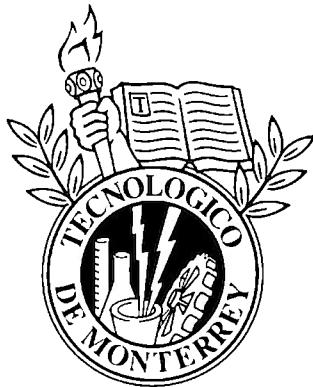


INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS ESTADO DE MÉXICO



TECNOLÓGICO DE MONTERREY®

ALGORITMO GENÉTICO MULTIOBJETIVO AUTOEVOLUTIVO PARA RESOLVER EL PROBLEMA DE SECUENCIACIÓN DE TRABAJOS EN UNA LÍNEA DE ENSAMBLE CON SISTEMA DE PRODUCCIÓN JUSTO A TIEMPO.

TESIS PARA OPTAR EL GRADO DE
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN
PRESENTA

NESTOR VELASCO BERMEO

Asesor: Dr. JAIME MORA VARGAS

Comité de tesis:

Jurado:	Dr. RODOLFO ESTEBAN IBARRA O. MSc. FERNANDO GUDIÑO PEÑALOZA Dr. MIGUEL GONZALEZ MENDOZA Dr. JAIME MORA VARGAS	Presidente Secretario Vocal Vocal
---------	--	--

Atizapán de Zaragoza, Edo. Méx., Abril de 2009

RESUMEN

Para una empresa que su principal actividad sea el producir o ensamblar la complejidad de sus procesos se torna más complejo al incrementarse la variabilidad de sus productos. Dado que se incluyen mayores variantes y características particulares en los productos el tiempo de proceso deja de ser finito y se torna en variable. Si una empresa implementa la filosofía Lean Six Sigma para satisfacer las necesidades de sus clientes debe ser capaz de ser flexible en términos de sus estrategias de manufactura y políticas de distribución según George [1]. De esa forma la compañía podrá adoptar una estrategia de producción de Fabricar a la Orden y dependerá menos de los pronósticos de la demanda. Es importante tener en mente que la personalización de un producto juega un papel muy importante en su ciclo de vida ya que le permite permanecer en el mercado por un tiempo más largo.

La necesidad de una solución computacional que considera todas las variables que la personalización de un producto representa un reto para los paquetes comerciales. La inclusión de variables y objetivos mutuamente excluyentes compitiendo por un mismo recurso hacen que el modelo matemático planteado sea computacionalmente complejo para resolver. Los problemas a resolver partiendo de lo antes descrito resultan ser multiobjetivo y de varias dimensiones lo que hace que sean definidos como NP-duros. La eficiencia computacional se vuelve un aspecto importante a considerar al resolver problemas con tantas soluciones posibles, el departamento de producción debe por tanto contar con una herramienta que les permita manejar tantas variables y objetivos conflictivos para construir secuencias factibles. Es de igual importancia el considerar nuevas órdenes de clientes que arriban una vez se definieron los planes de producción y que deben ser cumplidas en un tiempo y fecha establecidos.

Con base en lo antes expuesto se desarrolló un Algoritmo Genético Multiobjetivo para resolver el problema de secuenciación de trabajos en una línea de ensamble que opera bajo el sistema de producción Justo a tiempo (*Just in Time*). Dicho sistema de producción exige que una secuencia de producción no tenga interrupciones en ningún momento y requiere mantener al mínimo los tiempos muertos. Siguiendo dicha política se busca el reducir la cantidad de operaciones de ajuste y preparación de las maquinas que intervienen en el proceso de ensamblaje cada vez que se presenta un producto distinto. La relevancia en el método propuesto reside en la generación de un módulo autoadaptable para definir el valor de la razón de mutación en cada generación de acuerdo a la aptitud que presenta una población y de ésta manera acelerar el proceso evolutivo y llegar a una solución óptima en un menor número de generaciones.

CONTENIDO

RESUMEN	3
ÍNDICE DE FIGURAS	6
ÍNDICE DE TABLAS	7
CAPÍTULO I	8
1.1 INTRODUCCIÓN	8
1.3 JUSTIFICACIÓN	10
1.4 OBJETIVO GENERAL	14
1.4.1 OBJETIVOS ESPECÍFICOS	14
1.5 HIPÓTESIS	15
1.6 ALCANCES Y LIMITACIONES	15
1.7 ORGANIZACIÓN DE LA TESIS	15
1.8 APORTE	16
CAPÍTULO II	17
2.1 OPTIMIZACIÓN MULTIOBJETIVO	17
2.2 ELEMENTOS QUE CONFORMAN UN AG	22
2.3 AG MULTIOBJETIVO REPRESENTATIVOS	25
CAPÍTULO III	39
3.1 DESCRIPCIÓN DEL PROBLEMA	39
3.2 MODELACIÓN MATEMÁTICA DEL PROBLEMA	40
3.3 VARIABLES Y FUNCIONES OBJETIVO	43

CAPÍTULO IV	45
4.1 DESCRIPCIÓN DEL MODELO	45
4.2 PARTICULARIDADES DEL MODELO	46
4.3 OPERADORES GENÉTICOS	47
4.3.1 OPERADOR AUTOADAPTABLE DE MUTACIÓN	48
4.4 ESQUEMA GENERAL DE OPERACIÓN DEL MODELO	49
4.5 ALGORITMOS GENÉTICOS MULTIOBJETIVO ADICIONALES	51
4.6 INSTANCIAS DE PROBLEMAS A RESOLVER.....	52
 CAPÍTULO V	 53
5.1 PRUEBAS Y RESULTADOS.....	53
 CAPÍTULO VI	 59
6.1 CONCLUSIONES	59
6.2 TRABAJO FUTURO	60
 BIBLIOGRAFÍA Y REFERENCIAS	 61
 ANEXOS	 67

ÍNDICE DE FIGURAS

Figura 1 Cinco estrategias genéricas de Manufactura	13
Figura 2 Modelo esquemático de operación para un Algoritmo Genético tradicional.....	24
Figura 3 Ejemplo de secuencia de trabajos para un problema de secuenciación 3 trabajos en 3 máquinas.....	29
Figura 4 Cromosoma que utiliza una representación entera.....	45
Figura 5 Ejemplo de un cromosoma que contiene 5 trabajos para una demanda total de 15 productos	46
Figura 6 Representación de una secuencia de producción en números enteros	46
Figura 7 Estructura del archivo de configuración para el AG.....	50
Figura 8 Codificación de las funciones objetivo en Matlab	53
Figura 9 Razón de uso a lo largo de 100 generaciones	55
Figura 10 Razón de uso a lo largo de 500 generaciones	55
Figura 11 Cantidad de operaciones de ajuste después de 100 generaciones	57
Figura 12 Cantidad de operaciones de ajuste después de 500 generaciones	57
Figura 13 Razón de uso después de 100 generaciones.....	58
Figura 14 Razón de uso después de 500 generaciones.....	58

ÍNDICE DE TABLAS

Tabla 1 Secuencia de Máquinas y Trabajos	Error! Bookmark not defined.
Tabla 2 Conjunto de Problemas 1	52
Tabla 3 Conjunto de Problemas 2	52
Tabla 4 Resultados obtenidos con 100 generaciones	54
Tabla 5 Resultados obtenidos con 500 generaciones	54
Tabla 6 Cantidad de Ajustes obtenidos después de 100 generaciones	56
Tabla 7 Cantidad de Ajustes obtenidos después de 500 generaciones	56

CAPÍTULO I

1.1 INTRODUCCIÓN

En la actualidad la mayoría de los problemas que enfrenta una empresa de cualquier ramo tiene que ver directamente con la asignación y uso eficiente de sus recursos, el aprovechamiento al máximo de sus máquinas, materia prima, espacios, personal, etc. de manera dinámica y eficiente de forma tal que no se vea afectada la producción o el servicio que presta. Al intentar alcanzar dicho objetivo se intenta tomar en consideración la mayor cantidad de variables que se ven directa o indirectamente relacionados en el proceso con el fin de satisfacer las necesidades del cliente, sin embargo en ocasiones se hacen omisiones ya sea por desconocimiento u conveniencia práctica por lo que el resultado final no es favorable para la empresa al hacer uso de las herramientas computacionales que tiene a su alcance dado que las mismas no proveen de un apoyo con relación al problema que está atacando.

En el caso de las empresas manufactureras es crítico el uso eficiente de sus equipos y recursos de forma tal que puedan cumplir con la producción que se tiene planeada. Debido a la diversidad de productos a producir y al hecho de que en ocasiones se tiene que ajustar dicha planeación al integrar peticiones o pedidos extraordinarios a lo que se tenía contemplado previamente se debe ajustar dicha planeación para cumplir con los nuevos requerimientos y el plan de producción o la asignación de trabajo a cada máquina disponible se ve modificada sobre la marcha o con base en la experiencia del jefe de producción dada la premura, rapidez del ajuste y el poco tiempo disponible. Por tal motivo en la mayoría de las veces y con base en el hecho de que la toma de decisiones se lleva a cabo basada en la experiencia o consideración de una persona los resultados no siempre son los más favorables para la empresa.

Cuando se intenta atacar un problema de decisión que involucra más de una solución óptima se describe un problema multiobjetivo que por su naturaleza contempla distintas soluciones óptimas. Al resolver los tales no se obtiene una solución única que satisface todas las condiciones o variables en él, ya que las mismas pueden llegar a ser conflictivas entre sí (mutuamente excluyentes o compiten por el mismo recurso). Dado lo anterior se obtiene un conjunto de escenarios dentro de los cuales se debe considerar un cierto grado de ganancia o pérdida entre las soluciones. Esto llevará al tomador de decisiones a elegir entre las distintas soluciones óptimas de acuerdo a su preferencia, información adicional no técnica o cualitativa.

Dentro de algunos de los primeros acercamientos que se han manejado para atacar el problema antes mencionado podemos encontrar a los métodos exactos como la investigación de Operaciones [1] o Métodos Matemáticos Discretos [2], de igual forma podemos mencionar a los métodos meta heurísticos provenientes de la Inteligencia Artificial como la búsqueda tabú [3], Recocido Simulado[4] y del Cómputo Evolutivo [5] los Algoritmos Genéticos(AG) [6][7] la Programación Evolutiva [8], la Lógica Difusa [9] , las Redes Neuronales [10] y las Estrategias Evolutivas [11].

Al modelar una situación problemática real en un problema multiobjetivo nos enfrentamos al hecho de tratar simultáneamente varios objetivos bajo ciertas restricciones para los cuales obtendremos una serie de soluciones óptimas que nos presentan distintos escenarios en los cuales un objetivo tiene un mejor desempeño que otro pero sin sacrificar el resultado óptimo general del problema. Dicho conjunto de soluciones óptimas se conoce como frontera óptima de Pareto, y se definen a tales soluciones como aquellas que dan el mejor resultado en términos de una variable sin afectar considerablemente a las otras, se dice que son una serie de compensaciones entre cada una de las variables.

Durante los años 80 las técnicas de optimización relacionadas a la secuenciación de trabajos estaba enfocada en optimizar solo un objetivo a la vez (tiempo; máximo para terminar todas las tareas, de flujo de las tareas, de demora, el número de trabajos demorados, etc.). La concepción que se tenía sobre optimización de problemas multiobjetivos estaba directamente vinculada con hallar las soluciones óptimas de Pareto, tales soluciones tenían la característica de ser no dominadas (no existe otra solución mejor cuando se toman en cuenta todos los objetivos) y la elección de una solución de todas las presentadas recaía en el tomador de decisiones [12] (a esto se le conoce como un acercamiento *a posteriori*) [13]. En el caso del enfoque *a priori*, se tienen dos casos, en el primero el tomador de decisiones define a través de una serie de pesos (que son asignadas a cada una de las funciones objetivo) y con base en dicha asignación se define la preferencia sobre cada una de las funciones uniendo todos los objetivos en una función lineal, el segundo caso parte de una jerarquización de los objetivos y la estrategia de solución se enfoca en obtener una solución para el primer objetivo, una vez cumplida dicha condición se prosigue a obtener el resultado del segundo objetivo sin transgredir o afectar a la primera solución y esto continúa hasta haber resuelto el último objetivo

De los distintos métodos de solución relacionados a los problema multiobjetivo con los que cuenta la Inteligencia Artificial los algoritmos genéticos presentan una considerable ventaja por sobre los demás estrategias de solución pues por su naturaleza trabajan con un conjunto de soluciones (población) por lo que le permite probar con varias soluciones a la vez en lugar de intentar trabajar con una a la vez [14], además de que no necesitan hacer adaptaciones o ajustes a las funciones objetivo (al utilizar por ejemplo el operador ϵ como restricción) o también el tratar a las distintas funciones objetivo en una suma ponderada con pesos asignados.[15]. Un AG al trabajar con varias soluciones a la vez tiene la capacidad de encontrar el conjunto de soluciones que satisfagan las funciones objetivo propuestas y que a su vez sean soluciones no dominadas, esto es que no se pueden mejorar más una función objetivo sin sacrificar el resultado de otra función.

1.3 JUSTIFICACIÓN

El problema de secuenciación tiene una presencia considerable en muchos problemas y situaciones de la vida real [16] ahora bien si nos enfocamos en una industria podemos encontrar una instancia de dicho problema, la secuenciación de trabajos con ruta variable (*Job Shop Scheduling Problem* por su término en inglés) que se define como una situación ante la cual se desea obtener la secuencia óptima para una serie de trabajos j en una serie de máquinas m disponibles, cada uno de dichos trabajos requiere un tiempo distinto y se debe tomar en cuenta la precedencia entre los mismos así como la distinta duración que cada uno posee, por lo anterior el obtener una secuencia debe de tomar al menos los tres factores antes mencionados, sin embargo ésta es una idealización del problema pues en condiciones reales se deben tomar en cuenta otros factores, luego entonces, no solo se busca tener una secuencia que minimice el tiempo total de ciclo (*makespan* por su término en inglés) sino que se deben de considerar aspectos como el tiempo de preparación de cada máquina, el costo en el que se incurre en las demoras, el ajuste de los tiempos en cada trabajo si es que la demanda aumenta y los ajustes que se necesiten hacer sobre la marcha para poder cumplir con la demandas y si además tales situaciones se desean modelar matemáticamente la complejidad para el adecuado planteamiento del problema se eleva considerablemente al tomar en cuenta las variables antes mencionadas y expresar tales en funciones matemáticas.

Por lo anterior el problema original en el cual solamente se buscaba obtener una secuencia óptima que resultará en un mínimo de tiempo muerto total ahora se torna en un problema multiobjetivo al considerar todas las posibles variables que intervienen en la solución del problema y que en muchas ocasiones son conflictivas entre sí o compiten por los mismos recursos. Al hablar de un problema multiobjetivo la definición de “óptimo” se vuelve un tanto vaga y no es fácil de definir pues no se puede obtener una solución que cumpla con todas las restricciones además de estar basado en las preferencias y consideraciones de un tomador de decisiones, es por lo anterior que se parte de una definición de óptimo propuesta por Edgeworth al plantear que se generan distintos “escenarios” dentro de los cuales se tienen una serie de combinaciones de valores para satisfacer las variables modeladas. Debido a que no se puede obtener una solución que satisfaga todas las restricciones se maneja una serie de posibles respuestas dentro del espacio de solución factible, a ésta serie de respuestas óptimas se les denomina “*Set óptimo de Pareto*” gracias a Vilfredo Pareto al establecer que un vector \vec{x}^* es Pareto Óptimo si no existe otro vector dentro del espacio de soluciones disponible que ofrezca mejores resultados cuando se toman en cuenta todos los objetivos y a tales soluciones se les denomina “*no dominadas*”.

Haciendo referencia a la definición del problema que se presentó al inicio del capítulo se puede notar que se está describiendo una instancia o un caso en particular del problema de secuenciación con multiprocesadores el cual se define como un problema NP¹ Completo [17], de igual forma se puede identificar la característica de que dicho problema se puede plantear como un problema de decisión² (los cuales por naturaleza se consideran de tipo NP). Es importante resaltar que un problema con tal complejidad requiere de una mayor cantidad de recursos computacionales como el uso del procesador, memoria y tiempo de procesamiento puesto que para conocer una respuesta correcta para los tales no existe un algoritmo que pueda resolverlos en un tiempo polinómico pues se debe hacer una búsqueda en todo el espacio de solución aunque la respuesta pueda parecer obvia para algunos problemas (como por ejemplo el problema de subconjuntos³) el tiempo que se necesitará para resolverle será no menos que exponencial. Cabe resaltar el caso que Jackson [18] plantea de un problema de secuenciación con un procesador al contemplar 2 máquinas y al menos 2 trabajos, de lo que podemos concluir que para un problema que se defina con un número mayor de máquinas y trabajos se tratará de un problema NP Completo.

Existen otros autores que han comentado que el problema de secuenciación es de tipo NP Duro [19][20] al ser una subclase de los problemas combinatorios y por lo tanto solo existen aproximaciones heurísticas⁴, de igual forma [8] hace la misma afirmación basado en el trabajo de Lawler, et al. [21] que trata sobre un caso específico de secuenciación donde contemplaban un total de 3 trabajos y 3 máquinas y un tiempo máximo de procesamiento para cada operación igual para todos los trabajos tomando en consideración de igual forma el hecho de que las máquinas pueden detener el trabajo que están llevando a cabo en operación y cambiar a otras mientras que más tarde pueden reanudar las operaciones que suspendieron.

Ya sea que clasifiquemos al problema de secuenciación como NP Completo o NP Duro por su complejidad computacional no existen hasta la fecha métodos exactos eficientes para dar una solución en tiempo polinómico y las propuestas de solución presentadas hasta el momento toman en cuenta algoritmos que proveen de la mejor aproximación para resolver dichos casos. Es importante resaltar el hecho de que la mayoría de los problemas encontrados en la industria, y que van desde producción hasta secuenciación de trabajos, se pueden clasificar de tipo NP[22], por lo que el poder encontrar una estrategia de solución a dichos problemas no solo ofrecerá una mejor y más eficiente manera para las empresas de atacar estos problemas sino que permitirá que puedan modificar sus esquemas de producción y manejo de inventarios para ampliar su gama de productos o reducir sus tiempos de entrega.

¹ Por sus siglas en inglés *Nondeterministic Polynomial* (No determinista Polinómico)

² Basado en el hecho de hacer la pregunta sobre si existe una secuencia que sea menor que un valor B (óptima o de valor mínimo)

³ Que se enuncia como sigue: “dado un conjunto S de enteros, ¿Existe un conjunto no vacío de S cuyos elementos sumen cero?” y que tiene un número de $2^n - 1$ subconjuntos posibles.

⁴ Coello define a una Heurística como una técnica de solución de problemas en la cual la solución o su aproximación más apropiada es elegida utilizando reglas de comparación.

La relevancia de solucionar el problema de secuenciación de trabajos de una manera eficiente y rápida no solo es de gran importancia en términos de investigación sino que representa una gran herramienta para las empresas manufactureras que puede cambiar la estrategia de negocio de la misma. Debido a que permite adaptar su estrategia de manufactura y administración de inventarios dentro de su sistema de cadena de suministro para cumplir con la demanda que se tiene además de pedidos extraordinarios no contemplados dentro de su planeación.

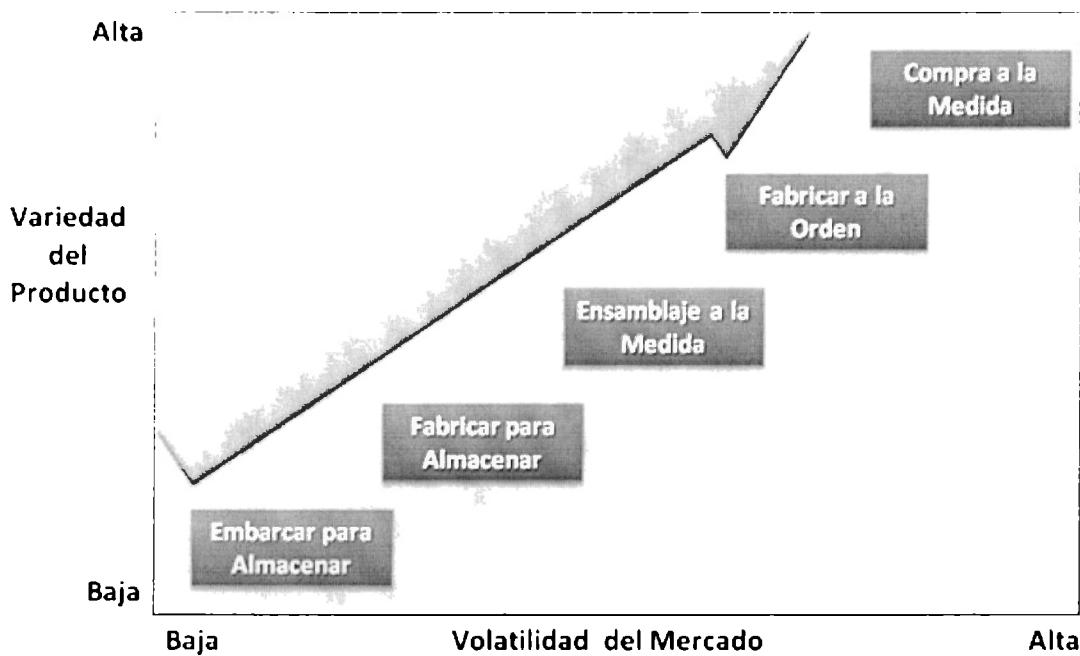
Lambert [23] define al respecto cinco tipos de cadenas de suministro que varían de la más flexible a la más rígida en términos de estrategias de manufactura:

- I) **Compra a la Medida** (*Buy to Order*) se basa en el hecho de que todos los productos son únicos y el cliente está consciente de tener que esperar un largo periodo de entrega además de que no se maneja un inventario pues se correría el riesgo de que tal se vuelva obsoleto.
- II) **Fabricar a la Orden** (*Make to Order*) tiene como principal ventaja la capacidad de fabricar nuevos productos siempre y cuando se basen en la misma materia prima, de igual forma permite una variabilidad en los volúmenes y las mezcla de productos,
- III) **Ensamblaje a la Medida** (*Assemble to Order*) tiene como ventaja principal el hecho de ofrecer al cliente un elevado grado de personalización a los productos si y solo si éstos parten de una plataforma estándar, el tiempo de ciclo se reduce de manera considerable y dependerá directamente de el momento en el que el ensamblaje final se lleve a cabo considerando la cadena de suministro.
- IV) **Fabricar para almacenar** (*Make to Stock*) parte de la fabricación de un producto estándar, de una demanda bien predicha y estable sin estar sujeta a una región específica.
- V) **Embarcar para almacenar** (*Ship to Stock*) provee de un producto estándar a puntos de venta específicos y se basa en una demanda permanente de parte del cliente hacia un producto pre-posicionado y estable en el mercado.

Lo anterior cobra aún más importancia al considerar el hecho de que si una empresa desea contar con una estrategia de manufactura flexible y así pueda cumplir con las demandas de los clientes o pueda producir nuevos productos o líneas de productos ésta debe tener la capacidad de adaptar los recursos con los que cuenta (máquinas, materia prima, trabajadores, etc.) y encontrar la mejor forma de cumplir con la demanda que recibe, distintas combinaciones de los mismos de forma eficiente y eficaz para no perder su posición en el mercado y que dicho cambio en la estrategia de manufactura no resulte perjudicial al no ser implementada correctamente de forma tal que no represente gastos innecesario al implementar cambios dentro de su orden de producción, esto es en otras palabras, el poder incluir nuevas operaciones o tareas en su estrategia de producción o ensamblaje dependiendo el caso para así tener la capacidad de crear nuevos productos con las máquinas y equipos con los que cuenta.

Para una empresa que ha adoptado en su totalidad cualquiera de las estrategias de manufactura antes mencionadas el migrar de una a otra puede resultar una decisión muy riesgosa a pesar de que pueda marcar una diferencia o incremento considerable en sus ventas, es por tal motivo que si tal incertidumbre pudiese reducirse al contar con un sistema que asegure el ordenamiento de las tareas de manera óptima con los equipos y máquinas que se tienen actualmente y si además se asegura que dicha secuencia aprovechará al máximo los equipos y recursos existentes el tomador de decisiones podrá enfocarse en otros aspectos relacionados con la implementación y no dedicar todo su tiempo al proceso de manufactura. Si se cuenta con un enfoque de manufactura esbelta o se quiere migrar a tal, se debe de tener en cuenta las adecuaciones que se deben llevar a cabo para cumplir con los requerimientos que dicho sistema de producción plantea. De acuerdo a George [24] y al enfoque de producción denominado “*Lean Six Sigma*” una empresa que desea tornarse competitiva para satisfacer mejor las necesidades de sus clientes además de reducir los costos relacionados con inventarios y distribución a la par debe ser capaz de llegar a ser tan flexible (en términos de manufactura y distribución) que pueda operar bajo el esquema de Fabricar a la Orden y así disminuir la dependencia de un pronóstico de la demanda.

Rajagopalan [25] menciona que la decisión entre Fabricar a la Orden o Fabricar para Almacenar está en función de la demanda y la capacidad de procesamiento disponible, además la demanda de un producto será mayor si se considera que existe la posibilidad de que el cliente lo personalice. Como se puede notar, el éxito que tenga una empresa para posicionarse dentro del mercado depende en gran medida de la posibilidad de brindar al cliente la posibilidad de personalizar un producto sin que esto afecte de manera considerable el tiempo de entrega del mismo. Lambert [23] menciona también la tendencia a que el ciclo de vida de un producto se reduce cada vez más una flexibilidad mayor es necesaria para poder manejar la incertidumbre de la demanda.



Ahora bien en el mercado existen distintas soluciones computacionales que ofrecen a una empresa manufacturera la posibilidad de manejar aspectos relacionados con su producción, almacén, manejo de inventarios (planeación, administración, control, etc.) Dentro de las soluciones comerciales que podemos encontrar una de las más conocidas a nivel internacional podemos encontrar a SAP⁵ que brinda un software que brinda un enfoque orientado al manejo de costos, conocer el valor del inventario o procesar órdenes de producción; *EVOLVER* es otra solución desarrollada por la compañía Palisade la cual se caracteriza por ofrecer un complemento para Excel que permite solucionar problemas de optimización como la secuenciación de trabajos, planeación de la producción, manejo del inventario, planeación de la capacidad entre otros y todo basado en la optimización basada en algoritmos genéticos. Siemens ofrece su suite *SIMATIC IT Production Suite* que ofrece la posibilidad de mejorar los procesos con los que cuenta la empresa al reducir los tiempos muertos e incorporar flexibilidad y al mismo tiempo asegurar la eficiencia y calidad en todo momento.

1.4 OBJETIVO GENERAL

Desarrollar un Algoritmo Genético que sea capaz de resolver problemas multiobjetivo relacionados directamente con la secuenciación de trabajos en una línea de ensamblaje/producción bajo el esquema de Justo a Tiempo con un módulo de mutación autoadaptable que pueda ser aplicado a una empresa que maneje el esquema fabricar a la orden (*make to order*).

1.4.1 OBJETIVOS ESPECÍFICOS

Desarrollo de un algoritmo genético multiobjetivo que permita:

- Implementar un esquema de autoajuste para la razón de mutación y conocer el comportamiento y efecto durante un tiempo determinado (100, 500 generaciones)
- Conocer los resultados de la solución propuesta en dos conjuntos de problemas de secuenciación de operaciones para una línea de ensamble que opera bajo el esquema de Justo a Tiempo.
- Comparación de resultados contra otras heurísticas de solución; AG típico, AG autoadaptable, AG típico con búsqueda local.

⁵ De la suite *SAP Business One* en específico se hace referencia a los módulos relacionados con la Administración del almacén y de la producción.

Con base en los objetivos antes propuestos se derivan las siguientes hipótesis que serán resueltas al final de ésta investigación:

1.5 HIPÓTESIS

- Es posible desarrollar un módulo de mutación autoadaptable para un algoritmo genético multiobjetivo que pueda minimizar de manera simultánea la razón de uso de la materia prima y la cantidad de operación de preparación y ajuste en cada máquina durante la producción/ensamble.
- El módulo autoadaptable que define la razón de mutación en el algoritmo mejorará de manera considerable la calidad de los resultados finales y reducirá el número total de evaluación de funciones al final de su ejecución en comparación con un AG cuya razón de mutación sea definida *a priori*.

1.6 ALCANCES Y LIMITACIONES

- Se desarrollará un prototipo funcional con un módulo autoadaptable para definir de manera dinámica la razón de mutación que ocupará el AG sin necesidad de definir un valor antes de ejecutar el algoritmo.
- En éste trabajo solo se modelarán problemas multiobjetivo de asignación de trabajos sin importar el número de máquinas dentro de la línea de ensamblaje sin embargo los problemas se limitarán a un total de 20 trabajos/estaciones de trabajo para fines de estudio.

1.7 ORGANIZACIÓN DE LA TESIS

La presente tesis se encuentra distribuida de la siguiente manera:

- En el capítulo 2 se presentan las estrategias clásicas de solución para problemas multiobjetivo, la complejidad computacional que cada uno de ellos comprende y se presentan los algoritmos evolutivos que ofrecen un mejor rendimiento y capacidad de solución para el caso de los problemas de secuenciación, así como algunas meta heurísticas para resolver dichos problemas.
- En el capítulo 3 se presenta la descripción del problema de secuenciación de trabajos en una línea de ensamblaje con un sistema de producción Justo a Tiempo, el modelo matemático que describe tal problema al igual que se presentan las variables y funciones objetivas involucradas en la formulación del problema.

- En el capítulo 4 se describe el modelo elaborado, los operadores genéticos que puede ocupar, se presenta el operador de mutación autoadaptable y las instancias de los problemas a resolver.
- En el capítulo 5 se discutirán los resultados obtenidos por el algoritmo propuesto contra los mejores resultados obtenidos por las otras estrategias utilizadas para la comparación.
- En el capítulo 6 se presentan las conclusiones y el trabajo futuro que de esta tesis se deriven.

1.8 APORTE

El presente trabajo de investigación tiene como objetivo el generar un algoritmo genético que resuelva problemas multiobjetivo manejando una estrategia de clasificación de soluciones no dominadas para elegir a los individuos más aptos y con base en el desempeño que presenten a lo largo del proceso evolutivo autoajustar de manera dinámica la razón de mutación en lugar de establecer un valor con base en la prueba y error.

CAPÍTULO II

2.1 OPTIMIZACIÓN MULTIOBJETIVO

Un problema multiobjetivo se puede definir [26] como el problema de encontrar:

“un vector de variables de decisión que satisface restricciones y optimiza una función vectorial cuyos elementos representan las funciones objetivo. Tales funciones componen una descripción matemática de criterios de desempeño que usualmente están en conflicto entre sí. Por lo tanto el término “optimizar” significa encontrar dicha solución de forma tal que arroje los valores aceptables para el tomador de decisiones de todas las funciones objetivo.”

Partiendo de la definición anterior es importante mencionar que las restricciones son características que describen las condiciones bajo las cuales el problema se está planteando y que las tales deben ser satisfechas para poder considerar a una solución aceptable (modelan limitaciones bajo las cuales el modelo matemático se rige). Tales restricciones se representan en términos de igualdades o desigualdades. Tomando en consideración todo lo antes mencionado la definición general [15] de problema multiobjetivo sería como sigue:

$$\begin{aligned}
 & \text{Minimizar / Maximizar } f_m(x), \quad m = 1, 2, \dots, M; \\
 & \text{Sujeto a: } g_j(x) \geq 0, \quad j = 1, 2, \dots, J; \\
 & \quad h_k(x) = 0, \quad k = 1, 2, \dots, K; \\
 & \quad x_i^{(L)} \leq x_i \leq x_i^{(U)}, \quad i = 1, 2, \dots, I.
 \end{aligned}$$

La solución x se define como un vector de n variables de decisión: $x = [x_1, x_2, \dots, x_n]^T$ que cumplan con j desigualdades y k igualdades, ahora bien el último conjunto de restricciones del modelo antes expuesto se denominan límites variables que definen bajo que rango se estará analizando la variable x_i , de igual manera describen el espacio de decisión bajo el cual se estará definiendo el problema. Dicho espacio se denomina espacio de decisión que en los problemas multiobjetivo resulta ser en ocasiones multidimensional por el número de funciones objetivo, de igual forma se cuenta con el espacio objetivo que mapea directamente un punto en el espacio de decisión con un valor determinado en el espacio objetivo, y con base en el hecho de que se debe de satisfacer varias funciones objetivo se construye en vector de objetivos.

Dentro de los métodos de solución clásicos para resolver un problema multiobjetivo [27] podemos encontrar los siguientes:

- a) Suma Ponderada
- b) Función de Utilidad
- c) Compromiso (comportamiento de búsqueda de un objetivo)
- d) Ordenamiento Lexicográfico
- e) Pareto

El primero de los métodos tiene como característica principal la integración de las distintas funciones objetivo en una función objetivo al asignar un valor numérico w o peso dependiendo de la preferencia del usuario entre los distintos objetivos, para el caso de la función de utilidad se intenta hacer un mapeo de los puntos del espacio de solución en valores reales mientras mayor sea el valor obtenido será colocado en el primer lugar de preferencia y así consecutivamente haciendo siempre la comparación en pares de valores. Para el caso del método de Compromiso se basa en la búsqueda de la menor distancia entre el punto a analizar y el punto ideal (el cual no es alcanzable pues no puede satisfacer simultáneamente todos los objetivos) y con base en una función de castigo se puede ordenar cada uno de los elementos del vector de criterio. El Ordenamiento Lexicográfico es un método que se basa en indexar los objetivos, cada uno de los cuales obtiene ese índice al compararlos entre sí. Por último el método de Pareto asume que no se tiene ninguna preferencia entre los distintos objetivos y solo el mayor valor es el preferido, para clasificar a los mismo de acuerdo a éste método se toman dos elementos y uno se dice que es mayor que otro en caso de que exista al menos un componente que sea mayor que el otro dentro de todo el espacio de solución. Deb [15] define el dominio de Pareto entre dos soluciones como sigue: Una solución $x^{(1)} \prec x^{(2)}$ (o $x^{(1)} \succ x^{(2)}$), si la solución $x^{(1)}$ es estrictamente mejor que la solución $x^{(2)}$ en todos los objetivos M.

Ahora bien de los acercamientos básicos antes descritos (excepto el de Pareto) tienen una característica en común al trabajar en la adaptación del problema de acuerdo a cada estrategia de solución por lo que la dificultad inherente del problema que está directamente relacionada con aspectos como: las propiedades de las funciones objetivo (si es continua, discontinua, lineal no lineal, etc.) , las funciones de restricción y el tamaño del problemas se duplica pues si no se hace dicha adaptación de manera adecuada el resultado final estará sesgado o incompleto, un ejemplo claro de lo anterior lo podemos encontrar para el caso de la suma ponderada que depende directamente de la preferencia del usuario en la asignación de pesos por lo que podemos encontrar demasiada variación que dependerá de los que al usuario le parezca importante y también del valor de los pesos que asigne teniendo una serie casi infinita de escenarios al resolver un problema multiobjetivo utilizando este acercamiento básico.

Podemos encontrar también estrategias que combinan varias funciones en una sola, a las tales se les conoce como *funciones agregadas* y resultan efectivas cuando se tiene una idea sobre comportamiento de las funciones además de que es fuerte computacionalmente hablando en términos de aprovechamiento de recursos dado que puede generar fuertes soluciones NO dominadas, y puede ser usada como solución inicial de otras técnicas, sin embargo una desventaja considerable recae en la asignación de los pesos q no se sabe como determinarlos apropiadamente cuando no se tiene suficiente información del problema.

La programación basada en metas [28] tiene como principal estrategia el definir metas para cada objetivo y se encarga de minimizar la diferencia entre los valores obtenidos y las metas de cada uno de ellos. Es eficiente en términos computacionales pero sigue dependiendo de la parte del decisor para definir las metas/prioridades a alcanzar y se puede lograr si se conoce la forma del espacio de búsqueda y si es no linear ya trono solo resulta bueno para aproximaciones donde el problema tenga comportamiento lineal. Una estrategia similar es la que parte del logro de metas al implementar el manejo de un vector que contiene pesos, pero además incorpora un vector de metas para cada uno de los objetivos, el tomador de decisiones define con él cual es la meta que se desea alcanzar para cada objetivo, el resultado de este algoritmo le permitirá conocer al decisor si la meta es alcanzable o no por el otro lado en caso de que exista una solución y quizás una mejor a la fijada ésta será arrojada. Las desventajas es que al igual que en los algoritmos anteriores se depende de las metas que se definan para los objetivos lo que puede ocasionar un resultado ambiguo o no utilizable para tomar una decisión, las ventajas que ofrece son su simplicidad en la implementación y su eficiencia computacional.

El Método de restricción ϵ se basa en minimizar un problema multiobjetivo y plantearlo como uno uniobjetivo partiendo de una función objetivo (la que se considere como la más importante) y contemplar a los otros objetivos como restricciones (dentro de ciertas fronteras definidas por niveles de ϵ_i). El conjunto de valores de ϵ_i nos permite definir la región de Pareto y el algoritmo se detiene cuando el decisor encuentra una solución satisfactoria. El estar codificando las funciones objetivo toma demasiado tiempo y en ocasiones puede llegar a ser imposible dada la gran cantidad de objetivos además de arrojar soluciones no dominadas débiles.

De la Investigación de Operaciones (IO) se han formulado más de 20 distintas propuestas [29] para resolver este tipo de problemas sin embargo dentro de las más comúnmente usadas podemos encontrar aquellas que parten de los principios de la teoría de juegos y la técnica de optimización Min Max. El primero es un método que plantea el problema desde la perspectiva de optimizar dos objetivos asignándolos a dos jugadores distintos que tienen 2 esquemas distintos de juego: cooperativo o no cooperativo, al momento de competir ambos jugadores llega un momento en el cual ninguno puede ir mas allá para conseguir su objetivo sin perder de manera considerable contra su oponente y a tal punto se conoce como solución de equilibrio de Nash. Ofrece la ventaja de ser eficiente en términos computacionales, sin embargo además de que se puede implementar un torneo entre un numero K de jugadores y obtener varios puntos de equilibrio de Nash que permitirían construir una frontera de Pareto.

La segunda propuesta de la IO que parte del enfoque Min Max proviene de la Teoría de Juegos al dar solución a situaciones conflictivas desde una perspectiva de juego. El modelo lineal fue propuesto por Jutler y Solich y ésta estrategia obtiene su óptimo al comparar las desviaciones obtenidas con relación al valor mínimo de cada función de manera individual. Si intentamos minimizar una función encontraremos los incrementos relativos mientras que al maximizar una función hallaremos los decrementos relativos. Dentro de las desventajas que se pueden encontrar podemos mencionar la presión para el cruce que se origina de ciertas combinaciones de pesos lo cual añade una variable no tan fácil de definir para el AG sin mencionar también el hecho de que no verifica el no dominio entre los resultados obtenidos. Existen además algunas variantes de tal propuesta, Coello propuso dos variantes al contemplar la participación del usuario al definir un conjunto pesos que permitirán la creación de una serie de subpoblaciones que evolucionarán de manera independiente y concurrente de forma tal que se pueda converger de manera individual a cada uno de los puntos que componen la frontera de Pareto. El segundo acercamiento basado en el método Min Max incorpora el manejo de un vector ideal en cada generación además del dominio de tipo Min Max (un individuo podría ganar un torneo por el hecho de que su máxima desviación con respecto al vector ideal sea la menor de todos aquellos individuos que estén compitiendo), lo anterior junto con condiciones de apareamiento permite mantener soluciones factibles en todas las generaciones. La desventaja que caracteriza este método gira en torno a la definición apropiada del factor que define la forma en la que se comparte información entre los individuos y así regir el cruzamiento de los individuos.

Por último podemos encontrar un acercamiento basado en el teorema de contacto para detectar soluciones óptimas de Pareto el cual es muy similar a utilizar la estrategia Min Max antes expuesto sin embargo tiene como característica especial el que intenta evaluar a un vector solución dependiendo de la distancia que tenga con respecto al conjunto de puntos que componen la frontera de Pareto, así también maneja la asignación de pesos para cada uno de los objetivos ni una función de compartición para mantener la diversidad entre la población aunque por desgracia este método no ha sido aplicado a problemas reales.

Los métodos clásicos antes mencionados se pueden clasificar en 2 categorías [30]; Métodos Directos y Métodos basados en un gradiente, para el primer caso solo la función objetivo y sus restricciones son usadas para dirigir la búsqueda mientras que para el segundo caso se hace uso de las derivadas de primer o segundo orden de la función objetivo y sus restricciones para guiar el proceso de búsqueda además de resaltar el hecho de que para resolver los problemas deben intentar una sola solución a la vez por lo que si se tiene un número grande de posibles soluciones posibles el encontrar el conjunto de candidatos adecuados tomará demasiado tiempo, además de que se basan en preferencias basadas en el tomar de decisiones por lo que la respuesta final puede ser un tanto sesgada y quizás no identificar alguna tendencia no conocida dentro del problema.



Si bien, los métodos anteriores ofrecen una respuesta a distintos problemas multiobjetivo en muchas ocasiones no son computacionalmente eficientes al requerir demasiados recursos (procesador, tiempo de ejecución, espacio en disco duro, etc.) para obtener una solución óptima conforme el tiempo necesario para realizar las operaciones necesarias y los problemas a resolver se tornan cada vez más complejos en términos computacionales ha sido necesario encontrar nuevas estrategias de solución para los problemas multiobjetivo.

A diferencia de los métodos de cómputo duros, el Cómputo Suave da cabida a imprecisión e incertidumbre [5], dentro de dicha clasificación podemos encontrar; a la Lógica Difusa, Redes Neuronales, Razonamiento Probabilístico, Algoritmos Genéticos y Teoría del Caos. Ahora bien, el cómputo evolutivo se considera una herramienta del cómputo suave basándose principalmente en conceptos de evolución artificial. Dentro del cómputo evolutivo se pueden encontrar una serie de técnicas de búsqueda adaptativa basadas en poblaciones que pueden ser perfectamente aplicadas a problemas de optimización.

Como se mencionó anteriormente los Algoritmos Genéticos son técnicas de cómputo evolutivo que toman como base de funcionamiento la adopción e implementación de principios que se observan en la naturaleza como son; la evolución, permanencia del individuo más apto, adaptación, aprendizaje, mutación, entre otros. Algunos de los acercamientos computacionales más conocidos dentro de esta área podemos mencionar:

- a) Algoritmos genéticos [6]
- b) Estrategias Evolutivas [11]
- c) Programación Evolutiva [31]
- d) Programación Genética [32]

Algunos casos de aplicación práctica en distintas ramas del conocimiento y en problemas reales se pueden mencionar los siguientes:

- a) Selección y Clasificación en el diagnóstico de cáncer cervical [33]
- b) Planeación de radioterapia por Optimización Multicriterio (Ehrgott & Burjony, 2001)
- c) Contención de contaminación de aguas subterráneas [34]
- d) Genetic Land: Modelación del cambio en el uso del suelo utilizando algoritmos evolutivos [35]
- e) Programación Genética Artificial para la predicción de crisis epilépticas [36]

John Holland [6] estableció los antecedentes formales de los AG a pesar de no tener la intención original para proponer un algoritmo que resolviera problemas en específico sino que intentaba modelar y estudiar el fenómeno de adaptación tal y como sucede en la naturaleza para incorporarlo a sistemas computacionales. Aún así los AG muestran una considerable ventaja en comparación de otros procedimientos de optimización basados en la búsqueda (Búsqueda Aleatoria, Hill Climbing, etc.) principalmente por su balance entre la exploración del espacio de resultados y la explotación de la mejor solución, razón por la cual se han tornado en una de las estrategias más utilizadas por algunos investigadores [37].

Los algoritmos genéticos se diferencian de manera considerable con respecto a otros métodos de solución [38] de cuatro formas principalmente:

- (1) Los AG trabajan en la codificación y no en los parámetros como tal. Por lo anterior los AG pueden manejar variables enteras o discretas.
- (2) Los AG buscan en una población de puntos, no un punto solamente y debido a tal propiedad pueden hallar soluciones globalmente óptimas.
- (3) Los AG usan solo información de la función objetivo, no necesitan conocimiento adicional (derivadas) lo que permite que puedan lidiar con funciones continuas, continuamente diferenciables y no diferenciables.
- (4) Los AG usan reglas de transición probabilísticas y no determinísticas.

2.2 ELEMENTOS QUE CONFORMAN UN AG

Al modelar la evolución que es un hecho que ocurre en la naturaleza los AG adoptan varios elementos de la misma dentro de su implementación y es importante definir de manera breve cada uno de ellos además de la participación que tienen y la función que desempeñan. Dentro de los más importantes podemos encontrar:

- **Gen.** Unidad fundamental que contiene información genética que se encarga de la herencia de una o varias características particulares, tomando un valor determinado (basado en la codificación que se utilice).
- **Cromosoma.** Se compone de una sucesión lineal de genes.
- **Alelo.** Representa el estado en el que se puede encontrar un gen o el valor que puede tomar en un momento determinado.
- **Locus.** Posición particular de cada gen en un cromosoma.
- **Genotipo.** Se define como la codificación de una solución del problema en cada individuo.
- **Fenotipo.** Estructura que define a un conjunto de soluciones, puntos o parámetros.
- **Individuo.** Comprende a cada una de las soluciones potenciales que pueden resolver el problema en cuestión.
- **Población.** Es el conjunto de individuos o soluciones disponibles.

Al tomar principios naturales los AG adoptan de igual manera tres procesos básicos y fundamentales conocidos como operadores genéticos que intervienen en el proceso de evolución de una población los cuales se describen a continuación:

- **Selección.** Se basa en el principio natural de la supervivencia del individuo más apto al tomar de la población a los individuos que muestren un desempeño sobresaliente para participar en el proceso de reproducción. Existen varios tipos de heurísticas en este proceso de los cuales se puede mencionar el modelo de ruleta, elitista, Boltzmann, etc.
- **Recombinación (Cruce).** Se deriva del fenómeno natural del apareamiento sin embargo en términos de aplicación de los AG se presenta cuando se toma información genética de dos individuos que se denominan padres y que se combina parte de dicha información de manera que se genera un nuevo fenotipo para dar paso a un individuo nuevo o “hijo” o nuevos dependiendo de la definición del operador, al igual que en el caso anterior existen variantes de este operador teniendo uno o varios puntos de cruce y siempre regidos por reglas de aleatoriedad para evitar una tendencia o repetitividad de características.
- **Mutación.** Tiene como primordial objetivo el introducir nuevos alelos a la población al dar paso a individuos que de manera tradicional (cruce) no se podrían originar. Mientras que los dos operadores anteriores producen variantes de las soluciones con las que se cuentan actualmente la mutación genera saltos nuevos dentro de dicha tendencia. Su aplicación comprende en seleccionar un individuo al azar y cambiar uno o más alelos del mismo en un locus también al azar. A diferencia de los dos operadores anteriores la Mutación sucede de manera esporádica con una probabilidad muy baja de ocurrir.

Para poder tener un criterio de comparación claro y homogéneo al momento de evaluar a los distintos individuos de una población y con base en tal evaluación aplicar los operadores genéticos es importante definir un procedimiento bajo el cual estaremos llevando a cabo tal evaluación. En la naturaleza a tal proceso se conoce como aptitud y partiendo del principio de la sobrevivencia del más apto tal y como lo definió Darwin en “El Origen de las Especies”; las soluciones se someten a una evaluación con base en una función matemática conocida como “*función de aptitud*” o “*función objetivo*” y con base en el resultado que obtengan se puede conocer que tan buena o mala una solución es. Cabe resaltar el hecho de que tal función de aptitud no se encarga de diferenciar las distintas soluciones disponibles y no muestra la solución óptima.

Partiendo de los conceptos y los operadores genéticos antes definidos la estructura bajo la cual un AG trabaja es la siguiente:

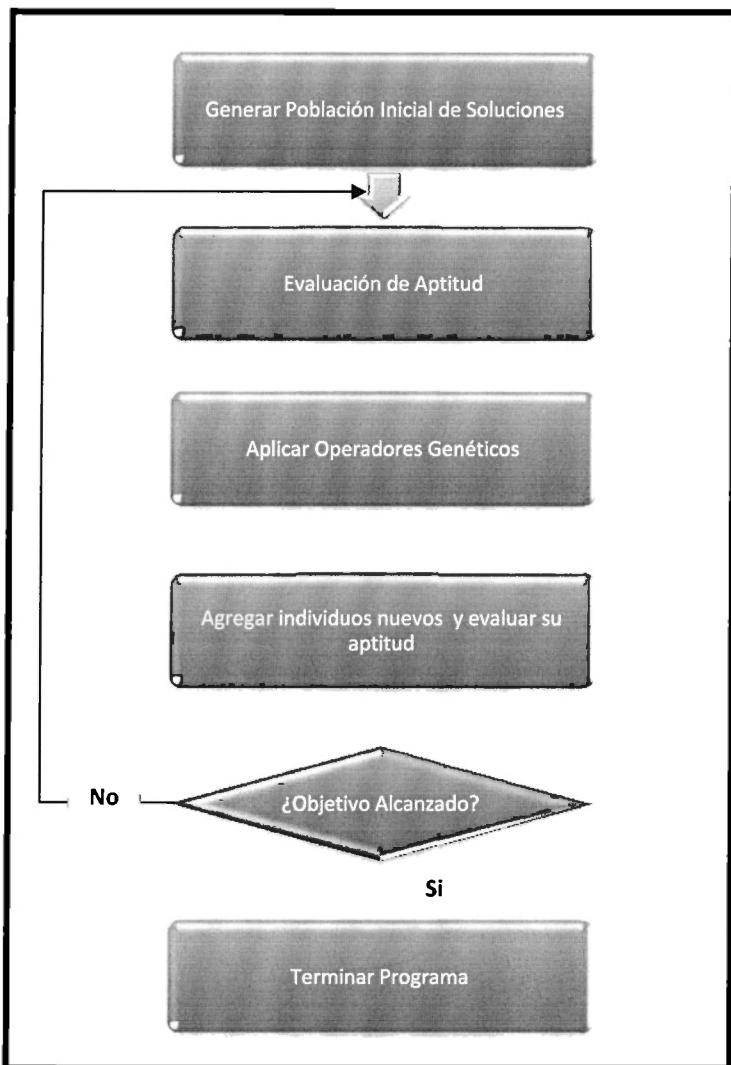


Figura 2 Modelo esquemático de operación para un Algoritmo Genético tradicional

Se parte de una población o grupo de soluciones generadas al azar (dependiendo del problema a resolver en ocasiones dichas soluciones no son factibles o adecuadas por lo que pueden eliminarse y necesitarse que se generen nuevos individuos que los reemplacen), calcula de toda la población su aptitud dependiendo de una función previamente definida y bajo un criterio de selección (elitismo, torneo, ranking, etc.) se seleccionan a los individuos más fuertes o aptos para que a partir de los mismos se generen nuevos individuos que hereden parte de la información de sus padres, los que no cumplen con dicho criterio de selección son descartados y la nueva población se conforma de los individuos producto de los más aptos. Lo anterior se repite consecutivamente dando paso a generaciones y ocasiona que las soluciones débiles o que no cumplen con un nivel de aptitud deseado o aceptable son descartadas.

Partiendo del hecho de que el operador de cruce es probabilístico puede darse el caso que algunos individuos permanezcan intactos lo que asegura la permanencia de las mejores soluciones aunque puede también presentarse el caso de que el nuevo individuo sea peor que sus predecesores pero tal situación ocurre con un poco probabilidad o con un espacio generacional muy amplio.

El pseudo código que describe el gráfico anterior sería como sigue:

- (Paso 1) Inicio
- (Paso 2) Genera población inicial
- (Paso 3) Para $i = 0$ hasta condición de paro haz:
 - (Paso 4) Para $j = 1$ hasta tamaño de población haz:
 - (Paso 5) Evaluar la Aptitud de cada individuo,
 - (Paso 6) Calcular la Aptitud de toda la población,
 - (Paso 7) Calcular la probabilidad de selección para cada individuo,
 - (Paso 8) Calcular la probabilidad q acumulada para cada individuo,
 - (Paso 9) Generar un número aleatorio r ,
 - (Paso 10) Si $r < q$ entonces seleccionar al primer individuo, si no seleccionar al siguiente,
 - (Paso 11) Aplicar operadores Genéticos,
 - (Paso 12) Seleccionar individuos para cruce de acuerdo a la probabilidad de cruce,
 - (Paso 13) Seleccionar individuos para mutación de acuerdo a la probabilidad de mutación,
 - (Paso 14) Obtener al nuevo individuo para crear la nueva población,
 - (Paso 15) Si la condición de paro se cumple entonces terminar, si no ir al Paso 3.

2.3 AG MULTIOBJETIVO REPRESENTATIVOS

Dentro de los AG más representativos para atacar problemas multiobjetivos se puede destacar la propuesta de Shaffer VEGA [39] (Algoritmo Genético basado en la Evaluación Vectorial) que tiene como estrategia de solución el generar una serie de subpoblaciones que depende directamente del número de objetivos, dichas soluciones son no dominadas en un ambiente local cuando se analiza en términos de cada una de las subpoblaciones, aunque si se analizan en un contexto general las mismas pueden llegar a ser dominadas. En este método se presenta un fenómeno muy característico y similar a lo que pudiera ocurrir en la naturaleza denominado “especiación” que hace referencia al surgimiento de soluciones que superan el desempeño de la media sin embargo no pasan de ese punto y terminan convergiendo en un punto en lugar de dibujar la región de Pareto, en esta estrategia de solución para corregir tal fenómeno se recurre a la implementación de heurísticas que refinen el proceso de cruce entre los individuos al aplicarlo entre las distintas subpoblaciones. La ventaja que ofrece este método recae principalmente en su simplicidad en términos de complejidad computacional aunque por desgracia tiene una gran deficiencia para generar soluciones óptimas de Pareto ante espacios de búsqueda no convexos.

También está la propuesta de Fonseca y Fleming [40] que parte de un ordenamiento basado en el número de individuos por los cuales es dominado. Todos los individuos no dominados se les asignan una clasificación como primer grupo y así consecutivamente hasta terminar de ordenar toda la población, en cada grupo se promedia su aptitud lo que permite que la aptitud de toda la población se mantenga constante y a la vez mantiene una presión adecuada en el proceso de selección.

De igual forma proponen una técnica de agrupamiento (nichos) lo que evita que el tomador de decisiones se vea abrumado con información inservible (cuando la frontera de decisión es muy amplia) por lo que se presenta información agrupada de forma tal que así aprenda sobre el comportamiento de dicha frontera y así refinar sus requerimientos.

Al igual que Fonseca y Fleming al manejar nichos Horn, Nafpliotis y Goldberg [41] parten del mismo esquema pero implementan una estrategia para ejercer mayor presión en el proceso de selección al desarrollar un método de selección basado en torneo, los individuos que participan en tal son elegidos al azar y compiten contra un porcentaje de la población total. Ahora si en tal torneo se presenta un empate, esto es que ninguno de los individuos sea dominado o se dominen entre sí y para no aplicar alguna técnica de degradación de aptitud el criterio de decisión parte del número de individuos que se encuentren en su nicho, por lo tanto aquél que tenga el menor número de individuos en su nicho es el ganador del torneo. A esta estrategia de reciprocidad la llamaron “reciprocidad equivalente de clase” y tal estrategia se puede aplicar a cualquier espacio de búsqueda no solamente aquél que es toma de un acercamiento de Pareto orientado a problemas multiobjetivo.

Otros acercamientos que han sentado base en la solución de problemas multiobjetivo se puede mencionar el trabajo de Zitzler y Thiele [42] al proponer el Algoritmo Evolutivo Fuerte de Pareto⁶ que tenía como principal ventaja el manejar un archivo externo para almacenar las soluciones no dominadas desde el inicio de la búsqueda además de implementar una técnica de agrupación llamada “método de vinculación promedio” cabe mencionarse que ambos autores también hicieron una segunda propuesta basados en el mismo algoritmo (SPEA2) que maneja una asignación de aptitud más detallada al igual que una estrategia de densidad de vecindarios. Ahora bien Deb partiendo del algoritmo NSGA hace una propuesta de una segunda versión (NSGA-II) [43] computacionalmente eficiente donde implementa elitismo y una comparación entre un conjunto de soluciones para así asegurar la diversidad durante todo el proceso.

Lo antes expuesto presenta algunas propuestas para resolver problemas multiobjetivo, sin embargo es importante no perder el enfoque en el tipo de problemas que se desea resolver en este trabajo de investigación, los problemas de secuenciación se enfocan en la asignación de recursos a una actividad o proyecto determinado y se conocen como problemas de secuenciación de trabajos con ruta variada (*Job Shop* por sus siglas en inglés) además de que comprende un área de investigación⁷ [44] importante que tiene que ver directamente con la teoría y metodología fundamental para la planeación de la producción y los problemas de secuenciación en general.

⁶ Strength Pareto Evolutionary Algorithm por sus definición en Inglés

⁷ Job Shop Scheduling Problems por su término en inglés.

Es por tal que toma gran relevancia para las empresas manufactureras al tornarse de una ventaja competitiva en un requerimiento imperativo si es que se desea mantener la competitividad en este mercado tan demandante y cambiante al abastecer los pedidos de los clientes aunque lleguen a destiempo y se le brinde un elevado grado de personalización en el producto a ofrecer.

Los problemas de secuenciación se definen como problemas de minimización cuya complejidad en términos computacionales es de tipo NP duro [45] por lo tanto cualquier instancia o caso específico es igual o tan complejo para ser resuelto por métodos tradicionales. Osman define a un problema de secuenciación como un problema que implica la asignación de tiempo para órdenes de trabajo en distintos esquemas de producción para minimizar al mínimo el tiempo de entrega o maximizar la utilización del equipo.

Según Pinedo [46] existen varias clases de problemas de secuenciación partiendo de las máquinas disponibles y su configuración:

- A) Una sola máquina o máquinas en paralelo
 - a. Una máquina.
 - b. Máquinas idénticas en paralelo.
 - c. Máquinas en paralelo con diferentes velocidades.
 - d. Máquinas no relacionadas en paralelo.

- B) Varias máquinas en serie
 - a. Flujo Continuo (Flow Shop)
 - b. Flujo Continuo Flexible (Flexible Flow Shop)
 - c. Ruta Variada (Job Shop)
 - d. Ruta Variada Flexible (Flexible Job Shop)
 - e. Ruta Abierta (Open Shop)

En dichos problemas un total de n trabajos deben ser terminados utilizando un número finito m de máquinas, sin embargo cada trabajo tiene un número o de operaciones que deben ser procesados en una máquina (ya sea distinta). Cada máquina solo puede procesar un trabajo a la vez en un tiempo definido y lo debe procesar hasta el final sin interrupción.

Para dar una solución óptima a un problema como el antes mencionado desde un enfoque uniobjetivo lo que se busca es elaborar una secuencia que minimice el tiempo total o quizás que minimice el tiempo muerto (tiempo en el cual una máquina no se está ocupando), sin embargo dichos planteamientos requieren que muchos aspectos directamente relacionados al problema sean fijos o se definan previamente lo que presenta un riesgo al obtener una respuesta si es que el planteamiento no fue el adecuado.

En un planteamiento clásico de un problema de secuenciación de trabajos donde se cuenta con toda la información necesaria Pinedo [46] la define como sigue:

- m denota el número de máquinas
- n denota el número de trabajos
- j el subíndice hace referencia a un trabajo
- i el subíndice hace referencia a una máquina
- $p_{(i,j)}$ se refiere al tiempo de procesamiento del trabajo j en la máquina i
- r_j tiempo de liberación del trabajo j , el tiempo en el que se puede empezar a procesar el trabajo j
- d_j fecha límite para la cual se debe entregar o terminar el trabajo j
- w_j peso asignado a cada trabajo que define la prioridad

Dentro de los elementos adicionales que componen un problema encontramos restricciones; de procesamiento como la interrupción del trabajo de una máquina en una tarea⁸ determinada, de precedencia en el caso que un trabajo deba respetar algunas consideraciones con respecto a su procesamiento, de elegibilidad de una máquina si es que un trabajo específico (M_j) debe no puede ser procesado por cualquier máquina disponible.

Tradicionalmente y bajo un esquema uniobjetivo al encontrar la solución del problema de secuenciación de trabajos de ruta variada se busca minimizar el tiempo para completar el trabajo j en la máquina i siendo denotado como C_{ij} , o el tiempo en el que el trabajo j abandona la última máquina del sistema C_j . Por lo anterior algunas posibles funciones objetivo que se derivan de dicho planteamiento serán según Pineda:

1. Minimizar el tiempo máximo de terminación de todas las tareas⁹ C_{\max} lo cual implica un uso eficiente de las máquinas.
2. Total ponderado del tiempo de terminación o tiempo de flujo ($\sum W_j C_j$) lo que permite conocer los costos de inventario en los que se incurre con tal secuencia.
3. Tardanza Total Ponderada ($\sum W_j T_j$) es una generalización de la función de costo del total ponderado del tiempo de terminación.
4. Ponderado de la cantidad de trabajos tardíos ($\sum W_j U_j$) se considera como una medida que muestra el estado de eficiencia del sistema o la secuencia actual.

⁸ Preemption por su nombre en inglés.

⁹ Makespan por su término en inglés.

Un ejemplo de un problema de secuenciación de 3 trabajos y 3 máquinas (o de 3×3) sería como lo muestra la Figura 2.

Trabajo	Secuencia de las Máquinas			
	M1	J1	J2	J3
1	1 (3)	2 (3)	3 (3)	
2	1 (2)	3 (3)	2 (4)	
3	2 (3)	1 (2)	3 (1)	

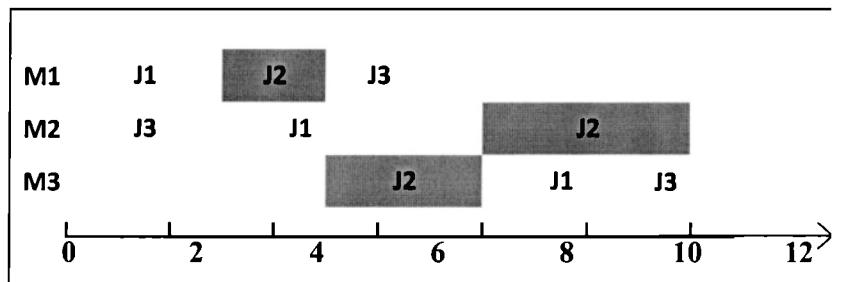


Figura 3: Ejemplo de secuenciación de trabajos para un problema de tres máquinas y 3 trabajos en 3 máquinas.

El primer número de cada columna representa la máquina M_i , en el cual la operación debe ser procesada mientras que el número que está dentro del paréntesis denota el tiempo requerido para procesar dicha operación.

Sin embargo lo antes expuesto hace referencia a un problema cuyo fin es optimizar una sola función objetivo, y ha sido exhaustivamente investigado. Giffler y Thompson [47] sentaron la base para resolver los problemas de secuenciación, en dicho trabajo el propósito final era el de encontrar una secuencia “correcta” de operaciones para ciertos trabajos considerando la precedencia y las restricciones de capacidad aunque dicha secuencia no fuera óptima en términos de los objetivos clásicos (minimizar tiempo muerto, maximizar el uso de los equipos, minimizar el tiempo de ciclo).

Garey y Jhonson [48] definieron desde los primeros inicios en el área estos los problemas de secuenciación de trabajos dentro del grupo de optimización combinatoria por lo que la complejidad computacional de este problema lo ubica en la clase NP duro, y por tal motivo no existe un algoritmo que lo resuelva en tiempo polinómico. Como se comentó anteriormente ésta es una de las principales razones de implementar técnicas de Cómputo Suave como lo son los Algoritmos Evolutivos al presentar una ventaja considerable.

Werner Aydin y Fogarty [49] combinan dos estrategias del cómputo suave al someter al proceso evolutivo un AG bajo la programación genética afectando directamente los operadores genéticos (cruce, mutación y selección) y el orden en el que se aplicarían para cada generación. Cada decisión de la programación genética se codificó en 8 bits definiendo para éste dos tipos de información: funciones matemáticas utilizadas en el modelo y terminales (valores de las variables y números aleatorios). Los operadores posibles obtenidos de la programación genética fueron; mutación usual, extender los límites de algunas decisiones en la generación de secuencias en un 10%, disminuir los límites de algunas decisiones en la generación de secuencias en un 10%, intercambiar alguna región del cromosoma de un parente, intercambiar el resto de la región del cromosoma de un parente. El desempeño de este acercamiento demostró ser superior a las técnicas tradicionales de solución llegándose a comparar inclusive con una técnica de recocido simulado modular (que parte de una población de individuos y no de uno solo) demostrando mejores resultados y menor número de evaluaciones comparado con el último método.

Aggoune [50] ataca el problema de secuenciación de trabajos en flujo continuo con restricciones de disponibilidad al incorporar tiempos de mantenimiento bajo dos esquemas (con tiempos definidos y en lapsos variables) siendo que para el segundo caso se incurre en tiempos muertos lo que ocasiona una secuencia no óptima, partiendo de lo anterior y para minimizar el tiempo total de ciclo a la vez que se minimiza el tiempo de inactividad de las máquinas se aplicó un refinamiento de secuencias utilizando un Algoritmo Genético para generar reglas de prioridad entre trabajos que servirían de directriz para un generador de secuencias óptimas que se construyó mientras que la Búsqueda Tabú se utilizó para obtener iterativamente una secuencia cuya factibilidad será probada por el generador hasta alcanzar 1000 generaciones, los resultados obtenidos mostraron una considerable estabilidad de ambas técnicas de cómputo suave en el caso de las ventanas de tiempo o lapsos variables.

Bertel y Billaut [51] presentan una heurística basada en un AG para resolver el problema de secuenciación de trabajos en un esquema de multiprocesadores considerando recirculación al tener que generar una secuencia con la posibilidad de que algún trabajo quede pendiente o quizás necesite pasar a otra máquina de nueva cuenta, el AG permite que el decisor defina los trabajos cuya prioridad es elevada al igual que aquellos que pueden dejarse pendientes en caso de algún sobre-trabajo inesperado para así generar la secuencia óptima.

Un AG multiobjetivo para resolver el problema de secuenciación de trabajos en una línea mixta de ensamblaje de distintos productos es usado por Ponnambalam [52] el cual ataca el problema haciendo uso de dos técnicas de selección distintas al implementar los nichos cúbicos de Pareto y el de selección de aptitud escalar considerando como objetivos el minimizar; la variación en el uso de las partes, los costos de preparación y el trabajo de nivelación de uso de partes mostrando mejores resultados la selección basada en nichos cúbicos de Pareto al encontrar más soluciones de tipo Pareto que el otro método de selección.

Ahora bien se puede notar en los trabajos antes expuestos que se enfocan en la aplicación de un AG para resolver problemas de secuenciación trabajos con al menos 2 objetivos como máximo sin embargo para atacar una instancia del mismo con más de 3 funciones objetivos en muchos de los casos es necesario fortalecer la heurística con otros elementos del cómputo suave(Lógica Difusa, Redes Neuronales, Razonamiento Probabilístico, etc.) o técnicas de búsqueda local (recocido simulado o búsqueda tabú) que permitan obtener mejores resultados (menor tiempo, mejores secuencias, mayor cantidad de escenarios, etc.) a lo que se conoce como hibridación

Algunas de las técnicas más conocidas basadas en algoritmos evolutivos para resolver problemas multiobjetivo son enlistadas por Coello [12], dichas técnicas manejan distintas formas de ocupar los operadores, funciones objetivo y la representación cromosómica:

- AG de Evaluación Vectorial (VEGA)
- AG de Ordenamiento Lexicográfico
- Optimización utilizando una Estrategia Evolutiva Vectorial (VOES)
- AG basado en pesos (WBGA)
- AG Multi-Objetivo (MOGA)
- AG de Nichos de Pareto (NPGA, NPGA 2)
- AG con Ordenamiento basado en No Dominio (NSGA, NSGA – II)
- AG Termodinámico (TDPGA)
- AG basado en la distancia de Pareto (DPGA)
- Algoritmo Evolutivo con Fuerza de Pareto (SPEA, SPEA2)
- AG desordenado Multi-Objetivo (MOMGA I, II, III)
- Estrategia Evolutiva con archivado de Pareto (PAES)
- Algoritmo de Selección Pareto con técnica de sobres (PESA, PESA II)
- Micro AG multiobjetivo(GA, GA2)
- Algoritmo de Optimización Multiobjetivo Bayesiano Multiobjetivo

Dentro de las propuestas para resolver un problema de secuenciación multiobjetivo e híbrido podemos destacar el trabajo de Lin y Liu [53] al trabajar con restricciones de precedencia y conflicto de recursos al combinar un algoritmo genético con un heurísticas de secuenciación basadas en reglas, el primer paso es dividir los trabajos de acuerdo a su nivel de conflicto de recursos en dos clases (con serio conflicto y sin serio conflicto) de forma tal que aquellos trabajos con serios conflictos de recursos sean atacados por el AG mientras los que no muestran conflictos serán resueltos por una serie de heurísticas basadas en reglas y al final para reducir el tiempo total se aplica de nueva cuenta el AG para optimizar la secuencia final.

Varadharajan y Rajendran [54] atacan el problema de permutación de secuencias en máquinas de flujo continuo (Flowshop) al minimizar el tiempo total de flujo y el tiempo muerto al desarrollar un algoritmo multiobjetivo con recocido simulado utilizando heurísticas tradicionales para generar las secuencias de arranque por permutación de forma tal que cumplan con los 2 objetivos del problema, el algoritmo obtiene secuencias no dominadas al respetar una función de probabilidad que selecciona entre los objetivos obteniendo secuencias no dominadas lo que permite cubrir el espacio de búsqueda uniformemente.

Por su parte Arroyo y Armentano [13] se basan en un enfoque *a posteriori* para optimizar un planteamiento multiobjetivo del problema de secuenciación con flujo continuo (*Flow Shop*) para optimizar dos instancias que ataquen un par de objetivos (Tiempo total de ciclo, máxima demora y tiempo total de flujo) con un algoritmo genético hibridizado con una técnica de búsqueda local paralela.

Su propuesta se caracteriza por la incorporación de; elitismo al separar un grupo de mejores soluciones que se incorporan de vez en vez en la población general, dominio de Pareto (clasificación rápida no dominante) para clasificar cada uno de los individuos dentro de la población además de promover la dispersión a través de la normalización de las distancias de saturación y al mismo tiempo asignar el valor de aptitud a cada individuo en la población mientras que para la búsqueda multiobjetivo local paralela (que se ejecuta con un intervalo de generaciones) se crean una serie de vecindarios o conjuntos que se generan a partir de un conjunto de soluciones no dominadas para encontrar nuevas soluciones no obtenidas por el algoritmo y que se incluirán en el set elitista para la nueva generación.

Gonçalves, Mendes y Resende [55] elaboran un AG híbrido para resolver el problema de secuenciación de ruta variada (*Job Shop*) aplicando un algoritmo genético para desarrollar las reglas de prioridad que se aplicarán a cada uno de los trabajos que se tienen que ordenar haciendo uso de la técnica de codificación “*llaves aleatorias*” para los cromosomas y A cuyo resultado se aplica a una algoritmo de ordenamiento que genera secuencias activas, ya por último a la secuencia obtenida se le aplica un algoritmo de búsqueda local (desarrollado por Nowicki and Smutnicki)de forma tal que se pueda buscar una alternativa para minimizar el tiempo total de ciclo, en caso de no conseguirlo introducir esta secuencia en el algoritmo para buscar la forma de mejorarla en alguna manera. Es importante mencionar que el algoritmo fue comparado mostró un desempeño considerable al obtener la mejor solución conocida en un 72% de un total de 43 problemas a los que se aplicó el problema contra 12 estrategias para resolver este problema.

Watanabe, Ida y Gen [56] proponen una técnica de solución del problema de secuenciación de trabajos en un esquema de ruta variada (*Job Shop*) al desarrollar un algoritmo genético que implementa la técnica de adaptabilidad del área de búsqueda al caracterizarse principalmente por satisfacer de la mejor manera los requerimientos de la optimización global y local, cabe destacar la generación de secuencias activas gracias al uso de una codificación de tipo “permutación” en los cromosomas los cuales muestran el secuencia para los trabajos a ordenar. Una vez que se aplican los operadores regulares genéticos de cruce y mutación se pasa a la fase de búsqueda al utilizar operadores de cruce y mutación creados para obtener individuos que sean al menos mejores que el peor individuo de la población. La comparación del desempeño se hace contra tres algoritmos que aplican también una estrategia de búsqueda adaptativa partiendo de dos problemas distintos (10 x 10 y 20 x 5) mostrando mejores resultados al minimizar el tiempo total de ciclo.

Mesghouni, Hammadi y Borne [57] proponen dos esquemas de desarrollar dos tipos de codificación genética juntos con sus respectivos operadores genéticos para la representación de trabajos y máquinas en paralelo al atacar la asignación de cada operación a una máquina y la secuenciación apropiada de las mismas de forma tal que se minimice el tiempo total de operación. Lo anterior debido a que tal acercamiento en la codificación da al usuario la información necesaria sobre la secuencia y permite tratar al mismo tiempo el problema de asignación de trabajos y la definición de un óptimo para el problema de secuenciación de trabajos con ruta variada.

Para el primer acercamiento donde tratan trabajos en paralelo al generar la población inicial utilizan los resultados de la Programación Lógica de Restricciones en conjunto con reglas de prioridad que contemplan; menor tiempo de procesamiento, mayor duración de procesamiento y operación que permita balancear la carga de la máquina. Por otra parte en cuanto al operador de cruce elige al azar dos padres y una máquina al azar lo que permite agregar intercambiar las operaciones que ambos padres comparten y que se aplican en la máquina seleccionada para generar una nueva secuencia (respetando reglas de precedencia y restricciones de disponibilidad de recursos) mientras que para el operador de mutación se tienen dos variantes, la primer contempla simplemente elegir al azar un cromosoma y una operación para ver si puede ser reasignada mientras que la segunda selecciona dos máquinas y una posición al azar para intentar hacer un cambio de operaciones siempre y cuando sea factible.

Para el segundo acercamiento se propone la “*codificación de trabajos en paralelo*” que considera desde un inicio las restricciones de precedencia al codificar que facilita la generación de la población inicial y los operadores genéticos son muy sencillos y generan una secuencia factible siempre al copiar información de precedencia en los nuevos hijos y cabe mencionar que el operador de mutación toma en cuenta a carga de trabajo que tienen las máquinas y con base en tal información realiza los cambios de operaciones para disminuir la carga en la máquina que presente la mayor carga.

Elaoud, Loukil y Teghem [58] proponen un algoritmo genético con cálculo de aptitud utilizando el enfoque de Pareto que presenta una innovadora forma doble de ordenar a los individuos que toma en cuenta la información de densidad al igual que una estrategia para estimar la densidad de la población lo que evita una búsqueda intensiva y que se actualiza a la par de los cambios que va sufriendo la misma. Para definir la posición del individuo¹⁰ en la población actual se suma el número de individuos que lo dominan junto con el de sus dominadores y con este acercamiento se combate el fenómeno de sesgo generacional cuando la población tiende a evolucionar a una región determinada.

Por otro lado al generar los nichos sobre los cuales se estará llevando a cabo la búsqueda para generar la frontera de Pareto durante el proceso de evolución es difícil definir las dimensiones de dichos nichos si el tamaño de la población cambia constantemente, por lo que se propone una estrategia de estimación de densidad de la población adaptativa¹¹ que define a un tamaño muy pequeño de los nichos para aquellos puntos que se acerquen a la frontera de Pareto, es importante resaltar que el número de divisiones en cada dimensión es el mismo. Para el caso de la asignación de aptitud de Pareto el valor se basa en el rango que tenga y el valor de densidad (que hace referencia al número de soluciones con las que comparte el nicho en el que se encuentra) por lo que una solución que se encuentre en un nicho sumamente saturado será penalizado, la medida anterior previene el sesgo generacional y promueve la búsqueda de zonas nuevas en el espacio de solución.

¹⁰ Estrategia de Clasificación Doble o “*Double Ranking Strategy*” por su nombre en inglés .

¹¹ *Population size adaptive density estimation strategy* por su nombre en inglés.

Para el proceso de selección de individuos de los que se obtendrán nuevas soluciones se utiliza un esquema nuevo de selección de individuos al generar un número al azar entre 0 y 1 y al individuo cuya aptitud sumada a tal valor sea mayor que dicho valor obtenido al azar y un proceso similar se sigue para el segundo individuo, y este procedimiento se sigue hasta obtener la cantidad de nuevos individuos nuevos que se desee. Por último es importante resaltar que junto con lo antes propuesto también se maneja un grupo elitista con las mejores soluciones cuyos integrantes se integran al azar a la población general para producir nuevas soluciones de mejor calidad.

Mattfeld y Bierwith [59] atacan el problema de secuenciación *Job Shop* considerando fechas de liberación y de entrega siendo consideradas como demoras en la definición de objetivos. Manejan el proceso de búsqueda en un espacio menor derivado heurísticamente del original lo que les permite ampliar la búsqueda para reducir potenciales pérdidas y complejidad del problema lo anterior se consigue siguiendo dos direcciones; primero reduciendo el espectro a nivel máquina por medio del constructor de secuencias, segundo al descomponer el problema a nivel de piso gracias a un acercamiento Multi-etapa. Permitiendo inactividad en las máquinas y así obtener secuencias activas o no permitiendo tal y obtener secuencias sin demoras se encontró que si se considera la inactividad en las secuencias se requiere de un mayor tiempo computacional al ampliar el espacio de solución siendo lo contrario para las secuencias sin demora.

Para el segundo acercamiento se define un grado de descomposición que se basa en el ordenamiento de los trabajos basados en su tiempo de liberación y se va trabajando con cada nivel que incluye a sus precedentes lo que también divide el espacio de solución con base en el número de sub problemas o clases que permite resolverlos de manera efectiva sin embargo este acercamiento mostró que mientras mayor sea la división (superando 6 grupos) la calidad de las soluciones se ve afectada considerablemente.

Zhou, Feng y Lang [60] incorporan reglas heurísticas de secuenciación como “tiempo de procesamiento más corto” y “primer orden a la actividad con mayor trabajo restante” al proceso evolutivo del AG además de una estrategia de búsqueda local de vecindad todo lo anterior con el fin de minimizar el tiempo total de ciclo en un problema tradicional de secuenciación de trabajos. La naturaleza de un algoritmo genético le da gran robustez dado que no necesita poseer conocimiento específico del problema a resolver y lo único que se necesita es la función bajo la cual se evaluará el desempeño de las soluciones con las que se estará trabajando. Sin embargo en ocasiones el no contar con mayor detalle y conocimiento sobre el problema a resolver sacrifica eficiencia por robustez. Para conseguir que el algoritmo sea más eficiente se hizo uso de las heurísticas antes mencionadas para restringir el espacio de solución y guiar el proceso de búsqueda.

El desempeño de la propuesta fue comparada contra estrategias comunes para resolver este problema tales como el recocido simulado y la técnica de búsqueda por vecindad demostrando un menor tiempo de convergencia a una solución satisfactoria en comparación con las otras estrategia en los problemas de 10 x 10, 15 x 15 y 20 x 20.

Osman, Abo-Sinna y Mousa [61] proponen un algoritmo genético para resolver el problema multiobjetivo de asignación de recursos a diferencia del acercamiento tradicional (Programación Dinámica) que descompone un problema complejo en una serie de pequeños problemas simples para darle solución, sin embargo la desventaja que se presenta al seguir este procedimiento es que es que en ocasiones para resolver el número final de problemas resultantes de dicha descomposición toma aún más tiempo que si se diese solución al problema original. El modelo que siguen los autores para darle solución a este tipo de problema es definiendo un modelo de red donde los suministros (trabajos) se representen como una etapas y los recursos (trabajadores asignados a los trabajos) serán representados por los estados de cada etapa.

El algoritmo entonces lo que busca es la distancia más corta con un mínimo de costo, mínimo de tiempo y máxima calidad, manejando una estrategia de pena de muerte para todos los individuos no factibles, para la evaluación se obedece el principio de óptimo de Pareto mientras que para la clasificación se obedece el principio de no dominio, por su parte la selección se deriva de una asignación aleatoria de pesos para los objetivos en una suma agregada y para mantener las mejores soluciones un principio de elitismo se aplica para construir la frontera de Pareto. La comparación de desempeño se basa en resolver el problema de asignar seis trabajadores a cuatro trabajos distintos basándose en una función de costo y eficiencia llegando a concluir la ventaja de un AG sobre la Programación Dinámica con respecto al tiempo y calidad de las soluciones a pesar de la complejidad o cantidad de variables que se utilicen en la formulación del problema

Por su parte Cheung y Zhou [62] desarrollaron un modelo híbrido de un Algoritmo Genético combinado con reglas heurísticas con el fin de dar solución a una generalización de los problemas de secuenciación de trabajos con preparativos dependientes de la secuencia. Las reglas heurísticas se enfocan principalmente en los tiempos de liberación y secuencia de los trabajos contribuyendo a reducir el número de tiempos muertos y contribuir en la generación de secuencias factibles, dichas reglas son probadas en el simulador incluido dentro de esta propuesta y se generan a su vez por el AG las cuales son codificadas utilizando el esquema de permutación de trabajos de forma tal que se pueda codificar inherente al cromosoma la información necesaria para generar la secuencia.

Zhao y Wu [63] se enfocan en el problema de secuenciación de trabajos dentro de un escenario más acorde a lo que sucede actualmente en varias empresas que se basan en la automatización de sus procesos. En este caso se contemplan rutas flexibles y máquinas flexibles (capaces de elaborar/ensamblar más de un tipo de producto) por tal motivo los tiempos de procesamiento pueden variar y no depender directamente de una máquina.

Partiendo de lo anterior la secuencia es dinámica en todo momento partiendo de las capacidades de las máquinas disponibles se pueden generar escenarios de máquinas en paralelo y la complejidad del problema se ve directamente afectada dada la variedad y cantidad de rutas que puede seguir una pieza/producto. Se recurre a la permutación de trabajos como estrategia de codificación del cromosoma, se ocupa el cruce ordenado lineal y la mutación tradicional.

Dichos operadores se aplicarán para las máquinas y las secuencias de procesamiento. Cabe resaltar la aportación en el planteamiento del problema al incluir un tipo de relajación en las restricciones de algunas piezas a procesar además de poder llevar a cabo la evolución de la selección de; rutas, máquinas y secuencias de operaciones al mismo tiempo con el AG.

Ruiz y Maroto [64] partiendo del análisis de la industria textil y manufacturera de pisos en España generan una propuesta de solución al problema híbrido de secuenciación de flujo continuo con tiempos de preparación dependientes y conflictos de elegibilidad de máquinas con el planteamiento de un algoritmo genético. Se parte del hecho que el calculo del tiempo total de ciclo está implícito en la función objetivo que contempla de manera simultanea la secuencia y la asignación de los trabajos, la selección se lleva a cabo combinando el ordenamiento por ranking y el torneo estocástico binario para el caso del cruce se implementó el denominado “*Cruce de Trabajos con Orden Similar*”.

Lo anterior evita el rompimiento de bloques con un valor elevado de aptitud. Para controlar el proceso de evolución e evitar que se estanke el mismo o se quede en un mínimo local se establece un contador si el valor mínimo del tiempo de ciclo no cambia de una generación a otra remplazando el 80% de los peores individuos de la población con nuevos individuos con mejores cromosomas y a su vez material genético. Cabe resaltar el hecho de haber aplicado la solución a problemas reales obteniendo un 9% de mejoría en las secuencias obtenidas contra resultados obtenidos por métodos tradicionales.

Torabi, Ghomi y Karimi [65] por su parte se enfocan en el problema de secuenciación de lotes y entregas con el objetivo de minimizar el promedio de espera, preparación y costos de transporte por unidad de tiempo en la cadena de suministro. Debido a la complejidad del problema para planteamientos de mediana y gran escala se optó por el desarrollo de un Algoritmo Genético Híbrido que implementa una técnica de intensificación como la “*búsqueda de vecindad*” basados en la idea de una búsqueda inteligente al azar para evitar caer en un óptimo local debido a no linealidad del problema. El problema se enfoca directamente a la asignación de máquinas en un esquema de múltiples máquinas en paralelo y la secuenciación de trabajos para cada etapa como sub-problemas que conforman la parte combinatoria del problema raíz mientras que el tamaño de lote y los problemas de secuenciación definen la parte continua del problema.

Cabe resaltar el hecho de que variables como la razón de la demanda, tamaño del lote y horizonte de planeación además del ciclo de producción son determinísticas y no cambian todo con el fin de construir un modelo estable. Se optó por utilizar para la codificación de los cromosomas; vectores de permutación de trabajos que luego se tradujeron en soluciones completas y discretas del problema. La propuesta antes expuesta demostró una superioridad considerable contra métodos de enumeración en términos de calidad y variedad de soluciones obtenidas no solo para problemas de mediana escala sino también para aquellos de gran escala.

Erol Pinto [66] por su parte adaptó la propuesta de Deb con su algoritmo NSGAII[43] a una cadena de suministro modelada en tres etapas (llegada del producto, ensamblaje-producción y entrega) para producir solo un producto compuesto por tres componentes que necesita de cinco proveedores distintos y da servicio a cuatro clientes.

El planteamiento de un algoritmo multiobjetivo es necesario para poder dar solución a una situación lo más real posible y más aún para el caso de una cadena de suministro en la cual siempre se debe satisfacer más de un objetivo. Por tal motivo se definieron varios conjuntos de objetivos a resolver para este problema dentro de los cuales se pueden mencionar; la minimización de costo total de operación y la razón entre los costos de manufactura y los costos totales de operación así como maximizar las ganancias y minimizar los costos de manufactura. En los cuatro sets de pares de funciones objetivo el algoritmo demostró construir fronteras de Pareto óptimas de forma tal que el tomador de decisión pudiera contar con información precisa sin preocuparse si tales eran óptimas o no pues todos los resultados obtenidos cumplían con tal requisito.

Es importante señalar que todo el análisis se aplicó a un solo producto y que en un planteamiento que comprenda más productos no es necesario afectar las ecuaciones ni los planteamientos matemáticos dado que un AG no trata el problema sino que se ocupa de la población y los operadores genéticos que lo rigen.

Un grupo de investigadores de la universidad de Carolina del Norte de los departamentos de Ingeniería Industrial e Ingeniería Textil [67] desarrollaron una solución robusta que se enfoca en las decisiones que una empresa debe tomar y que tienen que ver con el manejo de inventarios, cantidad de materia prima o productos a solicitar y que afectan directamente la cadena de suministro de una empresa y por consiguiente en la satisfacción de los pedidos de los clientes. A tal situación un algoritmo genético multiobjetivo junto con un simulador es desarrollado de forma con el fin de generar una frontera de Pareto óptima. Se busca maximizar el Margen del Retorno Sobre la Inversión con varios niveles de servicio (del 93% al 94%).

Se utilizó una combinación de cuatro distintos operadores de mutación (de frontera, uniforme, no-uniforme y Multi-uniforme) mientras que para el cruce se usaron tres tipos distintos (simple, aritmética y heurística). El proceso de selección de los individuos se lleva a cabo de acuerdo al desempeño que presente el simulador que también forma parte del método de solución del problema. Los resultados que se obtuvieron demostraron que se puede llegar a doblar el retorno sobre la inversión con hasta un 90% del servicio con errores en el volumen que iban desde un -20% hasta un 20% sobre la demanda real.

En el trabajo de Gen y Syarif [68] para dar solución al problema de la planeación de la producción y distribución en múltiples periodos de una empresa manufacturera desarrollaron un AG Híbrido que trabaja en conjunto con la técnica de expansión basada en un árbol y un controlador lógico difuso, éste último para afinar los parámetros del AG. La codificación de las soluciones se hace siguiendo lo propuesto por Prüfer en la que se representa una gráfica de la red que conecta J plantas con M clientes.

Los operadores de genéticos (mutación y cruce) tuvieron que ser adaptados a la codificación para obtener individuos factibles, para el caso del cruce se fija un punto a lo largo de los cromosomas de los padres y el contenido ubicado a la derecha de ambos se intercambia y se verifica la factibilidad de la nueva solución ahora bien para la mutación se utiliza el método de inversión partiendo de un punto fijo en el cromosoma e invirtiendo el resto del mismo.

Para afinar el comportamiento del AG en los parámetros de cruce y mutación se implementa el uso de un controlador lógico difuso que se basa en el desempeño de aptitud de la población a lo largo del tiempo, si la diferencia se encuentra dentro de unos rangos definidos previamente se toman 3 acciones (incrementar, reducir o mantener) los valores de ambos operadores en cuyo caso se aplican para la siguiente generación. Llegando a obtener con todo el anterior proceso mejores resultados que con el proceso normal de enumeración total o de construcción de árboles.

CAPÍTULO III

3.1 DESCRIPCIÓN DEL PROBLEMA

El proceso de secuenciación es un conjunto de decisiones que se llevan a cabo de manera natural y rutinaria por tomadores de decisiones (jefes de producción, jefes de planta, etc.) en muchas empresas ya sea de servicios o manufacturera. Lo que se busca es asignar recursos a una serie de tareas en períodos de tiempo definidos para satisfacer uno o más objetivos). Tales objetivos en la gran mayoría de las veces son mutuamente excluyentes o comparten varios recursos críticos entre sí por lo que es complicado cumplir con cada uno de ellos en su totalidad lo que hace este proceso complejo y difícil de resolver de una manera sencilla.,

Partiendo de lo anterior se identifica claramente la secuenciación como un proceso clave y vital en una empresa manufacturera, el asignar los recursos disponibles (máquinas, trabajadores, materia prima) a procesos clave de forma tal que se cumplan una serie de restricciones con relación a su uso y aprovechamiento. Las primeras estrategias para resolver estos problemas obedecían métodos matemáticos que intentaban enumerar o encontrar todas las posibles soluciones del problema y las evaluaban una a la vez al igual que la aplicación de reglas heurísticas que definirán las prioridades de los trabajos para poder elaborar la secuencia; el Menor Tiempo de Procesamiento (SPT¹²) o Primeras Entradas Primeras Salidas (FIFO¹³), Trabajo Total (TWK¹⁴), Tiempo de Holgura, etc. por mencionar algunas [69][70] .Sin embargo tales reglas son utilizadas en la mayoría de los casos para generar una primera propuesta de secuencia partiendo de su desempeño en el peor de los escenarios siendo después validadas y mejoradas por otras técnicas de optimización y refinamiento. De igual forma se pueden encontrar acercamientos de solución como el método de restricción ε , ponderación de objetivos, programación basada en metas, etc. Ahora bien como se comentó al inicio de este capítulo el acercamiento tradicional de enumeración total ha sido el más utilizado en muchas ocasiones, sin embargo si se recurriera a ésta técnica para resolver un problema de secuenciación siendo n el número de trabajos a ordenar y m las máquinas disponibles se necesitarían un total de $(n!)^m$ evaluaciones para abarcar el espacio total de solución aunque los valores de n y m fueran muy pequeños (10 por ejemplo).

El problema se complica aún más cuando se incorpora una política de calidad como lo es Justo a Tiempo¹⁵ que se relaciona directamente con reglas que definen el manejo de inventarios en una empresa y que tiene como principal objetivo mantener los mismos al mínimo y solo mantener lo estrictamente necesario.

¹² Shortest Processing Time, por sus siglas en inglés.

¹³ First In, First Out por sus siglas en inglés.

¹⁴ TotalWork por sus siglas en inglés.

¹⁵ Just in Time por sus siglas en inglés.

De igual forma se busca el balancear la carga de trabajo en los equipos a lo largo de la línea de forma tal que se eviten cuellos de botella y por tal motivo se retrase la producción, ahora bien si aunada a tal estrategia de producción la empresa rige sus operaciones bajo un esquema de producción “fabricar a la orden” para satisfacer los requerimientos y pedidos de los clientes.

Debido a lo anterior las exigencias y requerimientos de planeación aumentan considerablemente, sin embargo dicha planeación debe tener la posibilidad de ajustarse fácilmente y adaptarse a cambios drásticos o discretos que pudieran darse en la producción, basados en el hecho que las empresas de hoy en día ya no producen solamente un tipo de producto y las capacidades de los equipos con los que cuentan el dedicar una máquina a un solo tipo de producto sería un desperdicio es necesario contemplar los tiempos de preparación que se necesitan para ajustar y preparar la maquina con el fin de producir un tipo o variante de producto que tiene un impacto considerable e influye directamente en la planeación de los trabajos dando como resultado un incremento drástico en el tiempo total de ciclo o el tiempo total de producción.

3.2 MODELACIÓN MATEMÁTICA DEL PROBLEMA

Ahora bien si se parte de un esquema de producción de fabricar a la orden aunado junto con un esquema JIT las exigencias de la secuencia se incrementan considerablemente al ser necesario que no existan retrasos de ninguna índole en la entrega del producto terminado a la par de poder mantener el ritmo del consumo de la materia prima y materiales lo más estable posible a pesar de la combinación y variabilidad de productos que se pueda presentar considerando que no se debe tener un exceso de producto en proceso y mantener los márgenes de inventario (total y entre estaciones) al mínimo, aunado a lo anterior se busca de igual manera que la secuencia propuesta contemple la menor cantidad de ajustes y preparaciones en la máquina de forma tal que no se pierda tiempo y dinero en dichas operaciones. Para evaluar y definir la estabilidad en el uso de los materiales Miltenburg [71] definió una métrica para cuantificar ésta razón de uso (propuesta por Monden [72] y se define como sigue:

$$U = \sum_{k=1}^{D_T} \sum_{i=1}^a \left(X_{i,k} - k \frac{d_i}{D_T} \right)^2 \quad (\text{Ec. 1})$$

Donde U es la razón de uso de materia prima de una secuencia de producción, a es el número total de productos únicos o distintos a producir, D_T es la cantidad total de productos a producir definida también como Demanda Total, d_i es la demanda que se tiene por cada producto i mientras que $X_{i,k}$ es el total de productos necesarios i y que es producido en las etapas $1-k$ donde $k = 1, 2 \dots, D_T$.

La razón de uso es una forma que tienen las empresas para medir la capacidad que tiene su sistema de producción de forma tal que pueda mantener un nivel estable en su producción para cumplir con su demanda o también puede mostrar la capacidad que tiene para poder manejar un flujo estable de materia prima para los distintos productos que maneje la misma siempre asegurándose de que lleguen al sistema a una razón medianamente uniforme.

Cada que dos elementos consecutivos en la línea de producción es distinto se necesita un tiempo de preparación y ajustes en la máquina de forma tal que se pueda elaborar dicho producto distinto, y es necesario contabilizar la cantidad de veces que se necesita llevar a cabo dicha operación de preparación siendo definida por la siguiente expresión matemática:

$$S = 1 + \sum_{k=2}^{D_T} S_k$$

(Ec. 2)

Siendo S el número de ajustes necesarios para cada máquina en la secuencia de producción y S_k puede tomar un valor de 1 si se necesita dicho ajuste al detectarse una variación en la producción o 0 en caso de que no exista algún cambio en la secuencia. Sin embargo el contar con una variación mínima se puede alcanzar con un elevado número de ajustes en las máquinas pero que con los tales se puede incurrir en mayores costos o que no sea factible la secuencia de producción si se incluyen otras variables dentro de la formulación como son los cambios en la demanda o el mercado al igual que los costos que cada cambio genera a la empresa.

Como se mencionó anteriormente si se intentase dar solución a éste problema de secuenciación siguiendo la estrategia clásica de enumeración total nos encontraríamos con la naturaleza combinatoria del problema, llegando a un total de posibles secuencias con base en la siguiente ecuación:

$$\text{Cantidad de Soluciones} = \left(\sum_{i=1}^a d_i \right) / \left(\sum_{i=1}^a d_i \right)$$

(Ec. 3)

Tomando como base un ejemplo donde se cuente con 5 productos distintos (A, B, C, D, E) y la demanda que se tiene de cada producto es la siguiente: $d_A = 6, d_B = 3, d_C = 1, d_D = 1, d_E = 1$, dando como resultado una demanda total de 12 productos. Ahora bien si se analizan dos posibles secuencias que cumplan con la mezcla de productos y la demanda total como: $X_1 = BBBCAAAAAAED$ y $X_2 = EAAAAAACBBBD$ el número de ajustes necesarios en las máquinas (S) para ambas secuencias es de 5, sin embargo la razón de uso de la materia prima (U) para la secuencia X_1 es de 40.83 y un valor de 44.33 para la secuencia X_2 .

Si bien se consideran otras secuencias como: $X_3 = ABACADEABABA$ y $X_4 = AEABACABDABA$ podemos notar que para ambas secuencias X_3 y X_4 se necesitan 12 ajustes en las máquinas (S) mientras que las razones de uso de la materia prima (U) son de 7.67 y 8.83 respectivamente, si analizamos el desempeño de las secuencias antes analizadas bajo el concepto de *dominio* o *Pareto óptimo* se puede observar que es mejor la secuencia X_1 sobre X_2 o bien se puede decir que X_1 domina a X_2 -($X_1 > X_2$) ahora al igual que en el caso anterior se puede notar que X_3 domina a X_4 ($X_3 > X_4$) sin embargo a pesar de que X_3 tiene un valor menor concerniente al uso de la materia prima comparado con X_1 no se puede concluir que X_3 domina a X_1 dado que la última necesita un menor número de ajustes en las máquinas para cumplir con la demanda total.

Con un grupo de 4 productos únicos y una demanda total de 7 productos se obtiene un total de 420 secuencias posibles mientras que si la demanda se llegase a doblar se llegaría a un total de 1,261,260 posibles secuencias distintas. Es evidente con base en las cifras anteriores que una técnica clásica de solución tomaría demasiado tiempo para poder analizar y sintetizar el total de soluciones que se tienen para un problema de solo 4 productos sin embargo en una situación real es más que evidente que una empresa no maneja solo esa pequeña cantidad de productos sino que la cartera de productos sobrepasa al menos un mínimo de 10 o 20 productos (aunque las variaciones sean muy pequeñas entre sí, y que nos lleva a un total aproximado de 168,168,000 soluciones para 5 productos con una demanda total de 15 unidades).

Por lo anterior es evidente la necesidad de una solución computacional que brinde la posibilidad al tomador de decisiones de elegir de entre una amplia gama de distintos escenarios basados en la frontera de Pareto que se construye contenido todas las posibles soluciones óptimas que cumplan con los objetivos antes expuestos bajo el principio de que tal frontera se compone de soluciones que no pueden ser mejores sin que se afecte de manera considerable o directamente el desempeño de las otras variables.

El problema descrito presenta gran importancia para ser resuelto desde una perspectiva de la empresa sin embargo como se ha notado presenta un gran reto en términos computacionales por un exceso de recursos computacionales para evaluar y calcular todas las posibles soluciones en un problema con al menos 15 unidades y 5 productos por lo que cumple las características para pertenecer a la clase de problemas de tipo NP Duro, Jackson demostró la complejidad computacional que representa el generar una secuencia para 2 máquinas con al menos 2 operaciones, más tarde Papadimitrou, Goldberg y Lance [19] [7][20] establecieron que para los problemas de tipo combinatorio solo existen aproximaciones heurísticas, por su parte Varela resaltó el hecho de que la mayoría de los problemas encontrados en la industria pertenecen a la categoría de NP Duro y abarcan problemas de producción hasta secuenciación de trabajos[22].

Es por antes descrito que el propósito del presente trabajo de investigación es el desarrollar una herramienta de optimización para generar secuencias de producción en una línea de ensamblaje bajo el esquema de producción de justo a tiempo por medio de un algoritmo genético multiobjetivo cuyos objetivos de optimización sean el minimizar simultáneamente la cantidad de cambios en la secuencia y la razón de uso de las secuencias de producción y al mismo tiempo el modelo se apegue en lo más posible a un proceso natural de evolución que parte de un proceso de mutación y cruce autónomo.

Para conseguir tal objetivo se desarrolló un módulo de mutación evolutivo, que tiene como objetivo el simular un proceso auto adaptativo del proceso de mutación en una población determinada durante todo el proceso evolutivo contrario a los acercamientos clásicos directamente relacionados con los AG al definir *a priori* la razón de mutación bajo la cual se regirá todo el proceso evolutivo del algoritmo o en otras ocasiones basando el ajuste en varios ensayos y el ajuste será basado en prueba y error.

Los valores que tomarán estos parámetros estarán basados en un ajuste continuo con base en los cambios del horizonte a lo largo de las generaciones que presenten las soluciones en términos de su aptitud. De igual forma se hace un comparativo del desempeño de la propuesta contra las heurísticas más comúnmente utilizadas para éste tipo de problemas y que presentan un buen desempeño como un AG multiobjetivo con sus razones de cruce y mutación estacionarios y un AG multiobjetivo con búsqueda local como técnica de refinamiento con el fin de analizar el comportamiento de tales soluciones en un ambiente dinámico y constantemente en cambio.

3.3 VARIABLES Y FUNCIONES OBJETIVO

Se han definido las siguientes variables para las funciones objetivo que serán resueltas con base en la descripción del problema:

Variable	Descripción
U	Razones de uso de una secuencia de producción.
S	Número de preparaciones de máquinas en una secuencia de producción.
a	Número de productos únicos a producir.
D_T	Demandta total de unidades de todos los productos.
d_i	Demandta del producto i , $s = 1, 2, \dots, a$
S_k	Toma el valor de 1 si se requiere de una preparación en la máquina en la etapa k , 0 si no es el caso.
$x_{i,k}$	Número total de unidades del producto i producido en las etapas 1 a k , $k = 1, 2, \dots, D_T$

Tabla 2: Variables utilizadas en el problema

Con base en las variables descritas la siguiente formulación matemática define a las funciones objetivo del problema de secuenciación de trabajos en una línea de ensamble con un sistema de producción justo a tiempo y que será resuelto con un algoritmo genético multiobjetivo con módulo autoadaptable de mutación:

$$\text{Minimizar } \{f_1(X), f_2(X)\}$$

$$f_1(X) = S = 1 + \sum_{k=2}^{D_T} S_k,$$

$$f_2(X) = U = \sum_{k=1}^{D_T} \sum_{i=1}^a \left(x_{i,k} - k \times \frac{d_i}{D_T} \right)^2$$

(Ec. 4)

Siendo X el vector de las variables de decisión que en este caso hace referencia a los distintos tipos de productos que se realizarán en la línea. Buscando como resultado el valor mínimo de la razón de uso y su respectivo valor relacionado a la cantidad de preparaciones de máquina, obteniendo siempre una secuencia de producción que cumpla con el vector de demanda y la cantidad de productos únicos a producirse.

CAPÍTULO IV

4.1 DESCRIPCIÓN DEL MODELO

Dentro las estrategias que se han utilizado para dar solución al problema de secuenciación de operaciones en una línea de ensamblaje bajo un esquema de producción de Justo a Tiempo se pueden encontrar comúnmente las estrategias de solución mediante Recocido Simulado [73] y Búsqueda Tabú [74]. Sin embargo algunos de los acercamientos antes mencionados son estrategias para resolver problemas con un solo objetivo (minimizar el tiempo total de ciclo por ejemplo C_{max}) en cambio los AG multiobjetivo cuentan con una capacidad inherente para explorar y explotar el espacio de solución de cualquier problema a resolver de una manera eficiente [38]. El explorar el espacio de solución es posible sin importar si es el mismo es continuo, discontinuo o discreto de forma tal que pueda mantener una gran diversidad de individuos en la población mientras que a la par debe de ser capaz de explotar se refiere a la capacidad del algoritmo para encontrar soluciones óptimas y que a su vez cumplan con la definición de óptimo de Pareto y que al cumplir con el criterio de aptitud permanezcan a lo largo de las generaciones teniendo grandes posibilidades de sobrevivir el proceso de selección que se lleva a cabo durante las distintas generaciones.

En el capítulo anterior se definieron las variables y los objetivos que se desean alcanzar sin embargo al momento de construir un algoritmo genético es necesaria una representación adecuada que no retrase o inclusive prohíba el encontrar las soluciones óptimas con la precisión deseada [38] lo que puede llevar a una convergencia prematura a un óptimo local. Existen dos clases de representaciones comúnmente utilizadas para representar los datos e información de este problema:

- a) Representación Entera
- b) Representación Binaria

La primera de éstas es la más implementada al resolver este problema [75][76][77] pues brinda al tomador de decisiones un panorama claro en cada generación de la secuencia creada al desplegar en la estructura del cromosoma la lista de trabajos y con base en su orden de aparición se define la secuencia de producción o su asignación en cada máquina.

1	1	2	2	4	1	3	4	4	4
---	---	---	---	---	---	---	---	---	---

4	1	4	3	1	1	3	4	4	2
---	---	---	---	---	---	---	---	---	---

Figura 4 Cromosoma que utiliza una representación entera

Cabe resaltar que el hecho de utilizar dicha representación requiere de una serie de ajustes y adecuaciones de los operadores genéticos dado que se emplean números enteros y por tal motivo es necesario se adapten a la estructura del cromosoma.

Para el segundo caso, la representación de los elementos en la población (resultados potenciales) hace uso de elementos binarios aunque no necesariamente se utiliza solo para representar números como tal sino que se pueden utilizar algunos bits para definir si el número es negativo o positivo o también para incluir varios elementos dentro de una representación vectorial y se defina un par o número determinado de bits solo para separar las cadenas. Dicha representación trabaja con los operadores genéticos que intervienen de manera directa con la estructura de las soluciones como el cruce y la mutación siguiendo una implementación de manera transparente y sin necesidad de adecuaciones o adaptaciones como en el caso anterior.

4.2 PARTICULARIDADES DEL MODELO

Se ha definido para la presente investigación un total de cinco productos únicos con una demanda que puede ser de hasta 12, 15 y 20 unidades llegando a representar hasta 3.055×10^{11} secuencias posibles considerando cada uno de los tres casos de demanda. Con base en la codificación binaria elegida para el problema a resolver, cada producto que será elaborado y forma parte de una secuencia de producción se compone de 3 bits que a su vez representará el número entero del producto que será elaborado y el orden de los mismos va de izquierda a derecha, por ejemplo cuando se tienen 5 productos únicos (A, B, C, D y E) con una demanda para cada producto de 6 unidades del producto A, 3 del producto B, 3 del producto C, 2 del producto D y 1 del producto E se necesita de una cadena de 45 bits en total para representar una posible secuencia de producción, la Figura 5 muestra un ejemplo de un cromosoma que contiene una secuencia con los requerimientos antes descritos.

0	1	1	0	1	0	0	1	1	0	1	0	0	1	1	0	1	0	0	1	1	0	1	0	0	1	0	0	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Figura 5 Ejemplo de un cromosoma que contiene 5 trabajos para una demanda total de 15 productos

3	2	3	2	3	2	4	1	4	1	1	5	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Figura 6 Representación de una secuencia de producción en números enteros

La secuencia de producción antes descrita por la cadena de 45 bits se traduce en su representación de números enteros como se muestra en la Figura 6 definiendo el orden de producción de izquierda a derecha y de acuerdo a la demanda fija para cada producto se tiene una razón de uso de materia prima de 38.76 con un total de 12 ajustes de preparación para cumplir con la demanda definida para cada producto.

Para la presente investigación el modelo de solución se basa en la biblioteca de herramientas desarrolladas en el lenguaje de programación de C++ para resolver algoritmos genéticos uniobjetivo y multiobjetivo desarrollada por Sastry [78] para Matlab y utiliza la representación binaria de la información dada la facilidad que presenta en el tratamiento de los resultados al aplicarles distintos operadores genéticos y por tal motivo no se requiera de evaluaciones innecesarias para estar traduciendo los elementos de un sistema de codificación a otro solo para que se le apliquen determinados operadores genéticos.

4.3 OPERADORES GENÉTICOS

Los operadores genéticos que se utilizaron en la actual propuesta son los siguientes:

- a) Selección
 - a. Selección por Torneo con reemplazo [79]
 - b. Selección por Torneo sin reemplazo [79]
 - c. Selección por Ruleta [7]
 - d. Selección estocástica universal [7]

- b) Cruce
 - a. De un punto [80]
 - b. De dos puntos [80]
 - c. Uniforme [80]
 - d. Binario simulado (SBX) [81]

- c) Mutación
 - a. Polinomial [15]
 - b. Selectiva
 - c. Gaussiana
 - d. Autoadaptable*

Cada uno de los módulos antes mencionados opera bajo la estructura del algoritmo NSGA¹⁶ II propuesto por Deb [43] pero con las modificaciones y adecuaciones pertinentes realizadas por Sastry para que el algoritmo sea Multiobjetivo de igual forma se implementó una estrategia de Búsqueda Local *Nelder - Mead* [82] para refinar los resultados obtenidos al igual que un reemplazo elitista.

¹⁶ Non Sorting Genetic Algorithm II por sus siglas en inglés

Por otro lado el criterio de paro no se define estrictamente por el número de generaciones como tradicionalmente se define, para la presente propuesta se pueden elegir cinco condiciones de paro:

- 1) Número de evaluaciones de funciones
- 2) Variación de la aptitud
- 3) Aptitud Promedio
- 4) Objetivo Promedio
- 5) Cambio en la mejor aptitud

Para evaluar el desempeño de cada una de las secuencias de producción obtenidas en cada generación por el algoritmo desarrollado se definieron criterios que delimiten de manera clara el desempeño basado en el valor obtenido de dos ecuaciones que evalúan: (a) la razón de uso de la materia prima (Ec.1) y (b) la cantidad de ajustes necesarios para cada máquina (Ec2.) buscando que se minimicen a la par ambos criterios siempre y cuando se cumpla con las metas de producción previamente establecidas.

Con respecto a los operadores genéticos se desarrolló un operador de mutación autoadaptable que tiene como principal objetivo el regular el proceso de mutación de los individuos de manera adaptativa simulando un mecanismo natural sin que se deba definir un valor iniciar de manera arbitraria al inicio de la ejecución y modificar el mismo.

4.3.1 OPERADOR AUTOADAPTABLE DE MUTACIÓN

El operador autoadaptable que establece el valor de la razón de mutación, obedece al cambio en el desempeño o aptitud que presentan los individuos de una población durante una generación. Para detectar el comportamiento en la aptitud se realiza una igualación del valor actual con un histórico, para dicho análisis se consideran generaciones anteriores y la generación actual. Si existe un cambio considerable en términos de su desempeño, el valor de la razón de mutación se afecta de manera proporcional a dicho ajuste de forma tal que si el algoritmo está llegando a un mínimo o máximo local al cambiar el valor de la mutación se pueda dar un pequeño salto y así salir de dicho punto o si se ha llegado a un punto donde no se puede mejorar más la aptitud de un individuo el algoritmo no realice un cambio abrupto y tome más tiempo el obtener una solución.

Para conseguir lo anterior se ha definido un método que tiene como fin el analizar el comportamiento general de la población durante el proceso evolutivo y con base en el desempeño de los individuos en términos de su aptitud el valor de la razón de mutación se modificará y ajustará de manera automática y adaptativa. En el archivo de texto donde se registran los resultados de cada objetivo por generación se compara la media de las diez generaciones anteriores con la existente, y con base en el promedio de las diferencias entre ambas cantidades se afecta de manera proporcional el valor de la razón de mutación para la siguiente generación partiendo del valor que tenía anteriormente.

Se ha definido una consideración para las primeras diez generaciones de forma tal que se ajustó el criterio de comparación partiendo de un acumulado de 5 generaciones para que no se realicen demasiadas evaluaciones si se define un número menor al establecido y por lo tanto el tiempo que se requiera para resolver el problema sea mayor.

4.4 ESQUEMA GENERAL DE OPERACIÓN DEL MODELO

De manera general se podría describir lo antes expuesto en el algoritmo que se presenta a continuación:

1. Inicia la aplicación
2. Se leen los parámetros de operación obtenidos del archivo de configuración
 - a. Se define el tipo de AG, (uniobjetivo o multiobjetivo)
 - b. Se define el numero de variables de decisión
 - c. Se define el número de productos distintos a producir
 - d. Se define la cantidad de elementos requeridos de cada producto único
 - e. Se define el tipo de dato para representar la variable (entera o flotante) y los límites mínimos y máximos para cada una de las variables.
 - f. Se declara el número de objetivos y la meta a seguir (minimizar o maximizar)
 - g. Se especifica el tamaño de la población
 - h. Se define el criterio de selección
 - i. Se establece el criterio de cruce
 - j. Se define la razón de cruce
 - k. Se establece el criterio de cruce
 - l. Se define la razón de cruce
 - m. Se revisa si se realizará una búsqueda local y bajo qué criterios
 - n. Se verifica el criterio de paro del algoritmo
3. Se genera la población inicial
4. Se evalúa la aptitud de los individuos para cada uno de los objetivos
5. Se aplican los operadores genéticos a la población
6. Se registran los resultados en el archivo de texto
 - a. Se lleva a cabo la reevaluación de las generaciones
 - b. Si es necesario un ajuste se registra el valor en el archivo de configuración.
7. Se revisa si se ha cumplido con los criterios de paro
8. Se revisa si se cumplieron con los objetivos definidos para el problema
 - a. Si no se han cumplido se realiza de nuevo el paso 3
 - b. Si se ha cumplido con los objetivos definidos se termina la ejecución

El archivo de configuración en el que se definen los parámetros que se esbozan en el paso 2 puede ser modificado de manera sencilla por cualquier programa editor de texto sin embargo cabe mencionarse que se deben seguir una serie de consideraciones al respecto, en la Figura 7 se puede observar la estructura necesaria que deben seguir los argumentos de configuración.

```

#
# Tipo de AG: SGA o NSGA
# SGA para un solo objetivo
# NSGA para más de un objetivo
#
NSGA

#
# Número de Variables de decisión
#
#
#
# Número de productos únicos a elaborar
#
#
#
#
#
#Mix de products
# Basado en la cantidad total de productos únicos a producir
#   la demanda de cada producto se define como sigue:
#       [Número de Producto] [demanda]
#
#
#

```

Figura 7 Estructura del archivo de configuración para el AG

Es importante seguir una estructura ordenada dado que el programa obtiene los datos de manera ordenada, las líneas en blanco y las precedidas por el símbolo de gato (#) las salta de manera automática y empieza a leer los parámetros en el orden predefinido.

Una vez definidos los parámetros para la ejecución del programa es preciso definir las funciones objetivo que se utilizarán en el problema, las tales regirán la pertinencia y aptitud de los individuos en las poblaciones y de ésta manera se validará la presencia de cada individuo en la población. Dicho archivo es del tipo “m” de MatLab donde se definen las funciones objetivo y las variables de decisión que componen a las mismas. Para el caso del presente programa en la Figura 8 se muestra cómo se codificaron las Ecuaciones 1 y 2 que servirán de medida de aptitud para el AG propuesto.

4.5 ALGORITMOS GENÉTICOS MULTIOBJETIVO ADICIONALES

Partiendo de los distintos operadores genéticos y las condiciones de paro que ofrece la plataforma de solución se han definido dos modelos adicionales para poder evaluar el desempeño la propuesta del presente trabajo de investigación, se buscó comparar el modelo propuesto con las estrategias de solución multiobjetivo comúnmente empleadas; un AG básico y un AG que emplea una técnica de refinamiento de resultados basado en una búsqueda local y que de aquí en adelante se nombrarán como sigue:

1. MOGA Sencillo
2. MOGA con búsqueda local
3. MOGA autoadaptable

Para el primer modelo se definieron parámetros comunes de trabajo para un AG, una razón de mutación de 0.005 y una razón de cruce de 0.5[38][7][6], para el tamaño de la población se definió un tamaño de población de 100 individuos. Para el segundo modelo se tomaron los mismos parámetros de cruce y mutación del modelo anterior, sin embargo dado que además de lo anterior se ha de implementar una técnica de refinamiento basado en el método *Nelder - Mead* que se aplica con una probabilidad p/s a un individuo, la solución obtenida por la búsqueda local se maneja por dos estrategias; la estrategia propuesta por Baldwin determina que la solución hallada con una probabilidad pb remplaza al individuo pero la aptitud del mismo no se modifica, la estrategia de Lamarck establece con una probabilidad $1 - pb$ que se reemplazará tanto al individuo como su valora de aptitud.

Para el último modelo se ocuparán los mismos parámetros definidos anteriormente que están relacionados con el tamaño de la población y la razón de cruce sin embargo para la razón de mutación se implementará el método autoadaptable desarrollado en este trabajo que ajuste dicho valor a lo largo de la ejecución del algoritmo obedeciendo los cambios en la aptitud de la población para las funciones objetivo del problema.

De igual forma se definieron tres escenarios distintos para comparar el desempeño de los modelos antes descritos al resolver el problema de secuenciación de trabajos en una línea bajo dos conjuntos de nueve problemas distintos propuestos por que se tomaron como plataforma de comparación. Los tres escenarios se derivan de una variación en el total de generaciones a evaluar comprendidas en 100, 500 y 1000.

4.6 INSTANCIAS DE PROBLEMAS A RESOLVER

Los dos conjuntos de problemas (Tabla 2 y Tabla 3) contemplan dos escenarios distintos contemplando cinco productos únicos con una variación en la demanda de cada uno generando al final nueve variaciones distintas, de igual forma se muestra la cantidad total de soluciones que se pueden tener con la variación en la demanda de cada producto.

Problema	Tipo de Producto					Soluciones
	1	2	3	4	5	
B_1	8	1	1	1	1	11,880
C_1	7	2	1	1	1	47,520
D_1	6	3	1	1	1	110,880
E_1	6	2	2	1	1	166,320
F_1	5	3	2	1	1	332,640
G_1	5	2	2	2	1	498,960
H_1	4	3	2	2	1	831,600
I_1	4	4	2	1	1	415,800
J_1	3	4	2	2	2	1,663,200

Tabla 3 Conjunto de Problemas 1

Problema	Tipo de Producto					Soluciones
	1	2	3	4	5	
B_2	11	1	1	1	1	32,760
C_2	10	2	1	1	1	180,180
D_2	9	3	1	1	1	600,600
E_2	7	4	1	1	1	2,162,160
F_2	7	3	2	2	1	10,810,800
G_2	6	3	3	2	1	25,225,200
H_2	5	3	3	3	1	50,450,400
I_2	4	3	3	3	2	126,126,000
J_2	3	3	3	3	3	168,168,000

Tabla 4 Conjunto de Problemas 2

CAPÍTULO V

5.1 PRUEBAS Y RESULTADOS

En el capítulo anterior se definió el algoritmo genético multiobjetivo con un módulo de mutación autoadaptable para resolver dos conjuntos de problemas de secuenciación de trabajos en una línea de ensamble, de igual forma se definieron otras dos heurísticas para poder comparar el desempeño de la propuesta con heurísticas tradicionalmente implementadas en un problema como el antes descrito.

Para el desarrollo de las pruebas que se realizaron para solucionar el problema de secuenciación de trabajos implementando algoritmos genéticos multiobjetivos se utilizó una computadora de escritorio con las siguientes características.

- Procesador Intel Xeon a 2.4 Ghz. Con 2Gb de Memoria RAM
- Sistema Operativo Windows XP Professional Versión 2002 con Service Pack 3
- Matlab 7.1.0 (R14) Service Pack 3
- Compilador de Visual Studio .NET 2003

Las Tablas 2 y 3 del capítulo anterior muestran las 18 instancias del problema de secuenciación de trabajos propuesto por McMullen [83] y se siguió la clasificación que el autor definió para cada problema. Como se mencionó anteriormente la manera para evaluar la aptitud de un individuo dentro de la población se lleva a cabo al calcular el valor de la razón de uso de la secuencia de producción que presenta y la cantidad de operaciones de ajuste y preparación que requiere. Siguiendo la codificación que especifica Matlab ambas ecuaciones se definen como sigue:

```
function y = UR(S,d,a)
% Usage rate for x
[m,n]=size(S);
DT=sum(d);
tt=0;
for k=1:DT
    for i=1:a
        xik=0;
        t1=0;
        for l=1:k
            if (S(l)==i) xik=xik+1; end
        end
        t1=(xik-k*d(i)/DT)^0.5;
        tt=tt+t1;
    end
end
y=tt;
```

Figura 8 Codificación de las funciones objetivo en Matlab

Para la realización de las pruebas con las instancias antes mencionadas se comparó el resultado que se obtuvo de la propuesta y las dos heurísticas adicionales a lo largo de un total de 100 y 500 generaciones en total.

La tabla 4 y tabla 5 muestran los resultados obtenidos para el conjunto de problemas 1, se decidió agregar un 100 o 500 al lado de cada letra para diferenciar si el resultado se obtuvo al correr los algoritmos con un total de 100 y 500 generaciones, cabe mencionarse que los datos que se muestran hacen referencia al mejor resultado obtenido para la razón de uso de las secuencias de producción:

	MOGA	MOGA LOCAL	MOGA SELF
B100	19.341575555	15.00246124	14.161242
C100	11.32425337	12.00566206	6.52675838
D100	15.01780196	16.1127654	10.3714282
E100	17.20965568	15.73329082	10.8105685
F100	14.48001455	15.62082454	12.3147938
G100	13.20142658	10.54770893	8.12959793
H100	18.89624373	13.49758035	17.390135
I100	21.11377533	14.65436759	12.7661227
J100	17.1723976	16.39369876	14.2977877

Tabla 5 Resultados obtenidos con 100 generaciones

	MOGA	MOGA LOCAL	MOGA SELF
B500	3.112971980	8.922329	2.506574519
C500	4.38841545	7.351398	3.936523188
D500	3.75192329	5.721187	2.182951605
E500	9.9524029	10.9763	1.463450379
F500	3.51142831	3.635486	1.271197589
G500	4.03222894	2.215851	1.048961058
H500	7.27764118	4.16684	1.49885045
I500	10.2316827	4.486839	1.257344014
J500	8.49704812	7.858213	1.802026816

Tabla 6 Resultados obtenidos con 500 generaciones

Con respecto a los resultados que muestra la Tabla 4 se puede notar que el algoritmo que implementa la estrategia autoajustable (MOGA SELF) obtuvo para todas las instancias del conjunto de problemas 1 el menor valor en comparación con lo obtenido por las otras heurísticas MOGA, MOGA LOCAL).

De igual forma se puede observar que la misma tendencia se presenta para la ocasión que se analizaron 500 generaciones como máximo notándose una reducción considerable con respecto a los valores obtenidos para la razón de uso.

Las Figuras 9 y 10 muestran la representación de los datos antes presentados, y se puede identificar la manera en la que el algoritmo genético multiobjetivo obtiene un valor para la razón de uso por debajo de lo que las otras propuestas obtienen siendo más evidente cuando el algoritmo se ejecuta a lo largo de 500 generaciones.

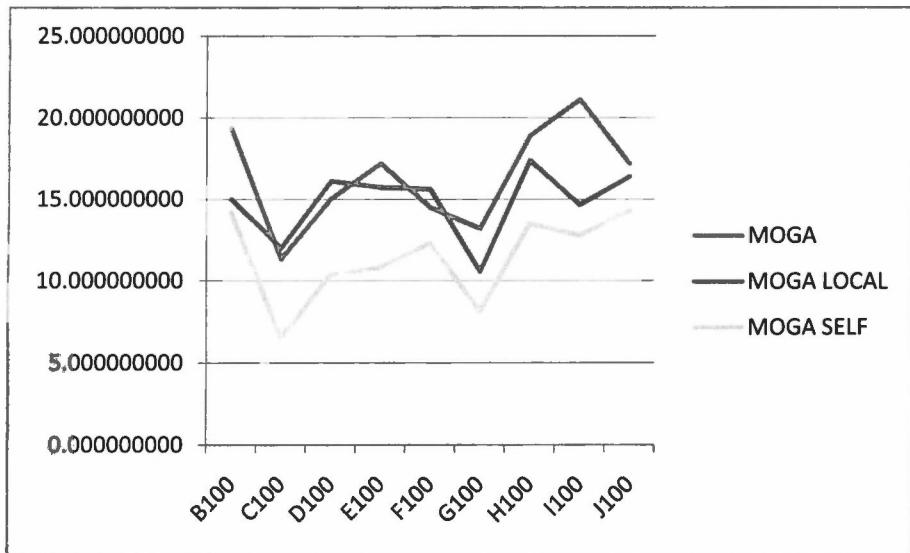


Figura 9 Razón de uso a lo largo de 100 generaciones

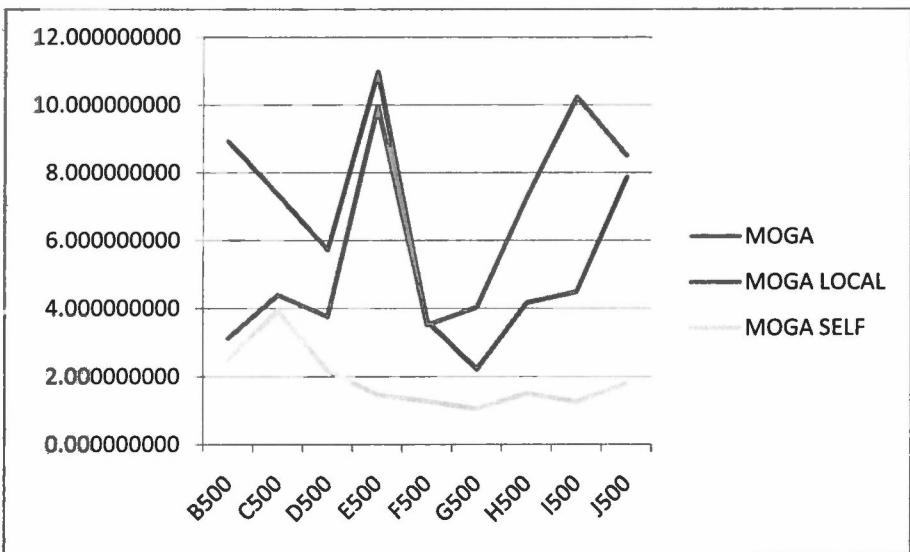


Figura 10 Razón de uso a lo largo de 500 generaciones

A continuación se presentan los resultados obtenidos en términos de la cantidad de operaciones de preparación en una secuencia de producción:

	MOGA	MOGA LOCAL	MOGA SELF
B100	6	8	5
C100	10	8	7
D100	8	9	5
E100	7	9	6
F100	10	9	8
G100	8	9	6
H100	8	10	7
I100	8	9	6
J100	9	10	8

Tabla 7 Cantidad de Ajustes obtenidos después de 100 generaciones

	MOGA	MOGA LOCAL	MOGA SELF
B500	7	9	5
C500	9	8	6
D500	8	10	7
E500	8	7	6
F500	9	8	5
G500	7	8	6
H500	11	10	9
I500	10	11	7
J500	7	9	6

Tabla 8 Cantidad de Ajustes obtenidos después de 500 generaciones

Es importante mencionar que se contabiliza un ajuste o secuencia de preparación cuando se encuentra un producto distinto en una secuencia de producción y por tal motivo se deba realizar una operación adicional no contemplada dentro de la producción. Lo anterior lo describe la ecuación 1 (Ec.1)

En la Figuras 9 y Figura 10 se puede notar la diferencia que presentan las tres heurísticas a lo largo de 100 y 500 generaciones mostrando una diferencia que va de 1 a 4 operaciones de preparación entre los valores obtenidos por el algoritmo MOGA SELF contra los otros algoritmos (MOGA, MOGA LOCAL) notándose una tendencia en la cual para éstas instancias el algoritmo propuesto muestra un mejor comportamiento a lo largo del tiempo.

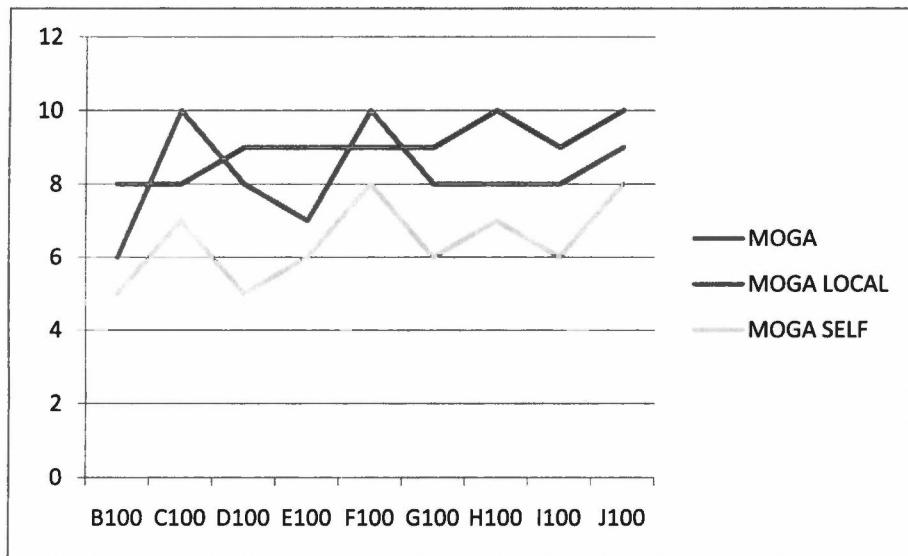


Figura 11 Cantidad de operaciones de ajuste después de 100 generaciones

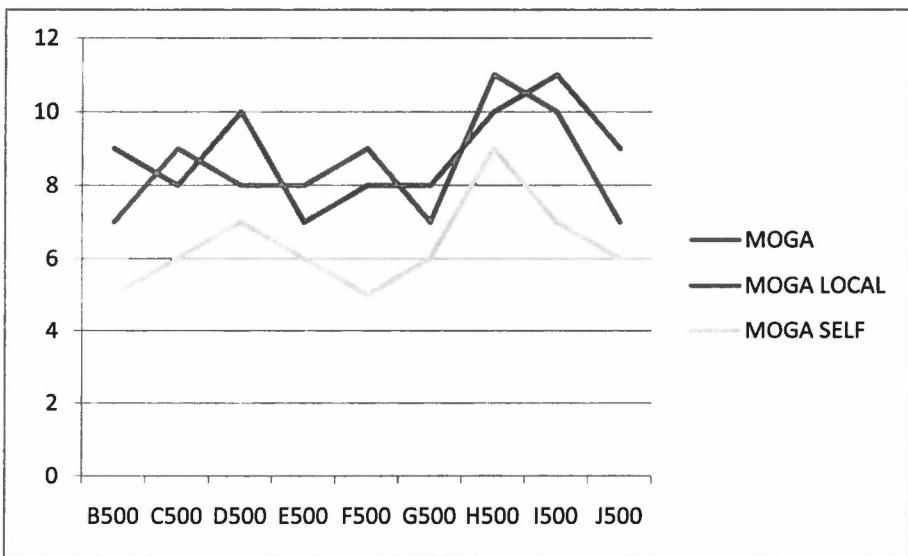


Figura 12 Cantidad de operaciones de ajuste después de 500 generaciones

De las pruebas realizadas a las instancias de McMullen [83] se observa que el algoritmo genético multiobjetivo con el módulo autoadaptable mostró un mejor desempeño en términos del valor obtenido para las funciones objetivo de la razón de uso y la cantidad de operaciones de preparación durante una secuencia de producción en las dos instancias de problemas elegidas para los casos de un total de 100 y 500 generaciones.

De igual forma se pudo notar con base en el desempeño a lo largo del tiempo que el módulo autoadaptable de mutación permitió cumplir con las dos funciones objetivo propuestas para esta investigación, minimizar de manera simultánea la razón de uso que una secuencia de producción presenta y la cantidad de operaciones de preparación.

Se puede observar que para ambos objetivos en todas las instancias en las que fue probado el algoritmo propuesto en esta tesis se pudo obtener una frontera de Pareto con soluciones no dominadas en todos los casos en comparación con las otras heurísticas (MOGA, MOGA LOCAL) implementadas para desarrollar el problema, lo anterior se puede observar en las Figuras 11 y 12 para el caso del número de operaciones de ajuste y preparación y en las Figuras 13 y 14 para el caso de la razón de uso.

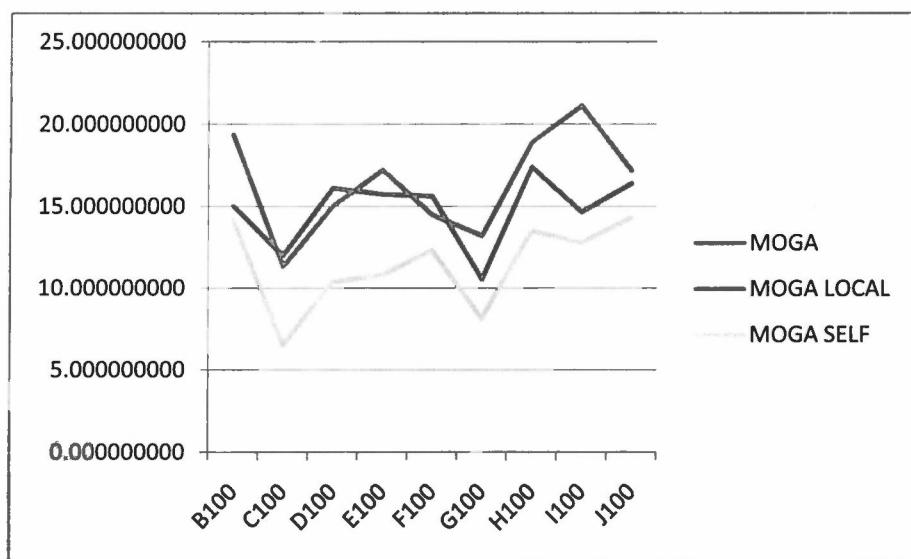


Figura 13 Razón de uso después de 100 generaciones

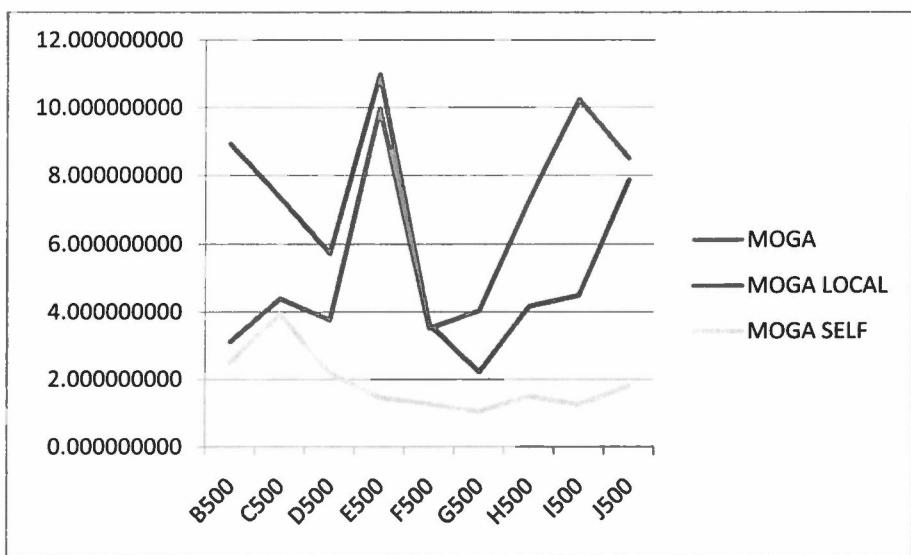


Figura 14 Razón de uso después de 500 generaciones

CAPÍTULO VI

6.1 CONCLUSIONES

El modelar matemáticamente un problema de la vida real no siempre nos asegura la factibilidad de resolverlo o de llegar a una solución óptima. Como se mencionó al inicio del presente trabajo el llegar a una solución óptima en ocasiones resulta ambiguo dado que si estamos atacando un problema en el cual intervienen más de dos variables de manera directa dado que pueden competir por el mismo recurso o resultar ser mutuamente excluyentes. Por lo anterior es que se necesitan estrategias de solución para tales problemas que puedan tener la capacidad de buscar de manera rápida y eficaz un espacio de solución complejo, discontinuo o considerablemente grande para llegar a una propuesta de solución.

El problema de asignación de recursos se ha descrito como un problema de tipo NP Completo dado que parte de un problema de decisión que por naturaleza son NP, esto implica que no existe un algoritmo que pueda resolver el problema en tiempo polinómico e implica que el tiempo necesario no será menos que exponencial. Tal característica aplica a cualquier instancia que se derive del problema de asignación de recursos que en este caso hace referencia a la secuenciación de trabajos en una línea de ensamble, para este caso asignamos los recursos disponibles (las máquinas necesarias para el ensamble) a determinadas tareas y actividades.

El resolver este problema tiene gran relevancia para las empresas que tienen un sistema de producción Justo a Tiempo pues les permitirá tener una ventaja competitiva con respecto a su competencia por la facilidad de ajustar su producción al incluir pedidos que no estaban planeados o contemplados y el seguir manteniendo políticas de calidad y reducción de costos sin afectar tampoco el tiempo estimado de entrega.

El cómputo evolutivo ha provisto a las empresas la posibilidad de contar con distintas técnicas de solución para problemas complejos que de manera tradicional tomarían demasiado tiempo para ser resueltos. Los algoritmos genéticos han demostrado ser una de las mejores estrategias de solución a problemas que contempla más de un objetivo de manera simultánea y con distintos propósitos de solución (maximización o minimización) dada su capacidad de exploración y explotación en un espacio de solución complejo y vasto.

Con la herramienta que se desarrolló para resolver el problema de secuenciación de trabajos en una línea de ensamble con un sistema de producción justo a tiempo se puede concluir que el uso de un Algoritmo Genético Multiobjetivo con un módulo de mutación autoadaptable permite encontrar soluciones que cumplen con la minimización de la razón de uso de la materia prima y la cantidad de operaciones de preparación y ajuste en una secuencia de producción además de presentar un mejor desempeño en todas las instancias probadas en comparación con un algoritmo genético multiobjetivo estándar o con una técnica de refinamiento, por lo que se puede considerar como una heurística viable para obtener secuencias de producción óptimas.

De igual forma se pudo observar la tendencia a encontrar un valor para ambos objetivos menor al que se obtiene por las otras dos heurísticas en las instancias de problemas pequeños dibujando una frontera de Pareto que contempla soluciones no dominadas.

6.2 TRABAJO FUTURO

Partiendo de los resultados obtenidos de la presente tesis y de la herramienta que se desarrolló para sustentar la misma se consideran los siguientes puntos como trabajos futuros para enriquecer el presente trabajo de investigación:

- Considerar dentro de la secuenciación de trabajos eventualidades como; reparaciones de máquina o descomposturas de la misma, ventanas de descanso por parte del operador, etc.
- Desarrollo de un módulo de cruce autoadaptable.
- Desarrollo de una implementación en paralelo para incluir secuencias más complicadas y con un mayor número de restricciones dentro de la planeación (precedencia, tiempo de ciclo, tiempo de preparación variable)
- Es necesario analizar más resultados experimentales y tomar modelos de secuenciación de una empresa que se rija bajo el esquema de producción Justo a Tiempo.
- Considerar una planeación de la producción que contemple un periodo de tiempo mayor al mínimo común (jornada laboral de ocho horas) poder trabajar con planes de producción semanales.
- Probar otros modelos autoadaptables que puedan incluirse dentro del algoritmo.
- Incluir más funciones objetivo que estén directamente relacionadas con el problema.

BIBLIOGRAFÍA Y REFERENCIAS

- [1]. *Algorithms for Solving Production-Scheduling Problems.* Giffler, B. y Thompson, G. L. 1960, Operations Research, págs. 487 - 503.
- [2]. *A Branch and Bound Algorithm for the Job-Shop Scheduling Problem.* Brucker, Peter, Jurisch, Bernd y Sievers, Bernd. 1994, Discrete Applied Mathematics, págs. 107-127.
- [3]. **Glover, Fred and Laguna, Manuel.** *Tabu Search.* s.l. : Kluwer Academic Publishers, 1997.
- [4]. *Optimization by simulated annealing.* Kirkpatrick, S., Gelatt, C.D. y Vecchi, M.P. 1983, Science, págs. 671-680.
- [5]. **Aliev, R.A. y Aliev, R. R.** *Soft computing and its applications.* Singapore : World Scientific, 2001.
- [6]. **Holland, John H.** *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* Michigan : University of Michigan Press, 1975.
- [7]. **Goldberg, David E.** *Genetic algorithms in search, optimization and machine learning.* Boston, MA : Addison-Wesley, 1989.
- [8]. *Esempi numerici di processi di evoluzione.* Barricelli, N. A. 1954, Methods, págs. 45 - 68.
- [9]. *Fuzzy algorithms.* Zadeh, L. A. 1968, Information and Control, págs. 94-102.
- [10]. *A logical calculus of the ideas immanent in nervous activity.* McCulloch, W. S y Pitts, W. H. 1943, Bulletin of Mathematical Biophysics, págs. 115-133..
- [11]. **Rechenberg, Ingo.** *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution.* Stuttgart : Frommann-Holzboog-Verlag, 1973.
- [12]. **Coello, Carlos A.** *Evolutionary algorithms for solving multi-objective problems.* Nueva York : Springer, 2007.
- [13]. *Genetic local search for multi-objective flowshop.* Arroyo, José Elias Claudio y Armentano, Vinícius Amaral. 2005, European Journal of Operational Research, págs. 717–738.
- [14]. *Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms.* Srinivas, N. y Deb, Kalyanmoy. 1994, Evolutionary Computation, págs. 221-248.
- [15]. **Deb, Kalyanmoy.** *Multi-objective optimization using evolutionary algorithms.* Nueva York : John Wiley & Sons, 2001.
- [16]. **Gen, M. y Cheng, R.** *Genetic algorithms and engineering design.* New York : Wiley, 1997.

- [17]. **Garey, M.R. and Johnson, D.S.** *Computers and Intractability - A Guide to the Theory of NP-Completeness*, s.l. : W.H Freeman, 1979.
- [18]. *An extension of Jhonson's results on job lot scheduling*. **Jackson, J. R.** 1956, Naval Research Logistics Quarterly, págs. 201-203.
- [19]. **Papadimitriou., Christos H.** *Computational complexity*. Massachussets : Addison-Wesley, 1994.
- [20]. **Chambers, Lance.** *The practical handbook of genetic algorithms : applications*. Boca Raton, Florida : Chapman & Hall, 2001.
- [21]. **Lawler, E. L., y otros.** Sequencing and scheduling: algorithms and complexity. *Logistics of production and inventory horm. 4 handbook in operations research and management science*. Mayo de 1993, págs. 287-326.
- [22]. **Varela, Leonilde, Aparicio, Joaquim y Carmo do Silva, Sílvio.** A Scheduling Web Service based on XML-RPC. *Multidisciplinary Scheduling: Theory and Applications*. Nottingham : Springer US, 2005, págs. 187-201.
- [23]. **Lambert, Douglas M.** *Supply chain management : processes, partnerships, performance*. Sarasota, Fla : Supply Chain Management Institute, 2004.
- [24]. **Michael, George.** *Lean Six SIGMA: Combining Six SIGMA Quality with Lean Production Speed*. Nueva York : McGraw-Hill, 2002.
- [25]. *Make to order or make to stock: Model and application*. **Rajagopalan, S.** 2002, Management Science, págs. 241-256.
- [26]. **Osyczka, A.** *Multicriteria Optimization in Engineering with FORTRAN programs*. Chichester : Ellis Horwood, 1984.
- [27]. **Mitsuo, Gen y Runwei, Cheng.** *Genetic algorithms and engineering optimization*. Nueva York : Wiley, 2000.
- [28]. **Charnes, A. y Cooper, WW.** *Management models and industrial applications of linear programming*. Nueva York : Wiley, 1961.
- [29]. *An updated survey of GA-based multiobjective optimization techniques*. **Coello, Carlos A.** 2000, ACM Computing Surveys (CSUR), págs. 109 - 143.
- [30]. **Deb, K.** *Optimization for Engineering Design: Algorithms and Examples*. Nueva Delhi : Prentice-Hall, 1995.
- [31]. **Fogel, L., Owens, A. y Walsh, M.** *Artificial Intelligence through Simulated Evolution*. Chichester : John Wiley, 1966.

- [32]. **Koza, J.** *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Massachusetts : The MIT Press, 1992.
- [33]. **Hallinan, J.** Feature selection and classification in the diagnosis of cervical cancer. [aut. libro] L. Chambers. *Practical Handbook of Genetic Algorithms, 2nd Edition*. Boca Raton, Florida : Chapman & Hall/CRC, 2001, págs. 167 - 202.
- [34]. *Using genetic algorithms to solve a multiple objective groundwater pollution containment problem*. **Ritzel, B. J., Eheart, J. W. y Ranjithan, S.** 1994, Water resources research, págs. 1589-1603.
- [35]. **Seixas, J., y otros.** Geneticland: Modelling Land-Use Change Using Evolutionary Algorithms. *Modelling Land-Use Change*. Netherlands : Springer Netherlands, 2007, págs. 181-197.
- [36]. *Epileptic seizure detection using genetically programmed artificial features*. **Firpi, H., Goodman, E.D and Echauz, J.** 2007, IEEE Transactions on Biomedical Engineering, pp. 212-224.
- [37]. **Osman, Ibrahim H. y Kelly, James P.** *Meta-Heuristics: Theory and Applications*. Norwell, MA : Kluwer Academic Publishers, 1996.
- [38]. **Zbigniew, Michalewicz.** *Genetic algorithms + data structures = evolution programs*. Berlin : Springer-Verlag, 1996.
- [39]. **Schaffer, J.D.** Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. *Tesis de Doctorado*. Nashville, Tennessee : s.n., 1984.
- [40]. *Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization*. **Fonseca, C.M. y Fleming, P. J.** s.l. : Morgan Kaufmann, 19993. Proceedings of the Fifth International Conference. págs. 416-423.
- [41]. *A niched pareto genetic algorithm for multiobjective optimization*. **Horn, J., Nafpliotis, N. y Goldberg, D. E.** Nueva Jersey : IEEE Press, 1994. Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Computation. págs. 82-87.
- [42]. *Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Evolutionary Algorithm*. **Zitzler, E. y Thiele, L.** 1999, IEEE Transactions on Evolutionary Computation, págs. 257-271.
- [43]. *A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II*. **Deb, K., y otros.** 2002, IEEE Transactions on Evolutionary Computation, págs. 182 - 197.
- [44]. **French, S.** *Sequencing and Scheduling: An Introduction to the Mathematics of*. s.l. : Wiley, 1982.

- [45]. **Lawler, E. L., y otros.** *The Traveling Salesman Problem*. Nueva York : Wiley, 1985.
- [46]. **Pinedo, M.** *Scheduling: Theory, Algorithms, and Systems*. s.l. : Prentice Hall, 1995.
- [47]. *Algorithms for Solving Production-Scheduling Problems*. **Giffler, B. y Thompson, G. L.** 1960, Operations Research, págs. 487-503.
- [48]. **Garey, Michael R. y Johnson, David S.** *Computers and intractability : a guide to the theory of NP-completeness*. Nueva York : W. H. Freeman and Company, 1979.
- [49]. *Evolving genetic algorithm for Job Shop Scheduling problems*. **Werner, James. C, Aydin, Mehmet E. y Fogarty, Terence C.** 2002. Adaptive computing in design and manufacture.
- [50]. *Minimizing the makespan for the flow shop scheduling problem with availability constraints*. **Aggoune, Riad.** 2004, European Journal of Operational Research, págs. 534-543.
- [51]. *A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation*. **Bertel, S. y Billaut, J.-C.** 2004, European Journal of Operational Research, págs. 651–662.
- [52]. *Genetic algorithms for sequencing problems in mixed model assembly lines*. **Ponnambalam, S.G., Aravindan, P. y Rao, M. Subba.** 2003, Computers & Industrial Engineering, págs. 669-690.
- [53]. *Parallel machine scheduling of machine-dependent jobs with unit-length*. **Lin, Y. y Liu, W.** 2004, European Journal of Operational Research, págs. 261-267.
- [54]. *A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs*. **Varadharajan, T.K y Rajendran, Chandrasekharan.** 2005, European Journal of Operational Research, págs. 772-795.
- [55]. *A random key based genetic algorithm for the resource constrained project scheduling problem*. **Mendes, J. J. M., Gonçalves, J. F. y Resende, M. G. C.** 2007, Computers and Operations Research, págs. 92-109 .
- [56]. *A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem*. **Gen, Mitsuo, Ida, Kenichi y Watanabe, Masato.** 2005, Computers & Industrial Engineering, págs. 743–752.
- [57]. *Evolutionary algorithms for job-shop scheduling*. **Hammadi, Slim, Mesghouni, Khaled y Borne, Pierre.** 2004, International Journal of Applied Mathematics and Computer Science, págs. 91-103.
- [58]. *The Pareto fitness genetic algorithm: Test function study*. **Elaoud, Semya, Loukil, Taicir y Teghem, Jacques.** 2007, European Journal of Operational Research, págs. 1703–1719.

- [59]. *An efficient genetic algorithm for job shop scheduling with tardiness objectives.* **Mattfeld, Dirk C. y Bierwirth, Christian.** 2004, European Journal of Operational Research, págs. 616-630.
- [60]. *The hybrid heuristic genetic algorithm for job shop scheduling.* **Zhou, Hong, Feng, Yuncheng y Han, Limin.** 2001, Computers and Industrial Engineering, págs. 191-200.
- [61]. *An effective genetic algorithm approach to multiobjective resource allocation problems (MORAPs).* **Osman, M.S., Abo-Sinna, M.A. y Mousa, A.A.** 2005, Applied Mathematics and Computation, págs. 755–76.
- [62]. *Using genetic algorithms and heuristics for job shop scheduling with sequence-dependent setup times.* **Cheung, Waiman y Zhou, Hong.** 2001, Annals of Operations Research, págs. 65-81.
- [63]. *A genetic algorithm approach to the scheduling of FMSs with multiple routes.* **Zhao, Chunwei y Wu, Zhiming.** 2001, International Journal of Flexible Manufacturing Systems, págs. 71-88.
- [64]. *A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility.* **Ruiz, Rubén y Maroto, Concepción.** 2006, European Journal of Operational Research, págs. 781-800.
- [65]. *A hybrid genetic algorithm for the finite horizon economic lot and delivery scheduling in supply chains.* **Torabi, S.A., Ghomi, S.M.T. Fatemi y Karimi, B.** 2006, European Journal of Operational Research, págs. 173–189.
- [66]. **Pinto, Errol G.** *Supply Chain Optimization using Multi-Objective Evolutionary Algorithms.* Pennsylvania : Pennsylvania State University, 2004.
- [67]. *Supply chain multi-objective simulation optimization.* **Joines, J. A, y otros.** San Diego, California : Winter Simulation Conference, 2002. Winter Simulation Conference. págs. 1306-1314.
- [68]. *Hybrid genetic algorithm for multi-time period production/distribution planning.* **Mitsuo, Gen y Syarif, Admi.** 2005, Computers & Industrial Engineering, págs. 799–809.
- [69]. *Two heuristics for scheduling multiple projects with resource constraints.* **Tsai, DM y Chiu, HN.** 1996, Construction Management and Economics, págs. 325 - 340 .
- [70]. *Genetic algorithm based on heuristic rules for high-constrained large-size single-stage multi-product scheduling with parallel units.* **Yaohua, He y Hui, David Chi Wai.** 2008, Chemical engineering and processing, págs. 3067-3083.
- [71]. *Scheduling mixed model multi-level just-in-time production systems.* **Miltenburg, J. y Sinnamon, G.** 1989, International Journal of Production Research, págs. 1487-1509.

- [72]. **Monden, Y.** *Toyota production system*. Norcross, GA : Institution of Industrial Engineering, 1998.
- [73]. *A Simulated Annealing Approach to Mixed-Model Sequencing with Multiple Objectives on a JIT Line*,. **McMullen, P. R. y Frazier, G. V.** 8, s.l. : IIE Transactions: Scheduling and Logistics, 2000, Vol. 32.
- [74]. *A transformed two stage method for reducing the part-usage variation of the product-level and part-level*. **Zhu, J. and Ding, F. Y.** 127, s.l. : A transformed two stage method for reducing the part-usage variation of the product-level and part-level, 2000, Vol. 1.
- [75]. *Multi-Objective Optimization of Flexible Manufacturing Systems*. **Chen, Jiang-Hung y Ho, Shinn-Ying**. San Francisco, California : Morgan Kaufmann Publishers, 2001. roceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001). págs. 1260--1267.
- [76]. *A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines*. **Hyun, C. J., Kim, Y. y Kim, Y. K.** s.l. : Computers & Operations Research, 1998, Vol. 25.
- [77]. **Groppetti, R. y Muscia, R.** On a Genetic Multiobjective Approach for the Integration and Optimization of Assembly Product Design and Process Planning. [aut. libro] P. Chedmail, J. C. Bocquet y D. Dornfeld. *Integrated Design and Manufacturing in Mechanical Engineering*,. Paises Bajos : Kluwer Academic Publishers, 1997, págs. 61-70.
- [78]. **Sastry, Kumara**. *Single and Multiobjective Genetic Algorithm Toolbox for Matlab in C++*. Illinois : University of Illinois at Urbana-Champaign, 2007.
- [79]. *Modeling tournament selection with replacement using apparent*. **Sastry, K. y Goldberg, D. E.** 2001, Intelligent Engineering Systems Through Artificial Neural Networks, págs. 129–134.
- [80]. **Sastry, K., Goldberg, D. E. y Kendall, G.** *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Berlin : Springer, 2005.
- [81]. *Real-coded genetic algorithms with simulated binary crossover: Studies on multimodal and multiobjective problems*. **Deb, K. y Kumar, A.** 1995, Complex Systems, págs. 431 - 454.
- [82]. **Press, W., y otros**. *Numerical recipes in C*. Cambridge : Cambridge University Press, 1989.
- [83]. *An Efficient Frontier Approach to Addressing JIT Sequencing Problems with Setups via Search Heuristics*. **McMullen, P. R.** 2001, Computers & Industrial Engineering, págs. 335 - 353.
- [84]. *A Genetic Algorithm for the Resource Constrained Multi-Project Scheduling Problem*. **Gonçalves, José Fernando y Mendes, Jorge José de Magalhães Mendes**. 2008, European Journal of Operational Research, págs. 1171-1190.

ANEXOS

Como se mencionó en el capítulo 5, el módulo de mutación autoadaptable se desarrolló bajo la arquitectura de Sastry[78] la cual se implementó en MatLab desarrollando sus módulos en C++.

Para compilar el programa una vez que se tienen todos los módulos del mismo se debe ocupar el comando mex dentro de MatLab con la siguiente estructura:

*mex -output MOGA *.cpp*

De igual forma el archivo nombrado ***mogaFitnessFunction.m*** contiene la(s) función(es) de aptitud que para el presente trabajo contiene codificadas la razón de uso y la cantidad de operaciones de preparación siguiendo la sintaxis que MatLab requiere quedando como se muestra en la Figura 15.

```
function y=UR(S,d,a)
% Usage rate for x
[m,n]=size(S);
DT=sum(d);
tt= ;
for k= :DT
    for i= :a
        xik= ;
        t1= ;
        for l= :k
            if (S(l)==i) xik=xik+ ; end
        end
        t1=(xik-k*d(i)/DT)^ ;
        tt=tt+t1;
    end
end
y=tt;
```

Se puede notar que la función objetivo toma como entrada la secuencia a analizar (**S**), la demanda total (**DT**) y la cantidad de productos únicos a producir (**a**).

Es importante resaltar de igual forma la estructura del archivo de configuración en el cual se definen los módulos y características más importantes que el Algoritmo Genético implementará.

A continuación la Figura 16 muestra las primeras 37 líneas con el contenido del archivo de acuerdo a lo que se utilizó en el presente trabajo de investigación, se puede notar la forma en la que se define la estrategia de solución, el número de variables y la configuración entre otras.

```

1   #
2   # Tipo de AG: SGA or NSGA
3   # SGA para un solo objetivo
4   # NSGA para más de un objetivo
5   #
6   NSGA
7
8   #
9   # Número de Variables de decisión
10  #
11
12
13  #
14  # Número de productos únicos a elaborar
15  #
16  #
17  3
18
19  #
20  #
21  #Mix de products
22  # Basado en la cantidad total de productos únicos a producir
23  # la demanda de cada producto se define como sigue:
24  #           [Número de Producto] [demanda]
25  #
26  #
27  1 2
28  2 1
29  1 1
30
31  # Por cada variable de decisión se debe definir:
32  # tipo de variable, Límite inferior, Límite superior
33  # El tipo de la variable de decisión puede ser doble(double) o entera (int)
34  #
35  int n
36  int l
37  double -5.0 5.0

```

Figura 16

En la siguiente figura se muestra la sección donde se define la cantidad de objetivos a resolver así como las restricciones a manejar, el tamaño de la población, número máximo de generaciones, etc.

```

40  # Objetivos:
41  #   Numero de objetivos
42  #   Para cada objetivo se debe definir el tipo de optimización: Min Max
43  #
44  #
45  Min
46  Min
47  #
48  #
49  # Restricciones
50  #   Numero de restricciones que seran penalizadas
51  #   para cada restriccion penalizada se debe asignar un peso
52  #
53  #
54  #
55  #
56  # Parametros Generales: Si no se define algun parametro especifico
57  #                         se tomaran valores por default. Si es el caso se debe escribir
58  #                         "default" en la posicion para cada parametro.
59  #   [Tamaño de la población]
60  #   [numero maximo de generaciones]
61  #   [porcion de remplazo]
62  #
63  #
64  #
65  default

```

Figura 17

Se puede notar lo fácil que es definir la cantidad de funciones objetivo a añadir, en la línea 44 podemos escribir el número de funciones objetivo y debajo definir si se va a minimizar o maximizar (Min, Max), de igual forma en la línea 63 se define el tamaño de la población y el número de generaciones que se desea utilizar para el estudio.

De la línea 67 a la 80 en el archivo de configuración se definen los distintos nichos que permiten mantener distintas soluciones dentro de la población, las distintas opciones disponibles que se pueden elegir en esta implementación son las siguientes; no implementar nichos, distribución, RTS y agrupamiento determinístico. Para la presente aplicación no se ocupará la estrategia de nichos.

En la figura 18 se puede notar la sección del archivo de configuración que define los principales operadores genéticos ocupados en el presente trabajo de investigación

```

82  # Selección
83  # Uso: Tipo de Selección , [parametro(s)...]
84  # Los tipos de selección disponibles y sus parámetros pueden ser:
85  # Ruleta: RouletteWheel
86  # SUS
87  # Torneo sin reemplazo [tamaño del torneo] : TournamentWOR [tournament size]
88  # Torneo con reemplazo [tamaño del torneo] : TournamentWR [tournament size]
89  # Truncado [# de copias] : Truncation [# copies]
90  #
91  # Cuando se implemente NSGA no se puede utilizar SUS o ruleta
92  #
93  RouletteWheel
94  #
95  # Cruce
96  # Probabilidad de cruce
97  #
98  # Uso: Tipo de cruce, [parametro(s)...]
99  # Tipos válidos de cruce y los parámetros opcionales son:
100 # Un punto : OnePoint
101 # Dos puntos : TwoPoint
102 # Uniforme [probabilidad de intercambio de genes] : Uniform [genewise swap probability]
103 # SBX [probabilidad de intercambio de genes][órden del polinomio] :
104 #           SBX [genewise swap probability][order of the polynomial]
105 #
106 default
107 default
108 #
109 # Mutación
110 # Probabilidad de Mutación
111 #
112 # Uso: Tipo de mutación, [parametro(s)...]
113 # Los tipos válidos de mutación y sus parámetros opcionales son:
114 # Selectivo : Selective
115 # Polinomial [órden del polinomio: Polynomial [order of the polynomial]]
116 # Cambio de génes [sigma para el gen#1][sigma para el gen#2][sigma para el gen#n] :
117 #           Genewise [sigma for gene #1][sigma for gene #2]...[sigma for gene #n]
118 # Autoajustable : Self
119 Self

```

Figura 18

Es fácil notar las distintas opciones que se tienen para cada uno de los operadores disponibles prestando especial atención al operador de mutación donde se muestra incluido el operador autoajustable que se desarrolló y que solo se necesita declararlo en el archivo de configuración para que el algoritmo lo implemente de manera transparente junto con las otras variantes disponibles (selectivo, polinomial y cambio de génes)

Las últimas líneas del archivo de configuración corresponden a la activación de los métodos de búsqueda local para refinar los resultados obtenidos además del criterio de paro que se ocupará para terminar la operación del programa.

Ahora bien, una vez que se definió el uso del módulo de mutación autoajustable dentro del algoritmo se debe incluir dentro del programa que se encarga de leer las opciones del archivo de configuración la rutina para reconocer la nueva estrategia de mutación a implementar, en la Figura 19 se puede notar en las líneas 647 y 648 del archivo “userDefinables.cpp” la declaración de esta nueva estrategia de mutación.

```

618 // tipo de mutacion
619 if ((pToken = readOneLine(caBuf, ciBufSize, fInput)) == NULL) {
620     fclose(fInput);
621     mexErrMsgTxt("Error in the input file, please refer to the documentation");
622 }
623 if(strcmp("default", pToken) == ) {
624     mexPrintf("# %s\n";
625     globalSetup->mutationType = Polynomial;
626 }
627 else if (strcmp("Selective", pToken) == ) {
628     globalSetup->mutationType = Selective;
629 }
630 else if (strcmp("Generwise", pToken) == ) {
631     globalSetup->mutationType = Generwise;
632 }
633 else if (strcmp("Polynomial", pToken) == ) {
634     globalSetup->mutationType = Polynomial;
635 }
636 else if (strcmp("Selfevolution", pToken) == ) {
637     globalSetup->mutationType = Selfevolution;
638 }
639 else {
640     fclose(fInput);
641     mexErrMsgTxt("Sorry, valid mutation types are: Selective, Generwise, Selfevolution and Polynomial");
642 }
643

```

Figura 19

Una vez definida la estrategia de mutación autoajustable en el archivo de configuración y la rutina de reconocimiento de preferencias se debe definir el método que contendrá los elementos más importantes de la estrategia de mutación autoajustable donde se puede notar la inicialización de un arreglo con diez espacios para registrar los valores aleatorios con los que se trabajará durante las primeras 10 generaciones

```

697 // self evolution method
698 case Selfevolution:
699 {
700     globalSetup->mutationParameters = new int[ ];
701     if ((pToken = strtok(NULL, BLANK_STR)) == NULL) {
702         mexPrintf("U need el modulo autoadjustable de mutation");
703         for(ii = ; ii < ; ii++) {
704             *(newselfparam[ii]) = *(tempselfparam[ii]);
705         }
706     }
707     else
708     ((int*)globalSetup->mutationParameters)[ ] = atoi(pToken);
709 }
710 break;

```

Figura 20