CIMAT

Centro de Investigación en Matemáticas, A.C.

OPTIMIZACIÓN DE LA SECUENCIA DE PRODUCCIÓN POR MEDIO DE UN ALGORITMO GENÉTICO

T E S I N A

Que para obtener el grado de

Maestro en Modelación Y optimización de Procesos

Presenta

Demetrio Rafael Briones Briones

Director de Tesis:

Dr. Jorge Raúl Pérez Gallardo

Autorización de la versión final

Aguascalientes, Ags., 18 de Febrero de 2021

DEDICATORIA

A mis familiares.

A mi hermano Roberto Arista Briones por ser el ejemplo de un hermano mayor y del cual aprendí aciertos y de momentos difíciles; a mi mamá Amanda Marina Briones Flores, a mi papá Demetrio Briones Arista por siempre apoyarme; a Iveth Hernández Chávez por brindarme su apoyo durante todo este tiempo y a todos aquellos que participaron directa o indirectamente en la elaboración de esta tesina.

¡Gracias a ustedes!

A mis amigos.

Que nos apoyamos mutuamente en nuestra formación profesional y que hasta ahora, seguimos siendo amigos: Guillermo Linares, Carlos Prieto, Juan Roman, Luis Rayon, Saul Domínguez, Yair Trejo.

Finalmente a los maestros, aquellos que marcaron cada etapa de nuestro camino universitario, y que me ayudaron en asesorías y dudas presentadas en la elaboración de la tesis.

Agradecimientos

A Dios por darme vida, salud y sabiduria.

A CONACyT por el apoyo economico brindado durante mis estudios de maestría en el CIMAT A.C. Unidad Aguascalientes.

A CIMAT A.C. y a la ESFM del IPN por todo el apoyo y conocimiento que me brindaron durante todos estos años que han sido escenciales para crecer profesionalmente y concluir esta estapa en mi vida.

También quiero agradecer a toda mi familia y mis amigos por todo el apoyo y amor brindado, en especial a mis padres que siempre estuvieron ante cualquier situación.

Al Dr. Raul Pérez Gallardo por haber aceptado ser mi director de tesina y haberme brindado asesoría durante este tiempo.

A los profesores de CIMAT por haber haberme brindado su apoyo y haber compartido conmigo su tiempo, sus conocimientos, su experiencia a través de este programa de maestría.

Índice general

Índice de cuadros **6**

Índice de figuras **7**

1. Introducción **10**
1.1. Antecedentes 10
1.2. Planteamiento del problema 12
1.3. Justificación 13
1.3.1. Alcances y limitaciones 16
1.4. Hipótesis 16
1.5. Objetivo general 16
1.5.1. Objetivos específicos 16

2. Marco teórico **18**
2.1. Programación y secuenciamiento de operaciones 18
2.1.1. Análisis combinatorio y complejidad computacional . . 21
2.1.2. Configuración Flexible Job Scheduling Problem 22
2.2. Métodos exactos de optimización 22
2.2.1. Modelo de programación entera mixta para el problema de secuenciamiento y programación de operaciones de corte de alambre 24
2.3. Algoritmos heurísticos usados en optimización 26
2.4. Algoritmos genéticos 27
2.4.1. Representación de individuos 29
2.4.2. Operadores genéticos 30
2.4.3. Selección 31
2.4.4. Cruzamiento 32

2.4.5. Mutación 33
2.4.6. Reemplazo 34
2.5. Prueba de hipótesis 34
2.5.1. Prueba de locación para dos muestras 35
2.6. Eficiencia y frente de Pareto 35
2.7. Software informático 36
2.7.1. Lenguaje Python 2.7 37
2.7.2. Lenguaje R 37
2.7.3. Software GAMS 38

3. Marco metodológico 39
3.1. Las instancias 39
3.1.1. Instancias reales 39
3.1.2. Instancias hipoteticas 39
3.2. Algoritmo genético para optimización de secuenciación 41
3.2.1. Función objetivo 42
3.2.2. Representación de los individuos 44
3.2.3. Ajuste de parámetros fijos 45
3.2.4. Operadadores genéticos 46
3.2.5. Selección de operadores genéticos 52
3.3. Estabilidad del algoritmo 56
3.4. Prueba del parámetro α 57

4. Análisis de resultados y conclusiones 58
4.1. Resultados de instancias reales 59
4.2. Resultados de instancias hipotéticas 60
4.3. Discusión de resultados y conclusiones 60

A. Instancias 63

B. Diagramas de Gantt 73

Bibliografía 79

Índice de cuadros

1.1. Autores y algoritmos propuestos 12
1.2. Factibilidad de Máquinas - Calibres 14

3.1. Construcción de instancias (tabla de pedidos) 40
3.2. Características de las instancias hipotéticas 41
3.3. Resultados de pruebas de hipótesis 56

4.1. Resultados de instancias reales 59
4.2. Resultados de instancias hipotéticas 60

Índice de figuras

1.1. Ejemplo de secuenciamiento de operaciones 15

2.1. Órdenes de complejidad computacional 21
2.2. Ejemplo de problema de secuenciamiento en configuración tipo FJSP 22
2.3. Ejemplo de cruza en un punto 33
2.4. Ejemplo de cruza uniforme 33
2.5. Gráfico de eficiencia de una función con dos objetivos 36

3.1. Resultados de algoritmos 53
3.2. Resultados normalizados de algoritmos 54
3.3. Desempeño del algoritmo $SR_CU_M2_RC$ 55
3.4. Desempeño del algoritmo $SB_CP_M2_RT$ 55
3.5. Frente de Pareto entre C_{max} y $Makespan$ 57

A.1. Factibilidad de operación para las 10 instancias reales, la instancia de prueba y las instancias hipotéticas Inst_M15_J5, Inst_M15_J10, Inst_M15_J15 y Inst_M15_J20 63
A.2. Pedidos de instancia de prueba 64
A.3. Pedidos de instancia real 1 65
A.4. Pedidos de instancia real 2 65
A.5. Pedidos de instancia real 3 65
A.6. Pedidos de instancia real 4 65
A.7. Pedidos de instancia real 5 66
A.8. Pedidos de instancia real 6 66
A.9. Pedidos de instancia real 7 66
A.10. Pedidos de instancia real 8 66

A.11.Pedidos de instancia real 9 67
A.12.Pedidos de instancia real 10 67
A.13.Factibilidad de operación para instancia hipotética Inst_M5_J5 67
A.14.Pedidos de instancia hipotética Inst_M5_J5 67
A.15.Factibilidad de operación para instancia hipotética Inst_M5_J10 68
A.16.Pedidos de instancia hipotética Inst_M5_J10 68
A.17.Factibilidad de operación para instancia hipotética Inst_M5_J15 68
A.18.Pedidos de instancia hipotética Inst_M5_J15 69
A.19.Factibilidad de operación para instancia hipotética Inst_M10_J5 69
A.20.Pedidos de instancia hipotética Inst_M10_J5 69
A.21.Factibilidad de operación para instancia hipotética Inst_M10_J10 70
A.22.Pedidos de instancia hipotética Inst_M10_J10 70
A.23.Factibilidad de operación para instancia hipotética Inst_M10_J15 70
A.24.Pedidos de instancia hipotética Inst_M10_J15 71
A.25.Pedidos de instancia hipotética Inst_M15_J5 71
A.26.Pedidos de instancia hipotética Inst_M15_J10 71
A.27.Pedidos de instancia hipotética Inst_M15_J15 72
A.28.Pedidos de instancia hipotética Inst_M15_J20 72

B.1. Solución instancia real 1 73
B.2. Solución instancia real 2 73
B.3. Solución instancia real 3 73
B.4. Solución instancia real 4 74
B.5. Solución instancia real 5 74
B.6. Solución instancia real 6 74
B.7. Solución instancia real 7 74
B.8. Solución instancia real 8 75
B.9. Solución instancia real 9 75
B.10.Solución instancia real 10 75
B.11.Solución instancia hipotética Inst_M5_J5 75
B.12.Solución instancia hipotética Inst_M5_J10 76
B.13.Solución instancia hipotética Inst_M5_J15 76
B.14.Solución instancia hipotética Inst_M10_J5 76
B.15.Solución instancia hipotética Inst_M10_J10 76
B.16.Solución instancia hipotética Inst_M10_J15 77
B.17.Solución instancia hipotética Inst_M15_J5 77
B.18.Solución instancia hipotética Inst_M15_J10 77

B.19.Solución instancia hipotética Inst_M15_J15 77
B.20.Solución instancia hipotética Inst_M15_J20 78

Capítulo 1

Introducción

1.1. Antecedentes

El problema clásico de programación de tareas en configuración Job Shop Scheduling (JSSP) tiene la característica principal que consiste en que cada trabajo o tarea puede ser elaborado de diferentes formas. Esta configuración consiste en un programar con N tareas a realizar en M máquinas. Este tipo de problema tiene muchas variantes, la cual depende directamente del problema que se busca solucionar. Algunas de las variantes son: Job Scheduling Problem (JSP), Flow Shop Scheduling Problem (FSP), Felixbles Job Shop Scheduling Problem (FJSSP), Flexible Job Scheduling Problem (FJSP), etc. Generalmente se ha encontrado que las investigaciones determinan como criterio de optimalidad para este problema la minimización del tiempo máximo de conclusión de todas las tareas también conocido como $Makespan$. Otros criterios de optimalidad son la minimización de tiempos de ajuste y la minimización del tiempo total de utilización de máquinas el cual llamaremos C_{max}, la diferencia entre optimizar el C_{max} y el $Makespan$ es que el primero se refiere a minimizar el tiempo de la máquina que tenga el mayor tiempo de utilización y el segundo se refiere a minimizar la sumatoria de los tiempos de todas las máquinas. El Makespan fue presentado por primera vez por los investigadores Fisher y Thompson (1963); y desde entonces este tipo de problema ha sido un tema importante en la programación de tareas y secuenciación de operaciones. Además, Morton y Pentico (1993) y Rinnooy Kan (1976); desarrollan un estudio para este problema donde se desmuestra de que el problema es NP-hard, es decir, las problemáticas de este tipo cuentan con una complejidad

computacional muy grande, esto debido a que el crecimiento de soluciones posibles aumenta de forma exponencial con respecto a la magnitud del problema.

Durante los ultimos años un considerable número de investigaciones han sido enfocadas en solucionar el problema en configuración JSSP, una variante del JSSP es el FJSP (Flexible Job Scheduling Problem). El FJSP es una variante del problema clásico de programación de Job Shop que permite que cualquier máquina procese una operación de un conjunto determinado, esto se refiere a que no todas las máquinas pueden realizar cualquier tipo de operación, así, imponiendo restricciones de producción y otra consideración de este tipo de problemas es que los trabajos son unicos e independientes. El problema consiste en adjudicar cada una de las operaciones a una máquina específica y al mismo tiempo secuenciar todas las operaciones que van a realizarse en las máquinas con el único objetivo de realizar la minimización del tiempo máximo de utilización conocido en la literatura como makespan.

El problema de secuenciación en configuración Flexible Job Scheduling (FJSP) presenta una serie de variantes, a diferencia del problema en configuración Job scheduling (JSP) que depende directamente de la naturaleza del caso de estudio y el tipo de comportamiento del ambiente donde se desenvuelve el mismo; considerando tanto las operaciones que se realizarán como el comportamiento de las máquinas. La configuración FJSP es una de las variantes más difíciles de resolver en la actualidad, debido a su alta complejidad computacional, esta problemática es donde todas las tareas pueden ser o no dependientes entre sí y las máquinas disponibles para procesar las tareas tienen características diferentes. En esta variante cada tarea requiere tiempos distintos de ejecución en cada máquina.

Han sido propuestos numerosos métodos concentrados en técnicas conocidas como algoritmos iterativos de búsqueda local o meta-heurísticas para resolver diferentes variantes del problema JSSP, como lo son: heurísticas constructivas [26], Búsqueda tabú [23], Algoritmos evolutivos [20], Algoritmos genéticos [17], entre otros. Cada uno de estos métodos varían dependiendo de las restricciones propuestas y del objetivo mismo.

Entre las propuestas de algoritmos para dar solución a este tipo de problemas podemos notar que se tienen diferentes tipos de metodologías y repre-

sentaciones de la solución. Las representaciones de las soluciones básicamente se reducen a dos tipos: representación vectorial y matricial. Estas representaciones ayudan a definir una secuenciación de tareas que pueden ser plasmadas de forma sencilla en un diagrama de Gantt, esto con el fin de facilitar el entendimiento de los resultados.

En el Cuadro 1.1 se mostrarán algunas metodologías usadas para dar solución a problemas similares al que se resuelve en este trabajo.

Autor	Algoritmo	Problema	Aplicación	Representación
Wannaporn et. al (2012)	Algoritmo genético modificado	FJSSP	benchmark	Vectorial
Khaled et. al. (2004)	Algoritmos evolutivos	JSSP	benchmark	Matricial
Ferdinando et. al. (2000)	Búsqueda tabú	JSSP	benchmark	Vectorial
Juan et. al. (2015)	Algoritmo genético	FFSSP	benchmark	Vectorial
Carlier et. al. (1989)	Algoritmo genético	JSSP	benchmark	Vectorial
Pedro et. al. (2013)	heurística constructiva	JSSP	Industrial	Vectorial

Cuadro 1.1: Autores y algoritmos propuestos

1.2. Planteamiento del problema

Con base en un problema de índole industrial que se presenta en el área de corte de alambre de cierta empresa ubicada en la ciudad de Aguascalientes, Ags, México se aborda un problema de programación de la producción de productos de alambre.

El problema surge en un proceso de corte de alambre que se usa para la fabricación de exhibidores comerciales para diversas empresas de renombre internacional como Sabritas©, Coca-Cola©, Barcel©, entre otras. En esta área se requiere programar de forma adecuada la secuencia en la que varios pedidos se procesarán en las diferentes máquinas que se tienen disponibles operando en paralelo. Se cuenta con 15 máquinas, cada una de estas máquinas cuenta con diferentes especificaciones, lo que indica que no todas las máquinas pueden realizar el proceso de corte con todos los calibres de alambre que se trabajan y que la velocidad de producción de las mismas es diferente dependiendo del tipo de calibre de alambre que se requiera cortar. Se tienen 17 diferentes tipos de calibres de alambre. Cada pedido tiene diferente longitud de corte,

diferente tipo de calibre de alambre y diferente cantidad de piezas a producir. Entre pedidos se tienen tiempos de reconfiguración de máquinas conocidos como tiempos de preparación o bien tiempos de set up, estos tiempos van a variar dependiendo de la secuencia de realización de los trabajos, los cuales van a depender directamente de los tipos de calibre y las longitudes de corte de cada pedido.

En este trabajo se considera que las máquinas trabajan de manera constante, refiriendonos a que la velocidad de corte no varía con el tiempo. Además, se considera que estas máquinas estan trabajando de forma correcta lo cuál evita fallas en el sistema, por lo que no se toman en cuenta tiempos de fallo y mantenimiento. Con respecto a los trabajos que deben ser asignados irán listados en orden de prioridad refiriéndonos a que el trabajo j no puede realizarce antes que el trabajo i, donde $j > i$ en la misma máquina, sin embargo, el pedido j puede iniciar antes que el i o en el mismo instante de tiempo en diferentes máquinas en caso de ser posible.

A continuación se muestra el Cuadro 1.2, dónde se ilustran los tipos de calibre de alambre y que máquinas pueden procesar estos diferentes tipos de calibres.

1.3. Justificación

El desarrollo actual de los métodos heurísticos y meta-heurísticos, principalmente se ve enfocado en los algoritmos genéticos ya existentes en la literatura para el problema de tipo Job Shop minimizando el makespan. El uso de algoritmos genéticos es eficiente debido a que nos ayuda a reducir tiempos de cómputo en problemas complejos asegurando la convergencia de nuestro objetivo. Un ejemplo de cómo un algoritmo genético funciona bien en este tipo de poblematicas es un caso basado en la industria de la indumentaria.

En está ocasión se usa un algoritmo genético para solucionar especificamente el problema descrito anteriormente, esto considerando la naturaleza del problema presentado en el proceso de corte de alambre.

	Calibres																
Máquinas	17	$12\frac{1}{4}$	12	11	$9\frac{1}{2}$	9	8	7	6	4	$3\frac{1}{4}$	3	2	$\frac{6}{16}$	$\frac{5}{16}$	$\frac{3}{16}$	“000
1													X	X			X
2										X	X	X	X				
3											X			X	X		
4		X	X		X	X			X							X	
5	X		X							X	X	X					
6		X	X						X			X				X	
7		X	X	X		X										X	
8									X								
9					X	X			X							X	
10								X	X		X	X				X	
11								X	X							X	
12				X	X	X	X										
13			X	X			X	X	X							X	
14				X	X												
15				X	X												

Cuadro 1.2: Factibilidad de Máquinas - Calibres

Este proyecto trae consigo dos beneficios, primeramente se ve beneficiada a la empresa y por otra parte se ve beneficiada la sociedad. Debido a que, al realizar la optimización de la secuencia de producción del proceso de corte de esta empresa, tendremos reducción en el tiempo de utilización de las máquinas, así, beneficiando a la sociedad disminuyendo la contaminación ambiental producida por las máquinas, ya que al reducir el tiempo de utilización de las máquinas se verá reducido el consumo de energía, por ende, reduciendo la huella de carbono producida por el uso de las mismas. También se observa que la empresa se ve beneficiada disminuyendo los costos de operación del proceso de corte de alambre.

A continuación se ilustra mediante figuras la naturaleza del problema, la Figura 1.1 representa un ejemplo de cómo podría realizarse la programación de 6 operaciones diferentes en las máquinas disponibles de la empresa (Parte superior), después de un proceso de optimización el tiempo de finalización de trabajos se reduce (Parte inferior).(Ver Figura 1.1).

Este proceso se modela y resuelve específicamente con un algoritmo genético que consiste en evolucionar una población de individuos sometiéndola a un conjunto de acciones aleatorias, haciendo referencia a las acciones que se lle-

van a cabo en la evolución biológica. Entre estas acciones se someten procesos conocidos como operadores genéticos los cuales realizar procedimientos de cruzamiento entre individuos, así combinando sus genes, y realizando mutaciones de forma aleatoria, esto con el motivo de buscar una buena solución en un tiempo de cómputo considerable.

Para poder modelar este proceso, se debe recabar información sobre el mismo, como datos de tipos de pedidos que realizan, las especificaciones sobre capacidad de producción de cada máquina y también obtener tiempos de reconfiguración de máquina que existe entre tareas.

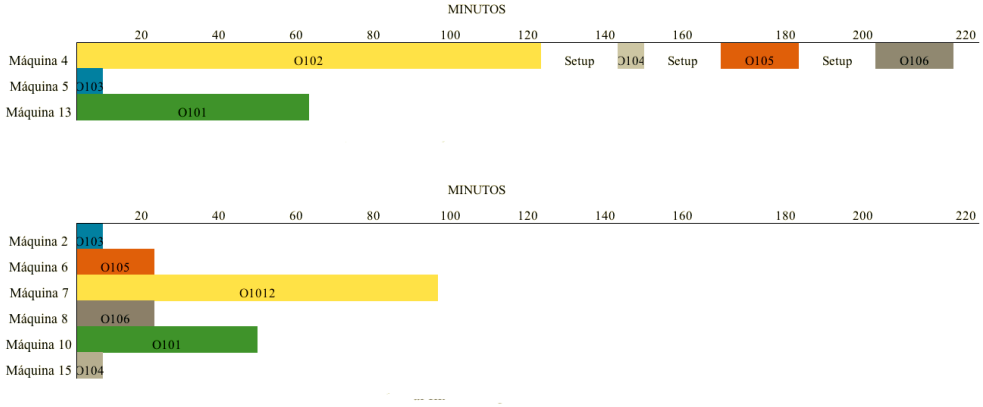

Figura 1.1: Ejemplo de secuenciamiento de operaciones

Como resultado obtendremos un algoritmo que ayuda a reducir los tiempos muertos entre trabajos, este algoritmo puede ser utilizado en procesos que cumplan con características parecidas a este proceso. También se tiene un beneficio económico y ambiental, reduciendo el consumo de energía electrica. Como se puede ver en la Figura 1.1, el diagrama de Gantt de la parte superior se muestra una posible solución factible, la cual no es la mejor y en la parte inferior se muestra cuál es una mejor solución, esto considerando que, al usar ésta asignación de tareas se reduce el $Makespan$ y el C_{max} debido a que las tareas se terminan en menor tiempo y se reducen los tiempos de reconfiguración

1.3.1. Alcances y limitaciones

El alcance del proyecto es realizar el modelamiento del algoritmo genético para optimizar un problema de tipo FJSP, se programa en Python 2.7 y se aplica en el proceso de corte de alambre de una industria de Aguascalientes.

La limitante de este proyecto es que se aplica solamente para ese proceso en específico, no pudíendose replicar en escenarios diferentes al FJSP.

1.4. Hipótesis

Como se sabe, el problema de tipo FJSP es considerado NP-hard, Morton y Pentico (1993) y Rinnooy Kan (1976); por lo cual es complicado dar una solución exacta al enfrentarse a un problema de gran magnitud. En esta ocasión se programa un algoritmo genético buscando minimizar el tiempo máximo de finalización de procesamiento de tareas ($Makespan$) y el tiempo total de ultilización de máquinas (C_{max}) en específico del proceso de corte de alambre y se compara con un modelo matemático que solo busca minimizar el $Makespan$, esto con el fin de notar la importancia de la optimización estocástica y los beneficios que trae consigo la optimización multi objetivo. Y también hacer notar la importancia de la optimización de procesos en la industria, aplicando la metodología desarrollada en un proceso industrial. En base a lo anterior se establecerá como hipótesis la siguiente sentencia:

"Se puede mejorar el secuenciamiento de trabajos-máquinas mediante el uso del algoritmo genético propuesto, tomando como base el valor óptimo encontrado con un modelo de programación matemática de un solo objetivo"

1.5. Objetivo general

Modelar y programar un algoritmo genético para la optimización de problemas tipo FJSP (Flexible Job Scheduling Problem).

1.5.1. Objetivos específicos

- Aplicación de algoritmo genético en la programación de producción en el proceso de corte de alambre de una empresa de Aguascalientes.

- Comparación de algoritmo genético contra modelo de programación matemática de un solo objetivo.

Capítulo 2

Marco teórico

2.1. Programación y secuenciamiento de operaciones

La programación y secuenciamiento de operaciones se sabe que es un proceso que consiste en la elección de diversas opciones dentro de un flujo de trabajo, este mismo es usado comúnmente en diversas compañías pertenecientes a la industria manufacturera y de servicios. En general, se trata de buscar y definir de forma correcta que recursos serán usados para la realización de operaciones específicas en un lapso de tiempo, de esta manera lograndro realizar la optimización de un objetivo establecido.

En este tipo de problemáticas los recursos a utilizar y las tareas a realizar son creadas de maneras diversas. Algunos ejemplos de recursos a utilizar podrían ser las máquinas que operan en cierto lugar de trabajo, los muelles de carga que se encuentran dentro de un centro de distribución, las cuadrillas de personal, entre muchos otros. Por otro lado, las tareas pueden definirse como las operaciones que serán realizadas dentro del proceso de producción, las salidas y las llegadas de productos o materiales en un CEDIS, las diferentes estapas que constituyen un proyecto de desarrollo tecnológico, entre otras. Dependiendo la naturaleza del problema, cada tarea puede ser definida un nivel de prioridad de realización sobre de otras dentro del mismo flujo de trabajo, por ejemplo, todas aquellas actividades que se pueden terminar más rapido, o bien, las que tienen un tiempo de expiración muy corto. Por la parte del

objetivo que se busca optimizar puede también tomar diferentes formas. Entre esas formas se suele hacer la minimización del tiempo requerido para concluir en su totalidad las tareas a realizar y otro sería la minimizción de la diferencia que hay entre el tiempo de completación de las tareas y la fecha de expiración.

Los problemas relacionados con la programación de la producción y secuenciamiento de operaciones son unos de los temas más retadores y complicados para la administración, estos probelmas han sido formulados a través del tiempo debido a su gran importancia e impacto que produce hacia las pequeñas y grandes industrias, por tanto, en la literatura existe una gran variedad de formulaciones de este tipo problemas y metodologías desarrolladas para la solución, esto con el fín de lograr diferentes objetivos. Su aplicación a configuraciones productivas y secuenciamiento de operaciones son cada vez más particulares y complejas.

Uno de los casos particulares más importantes utilizados para los diferentes sistemas productivos es la programación y secuenciamiento de tareas, también conocido como Job Shop Scheduling Problem (JSSP), el cuál podría traducirse como el problema de programación de operaciones dentro de un taller, el JSSP se puede ver como la forma general de problemáticas que se enfoca al secuenciamiento de operaciones.

En la actualidad existen diferentes opiniones sobre el origen de los problemas de tipo JSSP y aunque no está claro quién han sido el incursor en el tema, se conoce que el libro "Industrial Scheduling", realizado en 1963 por Muth y Thompson [21], contribuye directamente en la base teórica de investigaciones que siguieron sobre este tema [22].

La programación y secuenciamiento de las operaciones juega un papel relevante tanto parte de procesos de manufactura y de producción como en la mayor parte de sistemas de procesamiento de la información. De igual manera se reconoce la importancia que tiene en el área de logística, específicamente en el transportamiento y la distribución de productos, esto sin dejar fuera a otro tipo de compañias que ofrecen distintos servicios como las telecomunicaciones. Para poder demostrar dicho valor a continuación se menciona un ejemplo:

Considere una industria que se dedica a la producción de botellas de plas-

tico para bebidas carbonatadas, aguas, lacteos, etc. La materia prima más importante para la creación de este tipo de botellas es el plático. El proceso para la producción de estos productos consiste en tres etapas: la impresión de las etiquetas, la producción de botellas, y la producción de las tapas para las botellas. Cada una de estas etapas consiste en la operación de un número de máquinas la cuales pueden ser o no iguales. Las máquinas pertenecientes a cada etapa pueden ser diferentes en su velocidad de elaboración de operaciones, el número de imagenes que pueden plasmar sobre las etiquetas, o la variación en el tamaño y la forma de la botellas que pueden ser producidas. Cada una de las ordenes a despachar, nos especifica el tipo producto y la cantidad que debe ser producida y enviada de acuerdo a una fecha pactada con el cliente. Los tiempos para realizar el procesamiento de las diferentes tareas dentro del flujo de producción, son directamente proporcionales a las cantidades solicitadas de cada producto en el pedido, es decir, al número de botellas solicitadas por el cliente.

Realizar una entrega después de la fecha pactada con el cliente puede ocasionar una serie de penalizaciones, una de estas puede ser la pérdida de credibilidad e imagen de la empresa y otra puede ser una penalización impuesta directamente por el cliente en el trato pactado que dependera de la importancia de la orden. Un objetivo de la programación y secuenciamiento de tareas puede ser la minimización de estas penalizaciones.

Los sistemas en configuración tipo Job Shop son aquellos que se distinguen por la producción de una gran diversidad de productos en pequeños lotes, estos productos son los que pueden tener una secuencia diferente a la hora de ser elaborados, estos diferentes secuenciamientos pueden ser tanto en el orden de elaboración de las operaciones como en el flujo a través del proceso de elaboración, en su mayoría este tipo de sistemas funcionan bajo el esquema "Make To Order"(MTO), esto significa que se producen los productos especificados en el pedido realizado por el cliente.

Las características que se mencionaron anteriormente, hacen que los sistemas de configuración tipo Job Shop sean altamente complejos para la programación y secuenciamiento de tareas u operaciones. La importancia de los sistemas de este tipo se ve incrementada debido a que en su mayoría, las pequeñas y medianas empresas, suelen trabajar con este tipo de enfoque de

producción.

2.1.1. Análisis combinatorio y complejidad computacional

La dimensionalidad de los sistemas de tipo Job Shop Scheduling Problem está dada de la forma nxm, siendo n el número de pedidos a despachar y m el número de máquinas disponibles para operar, de tal forma que al menos tenemos $n*m$ soluciones factibles, así, generándose un problema combinatorio muy grande al momento de crecer los valores de n y de m,. El crecimiento del número de soluciones es exponencial con lo cual se dice que la complejidad computacional de este tipo de problemas resueltos por un algoritmo genético es $O(gnm)$, esto es lo que hace que los problemas de tipo JSSP sean catalogados como NP-hard [2].

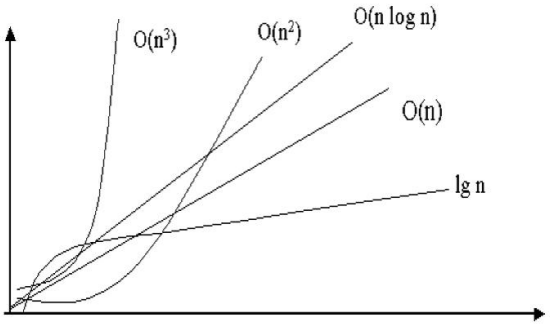

Figura 2.1: Órdenes de complejidad computacional

La enumeración de todas estas posibles soluciones para encontrar el programa o secuenciamiento óptimo no es práctico ya que es supremamente demandante de recursos computacionales. Se ha evidenciado que en este tipo de problemas se puede considerar con complejidad NP-Hard, aún siendo un caso muy básico como lo es el caso de minimización del makespan en un problema tipo Job Shop Scheduling con tres tareas a realizar y tres máquinas disponibles para operar, considerando que se permiten las paradas de producción entre operaciones. [1]

2.1.2. Configuración Flexible Job Scheduling Problem

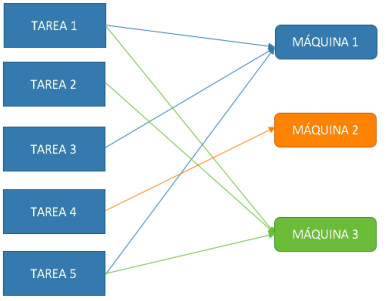

Figura 2.2: Ejemplo de problema de secuenciamiento en configuración tipo FJSP

El sistema con configuración tipo Flexible Job es una variante de la configuración Job Shop donde se tienen N tareas únicas e independientes las cuales pueden ser procesadas en M máquinas diferentes (Ver Figura 2.2). El problema básicamente es asignar cada operación de un conjunto determinado a una máquina y ordenar cada una de las operaciones asignadas a cada máquinas de modo que se uno o varios criterios de optimalidad.

2.2. Métodos exactos de optimización

Los métodos exactos para optimización son aquellos que nos permiten obtener una solución que es óptima para un problema específico. Cuando una solución es definida como óptima, es haciendo referencia a que dicha solución esta definida como la mejor solución entre todas las soluciones que son posibles para el caso de estudio, por lo tanto, se podría pensar, que los métodos que son exactos representan la mejor opción para obtener la solución de cualquier tipo de problemas de programación de tareas, pero realmente estos métodos son aplicables a instancias de tamaño pequeño, esto debido a que el tiempo computacional que se requiere para encontrar la solución de dicho problema aumenta exponencialmente cuando la dimensión del problema incrementa.

Teniendo como objetivo obtener el valor máximo o mínimo de una función

objetivo, se sabe que el primer acercamiento a una solución óptima, podría ser de forma analítica, utilizando las derivadas de la función en cuestión. Esto tomando en cuenta que indiscutiblemente la función que se desea optimizar deber ser continua, esta debe ser derivable y que su derivada también sea continua. Este tipo de métodos pueden ser utilizados, como se menciono anteriormente, para problemas pequeños que estén constituidos por funciones objetivo muy simples que esten definidas con pocas variables continuas. Por estos motivos, los métodos exactos no son recomendables para resolver los problemas de secuenciación y programación de tareas (Job Shop Scheduling Problem) [22].

Algunos de los métodos más usados para la solución de problemas combinatorios son los siguientes:

Enumeración total: este método es el más sencillo para solucionar el problema secuenciación y programación de tareas, consiste en enumerar y evaluar todas las soluciones posibles. Tomando en consideración lo anterior descrito, podemos decir que este método no puede ser considerado como eficiente debido a que la complejidad computacional del problema crece de forma exponencial.

Programación entera y entera mixta: Los problema de programación entera o de programación entera mixta se definen como aquellos problemas cuya solución óptima solo tiene sentido si algunas o todas las variables de decisión que están siendo consideradas en el modelo son capaces de tomar valores enteros. Uno de lo métodos más usados para resolver problemas que son modelados mediante la programación entera el es algoritmo de ramificación y poda. Casi la totalidad de los problemas combinatorios admiten una formulación con programación lineal o binaria, y especialmente como programación entera mixta. El inconveniente de hacer uso de este método es la gran cantidad de variables de decisión que surgen. [22]

Ramificación y poda: Este método de solución exacto es el más importante, propuesto por el investigador Bruker en 1994 [3], el cuál fue desarrollado tomando en cuenta metodologías y técnicas que fueron propuestas por investigadores como Carlier y Pinson en 1989 [5]. Este método forma parte de los algoritmos conocidos como algoritmos de enumeración implícita, estos son co-

nocidos así debido a que su objetivo es enumerar todas la soluciones factibles hasta encontrar la óptima, mediante el proceso de búsqueda se van excluyendo soluciones no óptimas, así, reduciendo la complejidad computacional, demostrando a priori su efectividad. Se sabe que en este algoritmo el universo de soluciones es dividido en ramas, a esta parte del algoritmo se le conoce como la ramificación de soluciones, cada una de estas ramas es evaluada y se siguen buscando soluciones solo sobre las ramas que son susceptibles a encontrar la solución optima, a esta fase se le conoce como proceso de poda.

2.2.1. Modelo de programación entera mixta para el problema de secuenciamiento y programación de operaciones de corte de alambre

Para el problema de programación de tareas se utiliza un modelo matemático de programación entera mixta, "los problemas de programación con enteros se formulan de la misma manera que los problemas de programación lineal, pero agregando la condición de que al menos alguna de las variables de decisión debe tomar valores enteros" [10].

El modelo de programación lineal entera mixta propuesto para esta problemática es más fácil de resolver que si fuera un modelo de programación no lineal, en este caso el uso de variables binarias elimina algunas operaciones aritméticas entre variables de decisión que lo convertirían en un problema no lineal, además del uso de variables discretas y continuas así, reduciendo la complejidad computacional del mismo. El modelo matemático propuesto por Diana Enríquez [8] para dar solución al problema de programación de la producción, fue implementado y resuelto en el lenguaje algebraico de modelación GAMS que hace uso del solucionador CPLEX.

El modelo matemático propuesto es el siguiente:

Índices

I, J : Conjunto de órdenes, $I = J = \{O_i\}$;

K : Conjunto de máquinas, $K = \{M_k\}$;

donde $i = 1, \cdots, n, j = 1, \cdots, n, k = 1, \cdots, l$.

Parámetros

t_{ik} Tiempo de procesamiento t para cada orden i en la máquina k;

a_{ij} Tiempo de $SetUp$ para cada orden i a j;

Variables de decisión

$x_{ik} = \begin{cases} 1 & \text{Si la operación } i \text{ es asignada a la máquina } k; \\ 0 & \text{Otro caso.} \end{cases}$

$y_{ijk} = \begin{cases} 1 & \text{Si se procesa la operación } i \text{ después de la } j \text{ en la máquina } k; \\ 0 & \text{Otro caso.} \end{cases}$

P_{ik} Tiempo de inicio de la orden i en la máquina k;

S_{jk} Tiempo de finalización de la orden j en la máquina k.

Función objetivo

$$minz = Makespan \tag{2.1}$$

Restricciones

$$S_{jk \leq C_{max}} \qquad \forall j, \forall k \tag{2.2}$$

$$\sum_{k=1}^{M} x_{ik} = 1 \qquad \forall i \tag{2.3}$$

$$P_{ik} + x_{ik} \cdot t_{ik} \leq S_{ik} \qquad \forall i, \forall k \tag{2.4}$$

$$S_{ik} + a_{ij} \cdot y_{ijk} \leq P_{jk} \qquad \forall i, \forall j, \forall k \tag{2.5}$$

$$y_{ijk} \geq x_{ik} + x_{jk} - 1 \qquad \forall i, \forall j, \forall k \tag{2.6}$$

La función objetivo (2.1) es minimizar el makespan dado por el $Makespan$, la restricción (2.2) asegura que se minimice el máximo de los tiempos de finalización, la restricción (2.3) asegura que una orden sea procesada solamente en una máquina, las restricciones (2.4) y (2.5) aseguran que una operación j se inicie después de que la operación i después de que se finaliza un tiempo ajuste S_{ijk}, la restricción (2.6) asegura que cuando existan dos operaciones consecutivas se agregue un tiempo de ajuste. [8]

2.3. Algoritmos heurísticos usados en optimización

Se le conoce como heurística al arte de inventar. En programación computacional se dice que un algoritmo es heurísstico cuando la solución de un problema dado no se encuentra de forma exacta, sino mediante ensayos, pruebas y reensayos.

Un método heurístico consiste en generar valores de forma aleatoria, que sean candidatos de posibles soluciones de acuerdo a un patrón establecido; luego estos candidatos son sometidos a un tipo de pruebas que de acuerdo a un criterio, sirven para caracterizar las soluciones. Si un candidato no es aceptado (infactible), se genera otro; y los pasos establecidos con el candidato anterior no son considerados. Es decir, existe inherentemente una vuelta atrás, para comenzar a generar un nuevo candidato.

Es así, que a partir de estos algoritmos heurísticos se desarrollan metodologías que ayudan a tener un eficiente y eficaz funcionamiento en la búsqueda de soluciones para resolver problemas de optimización, entre estas metodologías podemos encontrar la búsqueda tabú [23], los algoritmos evolutivos [20], los algoritmos genéticos [29], entre muchos otros.

2.4. Algoritmos genéticos

Los algoritmos genéticos fueron creados con un propósito, mejora el conjunto de soluciones generadas estratégicamente. Los algoritmos genéticos son métodos heurísticos de búsqueda apoyados en mecanismos propios del proceso de evolución y en la genética biológica. El interés por los algoritmos heurísticos creados con base en procesos físicos y biológicos parte de los años setentas cuando el investigador John Holland [11] propone por primera vez los algoritmos genéticos. Estos algoritmos son inspirados en aquellos procesos biológicos de búsqueda y selección que determinan la supervivencia de los individuos más aptos. El uso de procedimientos de búsqueda probabilística por parte de los algoritmos genéticos permiten reducir el trabajo que conlleva investigar soluciones al generar una población de individuos mediante los denominados operadores genéticos como la selección, cruza, mutación y reemplazo. Estos operadores posibilitan el intercambio de información entre las soluciones que representan cada individuo de la población, esto con el fin de buscar bien mejores individuos o bien dicho mejores soluciones.

Como bien se sabe, en la naturaleza, aquellos individuos que logran adaptarse a la competencia que se origina por la obtención de recursos generalmente escasos son los que logran sobrevivir. La adaptación a cualquier tipo de cambio que se presenta en el ambiente resulta primordial para la supervivencia de los individuos de todas las especies. Las características que son únicas y particulares de cada individuo son las que determinan sus capacidades de supervivencia, estas caracteristicas están dadas por sus contenidos genéticos. Cada una de las características del individuo está dada por una unidad básica la cual es conocida como gen. Los genes son los que controlan las características mas importantes para que un individuo pueda subsistir.

La evolución de una especie es un proceso universal que consiste en el cambio gradual. Dichos cambios se dan en la genética de la especie y estos suceden durante la selección natural y la recombinación de material genético que se da gracias a la reproducción biológica.

Solamente los individuos que sea más aptos son lo que van sobrevivir y son los que se van a reproducir. La selección natural es la que encausa la supervivencia de los individuos más aptos y esto involucra la subsistencia de

sus genes. Por otro lado, el proceso conocido como reproducción crea cierta diversidad sobre la genética de los individuos.

La evolución de la población se empieza cuando los genes de dos individuos, conocidos como padres, se combina a tavés del proceso de reproducción. Entonces a partir de los padres se crean nuevas combinaciones, así, creándose nuevo material genético de igual forma. Este proceso de intercambio de información entre genes es mejor como proceso de cruzamiento.

Entonces en base a todo lo mecionado anteriormente se dice que cada solución está asociada a un valor conocido como aptitud el cuál nos indica que tan bueno o malo es comparada con otras posibles soluciones dentro de la misma población. Los individuos que tengas una aptitud con los valores más bajos o altos, dependiendo del caso de estudio, son los que tienen una mayor probabilidad de sobrevivir y también de reproducirse en generaciones siguientes. La recombinación de los genes de los padres se simula a través de un proceso de cruzamiento que permite el intercambio de genes entre las estructuras genéticas de los individuos, esto con el fín de producir una nueva. Después de todos estos procedimientos, se sabe que en algunos entornos se lleva a cabo un proceso de mutación, la cuál causa cambios de forma esporádica y aleatoria en los elementos de la estructura genéticas que fue creada, así, creando un proceso de regeneración de perdida de información sobre el material genético que puede ocurrir a través de las generaciones. Por último, se lleva a cabo el ciclo generacional el cual se repite hasta que se cumpla un criterio establecido para detener el proceso.

El funcionamiento de un Algoritmo Genético puede apreciarse en el siguiente pseudocódigo:

- Crear población inicial aleatoriamente
- MIENTRAS no se cumpla el criterio de paro
    - Crear población temporal vacía
    - MIENTRAS la población temporal no esté llena
        - Seleccionar los padres
        - Cruzar a los padres con probabilidad Pc
        - SI se ha producido el cruce

- Mutar a los hijos con probabilidad Pm
- Evaluar a los hijos
- Añadir los hijos a la población temporal
- SI NO
- Añadir padres a la población temporal
- FIN SI
- FIN MIENTRAS
- Aumentar el contador generaciones
- Establecer como población inicial a la población temporal
- FIN MIENTRAS

2.4.1. Representación de individuos

Los algoritmos genéticos manipulan un conjunto de soluciones potenciales para encontrar la solución óptima de un problema. Esto sucede en representaciones codificadas de las soluciones, haciendo la equivalencia al material genético de los individuos. Los algoritmos genéticos codifican las soluciones de diversas formas como puede ser en cadenas de bits, números reales, letras, matrices, entre otros.

Con base a los trabajos mencionados anteriormente en el cuadro 1.1, podemos hacer mención de las siguientes representaciones usadas en diversos tipos de algoritmos, cada representación es la codificación de la solución:

- Representación 1:

$$[(O_1, O_2, O_5, O_4, O_3), (M_2, M_3, M_2, M_5, M_1)]$$

En esta representación se hace uso de un arreglo vectorial con dos componentes, la primer componente de esta representación se refieren al orden que en que se llevaran a cabo las operaciones y la segunda componente es el número de máquina donde se llevara a cabo.

- Representación 2:

$$[O_1, O_3, O_5, O_4, O_2]$$

$$[M_2, M_3, M_2, M_5, M_1]$$

En esta representación se hace uso de dos vectores, el primero es donde se tiene la secuencia en la que se llevarán a cabo las operaciones y el segundo en la máquina en la que se llevara a cabo.

- Representación 3:

$$[(1, 0, 1, 0), (0, 1, 0, 1), (0, 1, 0, 0)]$$

En esta representacion, cada componente representa la asignación de tareas a máquinas, por ejemplo, se desean asignar 3 trabajos a 4 máquinas, las componentes $(1, 0, 1, 0)$ se refieren al primer trabajo, donde se dice que se asignaras sus dos tareas a la máquina 1 y 3.

- Representación 4:

$$\begin{bmatrix} (t_{31}, M_1) & - & - \\ (t_{21}, M_2) & (t_{12}, M_1) & (t_{13}, 3) \\ (t_{11}, M_3) & (t_{22}, M_2) & - \end{bmatrix}$$

En esta representación se observa una forma matricial donde se cosideran 3 aspectos, los trabajos, las tareas que conllevan esos trabajos y las máquinas que realizan esos trabajos, Por ejemplo, en la columna 1, se refiere al trabajo 1, y cada elemento pareado de ese trabajo nos dice que tarea será realizada en que máquina.

2.4.2. Operadores genéticos

Se sabe que un operador genético, es una función utilizada en algoritmos genéticos que tienen la finalidad de conservar la diversidad genética dntro de una población.

La diversidad genética en este tipo de algoritmos es necesaria para el proceso de evolución. Los operadores genéticos que se usan en los algoritmos

genéticos son realizados analogamente a aquellos que suceden en el mundo real, entre estos operadores se encuentran: la selección de individuos, el cruzamiento entre individuos, la mutación y el reemplazo generacional.

2.4.3. Selección

Los algoritmos de selección se encargan de discriminar y escoger a los individuos que tendrán la oportunidad de reproducirse de los que no podrán.

Como se mencionó anteriormente, lo que se pretende es replicar lo que sucede en la naturaleza al otorgar a los individuos más aptos una mayor posibilidad de reproducirse. Por lo tanto, la selección de un individuo está directamente relacionada con la aptitud que lo identifica. Sin embargo, no hay que descartar por completo la posibilidad el procesos de reproducción de aquellos individuos catalogados como menos aptos con el fin de evitar que la población se vuelva homogénea después de pocas generaciones.

El proceso de selección comúnmente se realiza al elegir aquel individuo que participa en el proceso de cruzamiento mediante alguno de los métodos expuestos a continuación:

Selección por ruleta

Se asigna a cada individuo que conforma la población una parte de la ruleta proporcionalmente a su aptitud cuidando que la suma de los porcentajes sea la unidad. Cabe notar que aquellos individuos con mejor aptitud obtendrán una mayor porción de la ruleta. La selección ocurre al generar un numero aleatorio $X \sim U(0,1)$ y regresar al individuo que se encuentra ubicado en dicha posición de la ruleta. Esta posición se obtiene al recorrer a los individuos de la población acumulando sus proporciones asignadas en la ruleta hasta que se exceda el valor obtenido en la variable aleatoria.

Selección por torneo

Este método de selección realiza directamente la comparación entre un par de individuos. Existen dos versiones:

- Determinística

- Probabilística

La versión determinística, de manera aleatoria, selecciona un número n de individuos (usualmente n =2). Posteriormente, de los n individuos seleccionados se escoge al más apto para formar parte de la siguiente generación.

La versión probabilística se diferencia de la anterior en la forma de seleccionar al individuos ganador. En lugar de siempre escoger al individuo más apto, se genera número aleatorio $X \sim U(0,1)$. Este valor se compara con un parámetro Ps y si este valor es mayor se escoge al individuo más apto, en caso contrario se selecciona el individuo menos apto. De manera general, el rango de valores del parámetro Ps es 0,5 $<= P_s <= 1$.

2.4.4. Cruzamiento

Una vez que han sido seleccionados los individuos, éstos son cruzados para producir los descendientes conocidos como hijos, los cuales formarán la próxima generación. El cruzamiento es una proceso estratégico que busca replicar la reproducción sexual. Las tasa de cruzamiento que se suele usar ronda el 90 %.

Existe una gran variedad de algoritmos de cruzamiento. Sin embargo, algunos de los más utilizados son los siguientes:

Cruza de 1 punto

Este método de cruzamiento es el más sencillo de los métodos de cruzamiento existentes. Una vez que ya han sido seleccionados dos individuosson cortados en un punto determinado de manera aleatoria. El corte genera dos segmentos diferentes entre ellos denominados la cabeza y la cola. Después, se generan dos nuevos hijos al intercambiar las colas entre un par de individuos como se observa en la figura 2.3. Ambos hijos heredan la información genética de sus padres.

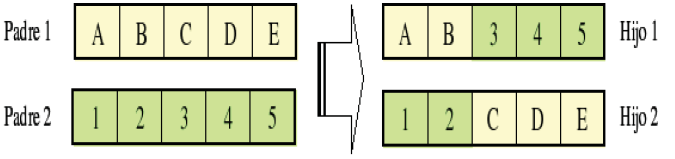

Figura 2.3: Ejemplo de cruza en un punto

Cruza uniforme

En este método de cruzamiento todos los genes de los hijos tienen las mismas posibilidades de provenir de alguno de los 2 padres. Aunque pueden implementarse de muy formas diversas, .esta técnica crea una máscara de valores binarios de cruzamiento. Si en cierta posición de la máscara se encuentra un valor de 1, se copia del primer padre el gen situado en la misma posición. En caso contrario, que haya un 0, se copia el gen ubicado en la misma posición del segundo padre.

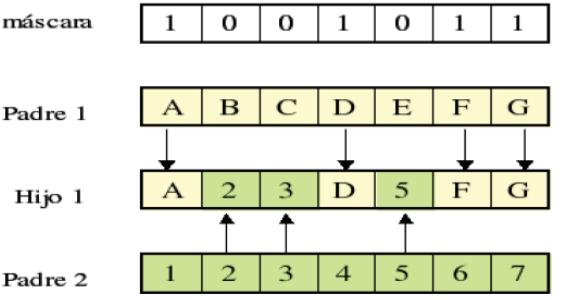

Figura 2.4: Ejemplo de cruza uniforme

2.4.5. Mutación

El proceso de mutación en individuos tiene el proposito de que aleatoriamente varíe el valor de uno de sus genes, generalmente sueles hacerse un sólo gen aunque pueden ser más.

La probabilidad de que ocurra una mutación suele ser muy baja, usualmente ronda el 1 %. Las mutaciones evitan que algún punto del espacio de búsqueda tenga una probabilidad nula de ser considerado. El remplazo aleatorio es la técnica de mutación más usada. En esta mutación un gen del cromosoma es variado aleatoreamente

2.4.6. Reemplazo

Cuando se trabaja con una población única en vez de una población temporal, sobre la que se realizan operaciones genéticas, se debe tener en cuenta que el insertar un individuo nuevo implica eliminar previamente un individuo proveniente de la población. Métodos de remplazo son los siguientes:

- Aleatorio: Un hijo se inserta de forma aleatoria en un lugar cualquiera de la población.
- Reemplazo total de padres: Todos los hijos generados llegan a reemplazar a los padres.
- Reemplazo de similares: una vez obtenido el ajuste de la descendencia se selecciona un grupo de individuos de la población con un ajuste similar. Se reemplazan aleatoriamente los que sean necesarios.
- Reemplazo de los peores: de entre un porcentaje de los peores individuos de la población se seleccionan aleatoriamente los necesarios para dejar sitio a la descendencia.
- Reemplazo con los mejores: se seleccionan de forma aleatoria un padre y un hijo y se realiza una comparación entre ellos, así, reemplanzando el padre por el hijo en caso de que la aptitud del hijo sea mejor que la del padre.

2.5. Prueba de hipótesis

Las pruebas de hipótesis se sabe que son reglas estadísticas que ayuda a saber si se puede rechazar o aceptar hechos específicos sobre una población dependiendo de las evidencias aportadas por una muestra de datos.

La prueba de hipótesis estudia dos aseveraciones confrontadas sobre una misma población denominadas hipótesis nula e hipótesis alterna. Generalmente, la hipótesis nula es la hipótesis que busca probar que no hay ningun efecto o no hay diferencia alguna. Por otro lado, la hipótesis alternativa busca concluir que es verídico a partir de la evidencia generada por los datos muestrales. En este trabajo de investigación se hace uso de una prueba de hipótesis con una prueba conocida como Test t o Prueba t de Student, esta misma se usa para saber si es significativa la diferencia entre las medias de dos muestras con diferentes tamaños, esto con el fin de poder demostrar la estabilidad del algoritmo genético desarrollado.

2.5.1. Prueba de locación para dos muestras

La prueba de locación para dos muestras, es una prueba con la cuál se puede comprobar si las medias de dos poblaciones normalmente distribuidas son idénticas. Todas estas pruebas suelen ser conocidas como prueba t de Student, aunque siendo estrictos, esta denominación debe ser empleada sí y solo sí las poblaciones que son estudiadas presentan varianza que puede considerarse como similar. En el caso de que el supuesto anterior no se cumpla, se le conoce como prueba de t de Welch. También conocida como prueba t desapareada o de muestras independientes, tiene su aplicación cuando las unidades estadísticas de ambas muestras sometidas a la comparación no se superponen.

2.6. Eficiencia y frente de Pareto

La eficiencia de Pareto, óptimo de Pareto u optimalidad de Pareto es un concepto aplicado en Economía así como en diversas áreas de la Ingeniería y en las Ciencias Sociales. El economista italiano Vilfredo Pareto da nombre a este término al realizar estudios relacionados con la eficiencia económica y la distribución de la renta.

Cuando se tiene una atribución inicial de bienes o ganancias entre un conjunto de individuos y posteriormente ocurre una nueva asignación que logra mejorar la situación actual de un individuo sin que con ello se altere de manera negativa la situación de los demás se esta enfrente de una mejora de Pareto.

El conjunto de soluciones denominadas óptimo de Pareto es conocido como frente de Pareto, este conjunto de soluciones es el que se usa para realizar una toma de decisiones, esto haciendo un análisis coste-beneficio entre diversas soluciones. Este tipo de análisis suele realizarse entre una función con dos objetivos (ver Figurra 2.5).

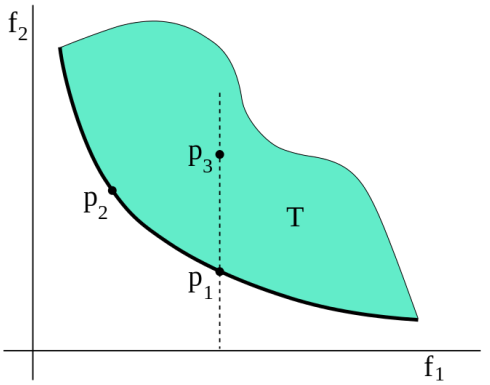

Figura 2.5: Gráfico de eficiencia de una función con dos objetivos

En ingeniería, específicamente en el área de optimización, se hace uso de esta técnica para la optimización multiobjetivo, esto con el fin de saber que configuración de entre 2 objetivos trae consigo el mejor benefico.

2.7. Software informático

Los lenguajes de programación son lenguajes formales con los cuales se puede especificar una serie de pasos a seguir para que una computadora pueda producir diversas clases de datos a través de diversos procedimientos. El uso de lenguajes de programación han permitido la creación de programas computacionales que ponen en práctica algoritmos específicos que posibilitan controlar el comportamiento físico y lógico de una computadora.

Lo programas computacionales están conformados por un conjunto de de-

notaciones, simbolos y reglas sintácticas y semánticas que definen la estructura y el significado de todos sus elementos y sus expresiones. Se le conoce como programación computacional al proceso de desarrollo por el cual se escribe, se prueba, se depura, se compila (de ser necesario) y se mantiene el código fuente de un programa computacional.

En esta ocasión como se hace el desarrollo de un programa computacional haciendo uso del lenguaje de programación Python 2.7, también se hace uso del lenguaje de programación R. El primero fue usado para la implementación del algoritmo genético y el segundo para realizar las pruebas estadísticas de estabilidad del algoritmo.

2.7.1. Lenguaje Python 2.7

Python es un lenguaje de programación interpretado de alto nivel para propositos generales de programación. Fue creado por Guido van Rossum y lanzado por primera vez en 1991, Python tiene una filosofía de diseño que enfatiza la legibilidad del código, notablemente el uso de espacios en blanco significativos. Proporciona construcciones que permiten una programación clara en escalas pequeñas y grandes. [15]

Python presenta un sistema de tipo dinámico y administración de memoria automática. Es compatible con múltiples paradigmas de programación, incluidos los orientados a objetos, imperativo, funcional y de procedimiento, y tiene una biblioteca estándar grande y completa.

Para más información sobre este lenguaje de programación puedes visitar `https://www.python.org`

2.7.2. Lenguaje R

R es un lenguaje de programación con un entorno de software libre usado para computación y gráficos estadísticos, este lenguaje es respaldado por la Fundación R para Computación Estadística. El lenguaje de programación R es ampliamente utilizado entre investigadores y desarrolladores estadísticos y mineros de datos para el desarrollo de software estadístico y análisis de datos. [6]

R es un paquete de GNU. El código fuente desarrollado para el entorno de software R está escrito principalmente en C, Fortran y R. R está disponible de forma gratuita bajo la Licencia Pública General de GNU, y se proporcionan versiones binarias precompiladas para varios sistemas operativos.
Mientras que R tiene una interfaz de línea de comando, hay varias interfaces gráficas y entornos de desarrollo integrados disponibles, un estorno muy conocido es R Studio.

Para más información sobre este lenguaje de programación puedes visitar `https://cran.r-project.org`

2.7.3. Software GAMS

GAMS es un provedor de herramientas de optimización utilizadas en la industria, el Sistema General de Modelaje Algebraico (GAMS) fue diseñado para realizar modelos de problemas de optimización y realizar programación matemática, ya sean problemas de programación lineales, no lineales o mixtos. El sistema es principalmente útil para obtener la solución de problemas que sean grandes y complejos, permite crear modelos de gran dimensionalidad que se pueden adaptar de forma sencilla a nuevas situaciones.

GAMS cuenta con más de 25 solucionadores, entre ellos podemos encontrar GUROBI, XPRESS, CPLEX, MINOS, SBB, entre otros.

Para más información sobre este software puedes visitar`https://www.gams.com`

Capítulo 3

Marco metodológico

3.1. Las instancias

En este trabajo se resuelven dos tipos instancias, las primeras instancias que se resuelven son las instancias reales otorgadas por la empresa, las segundas son instancias hipotéticas creadas a partir de las instancias reales. Se tienen 10 instancias reales y se generan 10 instancias hipotéticas.

3.1.1. Instancias reales

Estas instancias están registradas en Excel y fueron obtenidas de la empresa dónde se encuentra el caso de estudio, dentro de cada libro de Excel se tienen 2 hojas, la hoja de factibilidad que es donde se describen las máquinas disponibles y la hoja de pedidos que es donde se detallan las caracteristicas de los pedidos a realizar, estos pedidos están ordenados por prioridad, como se había comentado anteriormente, el pedido j, $\forall j > i$ no puede ser asinado antes del pedido i en la misma máquina, pero si pueden ser asignados antes o al mismo instante de tiempo o en diferentes máquinas en caso de ser posible.

3.1.2. Instancias hipoteticas

Como se comentó anteriormente, se realizan 10 intancias hipotéticas a partir de las intancias reales, estas fueron generadas siguiendo los siguientes

pasos.

1.- Seleccionar m máquinas de las 15 posibles

2.- Crear una lista de n trabajos

- Longitud de corte se asigna de forma aleatoria
  $U \sim (Min_{longitud}, Max_{longitud})$
- Calibre de corte se asigna de forma aleatoria
  $C_i \in Calibres_m, V.A.$
- Número de piezas se determina de forma aleatoria
  $U \sim (Min_{piezas}, Max_{piezas})$

A continuación se mostrará la composición de la lista de trabajos de las instancias hipotéticas:

n	5			
No. pedido	Pedido	Calibre	Medida en m	No. de Pzs.
1	J1	C_1	L_1	P_1
2	J2	C_2	L_2	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	J_n	C_n	L_n	P_n

Cuadro 3.1: Construcción de instancias (tabla de pedidos)

Donde:

C_i : El calibre del alambre del trabajo i

L_i : La longitud de corte por pieza del trabajo i

P_i : Número de piezas a realizar del pedido i

Las instancias fueron creadas con el número de trabajos y máquinas como se describe a continuación:

Nombre	No. máquinas	No. pedidos.
inst_M5_J5	5	5
inst_M5_J10	5	10
inst_M5_J15	5	15
inst_M10_J5	10	5
inst_M10_J10	10	10
inst_M10_J15	10	15
inst_M15_J5	15	5
inst_M15_J10	15	10
inst_M15_J15	15	15
inst_M15_J20	15	20

Cuadro 3.2: Características de las instancias hipotéticas

3.2. Algoritmo genético para optimización de secuenciación

En base a la teoría de los algoritmos genéticos antes descrita, se propone un algoritmo genético que sigue los siguientes pasos:

Entradas: Pedidos, factibilidad de máquinas, número de generaciones G, número de individuos N, probabilidad de cruzamiento P_c, probabilidad de mutación P_m, Error ε.

Salidas: Mejor solución encontrada para la programación de tareas

1.- Inicializar, $g = 0$, $n = N$.

2.- Crear población inicial de n individuos.

3.- Evaluar población inicial.

4.- Obtener el mejor individuo y guardar como $Elite_1$.

5.- Seleccionar individuos y crear nueva población.

6.- Cruzar los individuos seleccionados con una probabilidad de cruza P_c.

7.- Mutar los individuos cruzados con una probabilidad de mutación P_m.

8.- Evaluar los individuos mutados.

9.- Reemplazar los padres.

10.- Obtener el mejor individuo y guardar como $Elite_2$.

11.- Si $Elite_2$ es mejor que $Elite_1$, entonces $Elite_1 = Elite_2$.

12.- Calcular el error relativo: $\varepsilon_r = \frac{|Elite_1 - PromedioFO|}{Elite_1}$.

13.- $g = g + 1$

14.- Si $\varepsilon_r > \varepsilon$ o $g < G$ volver al paso 5, Si no ir al paso 15

15.- Devolver $Elite_1$ como mejor solución encontrada

16.- Terminar

3.2.1. Función objetivo

El propósito del algoritmo genético que se propone es la asignación de las N operaciones en las M máquinas, esto considerando el tiempo de procesamiento c_{im} y los tiempos de reconfiguración de máquina S_{ij} de tal forma que consiga reducirse el tiempo máximo de procesamiento de toda las operaciones ($Makespan$) y que se reduzca también el tiempo de utilización de las máquinas (C_max), así teniendo una función bi-objetivo, ésta función bi-objetivo está definida como:

$$\text{mín}\, Z = \alpha * Makesapn + (1 - \alpha) * C_{max}$$

$$0 \leq \alpha \leq 1$$

Donde a es un parámetro que se define en función de la prefencia del objetivo que se busca.

Para obtener el valor en función objetivo (aptitud) se hace uso del siguiente pseudocódigo:

Entrada: Población (padres o hijos), lista de pedidos (MatrizPedidos), número de individuos (L), número de pedidos (N), número de máquinas (M), valor del parámetro α

Salida: Lista con los valores de la función objetivo Z (SumaPonderada), lista con los valores del makespan (Cmax), lista con la suma de los tiempos de ultilización de todas las máquinas(TiempoDeUtil)

Crear una lista vacía, Cmax
Crear una lista vacía, TiempoTotalUtilizacion
Crear una lista vacía, SumaPonderad
for $i = 1, ...L$ **do**
    Crear lista vacía, IndicesOperacionesPorMaquina
    for $j = 1, ..., M$ **do**
        Crear lista vacía, indiceDummy
        Agregar el valor de j a la lista indiceDummy
        for k = 1,...,N **do**
            if $Padres[k][j] > 0$ **then**
                Agregar el valor de k a la lista indiceDummy
            end if
        end for
        Agregar los valores de la lista indiceDummy a la lista IndicesOperacionesPorMaquina
    end for
    Crear un vector nulo de tamaño M, setupTotalPorMaquina = [Tamaño M]
    for $j = 1, ..., M$ **do**
        if Tamaño de $IndicesOperacionesPorMaquina[j]) > 2$ **then**
            for k = 2,..., Tamaño de IndicesOperacionesPorMaquina[j] **do**
                if MatrizPedidos[IndicesOperacionesPorMaquina[j][k]][2] == MatrizPedidos[IndicesOperacionesPorMaquina[j][k+1]][2] **then**
                setupTotalPorMaquina[j] += 5.0
                else
                setupTotalPorMaquina[j] += 20.0

end if
end for
end if
end for
Crear una lista vacía, TiemposMaquinaTotal
for $j = 1, ..., L$ **do**
Agregar la suma de los tiempos de la máquina j más la suma de los tiempos de Setup en la máquina j a la lista TiemposMaquinaTotal
end for
Agregar el argumento máximo de la lista TiemposMaquinaTotal a la lista Cmax
Agregar la suma de los valores contenidos en la lista TiemposMaquina-Total a la lista TiempoTotalUtilizacion
Agregar la operación $[(\alpha * TiempoTotalUtilizacion[i]) + ((1-\alpha) * Cmax[i]]$ a lista SumaPonderada
end for
Regresar aptitudCmax, TiempoTotalUtilizacionPadres, SumaPonderada

3.2.2. Representación de los individuos

Para el algoritmo genético, se sabe que una parte muy importante es la codificación de los individuos, en este trabajo se propone una representación matricial, en esta representación se toma en cuenta cada reglon dentro de la matriz como un gen, así, cuando se aplique un operador genético, este debera ser considerar cada renglon como gen, a continuación se describe la propuesta de la representación de individuos.

Se tiene una población (Pob) de L individuos, dónde cada individuo consiste de N pedidos y M máquinas, mencionado lo anterior la población tiene la siguiente codificación:

$$Pob = \begin{bmatrix} x_{111}c_{11} & x_{112}c_{12} & \cdots & x_{11m}c_{1m} \\ x_{121}c_{21} & x_{122}c_{22} & & x_{12m}c_{2m} \\ \vdots & & \ddots & \vdots \\ x_{1n1}c_{n1} & x_{1n2}c_{n2} & \cdots & x_{1nm}c_{nm} \end{bmatrix}$$

$$\vdots$$

$$\begin{bmatrix} x_{l11}c_{11} & x_{l12}c_{12} & \cdots & x_{l1m}c_{1m} \\ x_{l21}c_{21} & x_{l22}c_{22} & & x_{l2m}c_{2m} \\ \vdots & & \ddots & \vdots \\ x_{ln1}c_{n1} & x_{ln2}c_{n2} & \cdots & x_{lnm}c_{nm} \end{bmatrix}$$

Dónde:

c_{nm} : Tiempo que tarda en se procesado el trabajo n en la máquina m

$$x_{lnm} = \begin{cases} 1 & \text{Si el trabajo } n \text{ del individuo } l \text{ se asigna a la máquina } m \\ 0 & \text{Otro caso} \end{cases}$$

$$\sum_{m=1}^{M} x_{lnm} = 1, \qquad n = 1, 2, \cdots, N, \qquad l = 1, 2, \cdots, L$$

El tiempo de procesamiento C_{nm} se calcula a partir de la longitud de cada pieza multiplicado por el número de piezas a producir dividido por la velocidad de procesamiento de cada máquina.

3.2.3. Ajuste de parámetros fijos

El funcionamiento de un algoritmo genético se puede ver como una exploración combinatoria que tiene un balance entre realizar la exploración en un espacio de búsqueda no explorado y la exploración en un espacio de búsqueda ya explorado. Este balance es controlado de forma crítica a través de la correcta elección de los parámetros de control, los cuales influyen directamente en el funcionamiento del algoritmo, estos parámetros de control son: la tasa de cruzamiento entre individuos, la tasa de mutación de los individuos y el tamaño de la población. La elección de un valor óptimo para los parámetros de control ha sido un incesante debate entre investigaciones empíricas e investigaciones analíticas. A continuación se mencionan 3 planteamientos sobre el balance de parámetros de contro:

- Cuando se incremente la probabilidad de cruza de forma excesiva se perturba la creación de soluciones favorables.
- Cuando se incrementa el valor de la tasa de mutación de forma excesiva, la búsqueda genética se convierte en una búsqueda sumamente aleatoria.

- Cuando se incrementa el tamaño de la población de forma excesiva se aumenta la diversidad de forma considerable y se reduce la capacidad de que el algoritmo tenga una convergencia rápida a un óptimo local, y por otro lado, también ésto ocasiona que se requiera un tiempo mayor para que la población llegue a converger a regiones óptimas.

Varios investigadores que han realizado propuestas de un conjunto de parámetros de control que pueden garantizar un buen rendimiento de los algoritmos genéticos al realizar pruebas. En base a estas premisas se ha llegado a 2 enfoques, el primero es considerar una población de tamaño relativamente reducido con tasas de cruzamiento y mutación relativamente altas. El segundo enfoque hace uso de una población grande pero usando tasas bajas de cruzamiento y mutación. Ambos enfoques se resumen en:

- Tasa de cruzamiento: 0.60, tasa de mutación: 0.001, tamaño de población 100 [7].
- Tasa de cruzamiento: 0.90, tasa de mutación: 0.01, tamaño de población 30 [13].

Entonces con base a lo anteriormente mencionado se decide fijar los parámetros de control de la siguiente forma:

- Parámetros de control del algoritmo genético
  - Tamaño de población: 200
  - Probabilidad de crzamiento: 0.80
  - Probabilidad de mutación: 0.01
- Parámetros de criterio de paro
  - Número máximo de generaciones: 250
  - Error: 0.01

3.2.4. Operadadores genéticos

En esta ocasión se proponen 8 operadores genéticos diferentes, 2 de selección, 2 de cruzamiento, 2 de mutación y 2 de reemplazo. Teniendo así 16 combinaciones posibles entre los operadores, para saber con cuál de estos

operadores genéticos se comporta mejor el algoritmo genético, se realiza una comparación entre 16 algoritmos genéticos diferentes.

Los operadores genéticos utilizados se describen a continuación.

Operadores de selección

El primer algoritmo de selección es por torneo binario, este algoritmo sigue el siguiente pseudocódigo:

Entrada: Población (padres), lista con la aptitud de los padres (aptitud), número de individuos (L)

Salida: Lista con los individuos seleccionado (sel)

Crear una lista vacía 'sel'
for i =1,...,L **do**
  Generar una variable aleatoria uniforme entera, $numb \sim U(1, L)$
  Generar una variable aleatoria uniforme entera, $numb2 \sim U(1, L)$
  if $aptitud[numb] > aptitud[numb2]$ **then**
    Agregar el individuo padres[$numb2$] a la lista sel
  else:
    Agregar el individuos padres[$numb$] a la lista sel
  end if
end for
Regresar sel

El segundo algoritmo de selección es por ruleta, este algoritmo sigue el siguiente pseudocódigo:

Entrada: Población (padres), lista con la aptitud de los padres (aptitud), número de individuos (L)

Salida: Lista con los individuos seleccionado (sel)

Crear una lista vacía 'sel'
for $i = 1, ..., L$ **do**

```
    Generar una variable aleatoria uniforme, numb ~ U(0, suma(aptitud))
    Iniciar un contador, suma_I = 0
    for j = 1, ..., L do
        suma_I = suma_I + aptitud[j]
        if suma_I > numb then
            Agregar el individuo padres[j] a la lista sel
            Detener
        end if
    end for
end for
Regresar sel
```

Operadores de cruzamiento

El primer algoritmo de cruzamiento es uniforme, este algoritmo sigue el siguiente pseudocódigo:

Entrada: Lista de individuos seleccionados (*sel*), número de individuos (L), número de pedidos (N)

Salida: Lista con los individuos cruzados (hijos)

```
Crear lista vacía 'hijos'
for i = 1, ..., L do
    Generar una variable aleatoria uniforme entera, numb ~ U(1, L)
    Agregar el individuo sel[Numb] a la lista hijos
    Generar variable aleatoria uniforme, prob ~ U(0, 1)
    if prob <= P_c then
        Generar una variable aleatoria uniforme entera, numb2 ~ U(1, L)
        for j = 1, ..., N do
            Generar variable aleatoria uniforme, tos ~ U(0, 1)
            if tos <= 0,5 then
                hijos[i][j] = sel[numb2][j]
            end if
        end for
    end if
end for
```

Regresar hijos

El segundo algoritmo de cruzamiento es en un punto, este algoritmo sigue el siguiente pseudocódigo:

Entrada: Lista de individuos seleccionados (sel), número de individuos (L), número de pedidos (N), número de máquinas (M)

Salida: Lista con los individuos cruzados (hijos)

Crear lista vacía 'hijos'
for $i = 1, ..., L$ **do**
  Generar una variable aleatoria uniforme entera, $numb \sim U(1, L)$
  Agregar el individuo sel[$Numb$] a la lista hijos
  Generar variable aleatoria uniforme, $prob \sim U(0, 1)$
  if $prob <= P_c$ **then**
    Generar una variable aleatoria uniforme entera, $numb2 \sim U(1, L)$
    Generar una variable aleatoria uniforme entera, $punto \sim U(1, M)$
    for $j = punto, ..., M$ **do**
      hijos[i][j] = sel[numb2][j]
    end for
  end if
end for
Regresar hijos

Operadores de mutación

El primer algoritmo de mutación (M1) sigue el siguiente pseudocódigo:

Entrada: Lista de individuos cruzados (hijos), número de individuos (L), número de pedidos (N), número de máquinas (M), Indices de factibilidad ($IndicesFactibles$), Matriz de factibilidad (Matrizindividuo)

Salida: Lista con los individuos Mutados (mutados)

for $i = 1, ..., L$ **do**
  Generar variable aleatoria uniforme, $prob \sim U(0, 1)$
  if $prob <= P_m$ **then**

```
        Generar una variable aleatoria uniforme entera, numb ~ U(1, L)
        Generar una variable aleatoria uniforme entera, IndicePedidoMutado ∈
IndicesFactibles[numb]
        for j = 1, ..., M do
            if j = IndicePedidoMutado then
                mutados[i][numb][j] = MatrizIndividuo[numb][IndicePedidoMutado]
            else
                mutados[i][numb][j] = 0
            end if
        end for
    end if
end for
Regresar mutados
```

El segundo algoritmo de mutación (M2) sigue el siguiente pseudocódigo:

Entrada: Lista de individuos cruzados (hijos), número de individuos (L), número de pedidos (N), número de máquinas (M), Indices de factibilidad ($IndicesFac$), Matriz de factibilida (Matrizindividuo)

Salida: Lista con los individuos Mutados (mutados)

```
for i = 1, ..., L do
    Generar variable aleatoria uniforme, prob ~ U(0, 1)
    if prob <= P_m then
        for j = 1, ..., N do
            Generar variable aleatoria uniforme, prob2 ~ U(0, 1)
            if prob2 <= 0,2 then
                Seleccionar aleatoriamente Indice ∈ IndicesFac[j]
                for k = 1, ..., M do
                    if k = Indice then
                        mutados[i][j][k] = MatrizIndividuo[j][Indice]
                    else
                        mutados[i][j][k] = 0
                    end if
                end for
            end if
```

end for
end if
end for
Regresar mutados

Operadores de reemplazo

El primer algoritmo de reemplazo es total (RT), éste es el algoritmo de reemplazo más sencillo, consiste en que la generación actual pasa a reemplazar a toda la población anterior.

El segundo algoritmo de reemplazo es por competencia (RC), este algoritmo sigue el siguiente pseudocódigo:

Entrada: Población de padres (padres), población de hijos mutados (mutados), lista con la aptitud de los hijos mutados (SumaPonderadaHijos, aptitudCmaxHijos, TiempoTotalUtilizacionHijos), lista con la aptitud de los padres (SumaPonderadaPadres, aptitudCmaxPadres, TiempoTotalUtilizacionPadres), número de individuos (L)

Salida: Población de padres reemplazados y aptitud (PadresReemplazados, SumaPonderadaReemplazado, FitnessUtilizacionTotalReemplazado ,FitnessCmaxReemplazado)

Crear lista vacía, PadresReemplazados
Crear lista vacía, SumaPonderadaReemplazado
Crear lista vacía, FitnessCmaxReemplazado
Crear lista vacía, FitnessUtilizacionTotalReemplazado
for $i = 1, ..., L$ **do**
Generar variable aleatoria uniforme entera, $Numb1 \sim U(1, L)$
if $SumaPonderadaPadres[Numb1] > SumaPonderadaHijos[Numb1]$ **then**
Agregar los valores de mutados[Numb1] en la lista PadresReemplazados
Agregar los valores de SumaPonderadaHijos[Numb1] en la listaSumaPonderadaReemplazado
Agregar los valores de aptitudCmaxHijos[Numb1] en la listaFitnessC-

maxReemplazado

Agregar los valores de TiempoTotalUtilizacionHijos en la lista FitnessUtilizacionTotalReemplazado

else

Agregar los valores de padres[Numb1] en la lista PadresReemplazados

Agregar los valores de SumaPonderadaPadres[Numb1] a la lista SumaPonderadaReemplazado

Agregar los valores de aptitudCmaxPadres[Numb1] a la lista FitnessCmaxReemplazado

Agregar los valores de TiempoTotalUtilizacionPadres[Numb1] a la lista FitnessUtilizacionTotalReemplazado

end if

end for

Regresar PadresReemplazados, SumaPonderadaReemplazado, FitnessUtilizacionTotalReemplazado ,FitnessCmaxReemplazado

3.2.5. Selección de operadores genéticos

Para hacer la selección de los mejores operadores genéticos para este problema se programan 16 algoritmos genéticos, las combinaciones programadas son las siguientes:

- SB_CU_M1_RT
- SB_CU_M1_RC
- SB_CU_M2_RT
- SB_CU_M2_RC
- SB_CP_M1_RT
- SB_CP_M1_RC
- SB_CP_M2_RT
- SB_CP_M2_RT
- SR_CU_M1_RT

- SR_CU_M1_RC
- SR_CU_M2_RT
- SR_CU_M2_RC
- SR_CP_M1_RT
- SR_CP_M1_RC
- SR_CP_M2_RT
- SR_CP_M2_RT

Dónde SB es selección binaria, SR es selección por ruleta, CU es cruza uniforme, CP es cruza en un punto, M1 es la mutación tipo 1, M2 es la mutación tipo 2, RC es reemplazo por competencia y RT es reemplazo total.

Se resuelve una instancia de prueba la cual está conformada por 620 operaciónes diferentes y 15 máquinas disponibles (Ver Figuras A.1 y A.2). Para ésta comparación se hace uso del parámetro a con un valor de 0,5. Los resultados obtenidos se muestran en la Figura 3.1

	Elite	Prom. F.O.	Prom. Makespan	Prom. Cmax	Error relativo porcentual
SR_CP_M2_RC	29919	30202.3885	10605.36	49799.417	0.947
SR_CP_M2_RT	39681.74	57086.295	39816.11	74356.48	43.86
SR_CP_M1_RC	43564.965	56364.725	39620.15	73109.3	29.38
SR_CP_M1_RT	43524.36	58762.81	41490.4	76035.22	35.0
SR_CU_M2_RC	27987	29581.57	10470.723	48692.417	5.697
SR_CU_M2_RT	39762.99	61391.455	41885.78	80897.13	54.39
SR_CU_M1_RC	43609.745	56250.97	39494.32	73007.62	28.98
SR_CU_M1_RT	43609.745	56250.97	39494.32	73007.62	28.98
SB_CP_M2_RC	29229	32610.1035	15070.79	50149.417	11.56
SB_CP_M2_RT	40310.24	61440.855	41779.67	81102.04	52.41
SB_CP_M1_RC	43637.685	56339.455	39843.17	72835.74	29.10
SB_CP_M1_RT	43373.24	58469.375	42892.62	74046.13	34.80
SB_CU_M2_RC	28922	33635.07	14470.723	52799.417	16.29
SB_CU_M2_RT	40473.985	57584.905	42133.73	73036.08	42.27
SB_CU_M1_RC	43366.495	57603.895	41428.67	73779.12	32.83
SB_CU_M1_RT	43500.365	57064.39	40477.8	73650.98	31.18

Figura 3.1: Resultados de algoritmos

Para tener una mejor visión sobre los resultados mostrados en la Figura 3.1 se procede a normalizar los datos y se realiza una suma ponderada uniforme (ver Figura 3.2).

	Min	Min	Min	Min	Min	
	0.2	0.2	0.2	0.2	0.2	1
	Elite	Prom. F.O.	Prom.Makespan	Prom.Cmax	Error relativo porcentual	Suma Ponderada
SR_CP_M2_RC	0.0479	0.0368	0.0196	0.0452	0.0019	0.03031
SR_CP_M2_RT	0.063	0.0695	0.0736	0.0675	0.0918	0.07321
SR_CP_M1_RC	0.0697	0.0686	0.073	0.0664	0.0614	0.06792
SR_CP_M1_RT	0.0696	0.0716	0.0766	0.0691	0.073	0.07207
SR_CU_M2_RC	0.0448	0.036	0.019	0.0442	0.0119	0.03127
SR_CU_M2_RT	0.063	0.0748	0.0774	0.0735	0.1138	0.08065
SR_CU_M1_RC	0.0698	0.0685	0.073	0.0663	0.0606	0.06768
SR_CU_M1_RT	0.0698	0.0685	0.073	0.0663	0.0606	0.06768
SB_CP_M2_RC	0.04	0.0397	0.027	0.0455	0.0242	0.036
SB_CP_M2_RT	0.0645	0.0748	0.0772	0.0737	0.1097	0.08001
SB_CP_M1_RC	0.0698	0.0686	0.0736	0.0661	0.0609	0.06786
SB_CP_M1_RT	0.0694	0.0712	0.0792	0.0672	0.0728	0.07202
SB_CU_M2_RC	0.0463	0.0409	0.0267	0.0479	0.0341	0.03922
SB_CU_M2_RT	0.0648	0.0701	0.077	0.0663	0.0884	0.07354
SB_CU_M1_RC	0.0694	0.0701	0.0765	0.0670	0.0687	0.07039
SB_CU_M1_RT	0.069	0.0695	0.0748	0.066	0.0652	0.06924

Figura 3.2: Resultados normalizados de algoritmos

Como se puede ver en los resultados presentados en la Figura 3.2 los algoritmos con mejores desempeños son el algoritmo $SR_CP_M2_RC$ y el algoritmo $SR_CU_M2_RC$.

Con base en los resultados obtenidos se decide usar el algoritmo $SR_CU_M2_RC$, ya que fue el que mejor desempeño, esto considerando que tiene una mejor convergencia. Otra opción, puede ser el algoritmo $SB_CU_M2_RC$. ya que en su desempeño general es mejor, aunque al revisar el error relativo porcentual de su poblacion final tiene mas variación. El desempeño y convergencia del algoritmo elegido se puede ver en la Figura 3.3.

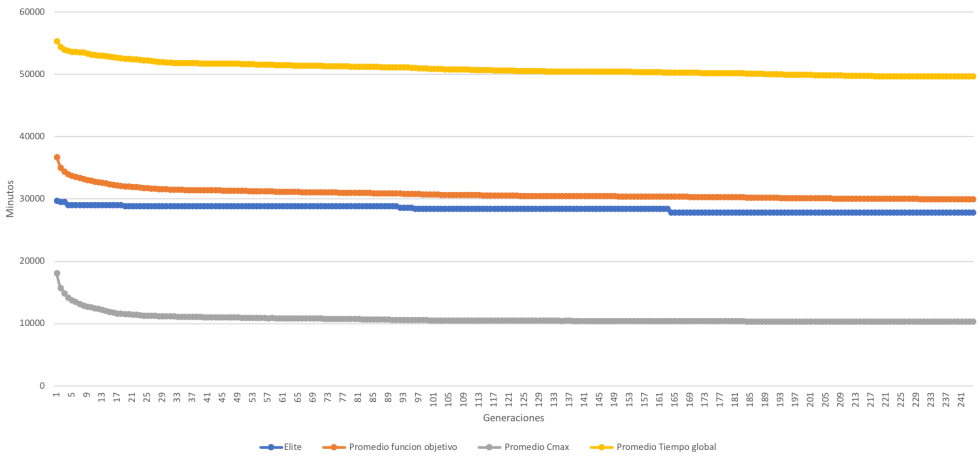

Figura 3.3: Desempeño del algoritmo $SR_CU_M2_RC$

Si se decidiera usar un algoritmo que no tuvo un buen desempeño podemos ver que realmente no llegariamos a ninguna solución, ya que dichos algoritmos tienen problemas con su convergencia (ver Figura 3.4).

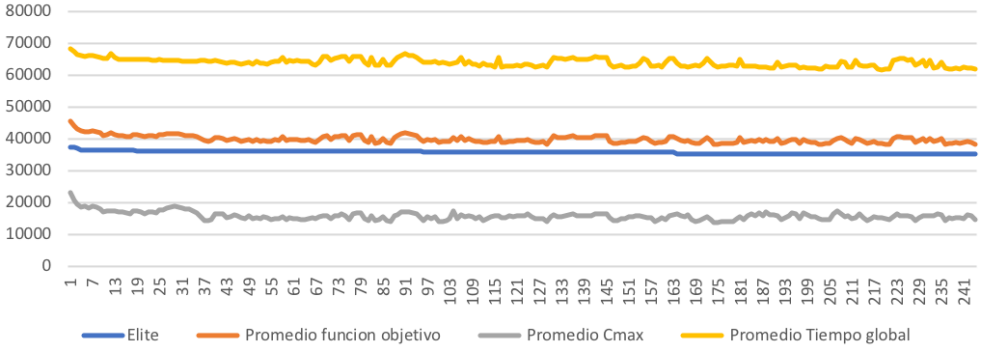

Figura 3.4: Desempeño del algoritmo $SB_CP_M2_RT$

3.3. Estabilidad del algoritmo

Para demostrar la estabilidad del algoritmo se resuelven las 10 instancias reales y la instancias de prueba 10 y 50 veces, después se hace un prueba de hipótesis sobre las medias de los resultados obtenidos para cada instancias con 10 y 50 corridas, estas prueba se hipótesis se hacen con un $\alpha = 0{,}05$.

Instancia	p-value	Significancia
Instancia de prueba	0.125	0.05
Inst1	0.449	0.05
Inst2	≈ 1	0.05
Inst3	0.893	0.05
Inst4	0.603	0.05
Inst5	0.19	0.05
Inst6	0.956	0.05
Inst7	0.26	0.05
Inst8	0.71	0.05
Inst9	≈ 1	0.05
Inst10	≈ 1	0.05

Cuadro 3.3: Resultados de pruebas de hipótesis

En base a los resultados en el $p-value$ para las pruebas, podemos concluir que los resultados obtenidos no tiene un diferencia significativa, por lo cual se puede decir que el algoritmo es estable ya que se garantiza una solución de calidad sin importar el número de veces que se realice una corrida.

Las pruebas de hipótesis son realizadas con el lenguaje de programación R, para realizar estas pruebas se hace uso del comando $t.test()$, para saber mas sobre este comando del lenguaje de programación R visite este url: `http://stat.ethz.ch/R-manual/R-devel/library/stats/html/t.test.html`

3.4. Prueba del parámetro α

Para saber que valor de α se toma, se realizan pruebas con el algoritmo seleccionado tomando $a \in \{0, 0{,}01, 0{,}02, ..., 0{,}99, 1\}$ y se realiza un frente de Pareto entre C_{max} y $Makespan$. (Ver Figura 3.5).

En base al frente de Pareto se decide tomar $a = 0{,}6$, esto debido a que, tomando en consideración las necesidades de las empresa de terminar mas rapido sus procesos y considerando que al rededor de este valor se encuentra el mejor equilibrio entre C_{max} y $Makespan$, así, dando más preferencia al $Makespan$ sin dejar de lado el C_{max}.

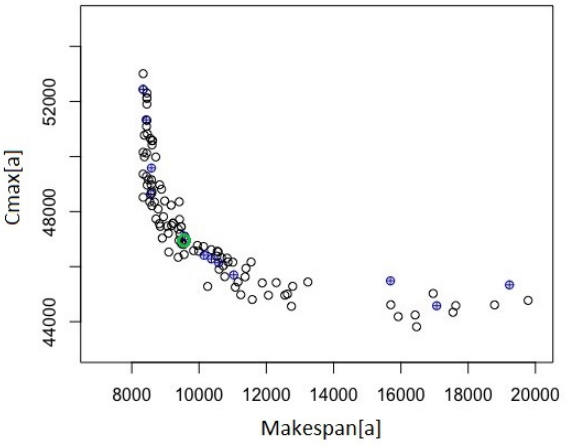

Figura 3.5: Frente de Pareto entre C_{max} y $Makespan$

Capítulo 4

Análisis de resultados y conclusiones

Se utilizó un equipo MacBook Air 11", con procesador Intel® Core i5, 1.7 GHz, 4 GB de RAM, MacOs High Sierra de 64 bits para ejecutar cada algoritmo. Se programó el algoritmo en el lenguaje de programación Python 2.7, se realizarón las pruebas estadisticas con el lenguaje de programación R.

En el Apéndice A se muestra la información contenida en cada instancia, las cuales estan compuestas como se explica en la sección 3.1. en el Apéndice B se muestran los diagramas de Gantt de la soluciones obtenidas en cada instancia.

A continuación se muestran los resultados obtenidos en las intancias reales e hipotéticas, éstos resultados son comparados en el objetivo de *Makespan* con los resultados obtenidos con el modelo de programación entrera mixta propuesto por Diana Enríquez et al. [8], el modelo esta programado en el lenguaje algebraico de GAMS haciendo uso de librerías de CPLEX, se definió como criterio de paro del solucionador el tiempo límite de 30 minutos o que el espacio de búsqueda que falta por explorar llegue a un 0 %, este porcentaje es conocido como GAP.

El algoritmo genético tardó 2 minutos con 13 segundos en solucionar la instancia de prueba, como se mencionó anteriormente esta instancia cuenta con 15 máquinas y 60 pedidos. Por consecuente el tiempo para solucionar las

instancias reales e hipotéticas es menor al mencionado.

4.1. Resultados de instancias reales

Los resultados obtenidos para las 10 instancias reales proporcionadas por la empresa pueden visualizarse en el Cuadro 4.1.

Instancias	C_{max} AG	*Makespan* AG	*Makespan* GAMS	GAP
inst1	631	213	213	0 %
inst2	300	100	100	0 %
inst3	79	58	58	0 %
inst4	795	308	308	0 %
inst5	1376	399	399	0 %
inst6	731	497	497	0 %
inst7	628	249	249	0 %
inst8	320	62	62	0 %
inst9	162	48	48	0 %
inst10	227	96	96	0 %

Cuadro 4.1: Resultados de instancias reales

Como se ve en el cuadro anterior, en la comparación de *Makespan* entre el resultado obtenido por el algoritmo genético y la solución exacta obtenida con el modelo de programación entera mixta a través de GAMS podemos notar que el algoritmo siempre alcanza el óptimo en esa función, por otra parte, la función C_{max} no es posible compararla contra una solución exacta debido a que el modelo de programación entera mixta no tiene ese alcance. Sin embargo como se mostró en el apartado 3.4 donde se realiza un frente de Pareto para ver la sensibilidad de las funciones objetivos, cuando se alcanza el *Makespan* puede darse el caso de tener varios óptimos cuando se trata de un solo objetivo, donde lo que podría definir una mejor solución es el C_{max}.

4.2. Resultados de instancias hipotéticas

Los resultados obtenidos para las 10 instancias hipotéticas que se proponen pueden visualizarse en el Cuadro 4.2.

Instancias	C_{max} AG	$Makespan$ AG	Makespan$GAMS$	GAP
inst_M5_J5	96	41	41	0 %
inst_M5_J10	271	151	151	0 %
inst_M5_J15	591	125	125	0 %
inst_M10_J5	160	48	48	0 %
inst_M10_J10	389	68	68	0 %
inst_M10_J15	799	142	164	26 %
inst_M15_J5	88	35	35	0 %
inst_M15_J10	341	64	68	22 %
inst_M15_J15	489	68	72	48 %
inst_M15_J20	733	109	115	63 %

Cuadro 4.2: Resultados de instancias hipotéticas

Como se ve en el cuadro anterior, en algunas de las instancias es dificil llegal al óptimo dentro del tiempo límite haciendo uso de GAMS y de la programación entera mixta, sin embargo como se mencionó anteriormente el tiempo de cómputo usado por el algoritmo genético no supera los 2 minutos. Otro punto destacable es que en las intancias donde no se encontró el óptimo se puede ver que la mejor solución encontrada por el algoritmo genético es mejor que la encontrada por GAMS.

4.3. Discusión de resultados y conclusiones

Con base en los resultados obtenidos podemos decir que el algoritmo genético puede brindarnos de forma confiable soluciones de calidad para este problema, esto debido a que realizando las pruebas estadísticas, se comprobó que el algoritmo es estable, además, en comparación al método de programación entera mixta es considerablemente más rapido, además, el algoritmo genético cumple su función de optimizar 2 objetivos a la vez. También es destacable que en las instancias donde se conoce el óptimo para el objetivo de $Makespan$ se puede ver que el algoritmo genético lo alcanza en un tiempo de

cómputo razonable.

Los diagramas de Gantt de cada una de las soluciones obtenidas por el algoritmo genético pueden verse en el Apéndice B.

Se puede concluir que hacer uso técnicas y metodologías meta-heurísticas como lo es programación genética, pueden tener buen desempeño en problemas complejos, este buen desempeño dependerá del tipo de metodología que se siga, la complejidad del problema y de los operadores genéticos que se usen, ya que, todos los problemas tienen diferentes caracteristicas y deben ser solucionados con diferentes procedimientos ya que no todos los operadores genéticos funcionan igual con todos los problemas.

Muchas veces suele ser conveniente hacer uso de técnicas heurísticas o meta-heuríticas debido a que, hacer uso de solucionadores como GAMS, CPLEX, GUROBI, entre otros, puede ser muy costoso computacionalmente. Como se demostró aquí, en algunas de las instancias tardaba hasta 30 minutos buscando soluciones y no se llegaba al óptimo. Entonces al momento de realizar un modelo donde se optimizan 2 objetivos, la complejidad de problema aumenta más, así, requiriendo mayor esfuerzo computacional.

Si se hace el uso correcto de técnicas heurísticas y meta-heurísticas podemos asegurar soluciones de calidad, esto dependera del tipo de problema, de que tipo de metodología sea utilizada y sus debidos análisis y pruebas pertinentes realizadas.

Para efectos de esta investigación se ve que el algoritmo genético se adaptó correctamente al problema en configuración Flexible Job Scheduling Problem, esto, haciendo uso de los operadores genéticos y parámetros adecuados, así asegurando una estabilidad en el algoritmo y logrando incrementar la eficiencia de productividad a través de la optimización del secuenciamiento de operaciones, también se ve que realizar la optimización en 2 objetivos ayuda a mejorar los resultados. Esto debido porque al reducir el tiempo de utilización de máquinas, se reducen costos de consumo y al reducir el $Makespan$ se reducen los tiempos de finalización de tareas. Con base en todos los resultados obtenidos se concluye que la hipótesis planteada al comienzo de la investigación se cumple.

Apéndice A

Instancias

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	15	17																
2										Calibre								
3	MÁQUIN	0.01	0.188	0.313	0.375	2.000	3.000	3.250	4.000	6.000	7.000	8.000	9.000	9.500	11.000	12.000	12.250	17.000
4	1	23.19			33.61	20.73												
5	2					31.35	27.53	26.00	26.76									
6	3			25.30	8.06			23.28										
7	4		19.72							26.16			26.06	20.45		26.35	24.04	
8	5						17.42	24.29	28.34							23.71		20.81
9	6		23.22				15.38			28.31						27.85	25.45	
10	7		25.18										28.11		45.30	34.60	28.89	
11	8									29.86								
12	9		27.61							25.14			28.71	29.52				
13	10		23.81				20.74	24.42		23.56	29.83							
14	11		25.86							24.68	24.50							
15	12											38.98	31.35	23.01	31.37			
16	13		27.05							21.32	20.87	27.85			33.14	14.05		
17	14												54.68	41.80				
18	15												43.58	47.47				

Factibilidad | Pedidos | +

Figura A.1: Factibilidad de operación para las 10 instancias reales, la instancia de prueba y las instancias hipotéticas Inst_M15_J5, Inst_M15_J10, Inst_M15_J15 y Inst_M15_J20

	A	B	C	D	E
1	62	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	P150T	9.00	0.25	25000.00
4	2	P150S	6.00	0.64	4000.00
5	3	P150L	6.00	0.24	2000.00
6	4	P200T	9.00	0.92	41665.00
7	5	P200S	6.00	0.64	20000.00
8	6	P200L	6.00	0.29	10000.00
9	7	PZ250T	9.00	0.81	75000.00
10	8	PZ250S	6.00	0.64	27000.00
11	9	PZ250L	6.00	0.34	6000.00
12	10	PZL	6.00	0.37	6000.00
13	11	P300350T	9.00	0.86	125000.00
14	12	P300350S	6.00	0.64	32500.00
15	13	P300350L	6.00	0.39	5000.00
16	14	P350L	6.00	0.44	5000.00
17	15	FS	9.00	0.61	14000.00
18	16	RS	3.00	0.61	2500.00
19	17	P176T	12.00	0.73	35658.00
20	18	P176S	9.00	0.51	11886.00
21	19	P176L	9.00	0.22	7924.00
22	20	P225T	12.00	0.73	35658.00
23	21	P225S	9.00	0.51	11886.00
24	22	P225L	9.00	0.27	7924.00
25	23	PTM	9.00	1.39	11886.00
26	24	PTS	9.00	0.50	11886.00
27	25	PTU	12.00	0.55	59430.00
28	26	EG	9.00	0.12	7924.00
29	27	EC	6.00	0.86	7924.00
30	28	ET	6.00	0.52	11886.00
31	29	PB60T	12.25	0.83	256000.00
32	30	PB60S	9.00	0.61	240000.00
33	31	CS	6.00	0.55	6000.00
34	32	P150T	9.00	0.25	25000.00
35	33	P150S	6.00	0.64	4000.00
36	34	P150L	6.00	0.24	2000.00
37	35	P200T	9.00	0.92	41665.00
38	36	P200S	6.00	0.64	20000.00
39	37	P200L	6.00	0.29	10000.00
40	38	PZ250T	9.00	0.81	75000.00
41	39	PZ250S	6.00	0.64	27000.00
42	40	PZ250L	6.00	0.34	6000.00
43	41	PZL	6.00	0.37	6000.00
44	42	P300350T	9.00	0.86	125000.00
45	43	P300350S	6.00	0.64	32500.00
46	44	P300350L	6.00	0.39	5000.00
47	45	P350L	6.00	0.44	5000.00
48	46	FS	9.00	0.61	14000.00
49	47	RS	3.00	0.61	2500.00
50	48	P176T	12.00	0.73	35658.00
51	49	P176S	9.00	0.51	11886.00
52	50	P176L	9.00	0.22	7924.00
53	51	P225T	12.00	0.73	35658.00
54	52	P225S	9.00	0.51	11886.00
55	53	P225L	9.00	0.27	7924.00
56	54	PTM	9.00	1.39	11886.00
57	55	PTS	9.00	0.50	11886.00
58	56	PTU	12.00	0.55	59430.00
59	57	EG	9.00	0.12	7924.00
60	58	EC	6.00	0.86	7924.00
61	59	ET	6.00	0.52	11886.00
62	60	PB60T	12.25	0.83	256000.00

Factibilidad Pedidos +

Figura A.2: Pedidos de instancia de prueba

	A	B	C	D	E
1	8	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	0.1875	0.377	2534.00
4	2	O2	11.0000	0.268	1950.00
5	3	O3	0.1875	1.541	56.00
6	4	O4	4.0000	1.402	4000.00
7	5	O5	7.0000	0.960	660.00
8	6	O6	0.1875	1.431	1139.00
9	7	O7	0.1875	1.025	5724.00
10	8	O8	7.0000	1.817	220.00

Factibilidad | Pedidos

Figura A.3: Pedidos de instancia real 1

	A	B	C	D	E
1	5	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	9.5000	0.967	4908.00
4	2	O2	3.0000	0.414	2000.00
5	3	O3	3.0000	0.953	1512.00
6	4	O4	7.0000	0.982	440.00
7	5	O5	8.0000	1.071	885.00

Factibilidad | Pedidos

Figura A.4: Pedidos de instancia real 2

	A	B	C	D	E
1	4	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	9.500	0.873	133.00
4	2	O2	9.500	0.715	70.00
5	3	O3	9.500	0.763	750.00
6	4	O4	9.500	0.642	4225.00

Factibilidad | Pedidos

Figura A.5: Pedidos de instancia real 3

	A	B	C	D	E
1	6	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	12.2500	0.660	3640.00
4	2	O2	9.5000	0.642	4211.00
5	3	O3	12.2500	0.660	2140.00
6	4	O4	12.2500	1.110	6000.00
7	5	O5	12.2500	1.110	8000.00
8	6	O6	0.1875	1.875	52.00

Factibilidad | Pedidos

Figura A.6: Pedidos de instancia real 4

	A	B	C	D	E
1	5	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	12.2500	0.922	10920.00
4	2	O2	12.2500	0.922	7800.00
5	3	O3	9.5000	0.642	29480.00
6	4	O4	9.5000	1.009	10440.00
7	5	O5	12.0000	0.433	5550.00

Factibilidad Pedidos +

Figura A.7: Pedidos de instancia real 5

	A	B	C	D	E
1	4	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O61	12.2500	0.660	21740.00
4	2	O62	9.5000	0.642	8100.00
5	3	O63	3.0000	2.245	1350.00
6	4	O64	0.3125	0.560	560.00

Factibilidad Pedidos +

Figura A.8: Pedidos de instancia real 6

	A	B	C	D	E
1	8	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	12.2500	0.258	10920.00
4	2	O2	12.2500	0.922	7800.00
5	3	O3	9.5000	0.642	2940.00
6	4	O4	9.5000	1.009	1040.00
7	5	O5	12.0000	0.433	5550.00
8	6	O6	3.0000	0.375	2940.00
9	7	O7	9.5000	0.730	2170.00
10	8	O8	0.3125	0.433	1720.00

Factibilidad Pedidos +

Figura A.9: Pedidos de instancia real 7

	A	B	C	D	E
1	7	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	12.0000	0.642	1060.00
4	2	O2	12.2500	0.922	1350.00
5	3	O3	9.5000	0.642	2980.00
6	4	O4	12.2500	0.922	1250.00
7	5	O5	3.0000	0.223	3650.00
8	6	O6	9.5000	0.642	4100.00
9	7	O7	12.2500	0.922	1920.00

Factibilidad Pedidos +

Figura A.10: Pedidos de instancia real 8

	A	B	C	D	E
1	5	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	12.2500	0.660	1840.00
4	2	O1	12.2500	0.260	5100.00
5	3	O2	9.5000	0.873	195.00
6	4	O4	9.5000	0.763	1250.00
7	5	O5	9.5000	0.642	1225.00

Factibilidad | Pedidos | +

Figura A.11: Pedidos de instancia real 9

	A	B	C	D	E
1	6	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	7.0000	0.568	2270.00
4	2	O2	12.2500	0.660	4200.00
5	3	O3	3.0000	0.873	133.00
6	4	O4	9.5000	0.362	70.00
7	5	O5	12.2500	0.660	750.00
8	6	O6	6.0000	0.763	750.00

Factibilidad | Pedidos | +

Figura A.12: Pedidos de instancia real 10

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	5	13												
2								Calibre						
3	MÁQUIN	0.1875	3.000	3.250	4.000	6.000	7.000	8.000	9.000	9.500	11.000	12.000	12.250	17.000
4	4	19.72				26.16			26.06	20.45		26.35	24.04	
5	5		17.42	24.29	28.34							23.71		20.81
6	9	27.61				25.14			28.71	29.52				
7	11	25.86				24.68	24.50							
8	12							38.98	31.35	23.01	31.37			

Factibilidad | Pedidos | +

Figura A.13: Factibilidad de operación para instancia hipotética Inst_M5_J5

	A	B	C	D	E
1	5	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	0.1875	0.682	970.00
4	2	O2	12.2500	0.820	1200.00
5	3	O3	3.0000	0.475	133.00
6	4	O4	9.5000	0.362	720.00
7	5	O5	9.0000	0.665	750.00

Factibilidad | Pedidos | +

Figura A.14: Pedidos de instancia hipotética Inst_M5_J5

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	5	14													
2		Calibre													
3	MÁQUIN	0.01	0.1875	0.3125	0.375	2.000	3.000	3.250	4.000	6.000	9.000	9.500	12.000	12.250	17.000
4	1	23.19			33.61	20.73									
5	2					31.35	27.53	26.00	26.76						
6	3			25.30	8.06			23.28							
7	4		19.72							26.16	26.06	20.45	26.35	24.04	
8	5						17.42	24.29	28.34				23.71		20.81

Factibilidad | Pedidos | +

Figura A.15: Factibilidad de operación para instancia hipotética Inst_M5_J10

	A	B	C	D	E
1	10	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	6.0000	0.568	2270.00
4	2	O2	12.2500	0.660	585.00
5	3	O3	3.0000	0.873	140.00
6	4	O4	9.5000	0.362	70.00
7	5	O5	3.2500	0.660	750.00
8	6	O6	6.0000	0.763	750.00
9	7	O7	2.0000	0.284	1300.00
10	8	O8	4.0000	0.923	250.00
11	9	O9	3.2500	0.185	1750.00
12	10	O10	17.0000	1.010	250.00

Factibilidad | Pedidos | +

Figura A.16: Pedidos de instancia hipotética Inst_M5_J10

	A	B	C	D	E	F	G	H	I	J	K	L
1	5	11										
2		Calibre										
3	MÁQUIN	0.1875	3.000	3.250	4.000	6.000	9.000	9.500	11.000	12.000	12.250	17.000
4	4	19.72				26.16	26.06	20.45		26.35	24.04	
5	5		17.42	24.29	28.34					23.71		20.81
6	6	23.22	15.38			28.31				27.85	25.45	
7	7	25.18					28.11		45.30	34.60	28.89	
8	9	27.61				25.14	28.71	29.52				

Factibilidad | Pedidos | +

Figura A.17: Factibilidad de operación para instancia hipotética Inst_M5_J15

	A	B	C	D	E
1	15	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	0.1875	0.568	2270.00
4	2	O2	12.2500	0.660	4200.00
5	3	O3	3.0000	0.873	133.00
6	4	O4	9.5000	0.362	70.00
7	5	O5	6.0000	0.660	750.00
8	6	O6	9.0000	0.763	750.00
9	7	O7	0.1875	0.364	1585.00
10	8	O8	4.0000	0.633	670.00
11	9	O9	0.1875	0.509	1810.00
12	10	O10	17.0000	0.374	620.00
13	11	O11	11.0000	0.919	1710.00
14	12	O12	6.0000	0.418	1090.00
15	13	O13	3.2500	0.352	860.00
16	14	O14	12.0000	0.500	2030.00
17	15	O15	17.0000	0.189	970.00

Factibilidad | Pedidos | +

Figura A.18: Pedidos de instancia hipotética Inst_M5_J15

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	10	16															
2		Calibre															
3	MÁQUIN	0.01	0.1875	0.3125	0.375	2.000	3.000	3.250	4.000	6.000	7.000	9.000	9.500	11.000	12.000	12.250	17.000
4	1	23.19			33.61	20.73											
5	2					31.35	27.53	26.00	26.76								
6	3			25.30	8.06			23.28									
7	4		19.72							26.16		26.06	20.45		26.35	24.04	
8	5						17.42	24.29	28.34						23.71		20.81
9	6		23.22				15.38			28.31					27.85	25.45	
10	7		25.18									28.11		45.30	34.60	28.89	
11	8									29.86							
12	9		27.61							25.14		28.71	29.52				
13	10		23.81				20.74	24.42		23.56	29.83						

Factibilidad | Pedidos | +

Figura A.19: Factibilidad de operación para instancia hipotética Inst_M10_J5

	A	B	C	D	E
1	5	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	0.1875	0.568	1270.00
4	2	O2	12.2500	0.660	2100.00
5	3	O3	3.0000	0.873	1350.00
6	4	O4	9.5000	0.362	1080.00
7	5	O5	9.0000	0.660	750.00

Factibilidad | Pedidos | +

Figura A.20: Pedidos de instancia hipotética Inst_M10_J5

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	10	15														
2									Calibre							
3	MÁQUIN	0.1875	0.3125	0.375	3.000	3.250	4.000	6.000	7.000	8.000	9.000	9.500	11.000	12.000	12.250	17.000
4	3		25.30	8.06		23.28										
5	4	19.72						26.16			26.06	20.45		26.35	24.04	
6	5				17.42	24.29	28.34							23.71		20.81
7	6	23.22			15.38			28.31						27.85	25.45	
8	7	25.18									28.11		45.30	34.60	28.89	
9	8							29.86								
10	9	27.61						25.14			28.71	29.52				
11	10	23.81			20.74	24.42		23.56	29.83							
12	11	25.86						24.68	24.50							
13	12									38.98	31.35	23.01	31.37			

Factibilidad | Pedidos | +

Figura A.21: Factibilidad de operación para instancia hipotética Inst_M10_J10

	A	B	C	D	E
1	10	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	7.0000	0.932	980.00
4	2	O2	12.2500	0.203	1310.00
5	3	O3	11.0000	0.395	2320.00
6	4	O4	9.0000	0.700	1210.00
7	5	O5	12.0000	0.884	1670.00
8	6	O6	6.0000	0.196	2530.00
9	7	O7	3.0000	0.982	1170.00
10	8	O8	0.1875	0.209	1930.00
11	9	O9	8.0000	0.569	1100.00
12	10	O10	3.2500	0.622	2640.00

Factibilidad | Pedidos | +

Figura A.22: Pedidos de instancia hipotética Inst_M10_J10

	A	B	C	D	E	F	G	H	I	J	K	L
1	10	11										
2							Calibre					
3	MÁQUIN	0.1875	3.000	3.250	6.000	7.000	8.000	9.000	9.500	11.000	12.000	12.250
4	6	23.22	15.38		28.31						27.85	25.45
5	7	25.18						28.11		45.30	34.60	28.89
6	8				29.86							
7	9	27.61			25.14			28.71	29.52			
8	10	23.81	20.74	24.42	23.56	29.83						
9	11	25.86			24.68	24.50						
10	12						38.98	31.35	23.01	31.37		
11	13	27.05			21.32	20.87	27.85			33.14	14.05	
12	14							54.68	41.80			
13	15							43.58	47.47			

Factibilidad | Pedidos | +

Figura A.23: Factibilidad de operación para instancia hipotética Inst_M10_J15

	A	B	C	D	E
1	15	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	7.0000	0.568	2080.00
4	2	O2	12.2500	0.660	2110.00
5	3	O3	3.0000	0.873	2640.00
6	4	O4	9.5000	0.362	1880.00
7	5	O5	12.2500	0.660	2330.00
8	6	O6	6.0000	0.763	1180.00
9	7	O7	0.1875	0.173	1000.00
10	8	O8	12.0000	0.903	1270.00
11	9	O9	0.1875	0.851	1770.00
12	10	O10	7.0000	0.440	650.00
13	11	O11	12.0000	0.623	2050.00
14	12	O12	9.5000	0.724	2100.00
15	13	O13	3.0000	0.982	1250.00
16	14	O14	8.0000	0.242	1670.00
17	15	O15	6.0000	0.543	2280.00

Factibilidad | Pedidos

Figura A.24: Pedidos de instancia hipotética Inst_M10_J15

	A	B	C	D	E
1	5	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	0.1875	0.682	970
4	2	O2	12.25	0.82	1200
5	3	O3	3	0.475	133
6	4	O4	9.5	0.362	720
7	5	O5	9	0.665	750

Factibilidad | Pedidos

Figura A.25: Pedidos de instancia hipotética Inst_M15_J5

	A	B	C	D	E
1	10	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	7	0.932	980
4	2	O2	12.25	0.203	1310
5	3	O3	11	0.395	2320
6	4	O4	9	0.7	1210
7	5	O5	12	0.884	1670
8	6	O6	6	0.196	2530
9	7	O7	3	0.982	1170
10	8	O8	0.1875	0.209	1930
11	9	O9	8	0.569	1100
12	10	O10	3.25	0.622	2640

Factibilidad | Pedidos

Figura A.26: Pedidos de instancia hipotética Inst_M15_J10

	A	B	C	D	E
1	15	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	7	0.25	3270
4	2	O2	12.25	0.02	3125
5	3	O3	3	0.58	2535
6	4	O4	9.5	0.01	2930
7	5	O5	12.25	0.83	2340
8	6	O6	6	0.26	2200
9	7	O7	0.1875	0.48	1330
10	8	O8	12	0.95	1300
11	9	O9	0.1875	0.24	1430
12	10	O10	7	0.01	2330
13	11	O11	12	0.6	2160
14	12	O12	9.5	0.35	2030
15	13	O13	3	0.55	730
16	14	O14	8	0.23	1550
17	15	O15	6	0.13	1390

Factibilidad Pedidos +

Figura A.27: Pedidos de instancia hipotética Inst_M15_J15

	A	B	C	D	E
1	20	5			
2	No. PEDIDO	CÓDIGO	CALIBRE	MEDIDA EN METROS	CANTIDAD DE PIEZAS NECESARIAS
3	1	O1	7	0.17	370
4	2	O2	12.25	0.76	1720
5	3	O3	3	0.75	395
6	4	O4	9.5	0.58	1450
7	5	O5	12.25	0.9	1980
8	6	O6	6	0.29	1360
9	7	O7	0.1875	0.25	1660
10	8	O8	12	0.69	2140
11	9	O9	0.1875	0.18	430
12	10	O10	7	0.83	1070
13	11	O11	12	0.3	720
14	12	O12	9.5	0.83	2150
15	13	O13	3	0.66	780
16	14	O14	8	0.39	1170
17	15	O15	6	0.6	1230
18	16	O16	6	0.6	680
19	17	O17	0.1875	0.94	1840
20	18	O18	12	0.25	1960
21	19	O19	0.1875	0.96	1410
22	20	O20	7	0.45	2320

Factibilidad Pedidos +

Figura A.28: Pedidos de instancia hipotética Inst_M15_J20

Apéndice B

Diagramas de Gantt

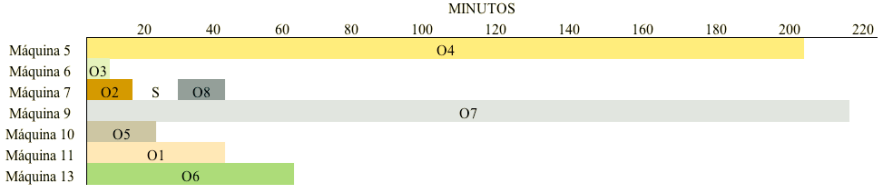

Figura B.1: Solución instancia real 1

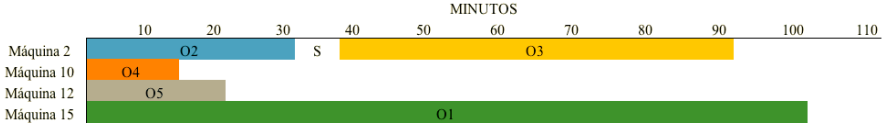

Figura B.2: Solución instancia real 2

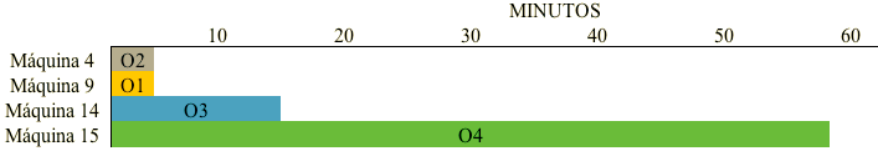

Figura B.3: Solución instancia real 3

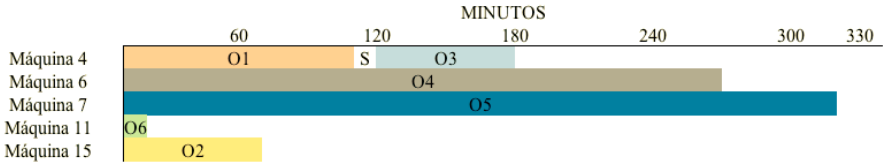

Figura B.4: Solución instancia real 4

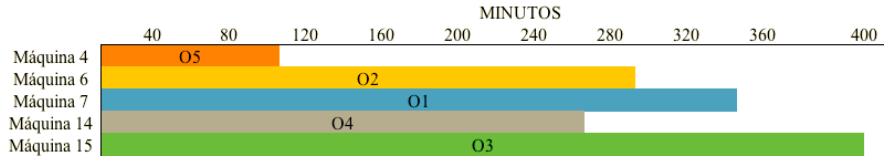

Figura B.5: Solución instancia real 5

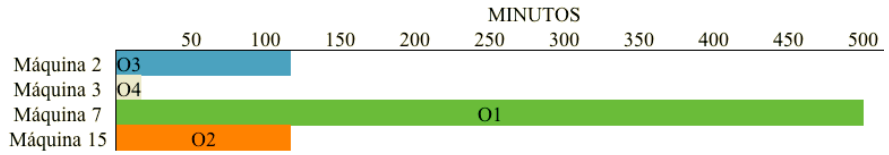

Figura B.6: Solución instancia real 6

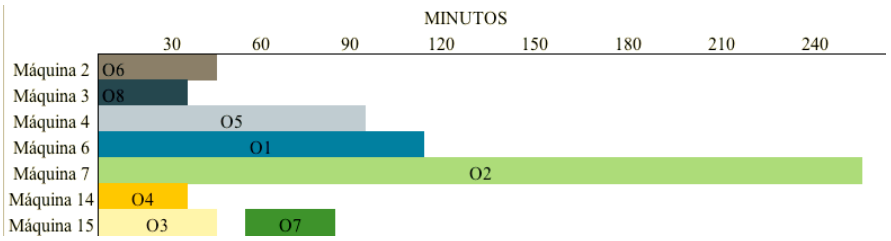

Figura B.7: Solución instancia real 7

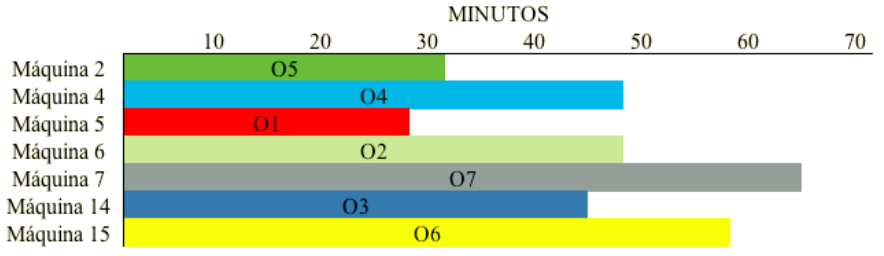

Figura B.8: Solución instancia real 8

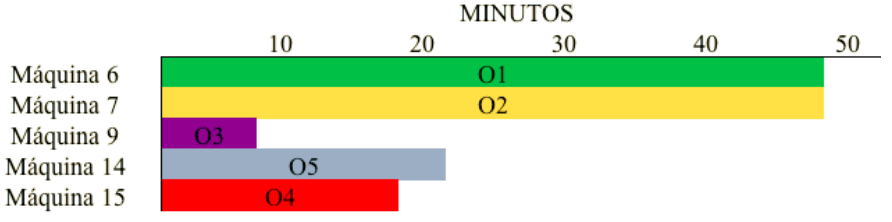

Figura B.9: Solución instancia real 9

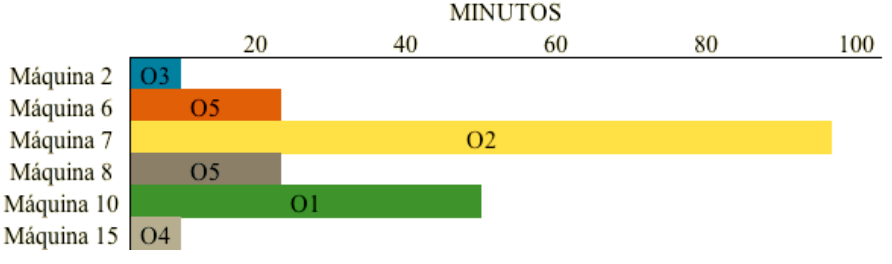

Figura B.10: Solución instancia real 10

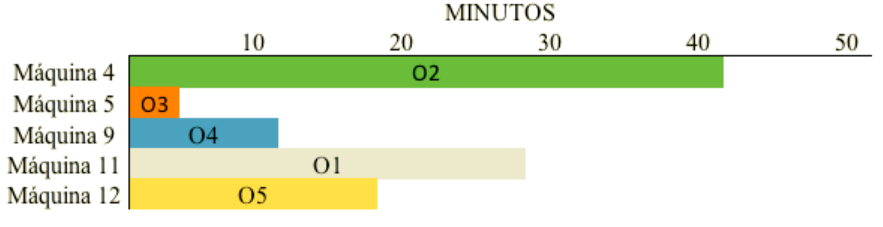

Figura B.11: Solución instancia hipotética Inst_M5_J5

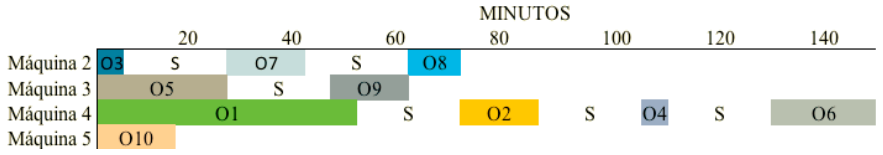

Figura B.12: Solución instancia hipotética Inst_M5_J10

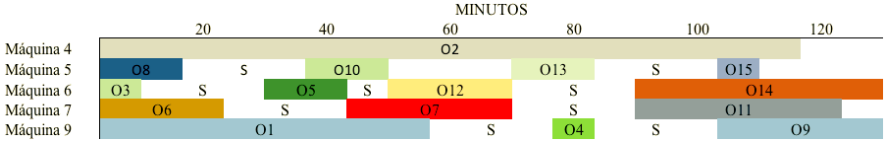

Figura B.13: Solución instancia hipotética Inst_M5_J15

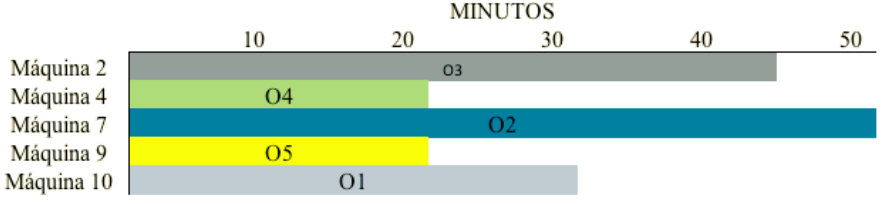

Figura B.14: Solución instancia hipotética Inst_M10_J5

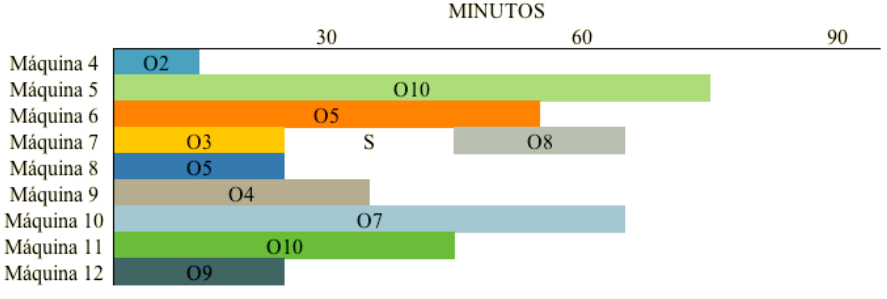

Figura B.15: Solución instancia hipotética Inst_M10_J10

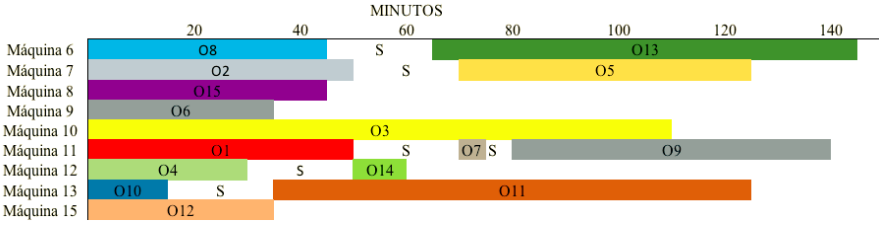

Figura B.16: Solución instancia hipotética Inst_M10_J15

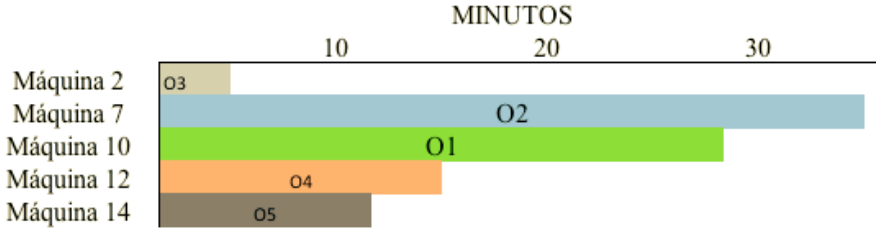

Figura B.17: Solución instancia hipotética Inst_M15_J5

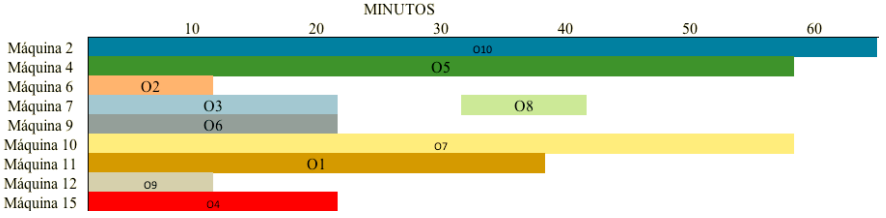

Figura B.18: Solución instancia hipotética Inst_M15_J10

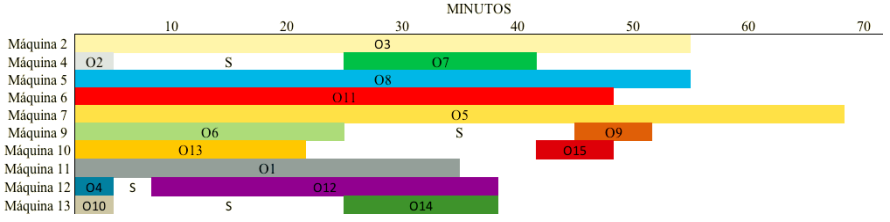

Figura B.19: Solución instancia hipotética Inst_M15_J15

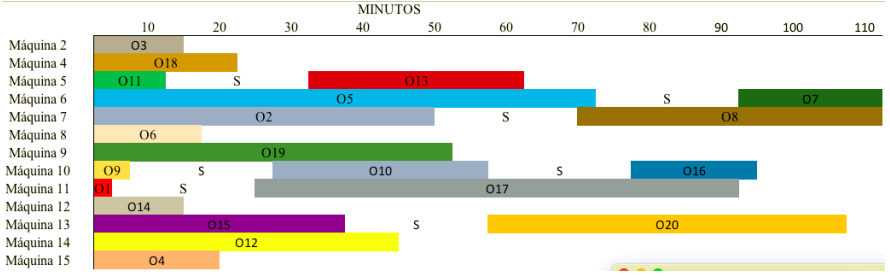

Figura B.20: Solución instancia hipotética Inst_M15_J20

Bibliografía

[1] Np-hardness of shop-scheduling problems with three jobs*. *Journal of Operational Research*, 50(3):237 – 266, 1995.

[2] Jain A.S. and Meeran S. Deterministic job shop scheduling: past, present and future. *Journal of Operational Research*, 113(2):390 – 434, 1999.

[3] P. Brucker, B. Jurisch, and B. Sievers. A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics*, 16(3):107–127, 1994.

[4] Ozguven C, Yavuz Y, and Ozbakr L. Mixed integer goal programming models for the flexible job-shop scheduling problems with separable and non-separable sequence dependent setup times. *European Journal of Operational Research*, 36(2):846—858, 2012.

[5] Jacques Carlier and E Pinson. An algorithm for solving job shop problem. 35:164–176, 02 1989.

[6] Peter Dalgaard. *Introductory statistics with R.* Springer, 2002.

[7] K. DeJong and W. Spears. An analysis of the interacting roles of population size and crossover in genetic algorithms; proceedings first workshop parallel problem solving from nature. *Springer-Verlag*, 16:38–47, 1990.

[8] Diana Enríquez. *Optimizacion de un problema de programación de la producción aplicado a un proceso de manufactura mediante técnicas heurísticas.* Instituto Tecnológico Superior de Arandas, 2017.

[9] Jose Fernando Gonçalves, Jorge Jose de Magalhaes Mendes, and Mauricio G. C. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, 167(1):77 – 95, 2005.

[10] F. Hillier and G. Lieberman. *Introducción a la investigación de operaciones.* Mc Graw Hill, 2002.

[11] John H. Holland. *Adaptation in Natural and Artificial Systems, An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* MIT, 1975.

[12] Carlier J. and Pinson E. An algorithm for solving the job shop problem. *Management science*, 35(2):164 – 176, 1989.

[13] Grefestette J. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, & Cybernetics*, 16(3):122–128, 1986.

[14] Baker Kenneth. R. *introduction to secuencing and scheduling.* New York, Wiley, 1974.

[15] Dave Kuhlman. *A Python Book: Beginning Python, Advanced Python, and Python Exercises.* https://web.archive.org, 2012.

[16] Eugene L. Lawler. *Combinatorial Optimization: networks and matroids.* Courier Corporation, 1976.

[17] Juan Camilo . L. V and Jaime Antero A. M. Algoritmo genético para reducir el makespan en un flow shop híbrido fexible con máquinas paralelas no relacionadas y tiempos de alistamiento dependientes de la secuencia. *Unilibre Cal, ingeniería y tecnología*, 11(1):250 – 262, 2015.

[18] Ye Li and Yan Chen. A genetic algorithm for job-shop scheduling. *Journal of Software*, 5(3):269–274, 2010.

[19] Pinedo M. L. *Scheduling: Theory, Algorithms and Systems.* New York, Springer, 2012.

[20] Khaled Mesghouni, Slim Hammadi, and Pierre Borne. Evolutionary algorithms for job shop scheduling. *Appl. Math. Comput. Sci.*, 14:91 – 103, 2004.

[21] John F. Muth and Gerald L. Thompson. *Industrial Scheduling.* Prentice-Hall, 1963.

[22] J. Márquez Delgado. *Optimización de la programación (scheduling) en talleres de mecanizado.* PhD thesis, UPM, 2012.

[23] Ferdinando Pezzella and Emanuela Merelli. A tabu search method guided by shifting bottleneck for the job shop scheduling problem. *European Journal of Operational Research*, 120:297 – 310, 2000.

[24] Ricardo Pérez Rodriguez. *Programación de operaciones en configuraciones job shop flexibles ultilizando un algoritmo de estimaciones de distribuciones.* PhD thesis, CIATEC, 2014.

[25] J. Russell and R. Cohn. *Pareto Efficiency.* Book on Demand, 2012.

[26] Pedro Pablo B. S., Diana Paola B. R., and Juan Eduardo B. B. Aplicación de una heurística constructiva en programación secuencial para asignación de varios trabajos a varias máquinas en paralelo. *Scientia et Technica Año XVIII*, 18(1):120–125, 2013.

[27] Choi S. H. and Wang K. Flexible flow shop scheduling with stochastic processing times: A decomposition based approach. *Computers & Industrial Engineering*, 63(2):362 – 373, 2012.

[28] Dirk Taeger and Sonja Kuhnt. *Statistical hipotesis testing with SAS and R.* WILEY, 2014.

[29] Wannaporn Teekeng and Arit Thammano. Modified genetic algorithm for flexible job-shop scheduling problems. *Procedia Computer Science*, 12:122 – 128, 2012.

SMIO
SOCIEDAD MEXICANA DE
INVESTIGACIÓN DE OPERACIONES

UNIVERSIDAD
PANAMERICANA®

Se otorga el presente diploma a:

Demetrio Rafael Briones Briones

Por su participación en el VI Congreso de la Sociedad Mexicana de Investigación de Operaciones, llevado a cabo en Zapopan, Jalisco del 4 al 6 de Octubre de 2017.

Dr. Elías Olivares Benítez
Presidente del Comité Organizador del Congreso

Dr. Miguel Ángel Moreno Míguez
Presidente del Comité Organizador del Congreso

SMIO
SOCIEDAD MEXICANA DE INVESTIGACIÓN DE OPERACIONES

UNIVERSIDAD PANAMERICANA

Se otorga la presente

CONSTANCIA A:

Saúl Domínguez Casasola

Jonás Velasco

Demetrio Briones

Con motivo de que el trabajo registrado con el No. 68, titulado "Una metaheurística para el diseño de plantas industriales con aplicación al sector lechero", bajo la autoría de los mismos, fue presentado en el VI congreso de la Sociedad Mexicana de Investigación de Operaciones, en Zapopan, Jalisco, del 4 al 6 de octubre de 2017.

Dr. Elias Olivares Benítez
Presidente del Comité Organizador del Congreso

Dr. Miguel Ángel Moreno Míguez
Presidente del Comité Organizador del Congreso

SMIO
SOCIEDAD MEXICANA DE INVESTIGACIÓN DE OPERACIONES

UNIVERSIDAD PANAMERICANA

Se otorga la presente

CONSTANCIA A:

Demetrio Briones

Saúl Domínguez Casasola

Jonás Velasco

Con motivo de que el trabajo registrado con el No. 65, titulado "Un modelo de programación binivel para la selección de paradas de autobuses y el ruteo de transporte escolar", bajo la autoría de los mismos, fue presentado en el VI congreso de la Sociedad Mexicana de Investigación de Operaciones, en Zapopan, Jalisco, del 4 al 6 de octubre de 2017.

Dr. Elias Olivares Benítez
Presidente del Comité Organizador del Congreso

Dr. Miguel Ángel Moreno Míguez
Presidente del Comité Organizador del Congreso

CIMAT

ABRIL 11-13, 2018

XVI ESCUELA DE PROBABILIDAD Y ESTADÍSTICA CIMAT, Guanajuato

El **Centro de Investigación en Matemáticas, A.C.**

otorga

la presente CONSTANCIA a

DEMETRIO RAFAEL BRIONES BRIONES

Por su asistencia a esta Escuela.

13 de abril de 2018

Guanajuato, Gto., México

Comité Organizador

Octavio Arizmendi Echegaray
CIMAT

Joaquín Ortega Sanchez
CIMAT

Emilio Joly
CIMAT

CIMAT

10° TALLER DE Solución DE Problemas Industriales
Guanajuato·Mérida·Monterrey

SPI

El Centro de Investigación en Matemáticas, A.C.

a través de la Coordinación de Servicios Tecnológicos

Otorga la presente CONSTANCIA a

Demetrio Rafael Briones Briones

por su participación en el
10° Taller de Solución de Problemas Industriales,
realizado en las instalaciones del Centro del 16 al 20 de enero de 2017.

Guanajuato, México. 20 de enero de 2017

M. en C. Ivete Sánchez Bravo
Comité Organizador

Dra. Graciela González Farías
Coordinadora de Servicios Tecnológicos