinzón, 2016).

El AG se define como una técnica de búsqueda cuyo fundamento es la teoría de la evolución de las especies de Charles Darwin, donde los individuos más aptos tienen más probabilidades de sobrevivir y aparearse, por lo que la próxima generación debe ser mejor (Salazar-Hornig & Medina, 2013). Este método permite la generación de poblaciones mediante operadores genéticos, creando poblaciones aleatorias cada vez de mejor calidad. Los operadores como la selección, la mutación y la recombinación o cruzamiento se aplican a la población (Anaya Fuentes et al., 2016; Gestal et al., 2005). En la Figura 2 se puede observar el Pseudocódigo del Algoritmo Genético en el que se menciona detalladamente cómo interfieren los operadores mencionados. Para completar la información, en las siguientes líneas se describen los anteriores operadores.

3.2.2. Operadores del Algoritmo Genético.

Los operadores de evolución buscan soluciones potenciales nuevas y / o mejores. La búsqueda revela dos aspectos diferentes: explotación de la mejor solución y exploración del espacio de búsqueda. El éxito de esta fase evolutiva está específicamente relacionado con la obtención de un equilibrio adecuado entre exploración y explotación que finalmente depende del conjunto seleccionado de evolución de operadores. Las nuevas soluciones potenciales se obtienen aplicando la evolución de operadores a los miembros de la

población; Cada uno de estos miembros es referido como individuo en la población (Sivanandam & Deepa, 2007).

Básicamente, hay tres operadores de evolución que funcionan para un código (cromosoma): selección, cruce y mutación.

3.2.2.1. Criterio de Selección.

La selección es el proceso en el que los cromosomas individuales son seleccionados de acuerdo al valor de la función objetivo (*makespan*). Esto significa que el cromosoma con mejor valor tiene más probabilidad de ser seleccionado para la próxima generación. Para realizar este proceso se utiliza el método de Torneo. En este método, un pequeño subconjunto de los cromosomas es seleccionado al azar, y de aquí, dos de los cromosomas que tienen el mejor *makespan* son tomados como padres.

Figura 2. Pseudocódigo del Algoritmo Genético Simple.

```
BEGIN /* Algoritmo Genetico Simple */
      Generar una poblacion inicial.
      Computar la funcion de evaluacion de cada individuo.
      WHILE NOT Terminado DO
      BEGIN /* Producir nueva generacion */
            FOR Tamaño poblacion/2 DO
            BEGIN /*Ciclo Reproductivo */
                  Seleccionar dos individuos de la anterior generacion,
                  para el cruce (probabilidad de seleccion proporcional
                  a la funcion de evaluacion del individuo).
                  Cruzar con cierta probabilidad los dos
                  individuos obteniendo dos descendientes.
                  Mutar los dos descendientes con cierta probabilidad.
                  Computar la funcion de evaluacion de los dos
                  descendientes mutados.
                  Insertar los dos descendientes mutados en la nueva generacion.
            END
            IF    la poblacion ha convergido THEN
                  Terminado := TRUE
      END
END
```

Fuente. (Goldberg & Holland, 1988).

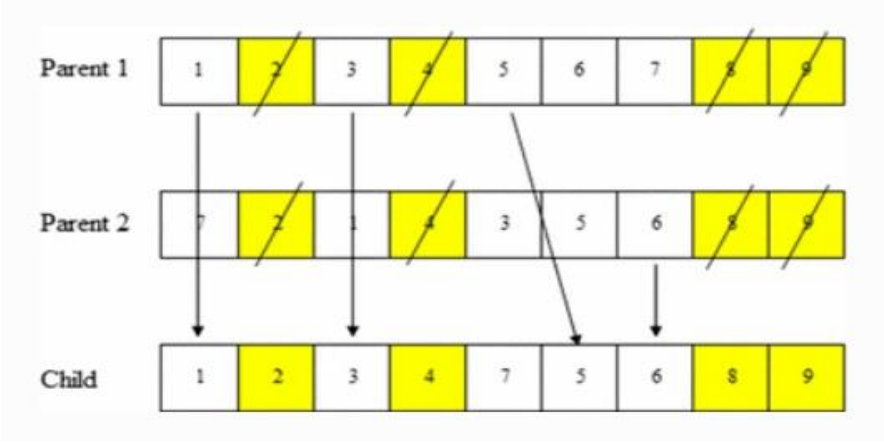
3.2.2.2. Operador de Cruce.

Este operador busca que los cromosomas de la generación existente (padres), se combinen entre sí y a partir de estos se creen los nuevos cromosomas (hijos) de la siguiente generación. Esta combinación puede darse por medio de puntos de corte.

Existen varios operadores de cruce entre ellos, y para este estudio, se tomarán básicamente cuatro de ellos los cuales servirán de apoyo para probar y decidir el más conveniente para ser aplicado en la investigación.

- *Homologous Crossover Genetic Operation (CS)*: una vez seleccionados los padres, este operador traslada por primera vez los genes homólogos de los dos padres al hijo, y luego el restante de los genes se seleccionan estocásticamente del padre 1 o 2 y se trasladan al hijo. La Figura 3 muestra la operación de la función.

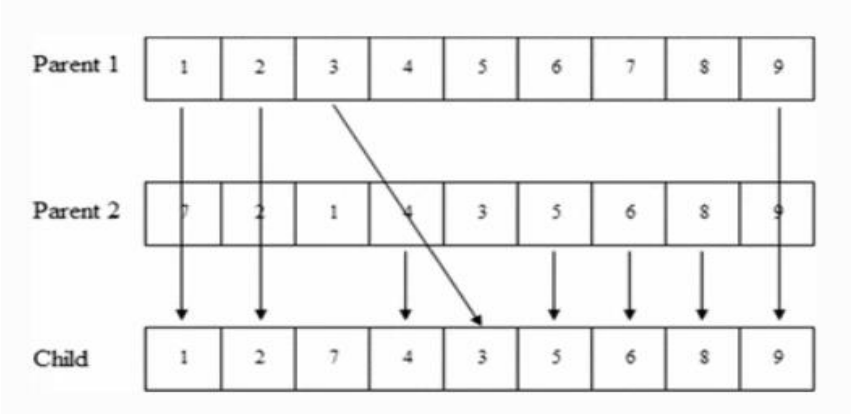
Figura 3. Operador de Cruce (C.S.).



Fuente (Rahmani Hosseinabadi et al., 2019).

- *Row Crossover Operation (CR):* una vez seleccionados los dos padres, la operación CR toma al azar a uno de los dos padres primero y le transmite su primer gen al hijo. Este gen luego se omite del otro padre. Este proceso continúa hasta que el hijo recibe todos sus genes. En este método de cruce, se generan diversos cromosomas. La figura 4 ilustra un ejemplo de la aplicación del C.R.

Figura 4. Operador de Cruce (C.R.)

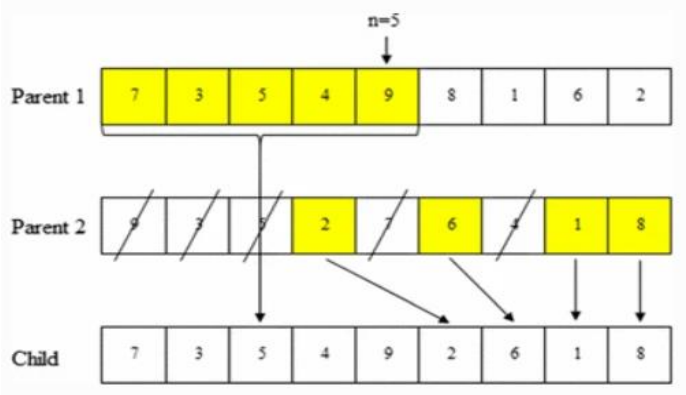


Fuente (Rahmani Hosseinabadi et al., 2019).

- *One-point Crossover Operation (CO)*: la operación de cruce de un punto (CO) primero selecciona un número estocásticamente del intervalo 1 hasta *n* (longitud del

cromosoma) y los genes en el intervalo de 0 al número seleccionado al azar se transmiten al hijo. A continuación, se omiten los genes homólogos del padre 2 y el resto de los genes se incrustan, respectivamente, en las células vacías del hijo. La Figura 5 ilustra la aplicación del CO.

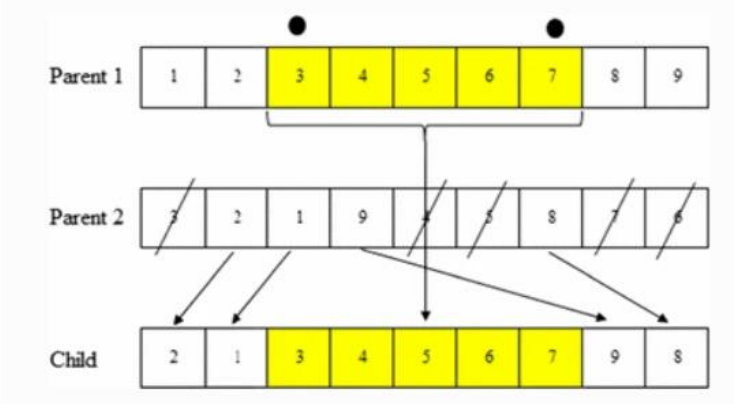
Figura 5. Operador de Cruce (C.O).



Fuente (Rahmani Hosseinabadi et al., 2019).

- *Two-Point Crossover Operation (CD)*: la operación de cruce de dos puntos (CD) primero selecciona dos números aleatorios en el intervalo de 0 a n (longitud del cromosoma). Los genes entre estos dos números seleccionados estocásticamente se transmiten del padre 1 al hijo, y se omiten los genes homólogos del padre 2. El resto de los genes están incrustados, respectivamente, del padre 2 en las células vacías del hijo. La figura 6 implica un ejemplo de la aplicación del C.D.

Figura 6. Operador de Cruce (C.D)



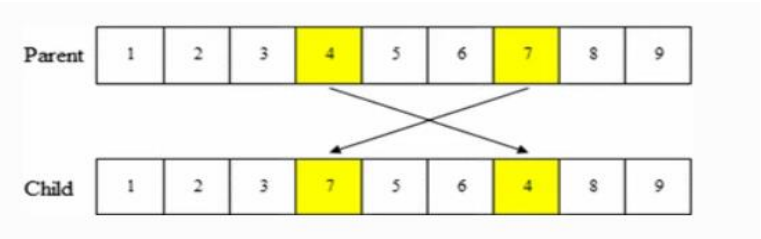
Fuente (Rahmani Hosseinabadi et al., 2019).

3.2.2.3. Operador de Mutación.

Existen diversos operadores de Mutación, en este caso se mencionarán dos específicamente, los cuales hacen parte de la investigación de este documento.

- *Displacement Mutation Operation (MA)*: después de la selección de los padres, la operación mutación de desplazamiento (MA) selecciona aleatoriamente dos genes y desplaza los desplaza. La Figura 7 indica la aplicación del MA.

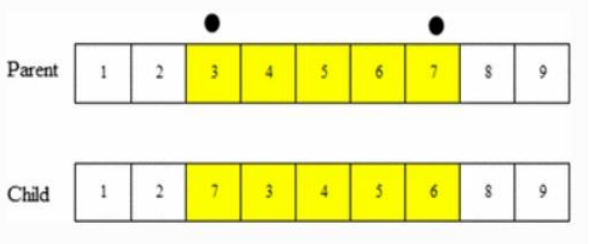
Figura 7. Operador de Mutación (M.A.)



Fuente (Rahmani Hosseinabadi et al., 2019).

- *Shift Mutation Operation (MS):* después de seleccionar el cromosoma original, se seleccionan al azar dos puntos en el intervalo de 1 a *n* (longitud del cromosoma) y los genes ubicados entre estos dos puntos se desplazan rotacionalmente hacia la izquierda. La Figura 8 muestra un ejemplo de MS.

Figura 8. Operador de Mutación (M.S.)



Fuente (Rahmani Hosseinabadi et al., 2019).

3.2.2.4. Criterio de Parada.

La condición de terminación de GA depende del tipo de problema y del conocimiento externo al respecto. Para éste estudio, el número máximo de generaciones especificado se considera como la condición de terminación, es decir, si se alcanza el número deseado de generaciones, se encuentra el mejor cromosoma y se termina el algoritmo (Jain & Deb, 2014).

3.2.3. Parámetros del Algoritmo Genético.

Los parámetros del GA se fijan de la siguiente manera: la probabilidad de cruce es igual a 0,8 y la probabilidad de mutación es igual a 0,2. El número de generaciones es 250, y el tamaño de la población se considera 92 según los autores (Jain & Deb, 2014). Estos valores se obtuvieron de forma experimental.

Capítulo 4.
Desarrollo del Proyecto.

4.1. Diseño del Modelo Matemático.

Dado que la cantidad de información de Modelos Matemáticos bajo la problemática *Open Shop* es escasa, se decidió adaptar y fortalecer un Modelo *Job Shop* experimentado con éxito, el cual fue desarrollado por (Ku & Beck, 2016). Estos autores evaluaron cuatro modelos de Programación Entera Mixta para el problema clásico de *Job Shop*, Disyuntivo, Disyuntivo de Liao, Tiempo Indexado y Basado en Rango; Adicionalmente compararon su rendimiento usando CPLEX, GUROBI y SCIP. Los resultados de CPLEX y GUROBI mostraron que, según lo anunciado (Liao & You, 1992), el modelo disyuntivo original de otro autor (Manne, 1960) es el más eficiente de todos ya que tienen la menor cantidad de variables binarias.

La conclusión del autor (Ku & Beck, 2016) sirvió para tomar como base el Modelo Matemático planteado por ellos, el cual fue transformado de *Job Shop* a *Open Shop* y adaptado a la necesidad motivo de estudio. La notación usada en este modelo es la siguiente.

- Conjuntos
  J_i = *tarea número: {1,…, i, … ,n}.*
  M_k = *número de máquina k: {1,…, k, …,m}.*

- Parámetros
  m = *número de máquinas.*
  n = *número de tareas.*
  M = *número positivo muy grande.*
  N_i = *número de operaciones de* J_i.
  P_{ik} = *tiempo de procesamiento de* J_i *en* M_k.
  r_{ik} = *1 si* J_i *requiere* M_k*; 0 de lo contrario; Binaria.* $N_i = \sum_{k=1}^{mi} r_{ik}$.

- Variables.

  X_{ijk} = *1 si* J_i *es programada en la j posición para ser procesada en* M_k*;*
  0 de lo contrario; Binaria.
  $z^k_{iji'j'}$ *= 1; si la actividad j de la tarea i precede a la actividad j'de la tarea i'en la máquina k (no necesariamente de manera inmediata); 0 de lo contrario; Binaria.*
  S_{ijk} = *tiempo de inicio de la actividad j de la tarea i en la máquina k.*

- Función Objetivo.
  C_{max} = *makespan;*
  $C_{max} = max_{i=1}^{n} C_{i;}$

La formulación del modelo matemático es la siguiente.

$$mín: \ Cmax \tag{1}$$

$$X_{ijk} <= r_{ik}; \ \forall i \in J, j, k \in \{1, \ldots, N_i\} \tag{2}$$

$$\sum_{j=1}^{Ni} X_{ijk} = r_{ik}; \ \forall i = 1, 2..n; \quad k = 1,2 \ldots . N_i; \tag{3}$$

$$\sum_{k=1}^{m} X_{ijk} = 1; \ \forall i = 1, 2..n; \quad j = 1, 2 \ldots Ni \tag{4}$$

$$S_{ijk} <= M * X_{ijk}; \quad \forall i = 1, \ldots, n, j = 1, \ldots, N_i, k = 1, \ldots, m \tag{5}$$

$$S_{ijk} + P_{ik} <= S_{i\,,j+1,k'} + M * (2 - X_{ijk} - X_{i,j+1,k'}); \quad \forall i = 1, \ldots, n, \quad j = 1, \ldots, N_i - 1, \quad k = 1, \ldots, m \tag{6}$$

$$S_{ijk} - S_{i'j'k} >= P_{i'k} - M * (2 - X_{ijk} - X_{i'j'k}) - M * (1 - Z^{k}_{iji'j'}); \quad \forall i = 1, \ldots, n, \quad j = 1, \ldots, N_i, \quad k = 1, \ldots, m \tag{7}$$

$$S_{i'j'k} >= S_{ijk} + P_{ik} - M * (2 - X_{ijk} - X_{i'j'k}) - M * (1 - Z^{k}_{iji'j'}); \quad \forall i = 1, \ldots, n, \quad j = 1, \ldots, N_i, \quad k = 1, \ldots, m, \quad i < i' \tag{8}$$

$$Cmax >= S_{iN_ik} + P_{ik} * X_{iN_ik}; \quad \forall i = 1,\ldots,n, k = 1,\ldots,N_i \tag{9}$$

$$Cmax >= 0; \ \ S_{ijk} >= 0; \ \forall i = 1, \ldots, n; \ j = 1, \ldots, Ni \tag{10}$$

$$Z_{\ iji'j'k} <= X_{ijk}; \ \forall i = 1, \ldots, n, \quad j = 1, \ldots, N_i, \quad k = 1, \ldots, m \tag{11}$$

$$Z_{\ iji'j'k} <= X_{i'j'k}; \ \ \forall i = 1, \ldots, n, \quad j = 1, \ldots, N_i, \quad k = 1, \ldots, m \tag{12}$$

En la restricción (2) se relacionan las variables X_{ijk} y los parámetros r_{ik}. Permiten que el modelo asigne la máquina disponible para el tarea y actividad correspondiente solo cuando es posible. La restricción (3) garantizan que se programen el total de las actividades de las tareas. La restricción (4) obliga a que solo una tarea sea ejecutada en una sola máquina.

La restricción (5) hace referencia al tiempo de inicio de la tarea *i*, número de operación *j* en la máquina *k*, S_{ijk} obligando a que sea igual a 0 cuando $X_{ijk} = 0$. Es decir, si no se asigna

esa operación de ese tarea en esa máquina no es posible que tenga un tiempo de inicio asociado a esa variable.

El conjunto de restricciones (6) garantiza que el procesamiento de $S_{i(j+1)k'}$ pueda ser iniciado sólo después de que S_{ijk} haya terminado (anula la simultaneidad operaciones de un mismo tarea y garantiza la precedencia de las tareas).

El conjunto de restricciones (7) y (8) juntos hacen cumplir el requisito de que sólo una operación puede procesarse en una máquina en cualquier instante de tiempo.

El conjunto de restricciones (9) otorga la definición de C_{max} en el que se especifica que debe ser minimizando la función objetivo (1). En la ecuación (10) se define la naturaleza de las variables.

El conjunto (11) y (12) especifican las restricciones binarias para X_{ijk} y $Z^{k}_{iji'j'}$ en las que se establece el cumplimiento cuando la actividad *j* de la tarea *i* precede a la actividad *j'* de la tarea *i'* en la máquina *k* no necesariamente de manera inmediata.

Capítulo 5.
Nuevo Algoritmo Genético.

5.1. Diseño del Algoritmo Genético.

Teniendo en cuenta la descripción del Algoritmo Genético realizada en el Capítulo 3, la metodología empleada para el desarrollo de la metaheurística involucra los siguientes operadores: Generación de la población inicial, evaluación del *makespan* de las soluciones generadas, selección de los padres, cruzamiento y mutación. Después de este procedimiento se establece un criterio de parada el cual fue tomado de experimentaciones exitosas de otros autores (Jain & Deb, 2014). Como resultado final, se obtiene un conjunto de mejores soluciones con su respectivo *makespan* siendo este el efecto de aplicar el AG al proceso de *scheduling*.

5.1.1. Población Inicial y Codificación.

El tamaño de población es determinado por *p* el cual representa el número de soluciones factibles al problema. La población está compuesta por un grupo de individuos o cromosomas de tamaño *m x n*, donde *m* es el número de máquinas y *n* es el número de tareas. El fenotipo y genotipo tienen una permutación sin repeticiones del tamaño *m x n*, en donde cada número corresponde a una activad pertenecientes a una tarea. Como resultado de lo anterior, cada cromosoma (individuo) tendrá la información necesaria del problema el cual será herramienta fundamental para que este modelo analice la combinación de tareas y máquinas que calcule mejores soluciones. En Figura 9 se observa la estructura de un cromosoma en el que los números de 1 a 9 hacen parte del conjunto de secuencias asociadas a la solución inicial, las cuales a su vez se encuentran ligadas a un Tarea y una máquina especifica. Es importante aclarar que esta codificación es usada para representar una solución mediante estas posiciones (1 a 9), pero que en ningún caso obligan a la solución a estar bajo una secuencia establecida sobre las tareas, puesto que ellos deben ser procesados por todas las máquinas, pero en cualquier orden.

Figura 9. Representación de Cromosomas

TAREAS / MÁQUINAS	1	2	3
1	1	2	3
2	4	5	6
3	7	8	9

Cromosoma 1	7	9	1	6	3	5	4	2	8

El AG busca encontrar soluciones haciendo una mejor secuenciación de las actividades del cromosoma (genes), su medición de desempeño es el tiempo empleado en completar tales actividades. De aquí surge la decodificación del AG con el *makespan* de cada cromosoma o conjunto de actividades.

La población inicial es determinada de manera aleatoria y el proceso de selección se realiza bajo una función distribución en la cual la probabilidad de ser seleccionado aumenta según mejore el *makespan* de cada individuo o cromosoma.

5.1.2. Medida de Desempeño.

Existen distintas alternativas de indicadores para medir el desempeño de una secuencia de operaciones en el problema *Open Shop*. En este caso dada la naturaleza del problema de reducir tiempos de operación mejorando la eficiencia en el uso de recursos, se escogió el indicador *Makespan,* el cual se define como el tiempo empleado para ejecutar las distintas operaciones de un proceso productivo desde el inicio hasta el fin del mismo. El menor *Makespan* será al objetivo a conseguir en el caso estudiado.

5.1.3. Selección de Padres.

EL AG utilizó el método de selección de Padres Torneo. El método de Torneo consiste en tomar aleatoriamente un grupo reducido de individuos de la población para que salgan de aquí los dos padres que serán encargados de producir nuevas poblaciones (Rahmani Hosseinabadi et al., 2019). Esta escogencia se hace teniendo en cuenta los mejores valores de *makespan* de cada individuo, en el que tendrá mayor probabilidad de ser escogido conforme el valor de *makespan* sea menor. En este método, un pequeño grupo de los cromosomas es seleccionado al azar y de estos, dos de los mejores cromosomas con mejores valores de *makespan* son los seleccionados como padres.

5.1.4. Operador de Cruce.

El operador escogido es *One-point Crossover Operation (CO)* el cual según investigaciones de varios autores (Rahmani Hosseinabadi et al., 2019), presenta mejor desempeño para este tipo de problemas *Open Shop*. La Figura 10 muestra la manera de aplicación de éste operador.

5.1.5. Operador de Mutación.

En el AG se utilizaron dos operaciones de mutación con distintos criterios cada uno para estudiar los efectos en la resolución del problema.

- **Displacement Mutation Operation (M.A).**
- **Shift Mutation Operation (M.S).**

Recopilando la investigación realizada por el autor (Rahmani Hosseinabadi et al., 2019) se encuentran los desempeños de los operadores de cruce y de mutación los cuales se pueden visualizar en la Figura 11. Como se puede apreciar, una vez más el uso del C.O y el M.A fue más adecuado y arrojó mejores soluciones, mientras que la operación de mutación M.S junto con la operación de cruce de C.O tuvo un desempeño deficiente. Estos últimos resultados fueron plasmados en la Figura 12 en donde el eje X corresponde a generaciones de tamaño 15x15 y el *makespan* (eje y) evaluando las distintas combinaciones de operadores de Mutación y de Cruce La experimentación anterior permitió la escogencia de los operadores del AG motivo de estudio: C.O y M.A.

Figura 10. Operador de Cruce

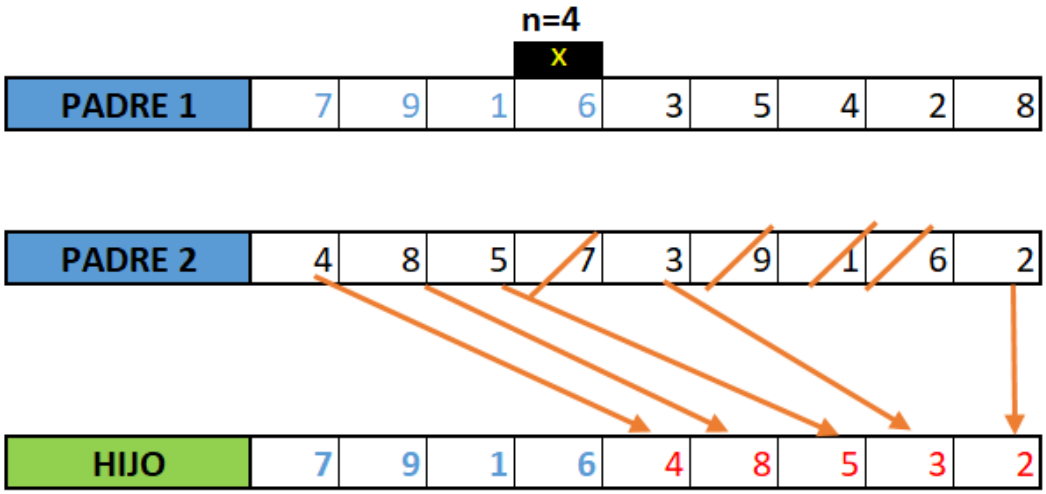
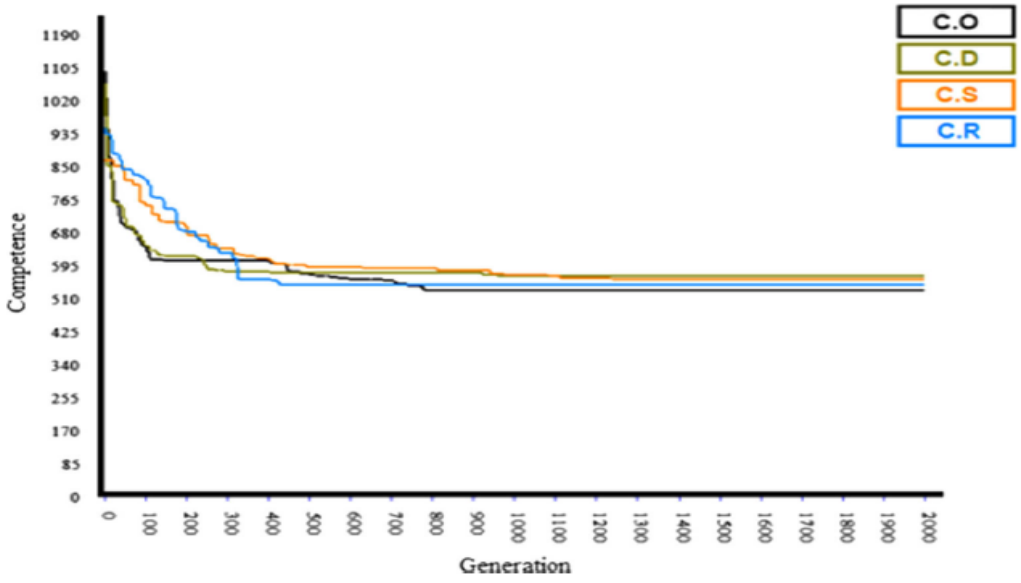


Figura 11. Operadores de Cruce y Operador de Mutación M.A. (7x7)

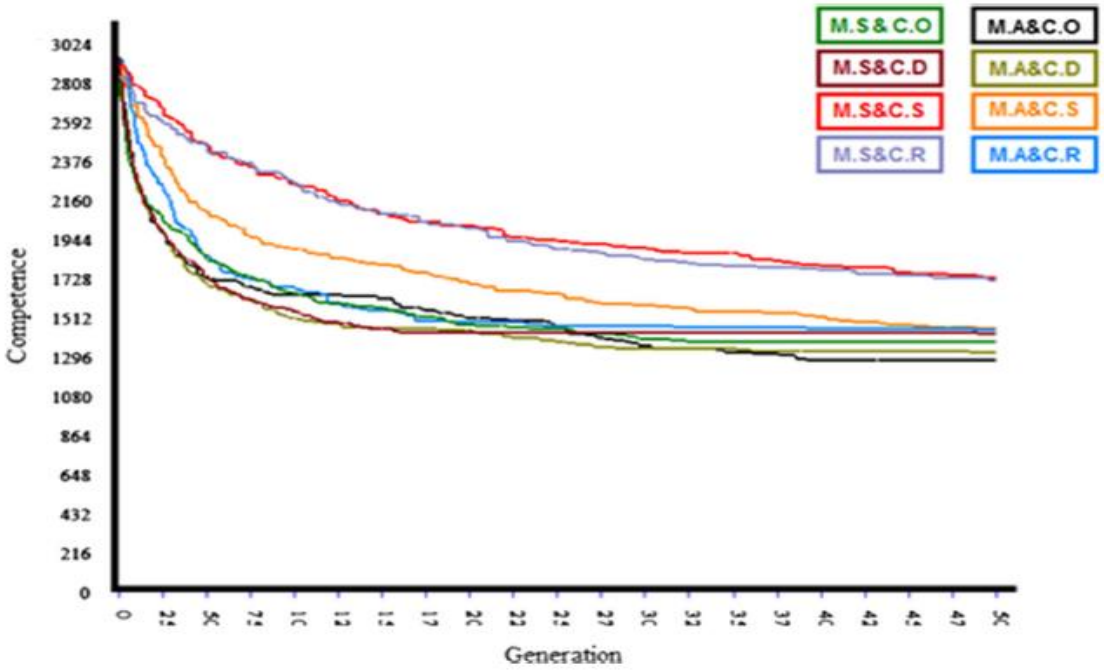


Fuente. Rahmani Hosseinabadi et al., 2019

5.1.6. Reemplazo de Población.

Los cromosomas para la próxima generación se seleccionaron en el AG utilizando un enfoque híbrido. Los cromosomas son ordenados en función de su *Makespan* inicialmente, los cromosomas repetitivos luego fueron omitidos. Solo el 10% de los cromosomas con

Figura 12. Operadores de Mutación y Operadores de Cruce (15 x 15)



Fuente. Rahmani Hosseinabadi et al., 2019

valores de aptitud más altos se seleccionaron para la próxima generación. El resto de los cromosomas se tomaron de forma aleatoria dado que este método pretende mantener la dispersión del cromosoma en cada iteración.

5.1.7. Condición de Terminación.

La condición de terminación del AG depende del tipo de problema y el conocimiento externo al respecto. En el AG el número máximo especificado de generaciones se considera como la condición de terminación, es decir, si el número deseado de generaciones se alcanza, se encuentra el mejor cromosoma y de esta manera se termina el algoritmo.

5.1.8. Parámetros de Ajuste.

Los parámetros del AG fueron determinados de la siguiente manera. La probabilidad de cruce es igual a 0,8 y la probabilidad de mutación es igual a 0,2. El número de generaciones es 250, y el tamaño de la población se considera 92 según estudios preliminares (Jain & Deb, 2014) Estos últimos autores desarrollaron un Algoritmo Genético Multiobjetivo en el que evidenciaron que aplicando un Óptimo de Pareto podrían encontrar mejores soluciones en el algoritmo de optimización, el resultado de éste optimo, sugiere una población de 92 para encontrar puntos de referencia útiles.

5.1.9. Pseudocódigo del AG.

El pseudocódigo del AG se puede apreciar en la Figura 13.

Figura 13. Pseudocódigo del Algoritmo Genético.

```
Function AG (problem)
Input: Populationsize, Problemsize, Pcrossover, Pmutation, StopConditionNumber
Output: Cmax
Population←Initalize Population (Populationsize, Problemsize);
Evaluate Population (Population);
Cmax ←Get Best Solution (Population);
int number Repeat=1;
While number Repeat <= Stop ConditionNumber  do
        Random ( ) = i;
        If i < =  Pcrossover  do
                Child = Parent1= Parent2= Ø;
                Select Parents (&Parent1, &Parent2, Population, Populationsize);
                Child ←Crossover C.O(Parent1, Parent2);
                Random ( ) = j;
                if j <= Pmutation;
                        Child ← Mutate M.A (Parent1, Parent2);
                        Evaluate Fitness (Child);
                        Population←Insert (Population, Child);
        end;
        Population←Select Generation (Population, Populationsize);
        Cmax ←Get Best Solution (Population);
        number Repeat + 1;
end;
Return Cmax;
```

Fuente. Adaptado de Rahmani Hosseinabadi et al., 2019

Capítulo 6.
Resultados.

6.1. Detalle de la Implementación.

El algoritmo desarrollado se implementó en C ++. Las pruebas computacionales se realizaron en una CPU Intel (R) Core (TM) i7 – 8550U CPU @ 1.80 GHz - 16GB de RAM y Sistema operativo Windows 10 pro de 64 bits.

6.2. Resultados Obtenidos.

6.2.1. Resultados del Modelo Matemático.

Se realizaron un total de 450 simulaciones en las cuales se programaron desde 3 hasta 15 máquinas, y desde 3 hasta 20 tareas. Los resultados obtenidos indicaron que el 34% de estas iteraciones resultaron viables tal como se puede ver en la Tabla 7. Se encontró adicionalmente que a partir de 5 tareas y 10 máquinas no es viable el modelo, al igual que desde 3 máquinas y 10 tareas, el 20% de las muestras no fueron viables. Esta conclusión se fundamenta teniendo en cuenta un escenario en el que cada tarea no usa la totalidad de las máquinas. Continuando con la revisión de las opciones óptimas, se puede concluir que el Nuevo Modelo Matemático de *Open Shop* puede obtener buenos resultados en instancias de máximo 5 máquinas x 5 tareas tal como aprecia en la Tabla 6. Como punto adicional, las Tablas 8 y 9 muestran un comparativo de los resultados obtenidos tomando las instancias mencionadas (5x5) al ser procesadas tanto en el AG como en el modelo matemático. Estos resultados fortalecen la calidad de las dos investigaciones al tener los mismos valores de *Cmax* en cada una de las instancias.

Tabla 7. Resultados de Iteraciones Modelo Matemático.

STATUS	**TOTAL**	**PORCENTAJE**
SOLUCIONES OPTIMAS	155	34.44%
DETENIDA POR LÍMITE DE TIEMPO	295	65.56%
Total general	**450**	**100.00%**

Tabla 8. Resultados Modelo Matemático Instancias 5 x 5.

Instancia	Cmax.	Número de variables	Número de restricciones	Tiempo de ejecución (sg)	Cantidad de máquinas utilizadas
Machine_05_tarea_5_Instance_3	193	3376	9475	13,109	TODAS LAS MQS
Machine_05_tarea_5_Instance_4	263	3376	9475	10,312	TODAS LAS MQS
Machine_05_tarea_5_Instance_6	179	3376	9475	8,594	TODAS LAS MQS
Machine_05_tarea_5_Instance_7	213	3376	9475	6,688	TODAS LAS MQS
Machine_05_tarea_5_Instance_8	228	3376	9475	1,703	TODAS LAS MQS
Machine_05_tarea_5_Instance_9	168	3376	9475	14,484	TODAS LAS MQS

Tabla 9. Comparativo Instancias 5x5 AG vs. Modelo Matemático.

INSTANCIA	Cmax AG	Tiempo (sg)	Cmax Mmat	Tiempo (sg)
Machine_5_tareas_5_Instance_3.dat	193	2.995	193	13,109
Machine_5_tareas_5_Instance_4.dat	263	2.994	263	10,312
Machine_5_tareas_5_Instance_6.dat	179	2.848	179	8,594
Machine_5_tareas_5_Instance_7.dat	213	2.919	213	6,688
Machine_5_tareas_5_Instance_8.dat	228	2.935	228	1,703
Machine_5_tareas_5_Instance_9.dat	168	2.806	168	14,484

6.2.1. Resultados del Algoritmo Genético.

Para validar el comportamiento del Algoritmo Genético implementado, se tomaron un total de 207 instancias de distintos autores (Taillard, 1993)(Brucker, 2007)(Guéret & Prins, 1999) las cuales inician con 3 máquinas y 3 tareas (3x3) y finalizan con 20 tareas y 20 máquinas (20 x 20). La medida de desempeño se denomina Desviación y es definida de la siguiente manera.

$$Desviación = (Cmax\ AG - Cmax\ Optimo) / Cmax\ Optimo \quad (13)$$

El desempeño del AG frente a las instancias mencionadas se puede apreciar en la Tabla 10. Los parámetros del AG son los mencionados en el punto 3.2.3 de este escrito.

Tabla 10. Resultados AG vs. Grupo de Instancias

GRUPO DE INSTANCIAS	DESVIACIÓN PROMEDIO
BRUCKER1997	0,119703
GUERET&PRINS1999	0,111745
TAILLARD1993	0,105125
Total general	**0,111832**

Los anteriores valores muestran promedios de las desviaciones en las que se puede comprobar que el mejor desempeño lo tuvo con las instancias de Taillard con un 10% de desviación lo que hace confiable al AG para las aplicaciones que se requieran.

Para validar el mejor desempeño del AG, se corrieron las instancias con los distintos parámetros mostrados a continuación.

- Cambios en el tamaño del Operador de Torneo. Este operador se aumentó hasta un tamaño de 65 cromosomas evidenciando mejores resultados de acuerdo a la Tabla 11. Por lo anterior, 65 será un buen parámetro de Tamaño de Torneo para el AG.

Tabla 11. Cambios en Operador de Torneo

GRUPO INSTANCIA/TAMAÑO TORNEO	21	35	40	65
BRUCKER1997	0,136	0,133	0,131	0,116
GUERET&PRINS1999	0,136	0,142	0,141	0,102
TAILLARD1993	0,126	0,127	0,122	0,101
Total general	**0,133**	**0,135**	**0,132**	**0,105**

- Probabilidad de Cruce de 0.8 y probabilidad de Mutación variable. Se realizaron nuevamente cálculos con el AG teniendo en cuenta cambios en el operador de Mutación obteniendo los resultados que se muestran en al Tabla 12. De este punto se puede concluir que, incrementado la probabilidad de Mutación, se obtienen mejores soluciones lo que quiere decir que implementando este operador al AG tiene a mejorar resultados.

Tabla 12. Cambios en Probabilidad de Mutación y Probabilidad de cruce 0,8.

PROBABILIDAD DE MUTACIÓN	BRUCKER1997	GUERET&PRINS1999	TAILLARD1993	Total general
0,1	0,1386	0,1243	0,1170	0,13
0,3	0,1122	0,1022	0,0975	0,10
0,4	0,1097	0,0998	0,0946	0,10
0,5	0,1119	0,0941	0,0950	0,10
0,6	0,1005	0,0962	0,0914	0,10
0,7	0,1091	0,0925	0,0909	0,10
0,8	0,1075	0,0953	0,0875	0,10
0,9	0,0985	0,0936	0,0870	0,09
1	0,0998	0,0887	0,0828	0,09
Total general	**0,1098**	**0,0985**	**0,0937**	**0,10**

- Sin Operador de Mutación con cambios en probabilidad de Cruce. Después de ejecutar el AG con estos parámetros, según se puede evidenciar en la Tabla 13, no hay variación significativa en los resultados finales, lo que permite concluir que, si bien el incluir operador de cruce ayuda a obtener mejores soluciones, no es Operador determinante para reducir el *makespan*.

- Sin Operador de Cruce con Probabilidad de Mutación Variable. Nuevamente se puede observar en la Tabla 14, la tendencia a disminuir el promedio de varianzas a medida que aumenta la probabilidad de Mutación. Este es un argumento adicional que permite afirmar que el AG obtiene mejores resultados explorando el operador de Mutación.

Tabla 13. Cambios en Probabilidad de Cruce sin Mutación.

GRUPO DE INSTANCIAS	Cruce = 0,4	Cruce = 0,6	Cruce = 0,8	Cruce = 1
BRUCKER1997	0,1714	0,1745	0,1692	0,1675
GUERET&PRINS1999	0,1706	0,1680	0,1649	0,1749
TAILLARD1993	0,1559	0,1441	0,1461	0,1439
Total general	**0,1662**	**0,1623**	**0,1602**	**0,1632**

Tabla 14. Cambios en Probabilidad de Mutación sin Cruce.

Grupo de Instancias/Probabilidad de Mutación	0,20	0,40	0,50	0,90	1,00
BRUCKER1997	0,1252	0,1098	0,1072	0,1006	0,1084
GUERET&PRINS1999	0,1084	0,1056	0,0970	0,0956	0,0900
TAILLARD1993	0,1066	0,0976	0,0881	0,0893	0,0841
Total general	**0,1124**	**0,1043**	**0,0970**	**0,0950**	**0,0931**

Como conclusión a los resultados obtenidos del AG se puede afirmar que el Operador dominante para obtener mejores soluciones del grupo poblacional es el Operador de Mutación. En esta investigación se utilizaron dos operadores, M.A y M.S, mostrando mejores resultados el operador M.A. Por tal motivo fue este el que se utilizó para calcular resultados teniendo variaciones de probabilidad de ocurrencia en el AG.

Para justificar el uso del Operador de Mutación M.A, en la Tabla 15 se muestran resultados de las desviaciones promedio en las instancias mencionadas, aplicando los operadores M.A y M.A bajo los siguientes parámetros:

- Probabilidad de cruce = 0,8
- Probabilidad de Mutación = 0,2
- Tamaño de torneo = 0,65

Tabla 15. Resultados Operadores de Mutación M.A y M.S.

Grupo de Instancias	Displacement Mutation Operation (M.A).	Shift mutation operation (M.S).
BRUCKER1997	0,11584	0,13189
GUERET&PRINS1999	0,10166	0,12836
TAILLARD1993	0,10142	0,11475
Total general	**0,10543**	**0,12506**

6.3. Caso Aplicado.

Tal como se mencionó en el capítulo 1, la finalidad de éste Algoritmo Genético es brindar mejores secuencias de operaciones en las distintas tareas o tareas generadas en un taller de mecánica automotriz. Adicionalmente obtener un beneficio económico en la Unidad Servicio ya que, al planear mejor las operaciones, el aprovechamiento de los recursos aumentaría. El caso aplicado fue codificado teniendo en cuenta que un vehículo es un Tarea con las actividades que requiera la orden de servicio solicitada. En la Tabla 15 se puede observar la codificación de las distintas operaciones con sus respectivas máquinas las cuales a su vez tienen un conjunto de actividades. Para complementar la información anterior, la Tabla 16 describe el nombre de las máquinas que intervienen en el procesamiento de las distintas actividades.

Tabla 16. Codificación de Actividades y Asignación de Máquinas

ACTIVIDAD	ID TAREA	MAQUINAS	TIEMPO (min)
1	LAVADO	M1	40
2	VULCANIZADO	M2	30
3	CAMBIO DE LLANTAS	M3	90
4	REVISIÓN DE VIAJE	M2	30
5	REVISIÓN 5000KM	M1, M2	M1(30) + M2(60) = 90
6	REVISIÓN 10.000 KM	M1, M3	M1(30) + M3(75) = 105
7	REVISIÓN 20.000 KM	M1, M3	M1(30) + M3(150) = 180
8	REVISIÓN BUSES	M2	180

Tabla 17. Nombre de Máquinas.

MÁQUINA	ID MÁQUINA
LAVADORA	1
BANCO DE INSPECCIÓN 1	2
BANCO DE INSPECCIÓN 2 COMPLEMENTOS	3

Los datos de entrada del Algoritmo Genético fueron tomados de un día de alta demanda de servicios en el cual fueron utilizados la totalidad del tiempo disponible diario para labores el cual es de 420 minutos. La Tabla 6 permite observar un total de 12 vehículos ingresados al Taller de Servicio de los cuales 3 quedaron sin finalizar ocasionando sobre costos de reemplazo de vehículos para desplazamiento de los usuarios y disminución de tiempo disponible en el día siguiente por culminación de trabajos pendientes.

Como resultado de solución a este problema *Open Shop*, el AG entregó las siguientes cifras.

Tabla 18. Resultados AG en Caso Aplicado

INSTANCIA	casociat.dat
CMAX Ag (min)	360
TIEMPO AG (Sg)	125,9
OPERADOR DE MUTACIÓN	M.A.
PROBABILIDAD DE CRUCE	0,8
PROBABILIDAD DE MUTACIÓN	1,00
TAMAÑO DE TORNEO	65
CANTIDAD DE PAREJAS	30
NÚMERO DE GENERACIONES	250

Como se esperaba, el AG encontró soluciones eficientes arrojando un mejor resultado de *makespan* de 360 minutos, lo que quiere decir, que las tareas anteriormente descritos pueden ser ejecutados en la disponibilidad habitual de tiempo de procesos diario (420 minutos).

Esta cifra traduce que los recursos se asignarían de una mejor manera y que además tendrían tiempo disponible de 60 minutos. La mejor asignación de máquinas y tiempos generada por el AG se pueden apreciar en el Tabla 18 en donde *id* corresponde al número de actividad, los números entre paréntesis son el período de tiempo en donde se programa, y *maq* corresponde a la máquina asignada. Adicionalmente, Tarea corresponde al vehículo que solicitó el servicio.

6.3.1 Beneficio Económico de la Propuesta.

Los beneficios económicos obtenidos por asignar de manera eficiente los recursos productivos de esta Unidad de Servicio, se pueden cuantificar con los valores que se evitaría incurrir gestionando de manera eficiente la correspondiente asignación.

- Costo de reemplazo de auto. US $35/día.
- Costo de reemplazo de bus. US $180/día.
- Inversión de maquinaria y equipo adicional.

La Unidad de Servicio de Mecánica automotriz debe continuar con el método de solicitud de atención bajo el sistema de cita previa de tal manera que el Programador de Tareas tenga el suficiente tiempo de asignar adecuadamente los recursos y poder cumplir con la promesa de servicio que ofrece esta Unidad a sus distintos usuarios.

Tabla 19. Asignación de Máquinas de AG en Caso Aplicado

ACTIVIDAD, TIEMPO, MÁQUINA	TAREA
{id=3 times=(150, 150) maq=3 }	1
{id=1 times=(105, 135) maq=1 }	1
{id=2 times=(180, 180) maq=2 }	1
{id=4 times=(0, 45) maq=1 }	2
{id=5 times=(180, 240) maq=2 }	2
{id=6 times=(330, 330) maq=3 }	2
{id=8 times=(60, 120) maq=2 }	3
{id=9 times=(150, 150) maq=3 }	3
{id=7 times=(300, 330) maq=1 }	3
{id=11 times=(0, 60) maq=2 }	4
{id=12 times=(150, 150) maq=3 }	4
{id=10 times=(75, 105) maq=1 }	4
{id=14 times=(0, 0) maq=2 }	5
{id=13 times=(105, 105) maq=1 }	5
{id=15 times=(150, 330) maq=3 }	5
{id=18 times=(0, 0) maq=3 }	6
{id=17 times=(180, 180) maq=2 }	6
{id=16 times=(240, 270) maq=1 }	6
{id=21 times=(150, 150) maq=3 }	7
{id=20 times=(240, 240) maq=2 }	7
{id=19 times=(270, 300) maq=1 }	7
{id=23 times=(180, 180) maq=2 }	8
{id=24 times=(150, 150) maq=3 }	8
{id=22 times=(135, 165) maq=1 }	8
{id=26 times=(60, 60) maq=2 }	9
{id=25 times=(45, 75) maq=1 }	9
{id=27 times=(150, 150) maq=3 }	9
{id=30 times=(0, 0) maq=3 }	10
{id=29 times=(240, 240) maq=2 }	10
{id=28 times=(330, 360) maq=1 }	10
{id=32 times=(120, 180) maq=2 }	11
{id=33 times=(180, 180) maq=3 }	11
{id=31 times=(210, 240) maq=1 }	11
{id=36 times=(0, 150) maq=3 }	12
{id=35 times=(180, 180) maq=2 }	12
{id=34 times=(165, 210) maq=1 }	12

Capítulo 7.
Conclusiones.

El desarrollo de la investigación se basó en encontrar mejores asignaciones de los recursos disponibles representados en máquinas para cumplir con las distintas solicitudes de servicio en un taller de mecánica automotriz. Inicialmente se adaptó un Modelo Matemático que permitiera obtener mejores soluciones, el cual resultó eficiente hasta un tamaño de 5 máquinas x 5 tareas, después de estos valores, el modelo no era capaz de encontrar soluciones óptimas en tiempos razonables (11 minutos), por este motivo se desarrolló un Algoritmo Genético (A.G.) que permitió obtener resultados confiables con instancias de tamaño mayor (20 máquinas x 20 tareas). El AG obtuvo resultados eficientes con un porcentaje de desviación promedio de 9% incluyendo probabilidades de mutación mayores al 90%.

Otro punto para concluir es que el Operador de Cruce ayuda a mejorar resultados, pero no es decisivo para acercarse a los valores óptimos ya que la experimentación mostró resultados desviaciones de 9.3% sin este operador y con probabilidad de mutación del 100%. Junto a lo anterior, el Operador de Torneo interactúa en buena parte para conseguir mejores resultados dado que al modificar este valor de 21 a 65 la desviación cambia de 13,3% a 10,5%.

Finalmente, se puede afirmar que este AG tiende a mejorar resultados una vez el Operador de Mutación sea diversificado, por tal motivo puede ser un caso interesante de estudio para futuras investigaciones que pretendan ayudar a solucionar el problema de *Open Shop.*

Capítulo 8.
Referencias Bibliográficas.

Anand, E., & Panneerselvam, R. (2015). Literature Review of Open Shop Scheduling Problems. *Intelligent Information Management*. https://doi.org/10.4236/iim.2015.71004

Anaya Fuentes, G. E., Hernández Gress, E. S., Seck Tuoh Mora, J. C., & Medina Marín, J. (2016). Solución al Problema de Secuenciación de Trabajos mediante el Problema del Agente Viajero. *RIAI - Revista Iberoamericana de Automatica e Informatica Industrial*. https://doi.org/10.1016/j.riai.2016.07.003

Beasley, J. E. (1990). A lagrangian heuristic for set-covering problems. *Naval Research Logistics (NRL)*. https://doi.org/10.1002/1520-6750(199002)37:1<151::AID-NAV3220370110>3.0.CO;2-2

Ben-Arieh, D., & Dror, M. (1991). Intelligent heuristic for FMS scheduling using grouping. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/BF01471941

Benziani, Y., Kacem, I., Laroche, P., & Nagih, A. (2014). Exact and heuristic methods for minimizing the total completion time in job-shops. *Studies in Informatics and Control*. https://doi.org/10.24846/v23i1y201404

Bishop, C., & Polya, G. (1991). How to Solve It. *The Mathematical Gazette*. https://doi.org/10.2307/3619013

Brucker, P. (2007). Scheduling algorithms. In *Scheduling Algorithms*. https://doi.org/10.1007/978-3-540-69516-5

Castillo, T. A., Díaz B., C. E., Gómez, J. D., Orduz, E. A., & Niño, M. L. (2018). Optimización del makespan en el problema de Job Shop Flexible con restricciones de transporte usando Algoritmos Genéticos. *Entre Ciencia e Ingeniería*. https://doi.org/10.31908/19098367.3820

Dorigo, M., Blum, C., Li, H., Lam, C. P., Dorigo, M., Socha, K., … Siarry, P. (2009). Swarm Intelligence: An Application of Ant Colony Optimization\n. *Ijmer*.

Dréo, J., Siarry, P., Pétrowski, A., & Taillard, E. (2006). Metaheuristics for hard optimization: Simulated annealing, tabu search, evolutionary and genetic algorithms, ant colonies,... Methods and case studie. In *Metaheuristics for Hard Optimization: Simulated Annealing, Tabu Search, Evolutionary and Genetic Algorithms, Ant Colonies,... Methods and Case Studies*. https://doi.org/10.1007/3-540-30966-7

Framinan, J. M., Leisten, R., & García, R. R. (2014). Manufacturing scheduling systems: An integrated view on models, methods and tools. In *Manufacturing Scheduling Systems: An Integrated View on Models, Methods and Tools*. https://doi.org/10.1007/978-1-4471-6272-8

Gendreau, M., & Potvin, J.-Y. (2010). Variable Nneighborhood search (chapter). *Handbook of Metaheuristics*.

Gestal, M., Dorado, J., Pazos, A., Gómez-Carracedo, M. P., Andrade, J. M., Fernández, E., & Prada, D. (2005). Selection of variables by genetic algorithms to classify apple beverages by artificial neural networks. *Applied Artificial Intelligence*.

Glover, F., & Kochenberger, G. A. (2003). Handbook of Metaheuristics International Series in Operations Research & Management Science. In *Kluwer Academic*

Publishers.
Goldberg, D. E., & Holland, J. H. (1988). Genetic Algorithms and Machine Learning. *Machine Learning*. https://doi.org/10.1023/A:1022602019183
Gonzalez, T., & Sahni, S. (1976). Open Shop Scheduling to Minimize Finish Time. *Journal of the ACM (JACM)*. https://doi.org/10.1145/321978.321985
Guéret, C., & Prins, C. (1999). A new lower bound for the open-shop problem. *Annals of Operations Research*. https://doi.org/10.1023/a:1018930613891
Holland, J. H. (1975). Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence. In *Ann Arbor University of Michigan Press 1975*.
Huang, Z., Zhuang, Z., Cao, Q., Lu, Z., Guo, L., & Qin, W. (2019). A survey of intelligent algorithms for open shop scheduling problem. *Procedia CIRP*. https://doi.org/10.1016/j.procir.2019.04.118
Jain, H., & Deb, K. (2014). An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*. https://doi.org/10.1109/TEVC.2013.2281534
Jiménez-Builes, J. A., Arango-Sanchez, R. E., & Jiménez-Pinzón, L. D. (2016). Métodos de búsqueda usando los algoritmos de enjambre de partículas y genético. *Lámpsakos*. https://doi.org/10.21501/21454086.1901
Kolen, A. W. J., Lenstra, J. K., Papadimitriou, C. H., & Spieksma, F. C. R. (2007). Interval scheduling: A survey. *Naval Research Logistics*. https://doi.org/10.1002/nav.20231
Ku, W. Y., & Beck, J. C. (2016). Mixed Integer Programming models for job shop scheduling: A computational analysis. *Computers and Operations Research*. https://doi.org/10.1016/j.cor.2016.04.006
Kubiak, W., Sriskandarajah, C., & Zaras, K. (1991). A Note On The Complexity Of Openshop Scheduling Problems. *INFOR: Information Systems and Operational Research*. https://doi.org/10.1080/03155986.1991.11732176
Kusiak, A. (1985). Flexible manufacturing systems a structural approach. *International Journal of Production Research*. https://doi.org/10.1080/00207548508904765
Liao, C. J., & You, C. T. (1992). An improved formulation for the job-shop scheduling problem. *Journal of the Operational Research Society*. https://doi.org/10.1057/jors.1992.162
Liaw, C. F. (2003). An efficient tabu search approach for the two-machine preemptive open shop scheduling problem. *Computers and Operations Research*. https://doi.org/10.1016/S0305-0548(02)00124-7
Lin, H. T., Lee, H. T., & Pan, W. J. (2008). Heuristics for scheduling in a no-wait open shop with movable dedicated machines. *International Journal of Production Economics*. https://doi.org/10.1016/j.ijpe.2007.01.005
Liu, C. Y., & Bulfin, R. L. (1987). Scheduling ordered open shops. *Computers and Operations Research*. https://doi.org/10.1016/0305-0548(87)90029-3
MacGregor, J. N., & Chu, Y. (2011). Human Performance on the Traveling Salesman and Related Problems: A Review. *The Journal of Problem Solving*.

https://doi.org/10.7771/1932-6246.1090
Manne, A. S. (1960). On the Job-Shop Scheduling Problem. *Operations Research*. https://doi.org/10.1287/opre.8.2.219
Prins, C. (1994). An Overview of Scheduling Problems Arising in Satellite Communications. *The Journal of the Operational Research Society*. https://doi.org/10.2307/2584452
Rahmani Hosseinabadi, A. A., Vahidi, J., Saemi, B., Sangaiah, A. K., & Elhoseny, M. (2019). Extended Genetic Algorithm for solving open-shop scheduling problem. *Soft Computing*. https://doi.org/10.1007/s00500-018-3177-y
Salazar-Hornig, E., & Medina, S. J. C. (2013). Minimización del makespan en máquinas paralelas idénticas con tiempos de preparación dependientes de la secuencia utilizando un algoritmo genético. *Ingeniería, Investigación y Tecnología*. https://doi.org/10.1016/s1405-7743(13)72224-8
Schuurman, P., & Woeginger, G. J. (1999). Approximation algorithms for the multiprocessor open shop scheduling problem. *Operations Research Letters*. https://doi.org/10.1016/S0167-6377(99)00005-X
Sivanandam, S. N., & Deepa, S. N. (2007). Genetic Algorithm Implementation Using Matlab. In *Introduction to Genetic Algorithms*. https://doi.org/10.1007/978-3-540-73190-0_8
Sivanandam, S. N., & Deepa, S. N. (2008). Introduction to genetic algorithms. In *Introduction to Genetic Algorithms*. https://doi.org/10.1007/978-3-540-73190-0
Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*. https://doi.org/10.1016/0377-2217(93)90182-M
Yin, Y., Stecke, K. E., & Li, D. (2018). The evolution of production systems from Industry 2.0 through Industry 4.0. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2017.1403664