CIATEC

PROGRAMACIÓN DE RUTAS DE MATERIALES EN CONFIGURACIÓN DE TRABAJO JOB-SHOP CON ALGORITMOS GENÉTICOS

TESIS

PARA OBTENER EL GRADO ACADÉMICO DE:

MAESTRO EN CIENCIA Y TECNOLOGÍA EN LA ESPECIALIDAD DE INGENIERÍA INDUSTRIAL Y MANUFACTURA

PRESENTA

I.I. DAVID ALEJANDRO VALLE MEDINA

DIRECTOR DE TESIS

DR. JAVIER YAÑEZ MENDIOLA

POSGRADO INTERINSTITUCIONAL CIENCIA Y TECNOLOGIA

LEÓN, GUANAJUATO, ENERO DE 2013

AGRADECIMIENTOS

A Dios, por ponerme en este camino y rodearme de gente tan maravillosa.

A mi Madre, Nina: gracias por ser mi soporte moral y financiero durante años. Siempre estaré en deuda contigo. Has confiado siempre en mí a pesar de los difíciles comienzos. Este posgrado es para ti.

En memoria del Dr. Pablo Valle, mi Padre. Mi ejemplo para toda la vida, un maestro de quién todavía aprendo las duras lecciones de la vida. Me enseñaste el valor de la congruencia y comienzo a darme cuenta que es la única ventana hacia un horizonte más brillante.

A mi esposa Yunuén, fuiste cómplice de los desvelos, frustraciones y logros. Eres quién siempre me impulsa a seguir de adelante. Gracias por tu cariño Flais.

Lily: me ensañaste con tu ejemplo, Dulce: supiste encausarme oportunamente y Helen: siempre admiraré tu firme perseverancia.

Agradezco profundamente al Dr. Javier Yáñez, director de esta tesis. Siempre tuvo un comentario claro y oportuno sobre la investigación y creyó en mí desde un principio. Sin su valioso apoyo en el campo de la programación matemática, no hubiese alcanzado los resultados esperados.

No puedo dejar de agradecer al CONACYT por darme la oportunidad de ingresar a este programa de posgrado y financiar mi beca de estudios. Al CIATEC que me abrió sus puertas y me brindó un espacio académico de alto nivel. Agradezco a mis profesores, compañeros y asesores. Gracias Toño.

Dedico un gran agradecimiento a la empresa Tecnomec Agrícola y a la familia Valdés Reynoso por su apoyo para la realización de esta investigación en nuestra planta, además de otorgarme el tiempo y recursos para la investigación que culmina con esta tesis.

A mis primos, tíos, compañeros y amigos, gracias por todo.

-"Si he podido ver más lejos es porque he subido a hombros de gigantes"

Sir Isaac Newton

RESUMEN

En este trabajo de investigación se estudiará la programación dinámica en un problema difícil de resolver, mediante la programación de los trabajos en un sistema de trabajo tipo taller o *job-shop*. El objetivo principal de esta investigación es generar una herramienta de programación que ayude a identificar la carga de trabajo de distintos centros productivos o maquinaria y entregue un sistema de secuenciación de estas tareas para facilitar la administración del departamento donde se ejecuta el sistema.

Se ha recurrido a un software de programación matemática para el desarrollo de un algoritmo evolutivo como método de optimización. El Algoritmo Genético que se estudia en esta investigación es capaz de generar una solución factible (disminución sustancial del tiempo de terminación de todos los trabajos) para este problema de secuenciación de trabajos en un sistema dinámico como el que presenta la empresa donde se realizó la implementación de esta investigación.

Palabras clave: Algoritmo Genético, Job-Shop Scheduling Problem, Optimización Combinatoria, Advanced Planning and Scheduling

INTRODUCCIÓN

En la industria de la transformación moderna se pretende reducir cualquier tipo de costo derivado de desperdicios del proceso; las materias primas así como los inventarios en proceso y productos terminados representan un activo que siempre está expuesto a los cambios de la demanda e ingeniería. Otro valioso recurso para cualquier organización es el tiempo productivo en las operaciones, maximizar el aprovechamiento de las máquinas cuello de botella es una actividad que favorece enormemente la productividad, por lo tanto resulta muy importante reducir este tipo de costos a través de una buena optimización de la programación de la producción y de las prioridades en el proceso productivo. Encontrar la mejor combinación de producción resulta indispensable cuando se quiere optimizar los recursos para la transformación y evitar grandes inventarios de productos terminados para el abastecimiento de la cambiante demanda global. El término APS por su siglas en inglés *Advanced Planning and Scheduling* es una expresión que recientemente comienza a utilizarse en el campo de la manufactura de clase mundial debido a que se basa en la administración de los procesos de manufactura donde la capacidad del proceso y las materias primas se deben sincronizar de manera óptima respecto a la demanda. Regularmente es utilizado en ambientes donde la programación de producción común no es capaz de direccionar adecuadamente las prioridades que se tienen en cada etapa del proceso productivo.

Los algoritmos evolutivos permiten abordar problemas complejos de búsqueda que surgen en las ingenierías y los campos científicos, sus aplicaciones van desde la planificación de tareas, horarios, tráfico, búsqueda de caminos óptimos hasta optimización de funciones complejas. Son utilizados en problemas difíciles de satisfacer, los cuales no pueden ser resueltos por métodos de optimización tradicionales. Los Algoritmos Genéticos forman parte de los sistemas de solución evolutivos; su objetivo es buscar un buen resultado entre una población de soluciones. Cada solución recibe el nombre de cromosoma y los símbolos que forman esta cadena son llamados genes. El algoritmo trabaja por medio de iteraciones llamadas generaciones, donde además se introducen variaciones aleatorias de mutación y cruce, los cromosomas con mejores características se seleccionan para la siguiente generación.

En este trabajo de investigación se aborda el problema de secuenciación de trabajos en un escenario de producción por lotes. El problema de secuenciación de trabajos en una configuración *job-shop* se caracteriza por requerir una distribución de trabajos efectiva en los recursos disponibles a fin de tener una producción dinámica que minimice los inventarios de producto. El objetivo principal es maximizar el tiempo disponible de la maquinaria a fin de cumplir con todas las demandas de trabajo, mientras que se reduce el *makespan* del proceso productivo.

TABLA DE CONTENIDO

1. INTRODUCCIÓN 11

2. PLANTEAMIENTO DEL PROBLEMA 15

3. ESTADO DEL ARTE 18

4. MARCO TEÓRICO 20

4.1 REPRESENTACIÓN DE UN MODELO DE I MAQUINAS CON J TRABAJOS 22
4.2 REPRESENTACIÓN DE UN MODELO PARA OPTIMIZAR UNA ASIGNACIÓN 23
4.3 MÉTODOS DE SOLUCIÓN 24
4.3.1 Técnicas de Optimización 25
4.3.2 Programación lineal entera mixta 25
4.3.3 Ramificación y poda 26
4.3.4 Algoritmos genéticos 27
4.3.5 Reglas de prioridad de despacho 28
4.3.6 Recocido simulado 29
4.3.7 Heurísticas basadas en cuellos de botella 30
4.3.8 Métodos de búsqueda local 31
4.3.9 El metaheurístico Particle Swarm Optimization 32
4.3.10 Búsqueda tabú 32
4.4 ALGORITMOS GENÉTICOS 34
4.4.1 Principios de los Algoritmos Genéticos (AG) 34
4.4.2 Representación de los individuos 35
4.4.3 Generación de la población inicial 36
4.4.4 Grado de adaptación de los individuos 37
4.4.5 Condiciones de terminación 37
4.4.6 Proceso de selección: Mecanismo de muestreo 37
4.4.7 Proceso de reproducción: Operadores genéticos 39
4.8 Proceso de remplazo 40

5. JUSTIFICACIÓN 42

6. OBJETIVO 44

7. HIPÓTESIS 46

8. DESARROLLO DE LA INVESTIGACIÓN 47

8.1 OBTENCIÓN DE DATOS 48
8.1.1 Tiempo ciclo 48
8.1.2 Tiempo de preparación 48
8.1.3 Recursos disponibles 49
8.2 PRUEBAS INICIALES 50
8.2.1 Fases de modelado para la experimentación 52
8.2.2 1° Etapa del proceso experimental. Población Inicial 53
8.2.3 Generación de la matriz de precedencias para el cromosoma seleccionado 56
8.2.4 2° Etapa del proceso experimental. Evaluación de adaptación en la función objetivo 57
8.2.5 Evaluación de la función objetivo en base al algoritmo de McNaughton 58
8.2.6 Evaluación de la función objetivo con Algoritmo List Scheduling 59
8.2.7 Evaluación de la función objetivo con Algoritmo de Tiempo más Largo de Proceso (LPT) 60
8.2.8 Evaluación de la función objetivo con Algoritmo de Brucker 61

8.3 PROCESO DE REPRODUCCIÓN Y MUTACIÓN 62
8.4 IMPLEMENTACIÓN GLOBAL BAPS 64
8.4.1 Implementación del sistema como programador de producción. 64
8.4.2 Implementación del sistema como guía para la distribución de los trabajos en planta. 65
9. ANÁLISIS DE RESULTADOS 67
10. CONCLUSIONES 72
11. REFERENCIAS BIBLIOGRÁFICAS 74
ANEXO A. PROGRAMACIÓN DE TOOLBOX DE OPTIMIZACIÓN MODIFICADO 76
ANEXO B. MATRIZ GENERAL DE DATOS 82
ANEXO C. DESCRIPCIÓN DE TOOLBOX UTILIZADOS 83

ÍNDICES PARA TABLAS Y FIGURAS GRÁFICAS

Ilustración 1. Modelo de Producción de Tecnomec Agrícola 12
Ilustración 2. Indicador de entregas contra programación de productos 13
Ilustración 3. Impacto Generado al Reducir las Actividades que no Agregan Valor 15
Ilustración 4. Distribución de trabajo en configuración tipo Job-Shop 17
Ilustración 5. Esquema Grafos para un problema de asignación de trabajos en máquinas 21
Ilustración 6. Representación de un cromosoma en un problema 5 X 3 36
Ilustración 7. Operador de cruce Monopunto 39
Ilustración 8. Operador de mutación 40
Ilustración 9. Ubicación en área de maquinados de los recursos para la prueba inicial 51
Ilustración 10. Fases del modelo heurístico de Algoritmos Genéticos 52
Ilustración 11. Representación gráfica de la evaluación del individuo con el Algoritmo de McNaughton 58
Ilustración 12. Representación gráfica de la evaluación del individuo con el Algoritmo List Scheduling 59
Ilustración 13. Representación gráfica de la evaluación del individuo con el Algoritmo LPT 60
Ilustración 14. Representación gráfica de la evaluación del individuo con el Algoritmo de Brucker 61
Ilustración 15. Toolbox para optimización en MATLAB con Algoritmos Genéticos 62
Ilustración 16. Ejemplo de sistema optimizado con toolbox de AG 63
Ilustración 17. Menú Principal del BAPS 64
Ilustración 18. Plan de secuenciación de trabajos 65
Ilustración 19. Área de maquinado de Tecnomec Agrícola S.A. de C.V. 66

Tabla 1. Recursos disponibles del área de maquinados 50
Tabla 2. Matriz de secuencia de operaciones para un producto 50
Tabla 3. Cromosoma aleatorizado 60X1. Resultado de la aplicación de la función aleatorioA. 55
Tabla 4. Población Aleatorizada 60X500. Resultado de la aplicación de la función Principal 55
Tabla 5. Cromosoma de un individuo con sus tiempos de proceso asociados 60X2 56
Tabla 6. Matriz de precedencias para la población 60X60 57
Tabla 7. Resultado de la evaluación del individuo con el Algoritmo de McNaughton 59
Tabla 8. Resultado de la evaluación del individuo con el Algoritmo List Scheduling 60
Tabla 9. Resultado de la evaluación del individuo con el Algoritmo LPT 61
Tabla 10. Resultado de la evaluación del individuo con el Algoritmo de Brucker 62
Tabla 11. Comportamiento del algoritmo para la población inicial. 67
Tabla 12. Comportamiento del algoritmo para la generación 25. 68
Tabla 13. Comportamiento del algoritmo para la generación 50. 68
Tabla 14. Comportamiento del algoritmo para la generación 100. 69
Tabla 15. Caracterización del sistema con la mejor combinación de cada iteración. 70
Tabla 16. Comparativo del makespan real en piso y la simulación con BAPS. 70

ABREVIACIONES Y SÍMBOLOS

FEMA. Farm Equipment Manufacturers Association

FEWA. Farm Equipment Wholesalers Association

BPS. Bison Production System

CNC. Control numérico computarizado

AVA. Actividades de Valor Agregado

ANVA. Actividades que No Agregan Valor

JSSP. Job Shop Scheduling Problem

AG. Algoritmo Genético

BAPS. Bison Advanced Planning Scheduling.

ERP. Enterprise Resource Planning.

TORSCHE. Time Optimization Resources Scheduling.

BAPS. Bison Advanced Production Scheduling

GLOSARIO

B

Bison Advanced Planning Scheduling (BAPS). Sistema computacional avanzado para la planeación de la secuenciación de trabajos.

E

Enterprise Resource Planning (ERP). Planeación de Recursos Empresariales

J

Job-Shop. Sistema de Manufactura que se caracteriza por producir lotes de tamaño muy pequeño con cambios de modelo constantes.

Job Shop Scheduling Problem (JSSP). Problema de secuenciación de la producción en un entorno productivo tipo taller.

M

Makespan. Tiempo de terminación de un proceso o un trabajo, se asocia con la finalización total de todas las operaciones de un trabajo o periodo establecido.

R

Restricción conjuntiva. La secuencia de las operaciones está definida previamente por el proceso de manufactura y ésta no pueda cambiarse en un sistema tipo job-shop.

Restricción disyuntiva. En un sistema *job-shop* limita a que cada recurso solo puede procesar un componente a la vez.

S

Setup. Tiempo transcurrido desde la terminación de la última pieza del lote, hasta el arranque del siguiente con la primera pieza validada contra especificaciones.

T

Tiempo Ciclo. Es el lapso transcurrido en una operación por cada pieza fabricada.

Time Optimization Resources Scheduling (TORSCHE). Es un sistema de optimización combinatoria para la secuenciación de tareas desarrollado por la Universidad Tecnológica Checa.

1. INTRODUCCIÓN

Tecnomec Agrícola S.A. de C.V. es un equipo de trabajo con 30 años de experiencia en diseño, manufactura y comercialización de equipo agrícola e industrial. Tecnomec forma parte del grupo VH integrado también por un centro de investigación privado apoyado por CONACYT nombrado IDEAA (Investigación y Desarrollo Aplicado de Aguascalientes), *Garfield Equipment* en los Estados Unidos y VH-Comercial. Su participación en el mercado nacional ha consolidado a la compañía como el líder en ventas de implementos agrícolas en el campo mexicano.

La empresa fue acreedora en el año 2003 del Premio Nacional de Exportación y es parte de organizaciones internacionales como: FEMA[1], FEWA[2] y Endeavor[3]. Sus instalaciones en el estado de Aguascalientes comprenden un espacio de 3 Hectáreas, con 9750m^2 techados y dedicados a la fabricación y 2,270m^2 de oficinas y áreas de servicio. Bison® cuenta con 450 colaboradores, que contribuyen a mantener rigurosos estándares de trabajo bajo la Norma ISO 9001 2008.

Esta visión emprendedora y de mejora continua en toda la cadena de valor, motivó a la dirección de la empresa a transformar el esquema de trabajo de Tecnomec en operaciones *Lean*[4] a través de la creación de un departamento de mejora denominado *Bison Production System (BPS)* en el año 2008. Desde entonces se han desarrollado mejores prácticas productivas basadas en los principios de ésta metodología, obteniendo muy buenos resultados en corto plazo de tiempo.

La empresa produce bajo un esquema de 6 días de *lead time*[5] para cualquier producto que se programe durante el día 0. Los 3 primeros días de transformación de un equipo son los más importantes ya que representan los procesos de cambio de conformación y

---

[1] *FEMA. Farm Equipment Manufacturers Association*, es una organización estadounidense encaminada a proveer oportunidades de negocios y mercado a través de una red de comunicaciones y tecnología para la adquisición de materiales y servicios requeridos por sus miembros.
[2] *FEWA. Farm Equipment Wholesalers Association*, es una asociación estadounidense que busca abrir mercados internacionales para la venta y distribución de maquinaria agrícola.
[3] *Endeavor*, es una organización privada que tiene como misión de transformar a México a través de empresas prosperas generadoras de empleo lideradas por emprendedores de alto impacto que incentiven a los demás a innovar y a tomar riesgos para generar un desarrollo económico sostenible.
[4] *Lean Manufacturing* es un conjunto de técnicas que sirven para mejorar y optimizar los procesos operativos de cualquier compañía, es una técnica de reducción de desperdicios.
[5] Tiempo de conducción o *lead time* es el tiempo que transcurre desde que se inicia un proceso de producción de un insumo terminado hasta que se completa, en ocasiones se debe tomar en cuenta el tiempo que le toma al producto llegar hasta el cliente e iniciarlo desde que se coloca la orden de compra.

preparación de los materiales antes de que pasen a los procesos de ensamble, pintura e instalación. Los primeros 3 días de proceso productivo tienen una configuración productiva tipo taller configurada por procesos o grupos tecnológicos, lo cual contrasta con el resto de la cadena productiva que se estructura mediante celdas de manufactura. La ilustración 1 que se presenta a continuación, muestra la estructura el sistema productivo actual:

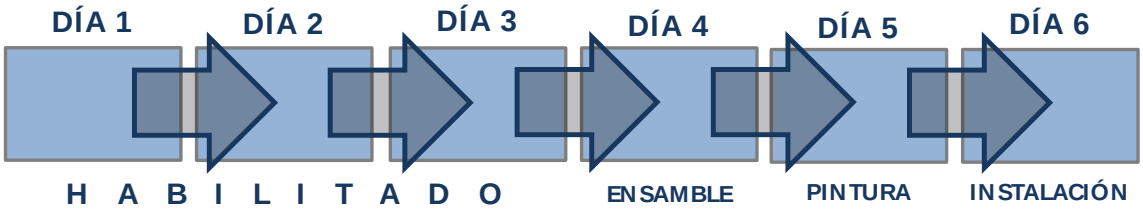


Ilustración 1. Modelo de Producción de Tecnomec Agrícola

El departamento de maquinado, que tiene una configuración en sus operaciones tipo taller, se caracteriza por tener una gran complejidad en la organización del trabajo debido a la enorme cantidad de combinaciones generadas durante cada periodo o programa de producción. Se pueden producir cerca 4000 componentes programándose en promedio 295 modelos distintos diariamente, los cuales pude ser procesados en 29 máquinas distintas.

Esta difícil tarea de secuenciar las operaciones, genera un problema de eficiencia en los equipos y ha despertado un enorme interés por la investigación de sistemas de programación dinámica que permitan calcular una secuencia de trabajos suficientemente buena como para reducir considerablemente el tiempo total de terminación de los trabajos y aumentar la disponibilidad de los equipos. Esta configuración de trabajo y la falta de un modelo de secuenciación en la empresa, ha generado un problema de retrasos consistentes en la entrega de partes a los procesos de fabricación posteriores. Es evidente que uno de los objetivos organizacionales encomendados por la Dirección de la compañía es: "*Aprovechar al máximo la capacidad de las máquinas en sus procesos automatizados o controlados numéricamente"* (ya que representan un costo hora-maquina mayor que resto de los procesos).

Este problema de retraso en producción se ha presentado recurrentemente desde hace más de 6 años y se tienen referencias del déficit en la entrega de partes desde hace 36 meses. Se han encontrado algunas soluciones parciales de forma empírica pero ninguna que evite que en este momento se tenga una situación de descontrol en las entregas de componentes a las líneas de ensamble que como se explicó representan el 4 día del proceso productivo. El efecto de este atraso en la entrega de partes hacia el cuarto día del proceso productivo, se mide diariamente con un control de resultados del departamento de maquinados. En la ilustración 2 se grafica la cantidad de piezas programadas diarias y la producción real diaria que percibimos el efecto de los equipos entregados a tiempo:

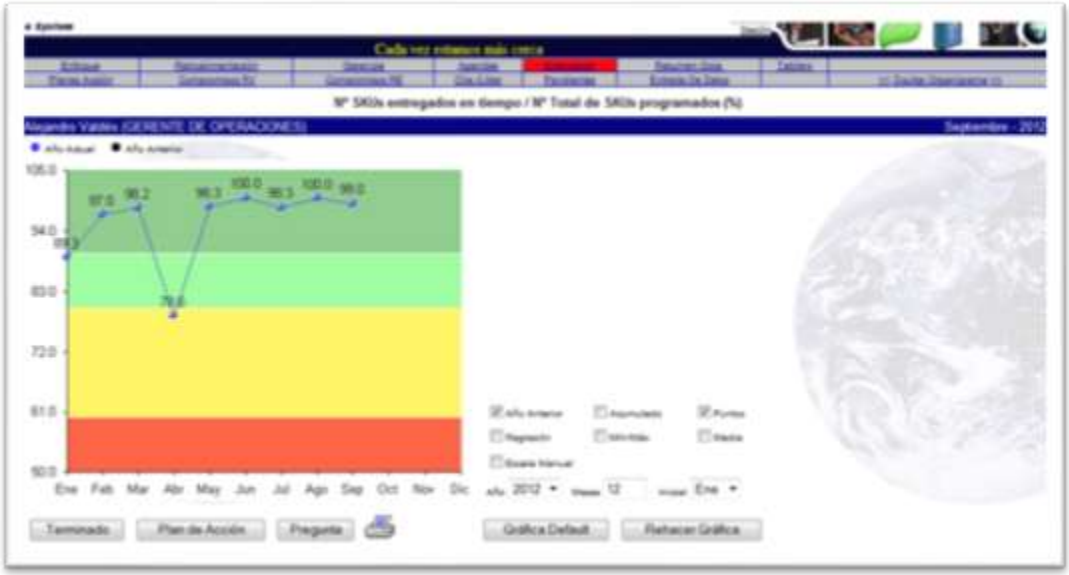


Ilustración 2. Indicador de entregas contra programación de productos

Algunas empresas no pueden adaptarse rápidamente a las nuevas tendencias de manufactura de clase mundial, incluso algunos procesos productivos presentan mayor resistencia a la transformación esbelta o simplemente algunas de las herramientas no son tan efectivas en todos los procesos. Tal es el caso de la configuración de producción tipo *Job-Shop* donde se producen lotes de piezas pequeños con variedades de producto relativamente altas, la programación de la producción resulta aún más compleja cuando se tienen que cumplir objetivos como calidad, costo y entrega, con restricciones y configuraciones en el proceso muy variadas. El área de maquinados en

Tecnomec Agrícola se clasifica como tipo taller o *Job-Shop*, por las características que ya se han expuesto, con la peculiaridad del sistema productivo con planeación de producción diaria.

Este reto de administración de la fabricación y el profundo interés de la compañía en resolver este problema han despertado un profundo interés de la Dirección de la empresa para la realización de este trabajo de investigación. El estudio también implica un desafiante nivel de investigación y aplicación de los conceptos de optimización combinatoria y los algoritmos genéticos los cuales forman parte esencial del sistema de solución del problema que se aborda y se profundiza en la sección 2.

2. PLANTEAMIENTO DEL PROBLEMA

Todos los procesos productivos tienen actividades que agregan valor *(AVA)* y otras que no lo hacen, las primeras son aquellas operaciones que realmente transforman el producto y que además el cliente está dispuesto a pagar por ellas. Por su parte las actividades que no agregan valor *(ANAV)* representan los desperdicios operacionales que además son generadores de costos para las organizaciones, lo anterior se resume de manera gráfica en la ilustración 3.

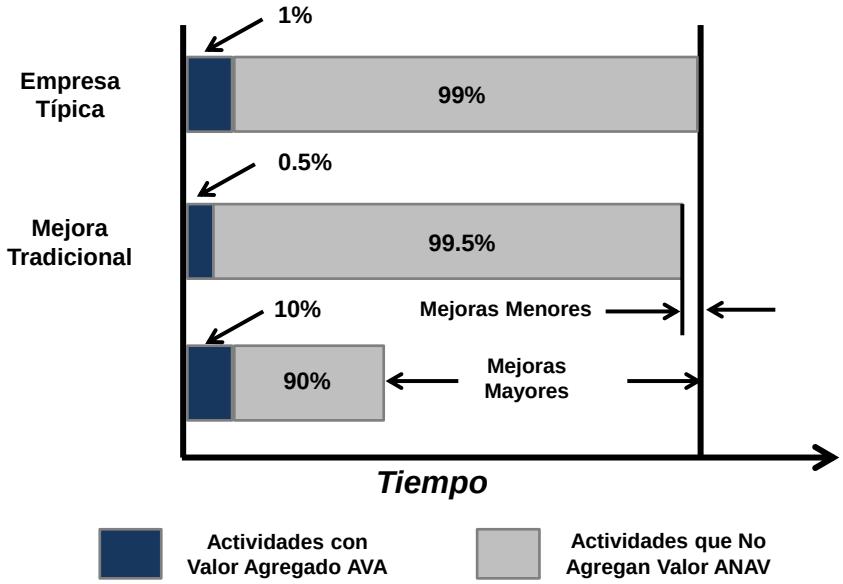


Ilustración 3. Impacto Generado al Reducir las Actividades que no Agregan Valor

El esquema productivo descrito en el apartado anterior en el área de maquinados más la tecnología utilizada en los procesos, pueden representar un costo muy alto que tiene que hacerse inherente al costo final del producto. Dentro del proceso productivo de un centro de maquinados de control numérico algunas de las actividades que no generan valor son; cambio de modelo, esperas por desconocimiento de programa productivo, desbalanceo de la carga de trabajo que implican esperas de los materiales para ser procesados.

Todas las dificultades anteriores además de ser actividades no generadoras de valor, son problemas a los que se enfrenta el departamento de maquinados durante cada periodo productivo. Finalmente a todo este conglomerado de dificultades se le puede

denominar *tiempo muerto no productivo*, y está estrechamente ligado a la falta de programación, organización y control del área, que se asocia a un *"problema de optimización de la programación de los trabajos".*

En los problemas de asignación de recursos, se dispone de un conjunto de operaciones, y para cada una de ellas se pueden utilizar un conjunto de máquinas que realizan la misma operación, pero pueden requerir tiempos distintos o costos de procesamiento diferentes, además no se conoce con anticipación que máquina se utilizará en cada operación, el problema radica en asignar las máquinas a tiempo para garantizar que se cumplan todas las demandas con el menor costo. En todos estos casos los trabajos tienen que pasar por ciertas estaciones donde *n* trabajos tienen que pasar por *m* máquinas bajo ciertas restricciones *t* tiempo y *c* costo (Sipper & Bulfin, 1997)[6]. Una forma simple de ejemplificar el problema en un sistema de 3 trabajos con 4 recursos se muestra en la ilustración 4.

La dimensión de una distribución tipo *Job-Shop Scheduling* como lo mencionamos es entonces *n x m*, las operaciones deben ser procesadas con el objetivo de minimizar el tiempo de finalización de los trabajos sujeto principalmente a dos restricciones (Adams, Balas, & Zawak, 1988)[7]:

1. La secuenciación de los trabajos (*n*) está definida (restricción conjuntiva). Esto se debe a la naturaleza de la transformación de los componentes según diseño.
2. Cada máquina (*m*) solo puede realizar una operación a la vez (restricción disyuntiva) y este proceso no se puede interrumpir (*non-preemptive scheduling*).

[6] Sipper, D., & Bulfin, R. L. (1997). *Planeación y Control de la Producción.* Mexico: McGraw Hill.

[7] Adams, J., Balas, E., & Zawak, D. (1988). The Shifting Bottleneck procedure for Job-Shop Scheduling. *Managment Science*.

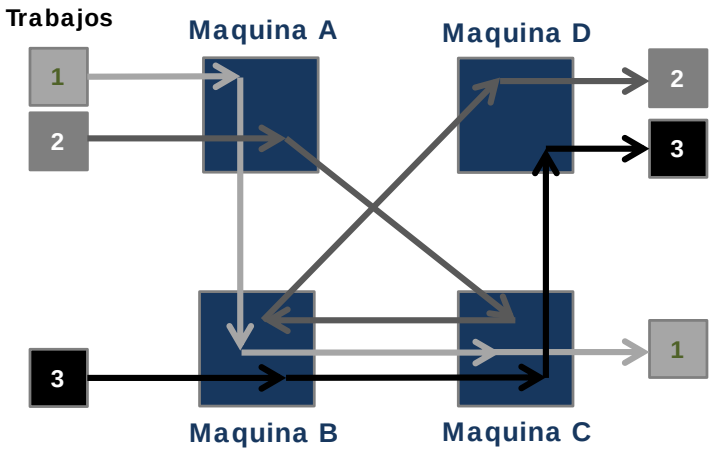


Ilustración 4. Distribución de trabajo en configuración tipo Job-Shop

La programación de la producción en Tecnomec se realiza diariamente a partir de los pedidos demandados por el cliente. Lo anterior genera un escenario de producción distinto regularmente. Lo anterior aunado a la enorme cantidad de configuraciones de producto que se tienen produce una gran cantidad y variedad de combinaciones todos los días.

Entonces, se dice que para el entorno productivo de maquinados en Tecnomec Agrícola se tiene un problema de secuenciación de operaciones de tamaño *295 x 30*, y de manera adicional la secuenciación debe ser dinámica, ya que se presentan distintas combinaciones de producción día con día. En el capítulo 3 de este documento se explicará a fondo el método de solución al problema, aun así es necesario explicar porque es útil buscar una solución a éste problema y cuáles son los beneficios que se obtienen con los resultados.

3. ESTADO DEL ARTE

Numerosas han sido las investigaciones que han utilizado la herramienta de Algoritmos Genéticos como método de optimización, y las variantes realizadas a este sistema han contribuido a perfeccionarlo a través de meta-heurísticos. En el *Job-Shop Scheduling*, las situaciones de programación de producción y por consecuencia la secuenciación de los trabajos son particulares para cada sistema, lo cual origina un sistema distinto y particular.

Recientemente se han publicado algunos trabajos que han contribuido como plataforma teórica de la presente investigación. Mohammad Ashshabi (Akhshabi & Akhshabi, Solving flexible Job-Shop scheduling problem using clonal selection algorithm, 2011), en su investigación *Solving flexible job-shop scheduling problem using clonal selection algorithm*[8], propone un sistema muy influenciado por el factor de clonación y contrasta sus resultados con un algoritmo anterior llamado *Lingo8*, el tiempo de búsqueda de una solución aproximada suficientemente buena es mucho menor y resuelve los problemas de tipo FJSP. El mismo autor (Akhshabi, Parallel Genetic Algorithm to solving Job-Shop Scheduling Problem, 2011) en su texto *Parallel Genetic Algorithm to Solving Job-Shop Scheduling Problem*[9], realiza trabajos con dimensiones hasta *100 x 20* y encuentra resultados satisfactorios en su afán de resolver el problema *Job-Shop Scheduling*.

Jianguo Jiang (Jiang & Wen, 2011), utiliza cromosomas híbridos para mejorar los resultados de la función objetivo, ha utilizado el sistema en entornos multi-objetivos con producción flexible en su investigación *Hybrid Genetic Algorithm for Flexible Job-shop Scheduling with Multi-objective*[10].

También se destaca la rapidez para encontrar la solución en el sistema en el documento *A Fast Algorithm for Solving JSSP*[11] de R. Murugesan (Murugesan & Navaneetha Kumar, 2011). El autor utiliza un sistema artificial inmune, utiliza algunos

[8] Akhshabi, M., & Akhshabi, M. (2011). Solving flexible jobshop scheduling problem using clonal selection algorithm. *Indian journal of Science and Tecnology*.
[9] Akhshabi, M. (2011). Parallel Genetic Algorithm to solving Job-Shop Scheduling Problem. *Journal of basic and applied scientific research*, 1484-1497.
[10] Jiang, J., & Wen, M. (2011). Hybrid Genetic Algorithm for flexible JobShop Scheduling with multiobjective . *Journal of information & computational science*, 2197-2205.
[11] Murugesan, R., & Navaneetha Kumar, V. (2011). A Fast Algorithm for solving JSSP. *European Journal of scientific research*, 579-586.

problemas de benchmarking para afirmar que la velocidad del algoritmo supera los métodos tradicionales.

Luego de tocar base en el estado del arte que se ha presentado y apreciar que los sistemas sugeridos no contemplan un sistema de producción ultra flexible con planeación de producción diaria como el que existe en Tecnomec Agrícola. Se tienen algunas referencias muy cercanas que facilitan la parte teórica de la investigación pero que tendrán sus diferencias durante la investigación, principalmente la idea de tener un sistema eficiente que se enfoque en la reducción de tiempos de *setup*. Éste método o procedimiento de investigación se amplía en la sección 4 de esta investigación.

4. MARCO TEÓRICO

Dentro del ámbito de la optimización combinatoria existen varios métodos para encontrar la solución a un objetivo, uno de ellos son los algoritmos heurísticos, su nombre proviene de la palabra griega *heuriskein* que significa encontrar o descubrir, esta forma de solución a un problema es usada recurrentemente (Marti, 2002)[12]. En un contexto científico se puede decir que la optimización es tratar de encontrar la mejor solución a determinado problema conocido.

A diferencia de los sistemas determinísticos, los heurísticos no requieren consumir gran cantidad de recursos para encontrar una solución que pudiese ser satisfactoria o útil para el problema que se quiere resolver. Estos problemas se pueden resumir como aquellos a los que se debe encontrar el valor de cierta variable donde la función objetivo alcanza su valor máximo o mínimo en un determinado número de intentos o iteraciones del método. Martí (Marti, 2002), también brinda una definición adicional para algunos algoritmos heurísticos de acuerdo a su complejidad, así la condición de un problema "difícil de resolver" se identifica con el término científico *NP-Hard* utilizado para definir su complejidad algorítmica, para este tipo de problemas no podemos garantizar el encontrar la mejor solución posible en un tiempo razonable.

La representación gráfica clásica de este tipo de problemas la define recientemente (Lopez de Haro, 2004)[13] en su publicación secuenciación de tareas mediante metaheurísticos. La idea puede ser representada bajo los siguientes términos:

En un conjunto G = {*N, A, E*} donde:

- **N,** Es el conjunto de nodos que representa a las operaciones, existen 2 nodos ficticios en cada operación, una fuente y un sumidero (inicio-fin), el peso positivo para cada nodo representa el tiempo de procesamiento d_{ij}, de la operación de la máquina *i* para el trabajo *j*. El nodo inicial se conecta con la primera operación de

[12]. Marti, R. (2002). Procedimientos Metaheuristicos en Optimización Combinatoria. *Departamento de Estadistica e Investigación Operativa. Universidad de Valencia*.
[13] Lopez de Haro, S. (2004). Secuanciación de Tareas Mediante Metaheuristicos. *Congreso de Ingenieria de Organización*.

cada trabajo, mediante un arco cuya longitud es su tiempo de liberación (r_i) y de igual forma la última operación de cada trabajo se conecta con el nodo final. En esta representación el nodo inicial es 0 y el final es *n*.

- **A,** Es el conjunto de arcos conjuntivos que representan las restricciones de precedencia entre las operaciones de cada trabajo

- **E,** Es el conjunto de arcos dirigidos que representan las restricciones de capacidad, arcos disyuntivos.

La ilustración 5 muestra la relación de los arcos y conjuntos de nodos de forma gráfica a través de un esquema grafos:

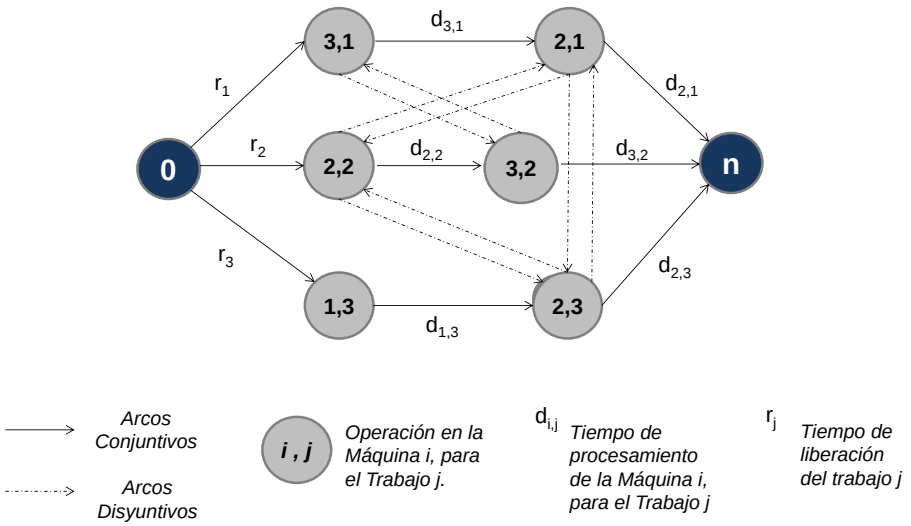


Ilustración 5. Esquema Grafos para un problema de asignación de trabajos en máquinas

Usando la teoría de grafos y las restricciones 1 y 2 algunos problemas de *Job-Shop Scheduling* se pueden definir matemáticamente de la siguiente manera:

$$\min t_n$$

$$t_{li} - t_{ki} \geq d_{ki} \quad donde \quad i,j \in A, \quad k,l \in M \tag{1}$$

$$t_{kj} - t_{ki} \geq d_{ki} \ \vee \ t_{ki} - t_{kj} \geq d_{kj} \quad donde \quad i,j \in E_k, k \in M \tag{2}$$

$$t_{kj} \geq r_j \quad donde \quad j \in N, k \in M \qquad (3)$$

$donde$:

t_n: $Tiempo\ de\ inicio\ de\ la\ operación\ n\ (final)\ o\ makespan$

d_{ki}: $Duración\ en\ la\ máquina\ k\ del\ trabajo\ i$

d_{kj}: $Duración\ en\ la\ máquina\ k\ del\ trabajo\ j$

t_{li}: $Tiempo\ de\ inicio\ en\ la\ máquina\ l\ del\ trabajo\ i$

t_{ki}: $Tiempo\ de\ inicio\ en\ la\ máquina\ k\ del\ trabajo\ i$

t_{kj}: $Tiempo\ de\ inicio\ en\ la\ máquina\ k\ del\ trabajo\ j$

A: $Conjunto\ de\ operaciones\ de\ un\ trabajo$

E_k: $Conjunto\ de\ operaciones\ a\ realizar\ en\ la\ máquina\ k$

M: $Conjunto\ de\ máquinas$

N: $Conjunto\ de\ trabajos$

P: $Cualquier\ solución\ factible$

r_j: $Tiempo\ de\ liberación\ del\ trabajo\ j$

La primera restricción representa a los arcos conjuntivos, mientras que la segunda a los disyuntivos, finalmente la última restricción indica que todos los trabajos tienen un tiempo de inicio mayor o igual que el tiempo de liberación del trabajo.

4.1 REPRESENTACIÓN DE UN MODELO DE I MAQUINAS CON J TRABAJOS

Una forma simple de representar un modelo de distribución de los trabajos hecho por José Salazar González (Salazar, 2001) en su texto *Programación matemática*[14] es considerar un conjunto *I* de máquinas y un conjunto *J* de trabajos a realizar. Para cada trabajo $j \in J$ se conoce la dedicación total d_j que demanda, cualquiera que sea $i \in I$.

El modelo matemático se representa para cada $(i, j) \in IXJ$ sea la variable continua x_{ij} que determina la dedicación de la máquina *i* sobre el trabajo *j*. Mediante la ayuda de una variable adicional, es posible formular el siguiente modelo:

[14] Salazar, G. J. (2001). *Programacion Matemática.* Madrid: Diaz de Santos.

$$\min\ t$$

Sujeto a:

$$\sum_{j \in J} x_{ij} = d_i \quad para\ todo\ i \in I$$

$$\sum_{j \in J} p_{ij} x_{ij} \leq t \quad para\ todo\ i \in I$$

$$\sum_{i \in I} p_{ij} x_{ij} \leq t \quad para\ todo\ j \in J$$

$$x_{ij} \geq 0 \quad para\ todo\ j \in J\ y\ todo\ i \in I$$

4.2 REPRESENTACIÓN DE UN MODELO PARA OPTIMIZAR UNA ASIGNACIÓN

En algunos problemas similares de asignación de recursos que sugiere José Salazar (Salazar, 2001), se ejemplifica con un problema de distribución de 7 asignaturas universitarias a 7 profesores donde además se busca que asignando una asignatura a cada profesor se encuentre el mínimo de deserción (costo) escolar según datos históricos.

Ahí si se consideran las variables decisionales:

$$x_{ij} := \begin{cases} 1 & si\ el\ profesor\ i\ imparte\ al\ curso\ j \\ 0 & otro\ caso \end{cases}$$

Se supone que c_{ij} representa el costo de que el profesor *i* imparta la asignatura *j* según la relación anterior. Entonces, un modelo matemático viene dado por:

$$min \sum_{i=1}^{7} \sum_{j=1}^{7} c_{ij} x_{ij}$$

Sujeto a:

$$\sum_{j=1}^{7} x_{ij} = 1 \quad para\ todo\ i = 1, \dots, 7$$

$$\sum_{i=1}^{7} x_{ij} = 1 \quad para\ todo\ j = 1, \dots, 7$$

$$x_{ij} \geq 0 \quad para\ todo\ i = 1, \dots, 7\ y\ j = 1, \dots, 7$$

$$x_{ij} \in Z \quad para\ todo\ i = 1, \dots, 7\ y\ j = 1, \dots, 7$$

Un problema de secuenciación en un entorno Job-Shop tiene *n* soluciones posibles (no necesariamente todas factibles) según Sipper (Sipper & Bulfin, 1997), y es considerado NP-Hard, característica que ha hecho que aún no se haya desarrollado un método que pueda optimizar una solución. Por esta razón en este tipo de problemas se utilizan algoritmos heurísticos con el fin de encontrar soluciones satisfactorias en cuanto a tiempo y ejecución.

La investigación en teoría de secuenciación o programación de los trabajos se ha desarrollado en la práctica desde hace ya más de 40 años, desde simples reglas de "despacho" hasta heurísticos sofisticados con algoritmos genéticos o de ramificación y poda.

4.3 MÉTODOS DE SOLUCIÓN

Diversas opiniones existen respecto al origen del *Job-Shop Scheduling Problem*. Saul I. Gass (Gass, 2005)[15], cita a B. Roy y B. Sussman en 1964 quienes fueron los primeros en proponer la representación mediante el grafo disyuntivo y a su vez comenta que Egon Balas en 1959 fue el primero en aplicar un acercamiento enumerativo basado en este grafo. Este autor asegura que existen trabajos anteriores: B. Gier y G. L. Thompson propusieron en 1960 un algoritmo de reglas de despacho de prioridad, J. R. Jackson en 1956 generalizó el algoritmo del *flow-shop* de S. M. Johnson de 1954 y en 1955, S. B. Akers y J. Friedman aplicaron un modelo de algebra booleana para representar secuencias de procesamiento. Estas investigaciones trataron con un

[15] Gass, S. I. (2005). *An annotated timeline of operations research.* USA: Kluwer Academic Publishers.

problema consistente de *n* trabajos, *m* máquinas y relaciones de precedencia, por lo cual cada trabajo se procesa a través de las máquinas en un orden diferente.

Las técnicas de resolución del problema tipo *Job-Shop* se pueden dividir en dos categorías: las técnicas de optimización, que producen una solución globalmente óptima, pero requiere un tiempo de computación muy alto, y las técnicas de aproximación, que proporcionan una buena solución en un tiempo aceptable. A continuación se describen las técnicas más importantes.

4.3.1 Técnicas de Optimización

En las técnicas de optimización, en general, la complejidad temporal crece exponencialmente con respecto al tamaño de la entrada. Los métodos obtenidos construyen una solución óptima a partir de los datos del problema siguiendo una serie de reglas que determinan de manera exacta el orden de procesamiento.

En muchas de las ocasiones, cuando se tiene un problema de optimización el primer intento de solución surge de técnicas de aproximación debido a su rapidez en encontrar soluciones satisfactorias, en la industria el tiempo de procesamiento para una solución es crítico y comúnmente no es conveniente buscar soluciones exactas. En las próximas secciones se explican algunas de las técnicas de aproximación más comunes.

4.3.2 Programación lineal entera mixta

Propuesto inicialmente en 1960 por Alan Mane (Mane, 1960)[16], este es un procedimiento muy utilizado en otros contextos de la programación matemática, sin embargo no resulta apropiado en este ámbito ya que su formulación da lugar a tiempos de ejecución elevados en problemas de tamaño real, lo cual la hace poco aconsejable.

[16] Mane, A. (1960). On the job-shop scheduling problem. *Yale University*, 40-42.

4.3.3 Ramificación y poda

Las técnicas de ramificación y poda hacen uso de una estructura de árbol construida dinámicamente, como forma de representar el espacio de todas las secuencias posibles. La búsqueda comienza en el nodo raíz y se continúa hasta llegar a un nodo hoja. Cada nodo en un nivel *p* de la búsqueda representa una secuencia parcial de *p* operaciones.

Desde un nodo no seleccionado, la operación de ramificación determina el siguiente conjunto de posibles nodos a partir del cual puede progresar la búsqueda. Las dos estrategias de ramificación más comunes son la Generación de Secuencias Activas (*Generating Active Schedules: GAS*), y Ajuste de Conflictos Secuenciales (*Settling Essential Conflicts: SEC*). Cada nodo consiste en un secuenciamiento parcial de las operaciones sobre los recursos, pero mientras que la primera estrategia de ramificación fija el conjunto de operaciones siguientes a secuenciar, en la segunda se determina si una operación debería secuenciarse antes o después de otra.

El procedimiento de poda selecciona la operación con la que continuará la búsqueda y se basa en una estimación de una cota inferior y la mejor cota superior alcanzada hasta el momento. En los métodos de ramificación y poda es esencial afinar las cotas, ya que eso previene la necesidad de buscar en secciones grandes del espacio de soluciones. Los investigadores Naendra y Fukunaga (Naendra & Fukunaga, 1977) desarrollaron a finales de los años setenta un modelo de búsqueda para *n* operaciones en *m* recursos, con la posibilidad de localizar soluciones óptimas en tiempos de computo muy cortos, en su texto *A Branch and bound algorithm for feature subset selection*[17], evalúan un experimento de este tipo donde se trabaja sobre 6000 combinaciones de un universo que originalmente tendría 2’700,000 posibles soluciones.

[17] Naendra, P. M., & Fukunaga, K. A. (1977). A branch and bound algorithm for feature subset selection. *Honeywell systems and research center*, 34-36.

4.3.4 Algoritmos genéticos

Los algoritmos genéticos imitan el procedimiento de la selección natural sobre el espacio de soluciones del problema considerado. Se basan en la creación de generaciones sucesivas de individuos representativos de posibles soluciones al problema. Los nuevos individuos se generan cruzando parejas seleccionadas dando mayor probabilidad a aquellas soluciones que mejor valor de la función objetivo han obtenido en la generación anterior (individuos más adaptados al entorno).

En la literatura existen numerosas propuestas que resuelven el *Job-Shop Problem* mediante el uso de Algoritmos Evolutivos. De acuerdo con este esquema un cromosoma es en principio una permutación del conjunto de tareas en el que cada tarea viene representada simplemente por el número del trabajo al que pertenece. Por ejemplo, si tenemos un problema con 3 trabajos de 3 tareas cada uno, un cromosoma puede ser el siguiente (1 3 2 2 3 1 1 3 2). En esta codificación el primer 1 representa la primera tarea del primer trabajo, el segundo cromosoma a la segunda tarea y así sucesivamente. De este modo las tareas de cada trabajo aparecen representadas en el orden de ejecución secuencial, mientras que las tareas que requieren al mismo recurso pueden aparecer en cualquier orden. La ventaja principal que presenta este esquema de codificación es que permite diseñar operadores genéticos eficientes, de cruce y mutación, que producen cromosomas factibles. Un buen ejemplo de este tipo de sistema de solución lo presenta Kobayashi (Kobayashi, 1995) en el texto *An Efficient Genetic Algorithm for Job-Shop Scheduling Problems*[18]*,* aquí se presenta la solución a un problema de programación de secuencias de trabajo en un sistema de producción tipo taller.

De igual forma en *A Hybrid Genetic Algorithm for the Job-Shop Scheduling Problem*[19], de Goncalves (Goncalves, 2002), se presenta una solución al problema de producción tipo *Job-Shop*, donde además se refinan los resultados aplicando un algoritmo extra de búsqueda local.

[18] Kobayashi, S. (1995). An Efficient Genetic Algorithm for Job-Shop Scheduling Problem. *Proc. of ICGA'95*, 506-511.
[19] Goncalves, J. F. (2002). A Hybrid Genetic Algorithm for the Job-Shop Scheduling Problem. *AT&T Labs Research Technical Report*.

4.3.5 Reglas de prioridad de despacho

Las reglas de prioridad son probablemente las reglas heurísticas aplicadas con mayor frecuencia para resolver problemas de secuenciación, esto se debe a su fácil implementación y a su baja complejidad de tiempo. Los algoritmos de Giffler y Thompson se pueden considerar como la base de todas las heurísticas basadas en reglas de prioridad.

Giffler y Thompson han propuesto dos algoritmos para generar secuencias: procedimientos de generación de secuencias activas y secuencias *nondelay*[20]. Una secuencia activa tiene la propiedad de que ninguna operación se puede iniciar tempranamente sin retrasar otro trabajo.

Las secuencias activas forman un conjunto más grande que incluye como subconjunto a las secuencias *nondelay*. El procedimiento de generación de Giffler y Thompson explora el espacio de búsqueda por medio de una estructura de árbol. Los nodos en el árbol representan las secuencias parciales, los arcos representan las posibles elecciones, y las hojas del árbol son el conjunto de secuencias. Para una secuencia parcial, el algoritmo esencialmente identifica todos los conflictos de procesamiento, es decir, identifica operaciones que compiten por la misma máquina, y en cada etapa usa un procedimiento de enumeración para resolver esos conflictos bajo todas las formas posibles. En contraste, las heurísticas resuelven esos conflictos con reglas de prioridad de despacho; las heurísticas especifican una regla de prioridad para seleccionar una operación entre las operaciones en conflicto. Interesante es el texto de los hindúes Natarajan, Mohanasundaram, y Shoban (Natarajan, Mohanasundaram, & Shogan Babu, 2007) *Performance evaluation of priority dispatching rules in multi-level assembly Job - Shops with jobs having weights for flowtime and tardiness*[21], donde se analizan los problemas de tardanza en sistemas de producción tipo *Job-Shop* basado en estructuras de árbol multinivel.

[20] Una secuencia nondelay tiene la propiedad de que ninguna máquina permanece ociosa si hay un trabajo disponible para su procesamiento.
[21] Natarajan, K., Mohanasundaram, K. M., & Shogan Babu, B. (2007). Performance evaluation of priority dispatching rules in multi-level assembly Job-Shops with jobs having weights for flowtime and tardiness. *Int J Manuf Technol*, 751-761.

4.3.6 Recocido simulado

La denominación del algoritmo, Recocido Simulado, o *Simulated Annealing*, proviene de la estrecha analogía que guarda con el proceso del recocido tal y como se usa en metalurgia. El método fue desarrollado por Daniel Gelatt & Mario Vecchi en 1983. Este consiste en que un metal fundido se va enfriando lentamente de manera que sus moléculas van adoptando poco a poco una configuración de mínima energía.

Cuando comienza el proceso, a alta temperatura, las moléculas vibran y se desplazan caóticamente adoptando todo tipo de configuraciones en la estructura del metal de la que forman parte. A medida que la temperatura disminuye se va ralentizando el movimiento de las moléculas y éstas, de acuerdo con la Termodinámica, tienden a adoptar paulatinamente las configuraciones de menor energía, siendo ésta nula en el cero absoluto.

El recocido Simulado intenta realizar numéricamente un proceso análogo al del recocido metalúrgico. El espacio de configuraciones no vendrá ya dado por las posiciones de las moléculas, sino por los valores de una variable de interés, que en el caso del *Job-Shop* será la secuencia de tareas que se desean procesar, mientras que el papel de la energía lo asumirá la función que se intenta minimizar, costo o tiempo de realización de tareas, por ejemplo. A decir verdad, muchos de las aplicaciones donde se ha utilizado este método datan de finales de los ochenta y principios de los noventa. Tal vez la complejidad con la que trabaja el modelo ha limitado su uso actualmente.

Actualmente en Alemania se ha desarrollado una investigación denominada *Tuning Simulated annealing Using the Sequential Parameter Optimization Tool Box (SPOT)*[22] por Thomas Bartz-Beielstein (Bartz-Beielstein, 2010), donde utiliza algunos métodos basados en análisis de regresión lineal clásicos y técnicas de análisis de varianza.

[22] Bartz-Beielstein, T. (2010). Tuning Simulated Annealing Using the Sequential Parameter optimization Toolbox. *Cologne University of applied Sciences*.

4.3.7 Heurísticas basadas en cuellos de botella

Si bien durante bastante tiempo los únicos métodos de aproximación viables han sido las reglas de prioridad de despacho, recientemente con la llegada de computadoras más potentes, así como en el énfasis puesto en nuevas técnicas cuidadosamente analizadas, diseñadas e implementadas, ha dado lugar a aproximaciones más sofisticadas, llamadas heurísticas basadas en cuellos de botella. Adams, Balas y Zawack (Adams, Balas, & Zawak, 1988), desarrollaron este tipo de modelos.

Un ejemplo de esta aproximación es el *Shifting bottleneck procedure (SBP)*. Un SBP se caracteriza por las siguientes tareas: identificación del subproblema, selección del cuello de botella, solución del subproblema, y reoptimización de la secuenciación. La estrategia real implica relajar el problema de secuenciación *n x m*, en *m* problemas de una máquina y resolver cada subproblema cada vez de forma iterativa usando la aproximación de Carlier. Cada una de las soluciones se compara con las demás y se ordenan las máquinas según su solución. La máquina no secuenciada que tiene la solución mayor se identifica como la máquina que provoca el cuello de botella. El SBP secuencia dicha máquina basándose en las que ya han sido calculadas, ignorando el resto de máquinas no secuenciadas. La selección de la máquina que constituye el cuello de botella viene motivada por la conjetura de que la secuenciación en etapas posteriores podría deteriorar el tiempo de realización de la programación (makespan).

Cada vez que una máquina se identifica como de cuello de botella, todas las máquinas que ya han sido secuenciadas, y que son susceptibles de mejoras, se reoptimizan localmente resolviendo el problema de una máquina de nuevo. La principal contribución de esta aproximación es la forma en que se usa la relajación de una máquina para decidir el orden en el que las máquinas deben ser secuenciadas. En *Programación de la producción en sistemas de manufactura tipo taller con algoritmo combinado cuello de botella móvil y búsqueda tabú*[23] de los colombianos Brito, Mejía y Caballero (Britto Agudelo, Mejia Delgadillo, & Caballero Villalobos, 2007) se utiliza la heurística del cuello de botella móvil y la búsqueda tabú, para reducir la tardanza ponderada total de los

[23] Britto Agudelo, R. A., Mejia Delgadillo, G., & Caballero Villalobos, J. P. (2007). Programación de la producción en sistemas de manufactura tipo taller con algoritmo combinado cuello de botella movil y busqueda tabu. *Universidad de Bogotá*, 203-221.

trabajos, ellos aplican dichas técnicas en alguno ejemplos clásicos de la literatura de optimización combinatoria.

4.3.8 Métodos de búsqueda local

Este procedimiento está basado en la suposición de que es posible encontrar una secuencia de soluciones entre la solución inicial y la final tal que cada una de ellas es ligeramente diferente a la inmediatamente anterior. Tiene la ventaja de que en poco tiempo puede encontrar soluciones suficientemente buenas para un conjunto amplio de problemas combinatoriales. Este procedimiento puede ofrecer una medida de la bondad de la solución encontrada, pero no garantiza que la solución obtenida sea el óptimo global del problema considerado.

Un concepto asociado a la búsqueda local es la descripción del conjunto de soluciones parecidas a una dada, denominado “vecindario". Dos soluciones son “vecinas" si una puede ser obtenida a partir de la otra intercambiando el orden entre dos tareas consecutivas en un mismo recurso perteneciendo ambas al camino crítico.

Una estrategia de búsqueda local bastante utilizada es la denominada “avariciosa", la cual parte de una solución inicial generada aleatoriamente o mediante un algoritmo heurístico y evalúa todas las soluciones que se encuentran en su vecindario, escogiendo de entre las que mejoren el valor de la función objetivo la más prometedora. La repetición de este proceso terminaría en un óptimo, local o global, pero en todo caso una solución mejor que cualquiera de las de su vecindario y lógicamente mejor que la solución inicial. Olli Bräysy y Michel Gendrau (Braysy & Gendreau, 2005) en el texto *Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search algorithm*[24]*,* trabajan con aproximaciones de las mejores rutas vehiculares con combinaciones entre búsqueda local.

[24] Braysy, O., & Gendreau, M. (2005). Vehicle routing problem with time windows. Part 1. Route construction and local search algorithms. *Transportation Science. Universidad de Montreal.*

4.3.9 El metaheurístico *Particle Swarm Optimization*

Es un método computacional que resuelve un problema iterativamente tratando de mejorar una solución de candidato con respecto a una determinada medida de calidad. El sistema optimiza un problema por tener una población de soluciones candidatas y se guían por las soluciones encontradas en el espacio que se actualizan a medida que se encuentran mejores posiciones por partículas. El proceso se atribuye a Kennedy y Eberhart en 1995. El trabajo Tasgerintein F., Sevklik M., y Yenisey M. (Tasgerintein, Sevlik, & Yenisey, 2006) en *A particle swarm optimization and differential evolution algorithms for Job-Shop scheduling problem*[25], trabaja con un sistema de optimización de la tardanza de los trabajos para un sistema de producción tipo *Job-Shop*.

4.3.10 Búsqueda tabú

Es una técnica de optimización iterativa global, que consiste en un procedimiento determinístico que restringe la búsqueda y evita los mínimos locales, almacenando la historia de búsqueda en memoria. Se prohíben movimientos entre vecinos que cumplan ciertas propiedades, con objeto de guiar el proceso de búsqueda para no duplicar soluciones previamente obtenidas. Una función de memoria a corto plazo permite “olvidos estratégicos" convirtiendo en “prohibido" los *t* movimientos más recientes. Sin embargo, el estado de un movimiento no es absoluto, ya que es posible seleccionar un movimiento tabú si alcanza un determinado nivel de calidad. También se dispone de funciones de memoria a largo plazo que pueden aplicarse para proporcionar una exploración más amplia del espacio de búsqueda. Finalmente existen estrategias a medio plazo o intermedias basadas en la modificación de las reglas de elección que favorecen la elección de movimientos y soluciones consideradas buenas históricamente, de forma que crean zonas de atracción del dominio de búsqueda e intensifican la búsqueda en dichas regiones. Los métodos a largo plazo diversifican la búsqueda en áreas no exploradas previamente.

[25] Tasgerintein, F., Sevlik, L., & Yenisey, M. (2006). A particle swarm optimization and differential evolution algorithms for Job-Shop scheduling problem. *International journal of operational research*.

El algoritmo de búsqueda tabú aplicada al *Job-Shop* genera vecinos mediante la inversión de arcos que unen tareas adyacentes en el camino crítico. Después de que un arco ($v \rightarrow w$) ha sido invertido, se introduce el paso inverso ($w \rightarrow v$) en la lista tabú, consistente en una lista de longitud definida que contiene los desplazamientos prohibidos.

Cada vez que se realiza un número *n* de iteraciones, se modifica aleatoriamente la longitud de la lista tabú. En cada iteración se evalúan todas las soluciones del vecindario que no se encuentran en la lista tabú, calculando para cada una el valor de *Cmax* y seleccionándose la más prometedora. La búsqueda tabú tiene varias investigaciones recientes, como *The Granular Tabu Search and Its Application to the Vehicle-Routing Problem*[26] de Paolo Toth y Daniele Vigo (Toth & Vigo, 2003), el cual se basa en las relaciones vecinales de las soluciones en un problema de ruteo vehicular con restricciones de distancias. Una reciente contribución de los colombianos Brito, Mejía y Caballero (Britto Agudelo, Mejia Delgadillo, & Caballero Villalobos, 2007), se basa en la combinación de los algoritmos de búsqueda tabú y el cuello de botella móvil que analizaremos más adelante, en su trabajo *Programación de la producción en sistemas de manufactura tipo taller con algoritmo combinado cuello de botella móvil y búsqueda tabú*[27]*,* los autores buscan minimizar la tardanza ponderada total del proceso, además se trabajó con algunos problemas clásicos de la literatura y se realizaron algunas mejoras al ya conocido algoritmo cuello de botella móvil.

Un trabajo clásico de aplicación de algoritmos de búsqueda tabú en distribuciones de trabajo tipo *job-shop*, fue el que desarrolló en los años noventa el suizo Eric Taillard (Taillard, 1993) en su texto *Parallel Taboo Search Techniques for the Job-Shop Scheduling Problem*[28]*,* el investigador asegura que el método tabú tiene ventajas claras sobre el algoritmo cuello de botella móvil y el recocido simulado, principalmente en la rapidez del cómputo y la simplificación del algoritmo.

[26] Toth, P., & Vigo, D. (2003). The granulat tabu search and it's placation to the vehicle-routing problem. *Informs journal of computing. Universidad de Bologna*.

[27] Britto Agudelo, R. A., Mejia Delgadillo, G., & Caballero Villalobos, J. P. (2007). Programación de la producción en sistemas de manufactura tipo taller con algoritmo combinado cuello de botella movil y busqueda tabu. *Universidad de Bogotá*, 203-221.

[28] Taillard, E. (1993). Parallel taboo search for the job-shop scheduling problem. ORSA Journal of computing. *Escuela Politécnica de Lausana*.

Existen fuertes razones por las que se ha escogido el heurístico de Algoritmos Genéticos como método de solución al problema de secuenciación de trabajos en Tecnomec Agrícola. Este fundamento que cimenta el desarrollo del presenta trabajo se presenta al final de esta sección siguiente, inicialmente se presentara el marco teórico fundamental de este sistema de optimización.

4.4 ALGORITMOS GENÉTICOS

Según Lourdes Araujo (Araujo & Cervigon, 2009) en su publicación *Algoritmos Evolutivos, un enfoque práctico*, "Los algoritmos genéticos suelen ser de los más estudiados por su eficiencia y sencillez de implementación, además de la existencia de numerosos estudios teóricos de los mecanismos subyacentes al funcionamiento de estos algoritmos para la resolución de problemas"[29].

4.4.1 Principios de los Algoritmos Genéticos (AG)

Como se ha mencionado, los Algoritmos Evolutivos constituyen una técnica de búsqueda y optimización de soluciones. Este método trabaja con una colección de posibles soluciones candidatas o individuos, para los que se calcula una medida de su adaptación o capacidad de ser solución al problema a resolver. La composición de la población cambia con un proceso iterativo donde cada iteración se denomina generación. Tal como ocurre en la naturaleza, con individuos con mayor adaptación tienen una probabilidad mayor de sobrevivir y permanecer en la población de la siguiente generación.

Araujo (Araujo & Cervigon, 2009), argumenta que los algoritmos evolutivos permiten abordar problemas complejos de búsqueda que surgen en las ingenierías y los campos científicos: planificación de tareas, horarios, tráfico, búsqueda de caminos óptimos, optimización de funciones, etc.

[29] Araujo, L., & Cervigon, C. (2009). *Algoritmos Evolutivos. Un enfoque practico.* Mexico: Alfaomega.

Los Algoritmos Genéticos (AG), se caracterizan por representar las soluciones al problema que se abordan en forma de cadena de bits, un algoritmo genético responde en general a este tipo de esquema:

```
%función algoritmo genético
TPoblacion pob;   %población
TParametros Parámetros  %tamaño población
obtener_parametros(parámetros);
pob = población_inicial( );
evaluación(pob, tam_pob, pos_mejor, sumadaptacion);
%bucle de evolución
Mientras no se alcanza la condición de terminación hacer {
        Selección(pob, parámetros);
        Reproducción(pob, parámetros);
        Evaluación(pob, parámetros, pos_mejor, sumadaptacion); }
Devolver pob[pos_mejor]
```

Existen algunos parámetros de funcionamiento indispensables en un Algoritmo Genético, como el tamaño de la población. En seguida se genera una población de individuos (todos ellos posibles soluciones del sistema), después la población de individuos se somete a un ciclo de evolución donde cada iteración incluye un proceso de selección que modifica la composición de la población, eliminando a ciertos individuos y reforzando la presencia de otros. Finalmente viene un proceso de reproducción que introduce nuevos individuos y una nueva evaluación es la que actualiza los datos de evolución tales como la adaptación media de la población o la posición del mejor individuo de la población.

4.4.2 Representación de los individuos

Como se hacía mención, los individuos son cadenas que denotaremos por *b* (genotipo), que representan a los puntos *x* (fenotipo) del espacio de búsqueda del problema. La nomenclatura biológica a veces se adopta también para otros datos del individuo. Así se usa gen para referirse a la codificación de una determinada característica del

individuo. En los algoritmos genéticos se suele identificar un gen con cada posición de la cadena. En el argot de este método se usa alelo para los distintos valores que puede tomar un gen y locus para referirse a determinada posición de la cadena binaria.

La sencillez de la representación anterior genera ventajas en este método, pero la contraparte aparece cuando es necesario disponer de un método para pasar de la representación binaria al espacio de búsqueda natural del problema. De ahí que para facilitar la codificación-decodificación de la cadena o cromosoma a la realidad en este trabajo se representa el orden de trabajo en cada máquina con un número que identifique al número de parte dentro del proceso.

Supongamos que se pretende procesar 5 trabajos distintos en 3 máquinas diferentes, la secuencia de los trabajos se discrimina. La representación del cromosoma se muestra en la siguiente ilustración:

Ilustración 6. Representación de un cromosoma en un problema 5 X 3

4.4.3 Generación de la población inicial

Los individuos de la población inicial de un AG suelen ser cadenas generadas de forma aleatoria, es decir, se va generado cada gen con una función asigna todas las posibles partes que se puedan procesas en una máquina con igual probabilidad. Si hay problemas donde se disponga de información adicional que nos permita saber con anticipación que determinadas cadenas tienen mayor probabilidad de llegar a ser solución, podemos favorecer su generación al crear la población inicial, pero cuidando que esta condición no genere soluciones u óptimos locales. Es importante destacar que este proceso debe ser completamente aleatorio y en el capítulo 3 se explicará el sistema utilizado durante la experimentación.

4.4.4 Grado de adaptación de los individuos

La evolución de la población depende de la calidad relativa de los individuos, en un problema de búsqueda u optimización. Dicha calidad se mide por la adecuación o adaptación de cada individuo a ser solución al problema (cumplir con la función objetivo). Es frecuente que los problemas se presenten como la optimización de una función matemática explícita. En dichos casos la función de adaptación coincide con la función a optimizar.

4.4.5 Condiciones de terminación

Es necesario especificar las condiciones en las que el algoritmo deja de evolucionar y se presenta la mejor solución encontrada. La condición de terminación más sencilla es alcanzar un número determinado de generaciones de evolución. Sin embargo también se pueden establecer parámetros de finalización cuando tengamos una solución suficientemente buena del sistema.

4.4.6 Proceso de selección: Mecanismo de muestreo

En la *selección proporcional o por ruleta*, la probabilidad de selección p_i de un individuo *i* con este método es proporcional a su adaptación relativa:

$$p_i = \frac{f(i)}{\bar{f}}$$

Siendo $\bar{f}$ la adaptación media de la población. Necesitamos generar un número aleatorio de acuerdo con la distribución de probabilidad dada por p_i.

- Definimos las puntuaciones acumuladas de la siguiente forma:

$$q_0 \coloneqq 0$$

$$q_i \coloneqq p_1 + \cdots + p_i \qquad (\forall i = 1, \dots n)$$

- Se genera un número aleatorio $a \in [0,1]$-

- Se selecciona al individuo *i* que cumpla: $q_{i-1} < a < q_i$

Este proceso se repite por cada individuo que se desee seleccionar.

Por su parte hay un sistema de selección denominado *muestreo estocástico universal* el cual es muy similar al muestreo por ruleta, pero en este caso se genera un solo número aleatorio y a partir de él se generan los *k* números que se necesitan (para generar *k* individuos) espaciados de igual forma. Los números se generan de la siguiente manera.

$$a_j \coloneqq \frac{a+j-1}{k} \quad (\forall j = 1, \dots, k)$$

Una vez generados estos números, el método funciona de la misma forma que la selección por ruleta, y es más eficiente que éste.

En el *método de selección por torneo*, se elige aleatoriamente una pequeña muestra de la población y de ella se selecciona el individuo de mejor valor de adaptación. Normalmente se utiliza un tamaño pequeño para la muestra. El proceso se repite hasta completar el número de individuos que se desee seleccionar, se puede hacer de forma probabilística o determinística.

Por último, también existe un método de *muestreo por restos*, el cual realiza una selección proporcional a la adaptación de los individuos garantizando copias de los mejores individuos sin intervención del azar. Concretamente de cada individuo x_i se seleccionan p_i*k copias de la muestra, siendo *k* el número de individuos a seleccionar y p_i la probabilidad de selección del individuo *i*. Los individuos que quedan hasta completar el tamaño de *k* de la muestra se seleccionan por alguno de los métodos anteriores.

4.4.7 Proceso de reproducción: Operadores genéticos

En cada nueva generación se crean algunos individuos que no estaban presentes en la población anterior. Los individuos se crean aplicando ciertos “operadores genéticos” a individuos de la población anterior. Los operadores que suelen estar presentes en todo algoritmo genético son los operadores de cruce y mutación.

El *operador de cruce* combina propiedades de dos individuos de la población anterior para crear nuevos individuos. El operador de mutación crea un nuevo individuo realizando algún tipo de alteración usualmente pequeña.

Para cada operador genético se establece una tasa o frecuencia, de manera que el operador sólo se aplica si un valor generado aleatoriamente está por encima de la tasa especificada. Por ejemplo, si la tasa de aplicación del operador de cruce es del 40%, entonces si al generar un número aleatorio entre 0 y 1 obtenemos 0.56, que es mayor que 0.4, aplicamos el operador de cruce a los dos individuos seleccionados a tal efecto.

El primer operador de cruce se denomina: *Operador de cruce monopunto*, es la forma más simple del operador de cruce en los algoritmos genéticos. Este consiste en seleccionar al azar una única posición en la cadena de ambos padres e intercambiar las partes de los padres divididas por dicha posición, como se muestra en la siguiente ilustración:

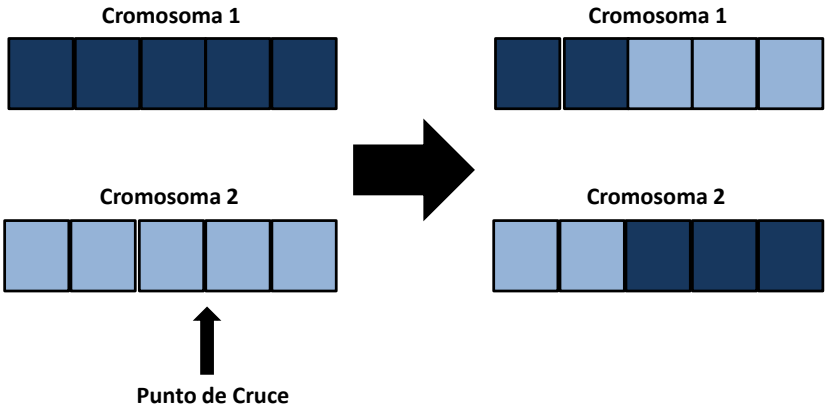


Ilustración 7. Operador de cruce Monopunto

Este operador produce dos hijos que combina propiedades de ambos padres, lo que puede llevar a una mejora de la adaptación de los hijos respecto a la de los padres.

El segundo tipo de cruce se llama; *Operación de mutación aleatoria bit a bit* y es la forma más sencilla de mutación que se puede aplicar a un individuo de un algoritmo genético consiste en cambiar el valor de una de las posiciones de la cadena: si es cero pasa a uno y si es uno pasa a cero. La siguiente ilustración muestra un ejemplo:

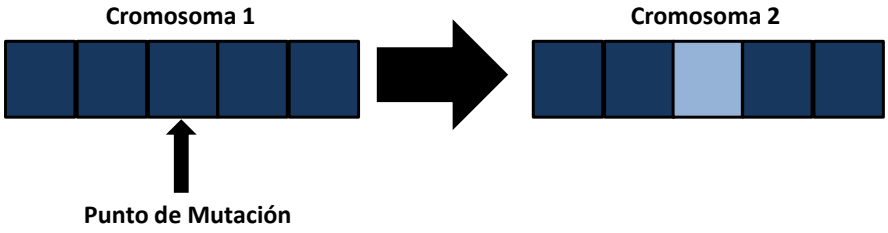


Ilustración 8. Operador de mutación

Para cada posición de cada individuo comprobamos la tasa de mutación, y si se cumple aplicamos el operador. Habitualmente la tasa de aplicación del operador de mutación es bastante pequeña (en torno al 0.1%) comparada con el operador de cruce. Sin embargo esto no es una norma general.

En muchos casos la mutación produce individuos con peor adaptación que los individuos originales, ya que la mutación puede romper las posibles correlaciones entre genes que se formaron con la evolución de la población.

4.8 Proceso de remplazo

Habitualmente los Algoritmos Genéticos mantienen un tamaño de población constante, aunque existen otras posibilidades. Para mantener el tamaño de la población, los nuevos individuos creados mediante los operadores genéticos deben remplazar a otros de la población anterior.

- *Generacionales.* Aquí la población se renueva por completo de una generación a otra.
- *Estado estacionario.* La descendencia de los individuos seleccionados en cada generación se incluye en la población remplazando algunos individuos de la población anterior. Existen diversas posibilidades para establecer los criterios de remplazo en los Algoritmos Genéticos con estado estacionario.
    - *Remplazo de los padres.* Los hijos sustituyen a los padres.
    - *Remplazo aleatorio.* Los individuos a eliminar se eligen aleatoriamente.
    - *Remplazo de los individuos peor adaptados.* Los individuos a eliminar se eligen aleatoriamente pero solo entre los que tienen el valor de adaptación más bajo.
    - *Remplazo de individuos de adaptación similar.* Cada nuevo individuo remplaza a un individuo de la población anterior que tiene un valor de adaptación similar al suyo.

5. JUSTIFICACIÓN

Un estudio inicial de costeo del proceso para la empresa, sugiere que en el área de maquinados se encuentra invertido el 65% del capital de trabajo de la compañía. El costo del proceso en estas operaciones, es igual al 43% del costo del producto final lo que denota principal atención a los resultados de este trabajo de investigación. En esta área de Tecnomec se procesa el 75% de los componentes manufacturados de las operaciones de preparación de componentes.

Lo anterior deja en claro la importancia en desarrollar un proyecto que ayude a incrementar el tiempo productivo del área y disponibilidad de maquinaria, el beneficio inmediato se centra en dos principales mejoras: completar los trabajos programados sin necesidad de tiempo extra y/o dar cabida a nuevos proyectos productivos sin invertir en nuevas máquinas y equipos.

En un esfuerzo por participar aún más en los mercados de consumo, la visión de la compañía para el año 2015, es duplicar la facturación total del grupo registrada en el año 2011. Este proyecto ha sido ya incluido dentro de los indicadores de mejora de la compañía y monitoreado a través del sistema de evaluación de objetivos estratégicos de la empresa.

Además de optimizar las rutas de materiales para reducir el impacto de las actividades que no agregan valor en el proceso productivo, existen varias razones que motivan al estudio de este problema en particular, empezando por el buen nivel de investigación científica de éste problema de producción clasificado como *"dificultad alta",* donde se pretende resolver el caso mediante un modelo basado en algoritmos de búsqueda. A pesar de que estos modelos se han utilizado ya por lo menos durante 4 décadas, siguen estando vigentes en la resolución de problemas complejos y tiene fundamento en un marco teórico muy completo de científicos que han dedicado su vida al desarrollo de este tipo de herramientas.

Las recompensas que se tendrán al desarrollar esta investigación pueden replicarse prácticamente en casi todas las áreas productivas de la organización una vez que se haya encontrado el algoritmo ideal para cada fase del proceso de fabricación. Se puede maximizar el beneficio de esta investigación al planear y generar un programa de rutas de materiales en toda la cadena de valor. Se tiene una facilidad relativa en acotar o agrupar el problema en pequeñas unidades de trabajo, para simplificar el algoritmo y trabajar a pequeña escala, antes de lanzar el proyecto a nivel planta.

La organización está dispuesta a invertir en acciones que generen productividad y ahorros, éste trabajo contribuye significativamente con ese esfuerzo y desarrolla una herramienta de planeación de producción en un campo de la Ingeniería Industrial que es todavía poco explorado. La parte fundamental del proceso de investigación y experimentación, inicia con una correcta definición del objetivo, el cual a su vez tiene estrecha relación con el problema y la justificación que ya se han explicado.

6. OBJETIVO

Como resultado del análisis del problema que se ha descrito para el lector y después de una amplia investigación del estado del arte en el estudio de optimización combinatoria y secuenciación de trabajos. Se presenta el objetivo de ésta investigación:

"Minimizar el tiempo total de terminación de todos los trabajos asignados al área de maquinados, mediante uso de Algoritmos Genéticos como método iterativo para la aproximación de una solución aceptable al problema de secuenciación de trabajos".

El modelo o función matemática se representa de la siguiente manera:

$$Minimizar\ C_{max} = \max(t_{ik} + T_{ik}) : \vee J_i \in J, M_k \in M$$

$$donde\ t_{ik}:\ Tiempo\ de\ inicio\ de\ cada\ operación.$$

$$J:\ Conjunto\ de\ n\ trabajos\ a\ ser\ procesados.$$

$$M:\ Conjunto\ de\ m\ recursos\ o\ máquinas.$$

$$O_{ik}:\ Operación\ de\ trabajo\ J_i, que\ debe\ ser\ procesado\ en\ la\ máquina\ M_k\ por\ un\ periodo\ ininterrumpido\ de\ tiempo\ T_{ik}$$

$sujeto\ a$:

$$tiempos\ de\ inicio \qquad t_{ik} \geq 0 \qquad i, p \in J \quad k, h \in M$$

$$restricción\ de\ precedencia \qquad t_{ik} - t_{ih} \geq T_{ih} \qquad si\ O_{ih}\ precede\ a\ O_{ik}$$

$$restricción\ disyuntiva \qquad t_{pk} - t_{ik} + K(1 - y_{ipk}) \geq T_{ik} \quad y_{ipk} = 1, si\ O_{ik}\ precede\ a\ O_{pk}$$

$$t_{ik} - t_{pk} + K(y_{ipk}) \geq T_{pk} \qquad y_{ipk} = 0,\ en\ otro\ caso$$

$$donde\ k > (\textstyle\sum_{i=1}^{n} \sum_{k=1}^{m} T_{ik} - \min(T_{ik}))$$

Como objetivos específicos de la investigación, destacan los siguientes:

- Proporcionar una herramienta al área de producción capaz de estimar la secuenciación de materiales para dejar de hacerlo empíricamente y reducir las actividades que no agregan valor.

- Estimar la capacidad de las máquinas con diferentes configuraciones de mezcla de producción, para obtener una referencia verdadera al programar los pedidos y evitar cargas excedidas o incompletas.

- Identificar y analizar las restricciones de materiales por máquina, para promover la flexibilidad de las mismas llevando estas restricciones a su mínimo nivel, y así incrementar las opciones de proceso para cada material que se introduzca al proceso.

- Estudiar el efecto y la eficacia de los Algoritmos Evolutivos para comprobar su efectividad y proponer soluciones de programación de producción en otras áreas productivas de la empresa.

- Adecuar el algoritmo de solución al sistema de control de producción ERP, para que se genere una interface de extracción de datos del sistema, procesamiento de información y entrega de resultados en tiempo para coordinar el flujo de materiales.

7. HIPÓTESIS

Se ha revisado intensamente el estado del arte, y definitivamente existen configuraciones de trabajo tipo taller donde se ha programado la producción de manera heurística con Algoritmos Genéticos, sin embargo debido a la gran versatilidad de las configuraciones productivas y a los métodos de programación de producción, así como a los esquemas de trabajo, no se ha encontrado un modelo de optimización combinatoria que resuelva el problema de programación de las rutas de materiales en un departamento de maquinados, como el que se tiene en Tecnomec Agrícola.

De ahí que este trabajo de investigación se fundamenta y persigue un resultado a partir de la siguiente hipótesis:

"El algoritmo de optimización combinatoria (Algoritmo Genético), es capaz de encontrar una solución factible al problema de asignación de recursos a los trabajos diarios de maquinado de materiales, en un esquema de producción por periodos diarios y programación de producción flexible"

La afirmación anterior se fundamenta en la revisión sistemática de cerca de 45 proyectos de investigación y divulgación científica, a través de plataformas de conocimiento certificadas, donde se ha constatado que el JSSP (problema de programación de tipo taller), ha sido resuelto de forma satisfactoria con modelado de funciones objetivo a través de algoritmos evolutivos. Pero sin condiciones que se asemejen a las que presenta en su producción la empresa donde se realiza el estudio.

Esta hipótesis se atañe al objetivo fincado en la sección anterior ya que encontrar una solución factible al problema de asignación de recursos está estrechamente ligado a la minimización de la terminación de todos los trabajo programados para el área de maquinados. En ese sentido al demostrarse que la hipótesis se cumple, se mejora la disponibilidad de los equipos lo cual se traducirá en incrementar la capacidad instalada en plata, disminuyendo los costos de proceso y las inversiones futuras.

8. DESARROLLO DE LA INVESTIGACIÓN

El desarrollo de esta investigación se divide en 3 etapas principales: Obtención de datos, Pruebas iniciales del algoritmo e Implementación global del sistema.

La primera etapa se fundamentó por la medición de las variables que afectan a la función objetivo. En esta etapa se obtuvieron tiempos ciclo del proceso, tiempo de puesta a punto o *setup*, maquinaría establecida dentro de la celda *job-shop* y factores de secuenciación de las operaciones dependiendo de la transformación tecnológica de los componentes.

Posteriormente se experimentó con algunos de los datos obtenidos en la primera etapa para un proceso de fabricación de un solo producto en la compañía, lo cual acotó el esfuerzo computacional y de obtención de datos a menor escala. Esta etapa resulta esencial en el proceso de investigación, ya que permite ajustar el modelo de simulación en función del resultado que se busca. En esta etapa se trabajó con distintos tipos de algoritmos de secuenciación, además se definió un sistema de identificación del tipo de cambio de modelo para simplificar la interacción del sistema y enfatizar la importancia de la secuenciación de trabajos basados en el agrupamiento de *setup* similares.

El proceso de implementación global del sistema se centra en la caracterización del sistema de secuenciación de tareas en toda la celda en configuración de trabajo tipo *job-shop* con todas sus combinaciones y restricciones. En esta sección se muestra el producto, resultado del proceso de investigación y aplicación industrial. En la organización hemos llamado *BAPS* al modelo de secuenciación de rutas generado en esta investigación por sus siglas en inglés *Bison Advanced Planning Scheduling* el cual se ha desarrollado (programado y ajustado) en la aplicación computacional MATLAB® y éste se enlaza con el software de ERP actualmente utilizado en la compañía llamado EPICOR[30].

[30] Epicor Software Corporation (recientemente combinada con Activant Solutions Inc.) es líder mundial en soluciones de software de negocios para las industrias de manufactura, distribución, retail y servicios.

8.1 OBTENCIÓN DE DATOS

Existen 3 etapas del proceso de manufactura en función al tiempo de terminación de los trabajos: el tiempo ciclo, el tiempo de *setup* y el recurso de trabajo. En cualquier sistema de organización de trabajo resulta indispensable trabajar con esta información actualizada y confiable. La obtención de información se generó a través del área de Ingeniería Industrial de la compañía y contó con la colaboración de algunos residentes de ingeniería industrial en planta. Se definieron cerca de 15,000 tiempos estándar para las 3 etapas del tiempo del proceso de transformación de cada operación de piso.

8.1.1 Tiempo ciclo

El tiempo ciclo del proceso se define como el intervalo de tiempo transcurrido en una operación desde que se coloca la pieza en máquina hasta que se retira y confirma su calidad, por cada pieza o ensamble que se realiza, está estrechamente ligado el tiempo estándar. Cada operación tiene un tiempo ciclo distinto dependiendo de la naturaleza de su transformación, el recurso establecido y la velocidad de procesamiento. Los componentes de los equipos fabricados en Tecnomec suelen tener en promedio de 3 a 4 operaciones por componente en el área de maquinados. Hasta el momento de esta redacción, se han contabilizado un total de 3963 trabajos distintos que pueden ser procesados con distintas combinaciones en función de la programación de los productos debido a la estacionalidad de los productos y la demanda del cliente.

8.1.2 Tiempo de preparación

El tiempo de preparación o *setup* es un elemento esencial de este trabajo de investigación. Lo anterior se fundamenta principalmente en la necesidad corporativa de producir únicamente los equipos que se encuentran vendidos al cliente, por lo cual se requiere una fabricación sumamente flexible que exige realizar muchos cambios de modelo cada turno de trabajo en lotes de producción con pocas piezas lo cual genera una configuración de trabajo tipo taller o *job-shop*. El tiempo de *setup* se define como el lapso que transcurre desde que se termina y valida la última pieza del lote anterior,

hasta que la máquina se encuentra herramentada y la primera pieza del siguiente lote esta validada respecto a las características de calidad y diseño. No todos los cambios de modelo consumen la misma cantidad de tiempo. La mayoría de los *setup* están vinculados con el tipo de herramienta de corte y sujeción que se montará en las máquinas, por lo que asociar componentes que se fabriquen con herramientas similares, permitirán que el tiempo total por turno dedicado a setup sea más corto. Para este trabajo se han definido grupos de herramentación dependiendo de las que requiera el proceso, en la matriz de datos este dato será conocido como “tipo *setup*”.

El principal recurso mecánico del área de maquinados está en operaciones de desbaste de materiales cilíndricos, generación de barrenos, corte lineal y desbaste en 3 ejes. La definición de los procesos corre a cargo del área de Ingeniería de Manufactura, la cual está en función de la complejidad del requerimiento geométrico y del tipo de tecnología de fabricación necesaria para el cumplimiento de las características del diseño, esta definición otorga la restricción conjuntiva del sistema. Un resumen de la maquinaria y equipo asociado a esta área se presenta en la Tabla 1.

8.1.3 Recursos disponibles

Una gran labor de obtención de datos para este estudio fue completar la matriz de características de las operaciones. En la siguiente tabla, se muestra un ejemplo de la matriz de datos donde se asocian algunas de las operaciones para la fabricación de un producto en particular. En ella se enlistan los componentes a fabricar, se asocian las precedencias de los procesos para cada pieza, en cada recurso disponible se definen el tipo setup, el tiempo de setup y el tiempo ciclo por cada proceso. Para este producto en particular solo se requieren 3 tornos distintos, un centro de maquinado y un taladro de banco para completar todas las operaciones de cada uno de sus componentes. La información de esta tabla se ha utilizado para realizar las pruebas iniciales de simulación de secuenciación de operaciones.

Tabla 1. Recursos disponibles del área de maquinados

MAQUINA	DESCRIPCIÓN	MARCA	MODELO
SG-001	SEGUETA	DoALL	C-916M
SG-002	SEGUETA	DoALL	500SNC
SG-003	SEGUETA	DoALL	500SNC
SG-004	SEGUETA	HE&M	VT120HA-60
TOR-002	TORNO CONVENCIONAL	NARDINI	437-HB
TOR-003	TORNO CONVENCIONAL	INCA	400
TOR-004	TORNO CONVENCIONAL	FANAMHER	TB-205
TOR-005	TORNO CONVENCIONAL	NARDINI	MASCOTE
TOR-006	TORNO CONVENCIONAL	NARDINI	ND 195 S
TRCN-001	TORNO CNC	YANG	SML-20
TRCN-002	TORNO CNC	HASS	SL-20 / ALIMENTADOR
TRCN-003	TORNO CNC	HASS	SL-20
TRCN-004	TORNO CNC	HASS	SL-20
TRCN-005	TORNO CNC	HASS	TL-3
TRCN-006	TORNO CNC	HASS	TL-3
TRCN-007	TORNO CNC	HASS	ST-30
TRCN-008	TORNO CNC	HASS	ST-30Y
CMQ-001	CENTRO DE MAQUINADOS	YANG	SMV 1000
CMQ-002	CENTRO DE MAQUINADOS	HASS	VF 3
CMQ-003	CENTRO DE MAQUINADOS	HASS	VF 5/40
TLD-001	TALADRO DE BANCO	FANAMHER	TCA-50
TLD-002	TALADRO DE BANCO	IBARMIA	B-50
TLD-005	TALADRO RADIAL	OOYA	RE2-1300A
TLD-006	TALADRO DE BANCO	CLAUSING	2286
TLD-007	TALADRO DE BANCO	CLAUSING	2286
TLD-008	TALADRO DE BANCO	ALLEN	A4A
TLD-009	TALADRO DE BANCO	ALLEN	A4A
TLD-011	TALADRO DE BANCO	CINCINNATI	3R5
FRE-001	FRESADORA	BRIDGEPORT	125610
MCH-001	BRAZO MACHUELADOR	FLEXARM	202

Tabla 2. Matriz de secuencia de operaciones para un producto

COMPONENTE	NOMBRE	PRECEDENCIA DE PROCESOS	LOTE	TOR-006			TRCN-002			TRCN-005			CMQ-002			TLD-011		
				TO			CN			CN			CM			BA		
				tipo setup	setup	ciclo	tipo setup	setup	ciclo	tipo setup	setup	ciclo	tipo setup	setup	ciclo	tipo setup	setup	ciclo
(1) AC12010001C	TUBO PRINCIPAL	TO-CN-BA	12	**1**	0.5	6				**54**	11	2						
(2) AC12010003C	TAPA SOLDADA	CN	12				**3**	5	7				**12**	7	0.5			
(3) AC12010004C	COPLE CONECTOR	CN-BA	24	**1**	0.5	10							**14**	9	11			
(4) AC12010007C	EMBOLO	BA-CN	12							**17**	8	4				**4**	2	3
(5) AC12010023C	TAPA BUJE	BA-CN	12	**4**	3	8							**23**	7	11			
(6) AC12010023C	FLECHA PRINCIPAL	CN-TO	12	**11**	2	6.6												
(7) AC03000001C	BUJE DE FLECHA	BA-TO	12	**4**	3	6.9				**17**	8	2.3						
(8) AC03000003C	BUJE DE CAMISA	BA-TO	12				**14**	8	9.2				**11**	12	0.5			
(9) AC03000004C	EXTENSION	TO	12			11.5									12.6			
(10) AC03000007C	RONDANA DE FIERRO	CN	24							**15**	4	4.6				**17**	1.5	3.4
(11) AC03000023C	ROTULA SOLDABLE	CM	24	**6**	1	9.2							**11**	12	12.6			
(12) AC03000023C	PIVOTE	CN-BA	12	**8**	2	7.6												

8.2 PRUEBAS INICIALES

Para experimentar con el modelo de secuenciación basado en el heurístico Algoritmos Genéticos, se toma solo una parte del problema de asignación global de Tecnomec

Agrícola con el objetivo de encontrar una solución local minimizada y conocer el método de trabajo del algoritmo tal como se ha mencionado en el apartado anterior. Se decide generar un problema de asignación para producir 2 cilindros hidráulicos de 2.000” de diámetro interior con capacidad de 3000psi, utilizados como sistema de levante en un cargador frontal. Los cilindros son equipos que cuentan con pocos componentes y requieren de varias máquinas que interviene en su proceso de fabricación. Entonces, se trabajará con un modelo *12 x 5* para el proceso experimental inicial de esta investigación. La siguiente ilustración muestra la ubicación en piso de planta de la maquinaria utilizada en esta primer fase del proceso de experimentación inicial en la distribución actual del área tipo *job-shop* de maquinados.

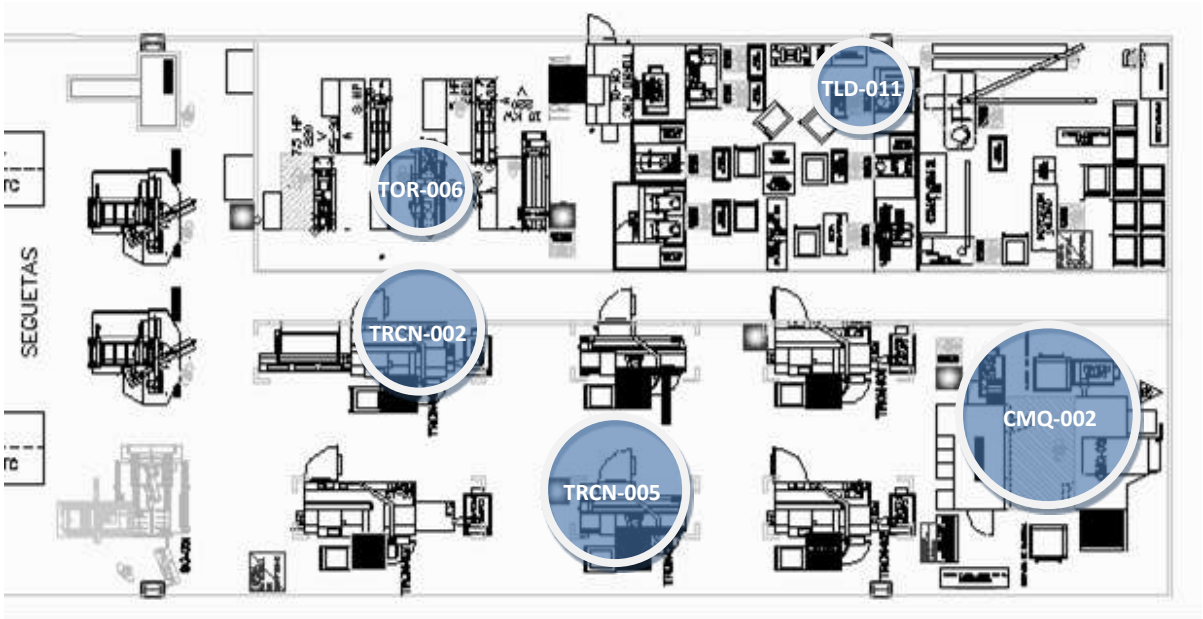


Ilustración 9. Ubicación en área de maquinados de los recursos para la prueba inicial

El flujo de los trabajos ya está definido y es teóricamente invariable, ya que responde a la secuencia natural del proceso y la tecnología instalada en planta, la información presentada en la tabla 2 es la base para la introducción de las variables en el proceso de secuenciación, la información se ha descargado en una base de datos en MATLAB®.

8.2.1 Fases de modelado para la experimentación

A partir de los principios investigados y estudiados sobre la estructura y funcionamiento de un algoritmo genético, a continuación se resume el método básico del proceso iterativo en la ilustración 10. Es importante destacar que como vanguardia de esta investigación durante la etapa de evaluación (paso #2 del proceso) se introduce al sistema la restricción de la priorización del sistema en función del tipo de *setup* asociado.

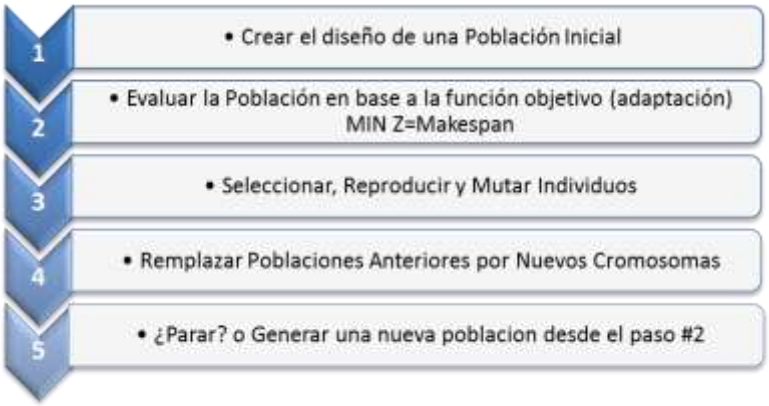


Ilustración 10. Fases del modelo heurístico de Algoritmos Genéticos

El proceso experimental se basa en la adaptación de los datos reales del proceso productivo en planta a un modelo computacional que cumpla con las 5 fases generales del proceso iterativo de búsqueda con un Algoritmo Genético como proceso de búsqueda de la solución. El autor de este trabajo ha decidido trabajar con el modelado de las variables de entrada (conversión de series de tiempo a sistema binario), así como la simulación del comportamiento para la función objetivo definida, por medio del software de programación MATLAB®.

La herramienta utilizada es un software matemático que ofrece un entorno de desarrollo de programación integral. Entre sus prestaciones básicas se destacan: manipulación de matrices, representación de datos y funciones, implementación de algoritmos pre-establecidos, librerías de funciones matemáticas y de programación precargadas, creación de interfaces de usuario (interfaces ejecutables y de comunicación con usuario) e interface de programas con otros lenguajes y con otros dispositivos, en el

caso de este trabajo de investigación para la implementación global se utilizará la interface MATLAB - EPICOR.

Como recursos definidos para esta etapa experimental, se trabajó con un PC con procesador Intel Core i3 M350 2.27 GHz, una memoria RAM de 4.00GB y Sistema operativo Windows 7 a 64bits SP1.

8.2.2 1° Etapa del proceso experimental. Población Inicial

En esta etapa se diseñó un algoritmo capaz de generar n números aleatorios sin repetición excluyendo al 0, en este caso *n*=12 (número de trabajos) que corresponde a la cantidad de componentes que tiene el producto de la fase experimental. El proceso de generación se replica *m* veces, ya que para este experimento se tienen 5 máquinas. El resultado es un vector de 60 posiciones, donde las primeras 12 posiciones representan la secuencia de entrada de los trabajos en la primer máquina, así como los últimos 12 representan la secuencia de operaciones de la máquina 5.

El algoritmo para la generación de dicho vector, se presenta a continuación, dicho vector resultante se identificará como el primer individuo de la población inicial del sistema evolutivo. Obsérvese que se utiliza la función rand, la cual genera números completamente aleatorios lo cual evita que se tengan soluciones locales en el sistema.

```
function [cromosoma]=aleatorioA(m)
maq=5;
job=12;
lcrom=maq*job;

t=1;
for j=1:maq
   n=0; % usaremos este contador n para saber cuántos elementos válidos llevamos en el vector
   while n<job, % ejecuta el ciclo tantas veces como sea necesario, mientras n no alcance los n números
      a=round(job*rand); % genera un número entero al azar.
      i=0; % usaremos este índice i para recorrer las posiciones ya guardadas del vector (desde 1 hasta n)
      existe=0; % usamos esta bandera para determinar cuándo debemos parar el ciclo (si se encuentra el
número)
```

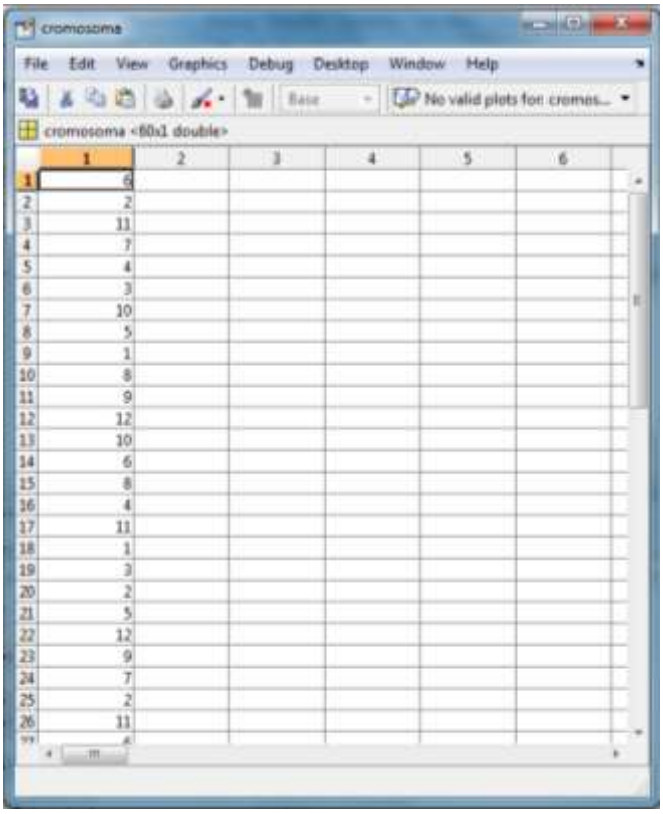
```
        while i<n && existe==0 && a~=0
            i=i + 1; % incrementa el índice de la posición del vector que estamos comparando
            if m(t,i)==a && a~=0
                existe=1; % esto cambiará la condición del ciclo, terminando la búsqueda antes de llegar a los n
números
            end % fin del if
        end % fin del While i
        if existe==0 && a~=0
            n = n + 1; % incrementamos el contador de números válidos en el vector
            m(t,n)=a; % asigna el valor encontrado a la nueva posición n del vector
        end % fin del i
    end % fin del while n
t=t+1;
end
o=m';
cromosoma=zeros(lcrom,1);

k=1;
for j=1:maq
    for i=1:job
        cromosoma(k)= o(i,j);
        k=k+1;
    end
end
```

El resultado es el vector para aleatorizar las carácterísticas de secuenciación del primer individuo de la población inicial. El que se muestra en la tabla 3, es un vector de tamaño 60x1, donde se establece la secuencia aleatorizada por cada máquina cada 12 celdas, las primeras 12 corresponden a las operaciones asinadas a la máquina 1, de la celda 13 a la 24 las que se programan para la segunda máquina y así sucesivamente.

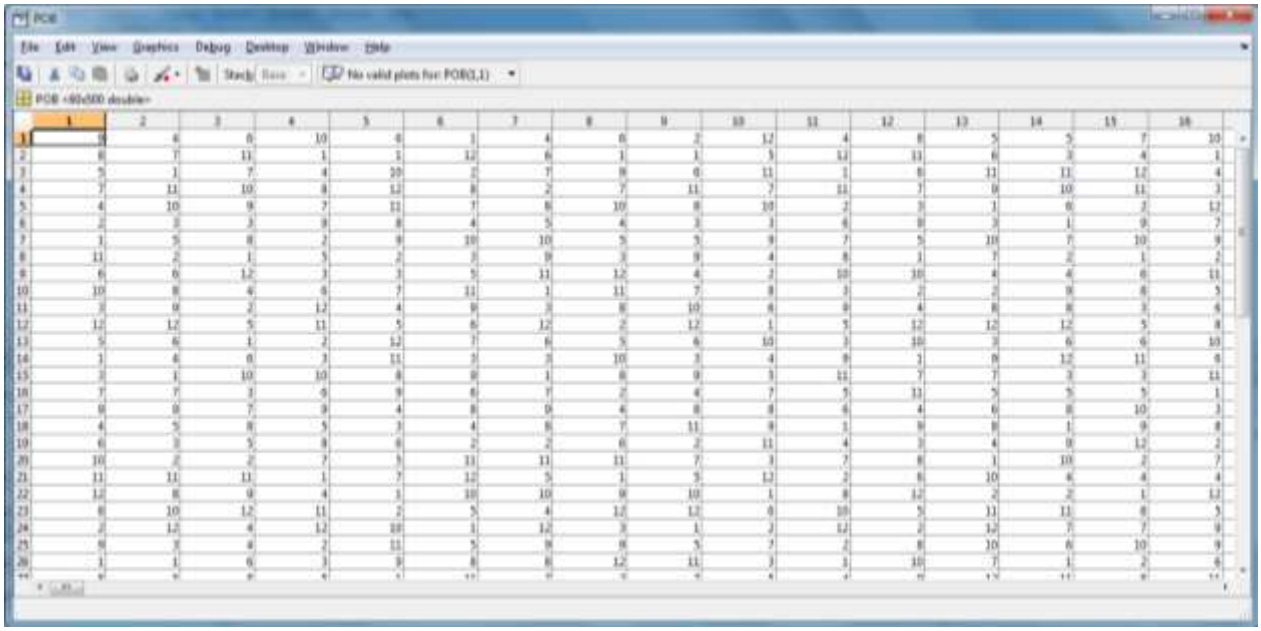
Para completar la primera generación del sistema evolutivo, se replica el método de aleatorización, para generar una población de individuos o soluciones factibles con una magnitud de 500 elementos, la matriz de la Tabla 4 representa la población inicial del sistema evolutivo, una parte del resultado de la réplica del algoritmo se muestra en dicha tabla.

Tabla 3. Cromosoma aleatorizado 60X1. Resultado de la aplicación de la función aleatorioA.



	1	2	3	4	5	6
1	6					
2	2					
3	11					
4	7					
5	4					
6	3					
7	10					
8	5					
9	1					
10	8					
11	9					
12	12					
13	10					
14	6					
15	8					
16	4					
17	11					
18	1					
19	3					
20	2					
21	5					
22	12					
23	9					
24	7					
25	2					
26	11					

Tabla 4. Población Aleatorizada 60X500. Resultado de la aplicación de la función Principal

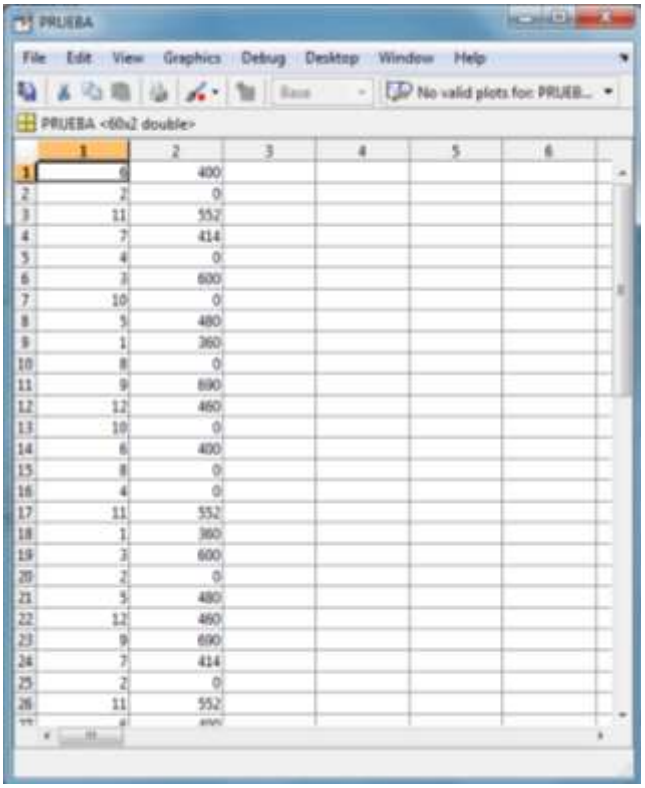


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	3	4	8	10	8	1	4	6	2	12	4	8	5	5	7	10
2	6	7	11	1	1	12	6	1	1	5	12	11	6	3	4	1
3	5	1	7	4	10	2	7	8	6	11	1	8	11	11	12	4
4	7	11	10	8	12	8	2	7	11	7	11	7	8	10	11	3
5	4	10	9	7	11	7	8	10	8	10	2	3	1	6	2	12
6	2	3	3	9	8	4	5	4	3	3	6	8	3	1	9	7
7	1	5	8	2	9	10	10	5	5	9	7	5	10	7	10	9
8	11	2	1	5	2	3	9	3	9	4	8	3	7	2	1	2
9	6	6	12	3	3	5	11	12	4	2	10	10	4	4	6	11
10	10	8	4	6	7	11	1	11	7	8	3	2	2	8	8	5
11	3	9	2	12	4	9	3	8	10	6	8	4	8	8	3	6
12	12	12	5	11	5	6	12	2	12	1	5	12	12	12	5	8
13	5	6	1	2	12	7	6	5	6	10	3	10	3	6	6	10
14	1	4	6	3	11	3	3	10	3	4	9	3	8	12	11	6
15	3	1	10	10	8	8	1	6	9	3	11	7	7	3	3	11
16	7	7	3	6	9	6	7	2	4	7	5	11	5	5	5	1
17	8	9	7	9	4	8	9	4	8	8	6	4	6	8	10	3
18	4	5	8	5	3	4	8	7	11	9	1	9	8	1	9	8
19	6	3	5	8	6	2	2	6	2	11	4	3	4	9	12	2
20	10	2	2	7	5	11	11	11	7	3	7	8	1	10	2	7
21	11	11	11	1	7	12	5	1	5	12	2	6	10	6	4	4
22	12	8	9	4	1	10	10	9	10	1	8	12	2	2	1	12
23	8	10	12	11	2	5	4	12	12	8	10	5	11	11	8	5
24	2	12	4	12	10	1	12	3	1	2	12	2	12	7	7	9
25	9	3	4	2	11	5	9	9	5	7	2	8	10	6	10	9
26	1	1	6	3	9	8	8	12	11	3	1	10	7	1	2	8

Antes de evaluar la adaptación del cromosoma frente a la función objetivo, se extrajo un cromosoma y se asignó el tiempo ciclo de proceso asignado por cada lote, se asoció el tiempo de preparación o *setup* relacionado a la máquina en donde se procesa, el resultado es una matriz *60 x 2*.

En la tabla que se mostrará a continuación se están integrando 3 elementos indispensables en la secuenciación de las operaciones: primero el tiempo ciclo del lote, el tiempo de preparación de la máquina previa a cada trabajo y la aleatorización de los trabajos. En la siguiente sección se introduce un elemento indispensable del sistema: las precedencias de las operaciones.

Tabla 5. Cromosoma de un individuo con sus tiempos de proceso asociados 60X2



	1	2	3	4	5	6
1	6	400				
2	2	0				
3	11	552				
4	7	414				
5	4	0				
6	3	600				
7	10	0				
8	5	480				
9	1	360				
10	8	0				
11	9	690				
12	12	460				
13	10	0				
14	6	400				
15	8	0				
16	4	0				
17	11	552				
18	1	360				
19	3	600				
20	2	0				
21	5	480				
22	12	460				
23	9	690				
24	7	414				
25	2	0				
26	11	552				

8.2.3 Generación de la matriz de precedencias para el cromosoma seleccionado

La generación de una matriz de precedencias permite “obligar” a la ordenación de trabajos según el esquema productivo actual, se mencionaba que la matriz de

precedencias estará en función del total de trabajos (*n*) y las máquinas (*m*). En este caso al tener un problema *5 x 12*, la matriz de precedencias deberá ser de *60 x 60*, ya que todas posiciones deberán relacionarse a su vez con cada una de ellas.

A continuación se presenta una parte de la matriz de precedencias que se asignó al vector de tiempos y posiciones para poder evaluarse con la función objetivo dentro de la sección 8.2.4 de esta investigación:

Tabla 6. Matriz de precedencias para la población 60X60

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
11	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	
12	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
15	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
16	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	
17	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
18	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	
19	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
22	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
27	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	

8.2.4 2° Etapa del proceso experimental. Evaluación de adaptación en la función objetivo

Haciendo una retrospectiva al fundamento principal de este trabajo, el objetivo del proyecto es la minimización del *makespan* o tiempo de finalización de todos los trabajos de un conjunto programado de componentes maquinados por cada día de trabajo o lo que es lo mismo; por cada periodo productivo. Las variables de entrada al proceso ya han sido definidas durante el proceso de generación de la población inicial. Se cuentan con los tiempos de inicio y finalización de cada trabajo (t_{ik}) se sabe del conjunto de trabajos a ser procesados (J), también se ha definido el conjunto de recursos o máquinas (M) y además se restringe el sistema a que cada trabajo (J_i) de ser

procesado en cierta máquina (J_i) por un tiempo ininterrumpido de tiempo (T_{ik}). La evaluación de la adaptación de cada cromosoma extraído de la población se hace a través del siguiente modelo matemático:

$$Minimizar\ C_{max} = \max(t_{ik} + T_{ik}) : \vee J_i \in J, M_k \in M$$

Existen distintas formas de evaluar la función objetivo a través de distintos algoritmos que dan prioridad a la entrada de trabajos, en seguida se evalúan algunos de ellos y su eficiencia en la minimización del makespan. Es importante destacar que estos sistemas de evaluación sistemática son parte del estado del arte en lo que respecta a secuenciación de trabajos, en secciones posteriores se explicará de donde proviene la programación de estos modelos.

8.2.5 Evaluación de la función objetivo en base al algoritmo de McNaughton

El Algoritmo de McNaughton ('P|pmtn|Cmax'), prioriza los trabajos para minimizar el tiempo de terminación, se asume que todos los recursos son idénticos y no respeta el hecho de las partes no pueden ser procesadas al mismo tiempo en 2 recursos diferentes.

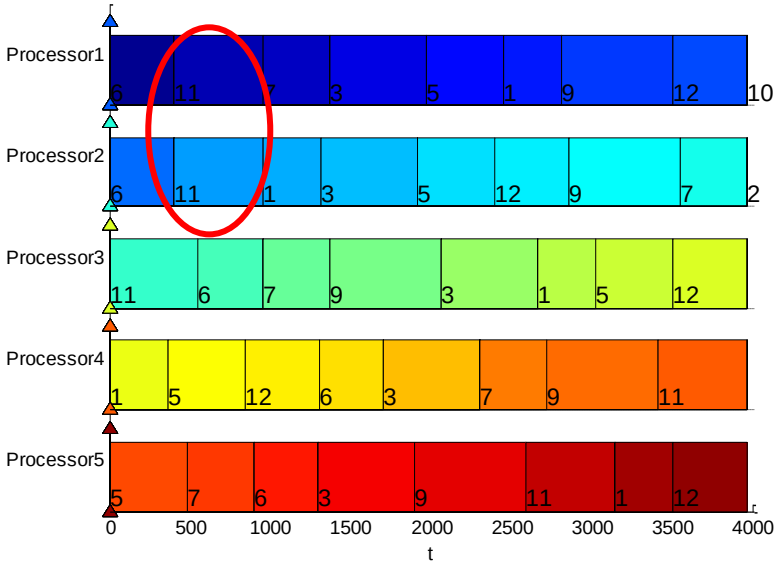


Ilustración 11. Representación gráfica de la evaluación del individuo con el Algoritmo de McNaughton

Tabla 7. Resultado de la evaluación del individuo con el Algoritmo de McNaughton

CARACTERÍSTICA	RESULTADO (tiempo, segundos)
cmax (makespan)	3.9560e+003 (65.93 minutos)

Como se observa en la tabla anterior, existen algunos trabajos que se traslapan durante su fabricación en distintas máquinas, lo cual no es permisible en un sistema de fabricación como el nuestro, en otras palabras: no se cumple la restricción disyuntiva del sistema JSSP. Por lo tanto se descarta el Algoritmo de McNaughton como un método efectivo de evaluación de la función objetivo.

8.2.6 Evaluación de la función objetivo con Algoritmo List Scheduling

En este heurístico ('P|prec|Cmax') las tareas son seleccionadas de una lista previa, cuando una maquina se desocupa la primer tarea disponible se programa y se elimina de la lista de espera. Esto elimina por completo la posibilidad de encontrar una pieza en 2 máquinas diferentes. El *makespan* se incrementa ligeramente como se observa en la ilustración 12.

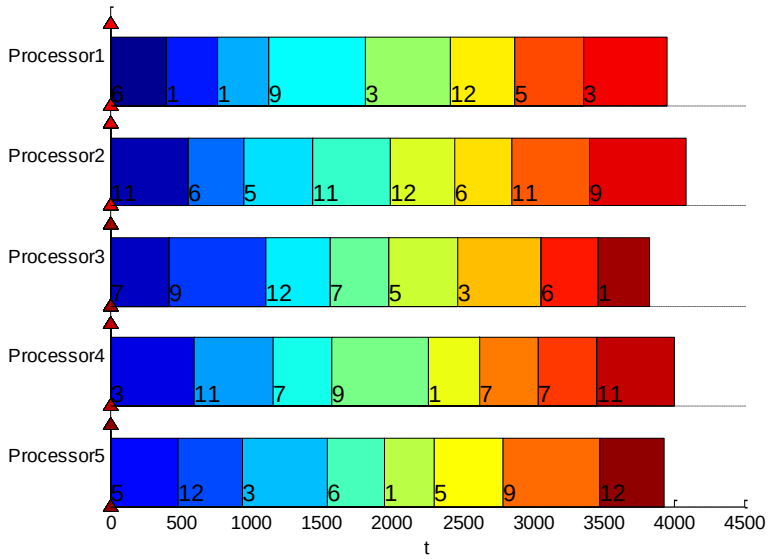


Ilustración 12. Representación gráfica de la evaluación del individuo con el Algoritmo List Scheduling

Tabla 8. Resultado de la evaluación del individuo con el Algoritmo List Scheduling

CARACTERÍSTICA	RESULTADO (tiempo segundos)
cmax (makespan)	4.086e+003 (68.1 minutos)

.

El algoritmo *List Scheduling*, es una buena opción como evaluador de la función objetivo y de medición de la adaptación del cromosoma.

8.2.7 Evaluación de la función objetivo con Algoritmo de Tiempo más Largo de Proceso (LPT)

Es un algoritmo muy limitado, donde las tareas son programadas de manera decreciente dependiendo del tiempo total de fabricación. Para este caso es muy impráctico y alejado de la realidad, ya que todas las tareas comparten recursos en el mismo periodo de tiempo. A pesar de que los resultados en el makespan son similares a los del Algoritmo de McNaughton, resulta imposible trabajar con éste método.

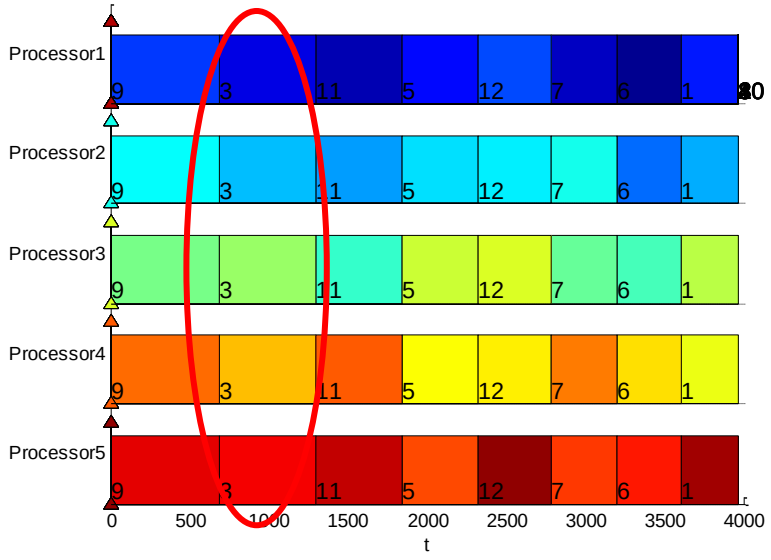


Ilustración 13. Representación gráfica de la evaluación del individuo con el Algoritmo LPT

Tabla 9. Resultado de la evaluación del individuo con el Algoritmo LPT

CARACTERÍSTICA	RESULTADO (tiempo, en segundos)
cmax (makespan)	3.9560+003 (65.98 minutos)

8.2.8 Evaluación de la función objetivo con Algoritmo de Brucker

El algoritmo de Brucker resuelve problemas de tipo Cmax, pero considerando los tiempos de finalización de cada uno y las precedencias entre cada operación. Los resultados del algoritmo son muy similares al algoritmo del *list scheduling*, pero el modelo no respeta las máquinas como exclusivas.

Hasta el momento se ha observado que el algoritmo de *List Scheduling* se aproxima bastante al tipo de problema real que se tiene en planta y se ha decidió por ahora trabajar con este método de adaptación para la función objetivo, mientras a la par se estudia el método: *Algorithm for Problem P|rj,prec,dj|Cmax*, que se presume es un algoritmo mucho más eficiente que respeta las asignaciones de los recursos y las precedencias en las operaciones.

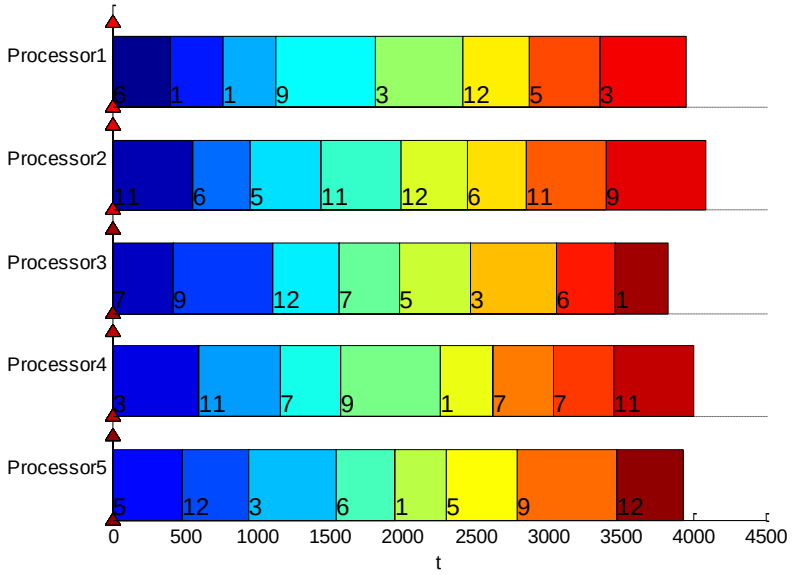


Ilustración 14. Representación gráfica de la evaluación del individuo con el Algoritmo de Brucker

Tabla 10. Resultado de la evaluación del individuo con el Algoritmo de Brucker

CARACTERÍSTICA	RESULTADO (tiempo, segundos)
cmax (makespan)	4.086e+003 (68.1 minutos)

Las pruebas experimentales hechas hasta el momento apoyadas del software de programación MATLAB®, se han apoyado del *ToolBox* desarrollado por la *Czech Technical University* en Praga, República Checa. Este complemento al software lleva el nombre de TORSCHE[31] (*Time Optimization Resources Scheduling*), éste ha sido diseñado específicamente para la investigación de operaciones complejas en la ingeniería industrial.

8.3 PROCESO DE REPRODUCCIÓN Y MUTACIÓN

Para el proceso evolutivo se recurrió al módulo de optimización en MATLAB® como recurso de investigación previo. Este *Toolbox* (Mathworks, 2010) contiene varios sistemas de optimización incluidos los Algoritmos Genéticos, en él se define la función objetivo principal así como las variables de la función y restricciones. Aquí se definen los niveles de reproducción y mutación así como la definición de la terminación del sistema evolutivo.

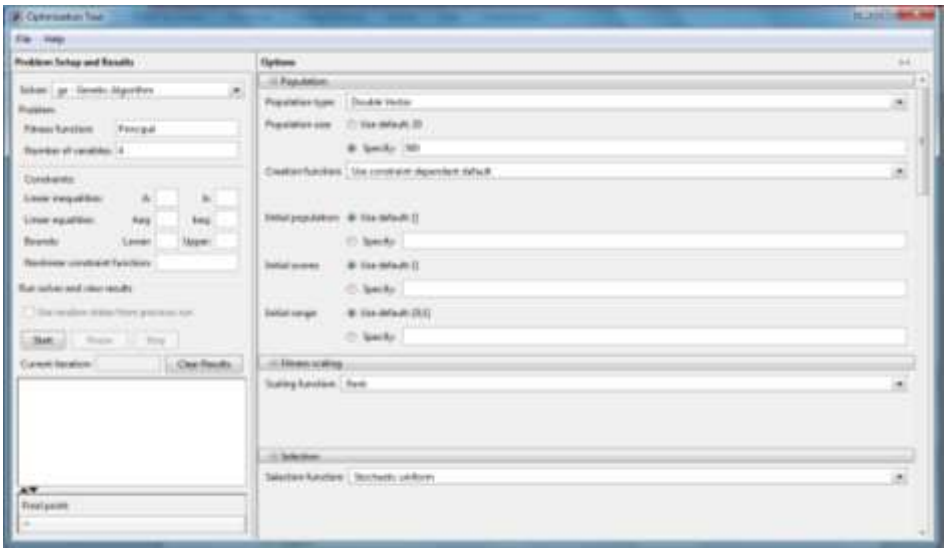


Ilustración 15. Toolbox para optimización en MATLAB con Algoritmos Genéticos

[31] TORSHE (time Optimization Resources Scheduling), Czech Technical University.

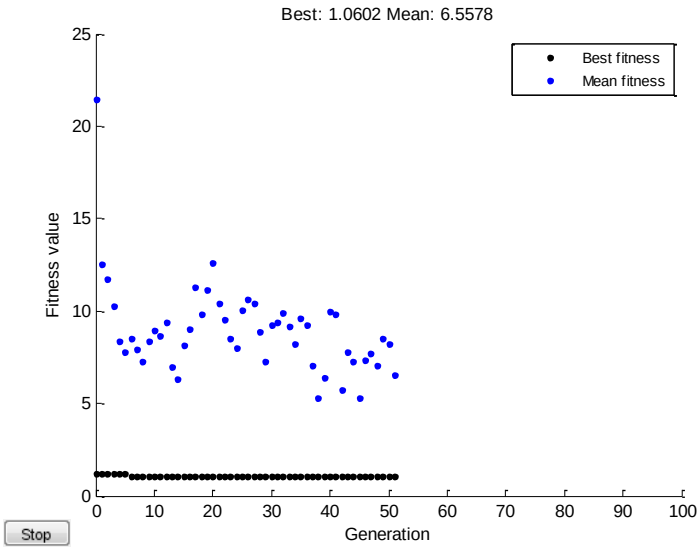


Ilustración 16. Ejemplo de sistema optimizado con toolbox de AG

Se analizó la programación del *Toolbox* para identificar los criterios de optimización y modificarlo para priorizar los trabajos por su tipo de setup. El programa modificado se muestra a detalle en el anexo A. En este programa se ha caracterizado la variable “tipo de setup” como una restricción al sistema y se ha modificado el *Toolbox*. Se ha adaptado la interface de usuario para que se pueda priorizar la ordenación para esta variable.

Como entregable de esta etapa del proceso de investigación, se tuvo éxito en la programación de la secuenciación de las actividades para la fabricación de componentes del cilindro hidráulico, encontrando que el algoritmo de secuenciación *list scheduling* favorece los resultados del sistema evolutivo. En los resultados finales de esta investigación se demostrará el impacto que tiene la secuenciación de las tareas con éste sistema comparado con el método tradicional utilizado. Araujo (Araujo & Cervigon, 2009), en su texto Algoritmos evolutivos, sugiere iniciar con una reproducción del 40% de los individuos y la eliminación generacional del 40% de los peores resultados. Para la mutación sugiere utilizar el 0.1% de la población.

Hasta este punto podemos concluir que podemos utilizar el método de secuenciación con el Algoritmo de *List Scheduling* y ejecutar el *Toolbox* de optimización para obtener la mejor secuencia de operaciones.

8.4 IMPLEMENTACIÓN GLOBAL BAPS

8.4.1 Implementación del sistema como programador de producción.

A nivel general en el área de maquinado, se implementó el algoritmo desarrollado en la etapa anterior de la investigación para todos los trabajos y maquinaria disponible del área. Se diseñó una interface para la entrada de datos mediante el *Toolbox* de creación de ejecutables en MATLAB, denominado GUI. La interface se ha denominado BAPS por sus siglas en inglés *Bison Advanced Production Scheduling*, el sistema permite cargar y configurar las cantidades y tipos de equipos a producir diariamente y mediante la ejecución del algoritmo muestra la disponibilidad teórica en cada unidad de trabajo.

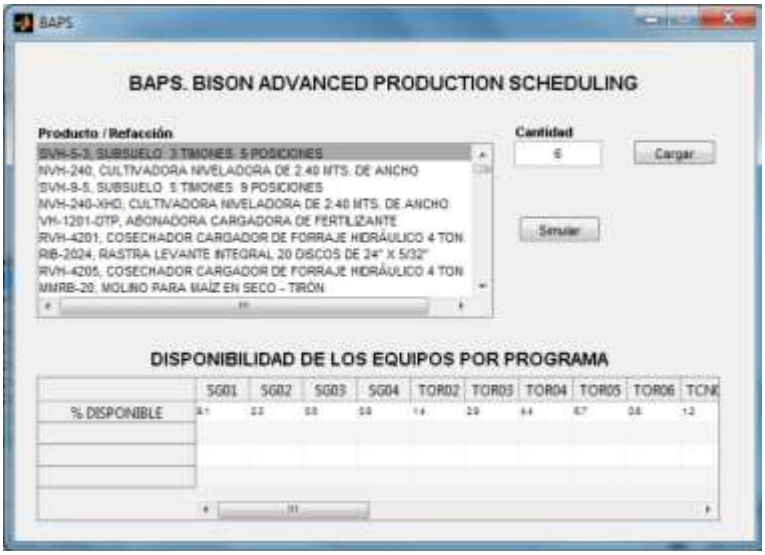


Ilustración 17. Menú Principal del BAPS

El sistema BAPS ha resultado una herramienta sumamente útil para el programador de producción, ya que elimina la programación de producción de forma empírica. Anteriormente este sistema generaba cargas de trabajo desbalanceadas, ocasionando que los trabajos no fuesen completados en tiempo durante la primera fase del proceso de transformación, o en ocasiones generar maquinas subutilizadas o sin carga de trabajo.

Según Jeffrey Linker (Linker, 2004) autor del texto *The Toyota Way*, la base para la implementación de las operaciones esbeltas, es la nivelación de la producción, también conocido por el termino japonés *heijunka*. El sistema para la distribución y secuenciación de los trabajos ha logrado como primer entregable balancear la carga de producción en el sistema dinámico con el que se produce en Tecnomec.

8.4.2 Implementación del sistema como guía para la distribución de los trabajos en planta.

El sistema de secuenciación de trabajos como entregable de este proyecto de investigación, permite al usuario final conocer cuál es la mejor forma de distribuir los trabajos a través del proceso productivo y pone a juicio la hipótesis de este trabajo.

Recordemos que se tiene un sistema de fabricación dinámico y se pretende que sea sumamente flexible. El sistema productivo de Tecnomec tiene la capacidad de procesar cerca de 4000 números de parte distintos en sus 29 máquinas para el área de maquinados, no está de más mencionar que las combinaciones de trabajo son infinitas. Una forma simple de secuenciar y mostrar las operaciones que se ejecutarán en cada máquina del área se muestra en la siguiente ilustración, aquí se muestra la máquina programada y con un color se identifica el trabajo que se procesa y el orden establecido.

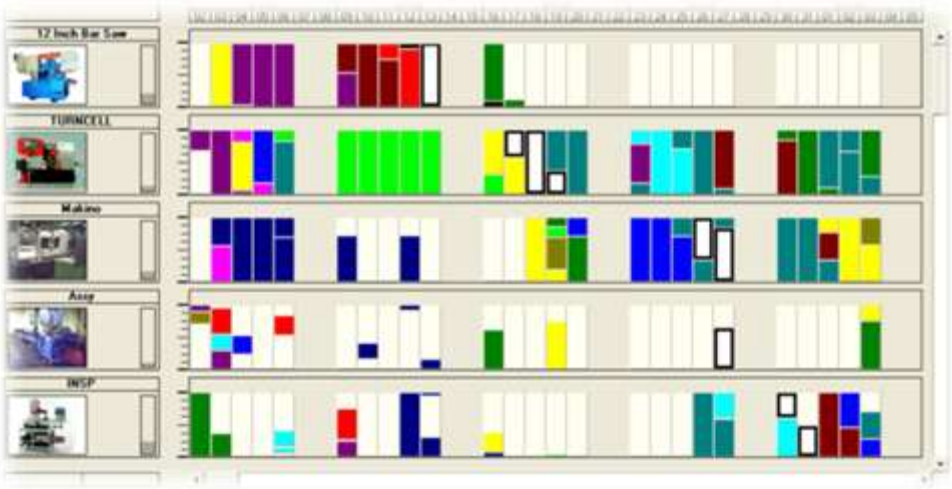


Ilustración 18. Plan de secuenciación de trabajos

El reporte que se ha mostrado le hemos llamado: Plan de secuenciación de trabajos y permite al supervisor de línea destinar cada trabajo a su respectiva máquina, se visualizan los tiempos muertos para cada recurso lo cual permite seguir la producción de manera efectiva, y como se demostrará en el capítulo de resultados, logra incrementar la disponibilidad de los equipos.

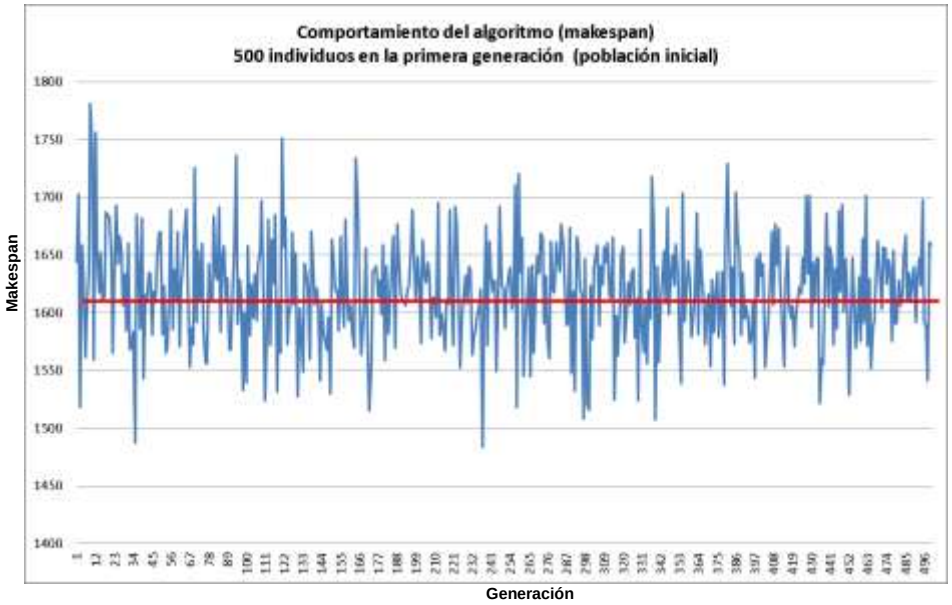
Ilustración 19. Área de maquinado de Tecnomec Agrícola S.A. de C.V.

9. ANÁLISIS DE RESULTADOS

Si partimos de revisar la hipótesis de este trabajo, tenemos que: *"El algoritmo de optimización combinatoria (Algoritmo Genético), es capaz de encontrar una solución factible al problema de asignación de recursos a los trabajos diarios de maquinado de materiales, en un esquema de producción por periodos diarios y programación de producción flexible".*

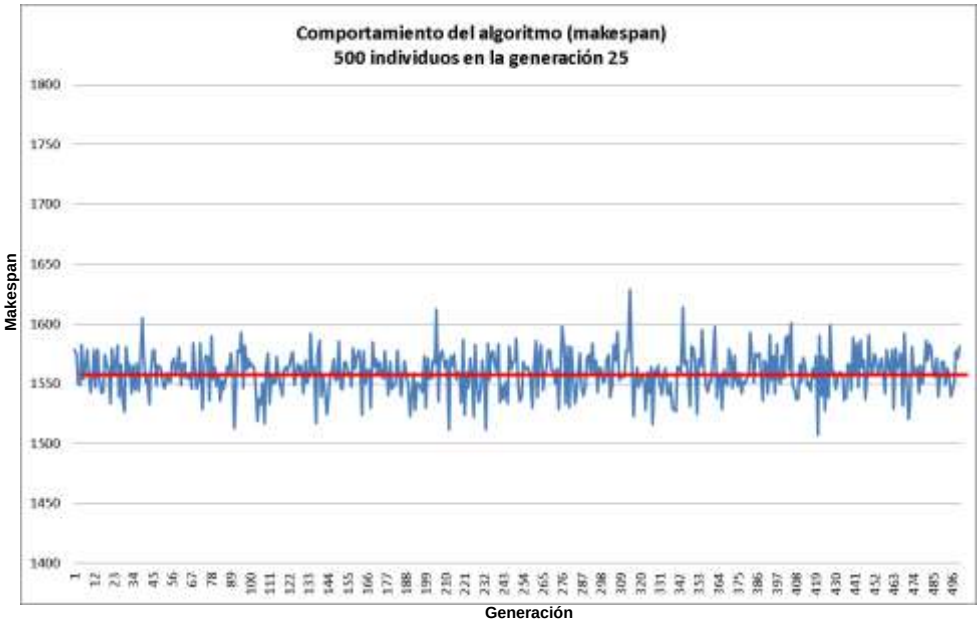
Los resultados de esta investigación se han caracterizado con la evaluación del algoritmo y el comportamiento de cada generación de resultados con el algoritmo genético como metaheurístico. La siguiente gráfica muestra el desempeño del algoritmo y la calidad de los individuos en la primera generación del sistema, que representa a la población inicial del sistema:

Tabla 11. Comportamiento del algoritmo para la población inicial.



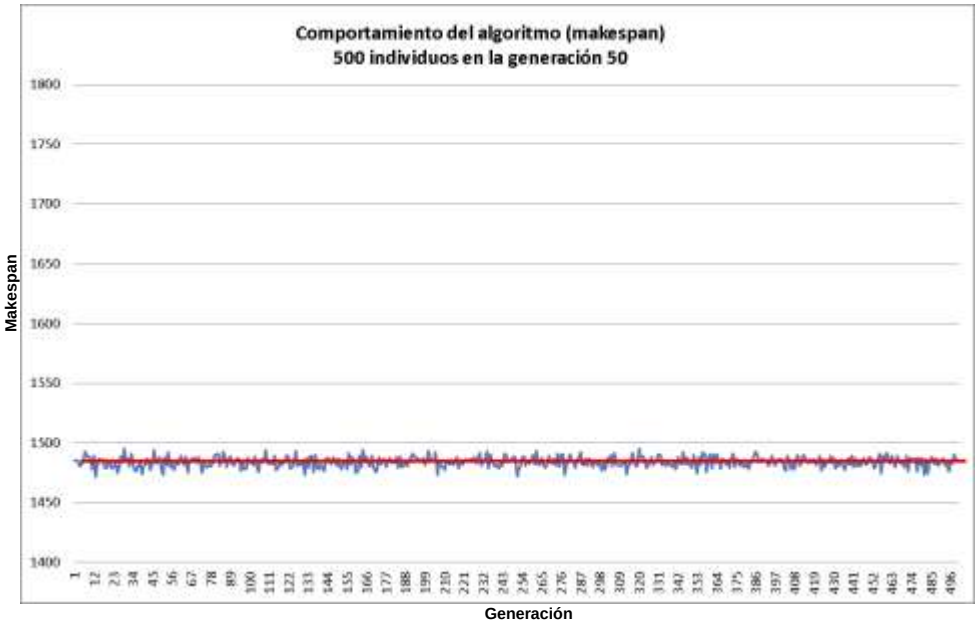
En la primera generación del sistema se obtiene un individuo mejor adaptado con un valor convertido de 1483 minutos de makespan. Los datos pueden observarse dispersos y normales lo cual confirma la aleatoriedad inicial del sistema. Se mostrarán los resultados del sistema después de 25 generaciones hasta completar 100 iteraciones del sistema evolutivo.

Tabla 12. Comportamiento del algoritmo para la generación 25.



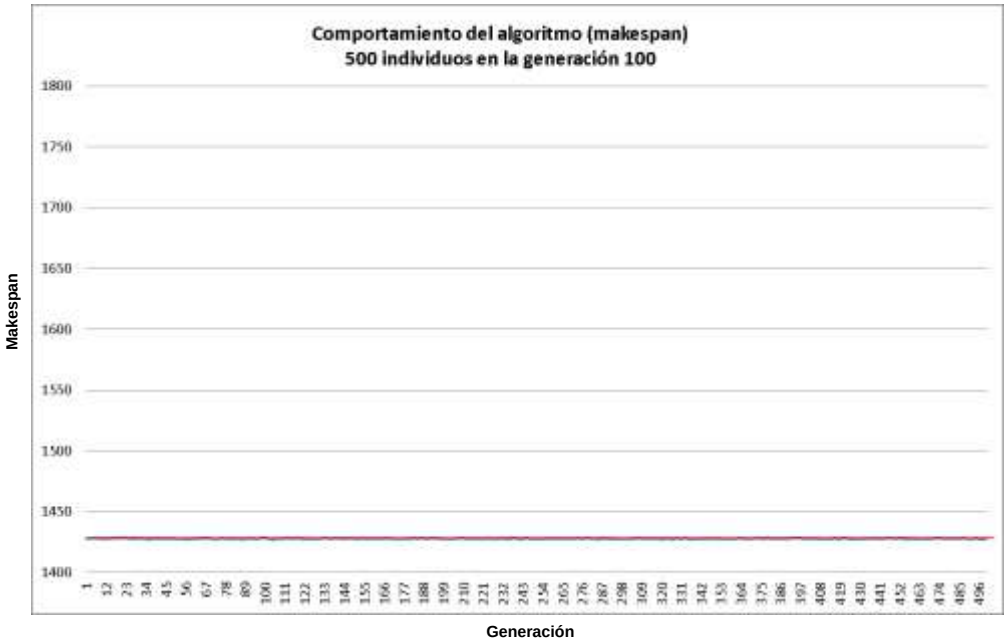
Para la generación 25 se observa una disminución de la media de todos los individuos cercano a 1560 minutos. La población pierde dispersión, es homogénea y tiene un valor mínimo de makespan de 1507 minutos. El sistema se comporta adecuadamente según lo esperado. En la siguiente tabla, observaremos el comportamiento del sistema en la generación 50.

Tabla 13. Comportamiento del algoritmo para la generación 50.



Para la generación 50 ya se tiene un makespan mínimo de 1471 y un promedio de los valores de la población de 1486. El algoritmo ha arrojado combinaciones con valores del makespan menores y la población se consolida. La última caracterización del comportamiento del algoritmo es al final del sistema iterativo, el cual se ha definido al completar 100 iteraciones.

Tabla 14. Comportamiento del algoritmo para la generación 100.

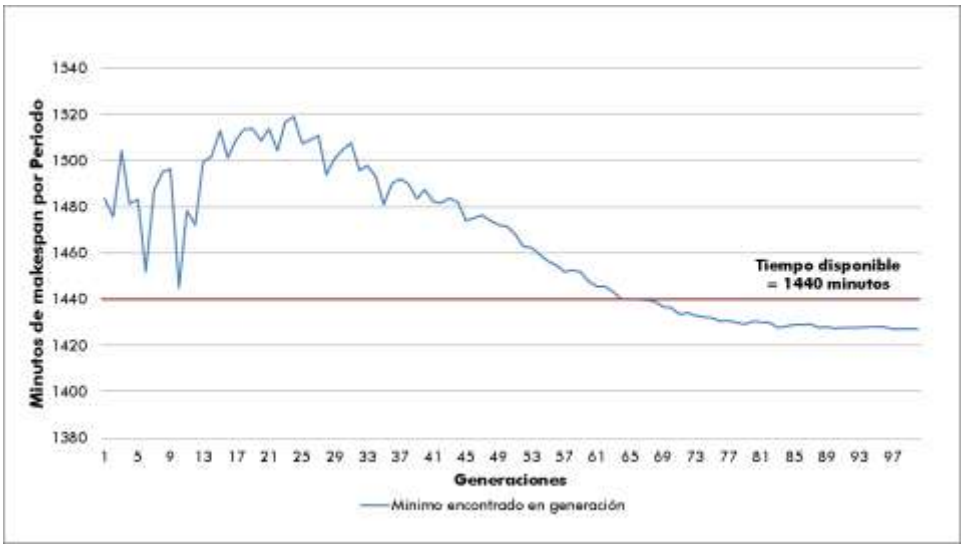


En la última generación definida para el sistema, se obtuvo un valor mínimo de la generación de 1427 minutos, se tiene una población muy homogénea con poca dispersión en sus valores. Para un programa productivo con 1440 minutos de trabajo disponibles, en teoría debe completarse el programa productivo sin inconvenientes.

Se ha extraído el mejor individuo de cada generación, es decir el que tiene el menor makespan durante cada uno de las 100 iteraciones. Esta caracterización del comportamiento del sistema nos permite identificar la tendencia del sistema y comprobar el número de iteraciones necesarias para encontrar una solución factible al problema de secuenciación y asignación de los trabajos. En la siguiente ilustración se ejemplifica esta caracterización para un periodo productivo completo, después de 100 iteraciones trabajadas. Observe que se han encontrado combinaciones de producción

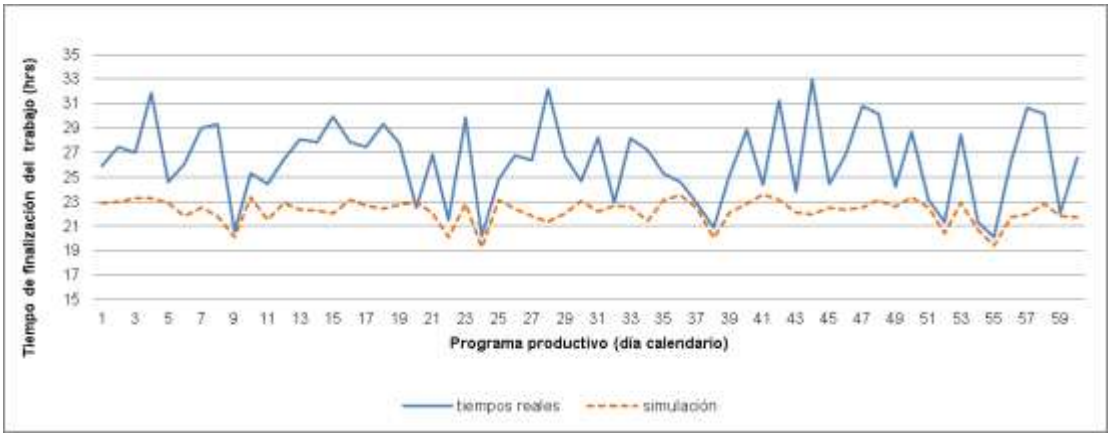
con makespan menor al tiempo disponible a partir de la generación 70, a partir de esta generación se tienen combinaciones exitosas.

Tabla 15. Caracterización del sistema con la mejor combinación de cada iteración.



Comprobar que la eficiencia del algoritmo fuera adecuada a través del tiempo, se comparó el resultado arrojado por el sistema y el makespan real del proceso durante 60 programas productivos de Tecnomec Agrícola. En la siguiente Tabla, la línea puntada representa el valor de makespan mejor adaptado en cada programa, la línea continua es el valor real del makespan en el proceso productivo de Tecnomec Agrícola en los meses de Abril a Julio de 2012.

Tabla 16. Comparativo del makespan real en piso y la simulación con BAPS.



Este impacto directo en la disminución del makespan o el tiempo de finalización total de todos los trabajos ha traído como beneficios económicos directos y comprobables los que se presentan a continuación:

- Eliminación de programación de tiempo extra en las líneas de maquinados de 3,248 horas en los últimos 6 meses, equivalente a $350,000.00 M.N.
- Introducción de una nueva línea de cargadores frontales para un nuevo cliente internacional, con abastecimiento de la misma línea de maquinados. Lo anterior evitó adquirir 2 tornos CNC y 1 centro de maquinados con valor aproximado de $480,000.00 USD.
- Se redujo el consumo de herramienta de corte al estandarizar los tipos de *setup* generando un ahorro anual proyectado de $180,000.00 M.N.

10. CONCLUSIONES

Como se mostró en la sección de Resultados, el entregable de esta investigación ha sido exitoso. Se ha cumplido la hipótesis propuesta proveniente del estudio del estado del arte durante la aplicación del sistema en el esquema productivo flexible de Tecnomec Agrícola S.A. de C.V.

Actualmente se trabaja con el sistema BAPS para la programación de la producción, lo cual ayuda a anticipar la carga de producción en las áreas de maquinado. Más provechoso ha sido aún el poder contar ahora con un sistema de secuenciación de los trabajos dinámico y en tiempo real que ayuda a la administración del área de habilitado de nuestra empresa.

El sistema implementado como entregable de este trabajo de investigación ha generado una cultura de identificación de las actividades que no agregan valor, lo cual obliga a realizar una revisión constante de los tiempos estándar establecidos y nos exige mantener vigente la información a fin de contar con un sistema eficiente y veraz en todo momento.

Respecto al aporte científico y académico de este trabajo, se ha confirmado que el sistema de optimización combinatoria programado a través de los Algoritmos Genéticos es funcional en sistemas de producción ultra flexibles como el que define a Tecnomec. Hasta ahora no hay una investigación aplicada de este sistema de solución en modelos dinámicos de mayores a *25 x 25*. Recordemos que para nuestro caso, se tienen programas con combinaciones hasta *29 x 295*, lo cual implicó un reto en programación para que el sistema entregue resultados en corto tiempo y además que sea útil para las necesidades actuales de la organización.

Se ha llegado a la conclusión de que sistemas dinámicos con esta cantidad de variables se vuelven complejos a través del tiempo y es difícil mantenerlos al día. Analizando la teoría de manufactura esbelta ser debe optar por sistemas de fabricación simples, lo cual obliga al análisis de la verdadera efectividad del sistema de producción que tiene

Tecnomec Agrícola en la actualidad y valorar si generar un inventario en proceso controlado minimiza la cantidad de cambios de modelo y permite generar mayor disponibilidad de los equipos.

Como una continuación de este trabajo de investigación, es importante analizar con detenimiento los sistemas productivos tipo *job-shop* y contrastarlos con los sistemas con inventarios tipo *kanban* o justo a tiempo. Es de interés del autor realizar un análisis profundo de los distintos esquemas productivos para analizar qué tipo de organización de trabajo resulta más efectivo en función de las características de la empresa y las que arrojen mejores resultados a la organización.

Se ha analizado que se pueden desprender nuevas líneas de investigación a partir de este trabajo. Primero, un contraste entre las ambigüedades del sistema de manufactura esbelto que propone la eliminación del inventario y a su vez la nivelación de la producción, con una demanda de producto tan cambiante se debe encontrar un punto en medio entre el stock y producir JIT.

Y segundo un análisis sobre los beneficios globales de la producción ultra flexible como se tiene ahora en Tecnomec vs la Manufactura celular por producto: Un análisis de costos, inversiones y demanda.

11. REFERENCIAS BIBLIOGRÁFICAS

Adams, J., Balas, E., & Zawak, D. (1988). The Shifting Bottleneck procedure for Job-Shop Scheduling. *Managment Science*.

Akhshabi, M. (2011). Parallel Genetic Algorithm to solving Job-Shop Scheduling Problem. *Journal of basic and applied scientific research*, 1484-1497.

Akhshabi, M., & Akhshabi, M. (2011). Solving flexible Job-Shop scheduling problem using clonal selection algorithm. *Indian journal of Science and Tecnology*.

Araujo, L., & Cervigon, C. (2009). *Algoritmos Evolutivos. Un enfoque practico.* Mexico: Alfaomega.

Bartz-Beielstein, T. (2010). Tuning Simulated Annealing Using the Sequential Parameter optimization Toolbox. *Cologne University of applied Sciences*.

Braysy, O., & Gendreau, M. (2005). Vehicle routing problem with time windows. Part 1. Route construction and local search algorithms. *Transportation Science. Universidad de Montreal.*

Britto Agudelo, R. A., Mejia Delgadillo, G., & Caballero Villalobos, J. P. (2007). Programación de la producción en sistemas de manufactura tipo taller con algoritmo combinado cuello de botella movil y busqueda tabu. *Universidad de Bogotá*, 203-221.

ENDEAVOR MEXICO. (s.f.). Recuperado el 15 de 08 de 2012, de http://www.endeavor.org.mx/misión.html/

FARM EQUIPMENT MANUFACTURERS ASSOCIATION. (s.f.). Recuperado el 15 de 08 de 2012, de http://www.farmequip.org/

FARM EQUIPMENT WHOLESALERS ASSOCIATION. (s.f.). Recuperado el 15 de 08 de 2012, de http://www.emda.net/

Gass, S. I. (2005). *An annotated timeline of operations research.* USA: Kluwer Academic Publishers.

Goncalves, J. F. (2002). A Hybrid Genetic Algorithm for the Job-Shop Scheduling Problem. *AT&T Labs Research Technical Report*.

Jiang, J., & Wen, M. (2011). Hybrid Genetic Algorithm for flexible Job-Shop Scheduling with multiobjective. *Journal of information & computational science*, 2197-2205.

Kobayashi, S. (1995). An Efficient Genetic Algorithm for Job-Shop Scheduling Problem. *Proc. of ICGA'95*, 506-511.

Linker, J. K. (2004). *The Toyota Way.* Madison WI: McGraw Hill.

Lopez de Haro, S. (2004). Secuanciación de Tareas Mediante Metaheuristicos. *Congreso de Ingenieria de Organización*.

Mane, A. (1960). On the Job-Shop scheduling problem. *Yale University*, 40-42.

Marti, R. (2002). Procedimientos Metaheuristicos en Optimización Combinatoria. *Departamento de Estadistica e Investigación Operativa. Universidad de Valencia*.

Mathworks, T. (2010). MATLAB Optimization Toolbox. U.S.

Murugesan, R., & Navaneetha Kumar, V. (2011). A Fast Algorithm for solving JSSP. *European Journal of scientific research*, 579-586.

Naendra, P. M., & Fukunaga, K. A. (1977). A branch and bound algorithm for feature subset selection. *Honeywell systems and research center*, 34-36.

Natarajan, K., Mohanasundaram, K. M., & Shogan Babu, B. (2007). Performance evaluation of priority dispatching rules in multi-level assembly Job-Shops with jobs having weights for flowtime and tardiness. *Int J Manuf Technol*, 751-761.

Salazar, G. J. (2001). *Programacion Matemática.* Madrid: Diaz de Santos.

Sipper, D., & Bulfin, R. L. (1997). *Planeación y Control de la Producción.* Mexico: McGraw Hill.

Taillard, E. (1993). Parallel taboo search for the Job-Shop scheduling problem. ORSA Journal of computing. *Escuela Politécnica de Lausana*.

Tasgerintein, F., Sevlik, L., & Yenisey, M. (2006). A particle swarm optimization and differential evolution algorithms for Job-Shop scheduling problem. *International journal of operational research*.

Toth, P., & Vigo, D. (2003). The granulat tabu search and it's placation to the vehicle-routing problem. *Informs journal of computing. Universidad de Bologna*.

ANEXO A. PROGRAMACIÓN DE TOOLBOX DE OPTIMIZACIÓN MODIFICADO

```
function [x,fval,exitFlag,output,population,scores] = ga(fun,nvars,Aineq,bineq,Aeq,beq,lb,ub,nonlcon,intcon,options)
%GA    Constrained optimization using genetic algorithm.
%   GA attempts to solve problems of the following forms:
%      min F(X)  subject to:  A*X  <= B, Aeq*X  = Beq (linear constraints)
%       X                     C(X) <= 0, Ceq(X) = 0 (nonlinear constraints)
%                            LB <= X <= UB
%                            X(i) integer, where i is in the index
%                            vector INTCON (integer constraints)
%
%   Note: If INTCON is not empty, then no equality constraints are allowed.
%   That is:-
%   * Aeq and Beq must be empty
%   * Ceq returned from NONLCON must be empty
%
%   X = GA(FITNESSFCN,NVARS) finds a local unconstrained minimum X to the
%   FITNESSFCN using GA. NVARS is the dimension (number of design
%   variables) of the FITNESSFCN. FITNESSFCN accepts a vector X of size
%   1-by-NVARS, and returns a scalar evaluated at X.
%
%   X = GA(FITNESSFCN,NVARS,A,b) finds a local minimum X to the function
%   FITNESSFCN, subject to the linear inequalities A*X <= B. Linear
%   constraints are not satisfied when the PopulationType option is set to
%   'bitString' or 'custom'. See the documentation for details.
%
%   X = GA(FITNESSFCN,NVARS,A,b,Aeq,beq) finds a local minimum X to the
%   function FITNESSFCN, subject to the linear equalities Aeq*X = beq as
%   well as A*X <= B. (Set A=[] and B=[] if no inequalities exist.) Linear
%   constraints are not satisfied when the PopulationType option is set to
%   'bitString' or 'custom'. See the documentation for details.
%
%   X = GA(FITNESSFCN,NVARS,A,b,Aeq,beq,lb,ub) defines a set of lower and
%   upper bounds on the design variables, X, so that a solution is found in
%   the range lb <= X <= ub. Use empty matrices for lb and ub if no bounds
%   exist. Set lb(i) = -Inf if X(i) is unbounded below;  set ub(i) = Inf if
%   X(i) is unbounded above. Linear constraints are not satisfied when the
%   PopulationType option is set to 'bitString' or 'custom'. See the
%   documentation for details.
%
%   X = GA(FITNESSFCN,NVARS,A,b,Aeq,beq,lb,ub,NONLCON) subjects the
%   minimization to the constraints defined in NONLCON. The function
%   NONLCON accepts X and returns the vectors C and Ceq, representing the
%   nonlinear inequalities and equalities respectively. GA minimizes
%   FITNESSFCN such that C(X)<=0 and Ceq(X)=0. (Set lb=[] and/or ub=[] if
%   no bounds exist.) Nonlinear constraints are not satisfied when the
%   PopulationType option is set to 'bitString' or 'custom'. See the
%   documentation for details.
%
%   X = GA(FITNESSFCN,NVARS,A,b,Aeq,beq,lb,ub,NONLCON,options) minimizes
%   with the default optimization parameters replaced by values in the
%   structure OPTIONS. OPTIONS can be created with the GAOPTIMSET function.
%   See GAOPTIMSET for details.
%
%   X = GA(FITNESSFCN,NVARS,A,b,[],[],lb,ub,NONLCON,INTCON) requires that
%   the variables listed in INTCON take integer values. Note that GA does
%   not solve problems with integer and equality constraints. Pass empty
%   matrices for the Aeq and beq inputs if INTCON is not empty.
%
%   X = GA(FITNESSFCN,NVARS,A,b,[],[],lb,ub,NONLCON,INTCON,options)
```

```
%   minimizes with integer constraints and the default optimization
%   parameters replaced by values in the structure OPTIONS. OPTIONS can be
%   created with the GAOPTIMSET function. See GAOPTIMSET for details.
%
%   X = GA(PROBLEM) finds the minimum for PROBLEM. PROBLEM is a structure
%   that has the following fields:
%       fitnessfcn: <Fitness function>
%            nvars: <Number of design variables>
%            Aineq: <A matrix for inequality constraints>
%            bineq: <b vector for inequality constraints>
%              Aeq: <Aeq matrix for equality constraints>
%              beq: <beq vector for equality constraints>
%               lb: <Lower bound on X>
%               ub: <Upper bound on X>
%          nonlcon: <Nonlinear constraint function>
%           intcon: <Index vector for integer variables>
%          options: <Options structure created with GAOPTIMSET>
%         rngstate: <State of the random number generator>
%
%   [X,FVAL] = GA(FITNESSFCN, ...) returns FVAL, the value of the fitness
%   function FITNESSFCN at the solution X.
%
%   [X,FVAL,EXITFLAG] = GA(FITNESSFCN, ...) returns EXITFLAG which
%   describes the exit condition of GA. Possible values of EXITFLAG and the
%   corresponding exit conditions are
%
%     1 Average change in value of the fitness function over
%       options.StallGenLimit generations less than options.TolFun and
%       constraint violation less than options.TolCon.
%     3 The value of the fitness function did not change in
%       options.StallGenLimit generations and constraint violation less
%       than options.TolCon.
%     4 Magnitude of step smaller than machine precision and constraint
%       violation less than options.TolCon. This exit condition applies
%       only to nonlinear constraints.
%     5 Fitness limit reached and constraint violation less than
%       options.TolCon.
%     0 Maximum number of generations exceeded.
%    -1 Optimization terminated by the output or plot function.
%    -2 No feasible point found.
%    -4 Stall time limit exceeded.
%    -5 Time limit exceeded.
%
%   [X,FVAL,EXITFLAG,OUTPUT] = GA(FITNESSFCN, ...) returns a
%   structure OUTPUT with the following information:
%             rngstate: <State of the random number generator before GA started>
%          generations: <Total generations, excluding HybridFcn iterations>
%            funccount: <Total function evaluations>
%        maxconstraint: <Maximum constraint violation>, if any
%              message: <GA termination message>
%
%   [X,FVAL,EXITFLAG,OUTPUT,POPULATION] = GA(FITNESSFCN, ...) returns the
%   final POPULATION at termination.
%
%   [X,FVAL,EXITFLAG,OUTPUT,POPULATION,SCORES] = GA(FITNESSFCN, ...) returns
%   the SCORES of the final POPULATION.
%
%
%   Example:
%     Unconstrained minimization of 'rastriginsfcn' fitness function of
%     numberOfVariables = 2
%      x = ga(@rastriginsfcn,2)
%
%     Display plotting functions while GA minimizes
%      options = gaoptimset('PlotFcns',...
```

```
%       {@gaplotbestf,@gaplotbestindiv,@gaplotexpectation,@gaplotstopping});
%      [x,fval,exitflag,output] = ga(@rastriginsfcn,2,[],[],[],[],[],[],[],options)
%
%   An example with inequality constraints and lower bounds
%    A = [1 1; -1 2; 2 1];  b = [2; 2; 3];  lb = zeros(2,1);
%    % Use mutation function which can handle constraints
%    options = gaoptimset('MutationFcn',@mutationadaptfeasible);
%    [x,fval,exitflag] = ga(@lincontest6,2,A,b,[],[],lb,[],[],options);
%
%     FITNESSFCN can also be an anonymous function:
%       x = ga(@(x) 3*sin(x(1))+exp(x(2)),2)
%
%   If FITNESSFCN or NONLCON are parameterized, you can use anonymous
%   functions to capture the problem-dependent parameters. Suppose you want
%   to minimize the fitness given in the function myfit, subject to the
%   nonlinear constraint myconstr, where these two functions are
%   parameterized by their second argument a1 and a2, respectively. Here
%   myfit and myconstr are MATLAB file functions such as
%
%       function f = myfit(x,a1)
%       f = exp(x(1))*(4*x(1)^2 + 2*x(2)^2 + 4*x(1)*x(2) + 2*x(2) + a1);
%
%   and
%
%       function [c,ceq] = myconstr(x,a2)
%       c = [1.5 + x(1)*x(2) - x(1) - x(2);
%            -x(1)*x(2) - a2];
%       % No nonlinear equality constraints:
%        ceq = [];
%
%   To optimize for specific values of a1 and a2, first assign the values
%   to these two parameters. Then create two one-argument anonymous
%   functions that capture the values of a1 and a2, and call myfit and
%   myconstr with two arguments. Finally, pass these anonymous functions to
%   GA:
%
%     a1 = 1; a2 = 10; % define parameters first
%     % Mutation function for constrained minimization
%     options = gaoptimset('MutationFcn',@mutationadaptfeasible);
%     x = ga(@(x)myfit(x,a1),2,[],[],[],[],[],[],@(x)myconstr(x,a2),options)
%
%   Example: Solving a mixed-integer optimization problem
%   An example of optimizing a function where a subset of the variables are
%   required to be integers:
%
%   % Define the objective and call GA. Here variables x(2) and x(3) will
%   % be integer.
%   fun = @(x) (x(1) - 0.2)^2 + (x(2) - 1.7)^2 + (x(3) -5.1)^2;
%   x = ga(fun,3,[],[],[],[],[],[],[],[2 3])
%
%   See also GAOPTIMSET, FITNESSFUNCTION, GAOUTPUTFCNTEMPLATE, PATTERNSEARCH, @.

%   Copyright 2003-2011 The MathWorks, Inc.
%   $Revision: 1.1.6.9 $  $Date: 2011/05/09 00:50:31 $

% If the first arg is not a gaoptimset, then it's a fitness function followed by a genome
% length. Here we make a gaoptimset from the args.
defaultopt = struct('PopulationType', 'doubleVector', ...
    'PopInitRange', [0;1], ...
    'PopulationSize', 20, ...
    'EliteCount', 2, ...
    'CrossoverFraction', 0.8, ...
    'MigrationDirection','forward', ...
    'MigrationInterval',20, ...
```

```
    'MigrationFraction',0.2, ...
    'Generations', 100, ...
    'TimeLimit', inf, ...
    'FitnessLimit', -inf, ...
    'StallGenLimit', 50, ...
    'StallTimeLimit', inf, ...
    'TolFun', 1e-6, ...
    'TolCon', 1e-6, ...
    'InitialPopulation',[], ...
    'InitialScores', [], ...
    'InitialPenalty', 10, ...
    'PenaltyFactor', 100, ...
    'PlotInterval',1, ...
    'CreationFcn',@gacreationuniform, ...
    'FitnessScalingFcn', @fitscalingrank, ...
    'SelectionFcn', @selectionstochunif, ...
    'CrossoverFcn',@crossoverscattered, ...
    'MutationFcn',{{@mutationgaussian 1  1}}, ...
    'HybridFcn',[], ...
    'Display', 'final', ...
    'PlotFcns', [], ...
    'OutputFcns', [], ...
    'Vectorized','off', ...
    'UseParallel', 'never');

% Check number of input arguments
errmsg = nargchk(1,11,nargin);
if ~isempty(errmsg)
    error(message('globaloptim:ga:numberOfInputs', errmsg));
end

% If just 'defaults' passed in, return the default options in X
if nargin == 1 && nargout <= 1 && isequal(fun,'defaults')
    x = defaultopt;
    return
end

if nargin < 11, options = [];
    if nargin < 10,  intcon = [];
        if nargin < 9,  nonlcon = [];
            if nargin < 8, ub = [];
                if nargin < 7, lb = [];
                    if nargin <6, beq = [];
                        if nargin <5, Aeq = [];
                            if nargin < 4, bineq = [];
                                if nargin < 3, Aineq = [];
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end

% Is third argument a structure
if nargin == 3 && isstruct(Aineq) % Old syntax
    options = Aineq; Aineq = [];
end

% Is tenth argument a structure? If so, integer constraints have not been
% specified
```

```
if nargin == 10 && isstruct(intcon)
   options = intcon;
   intcon = [];
end

% One input argument is for problem structure
if nargin == 1
   if isa(fun,'struct')
      [fun,nvars,Aineq,bineq,Aeq,beq,lb,ub,nonlcon,intcon,rngstate,options] = separateOptimStruct(fun);
      % Reset the random number generators
      resetDfltRng(rngstate);
   else % Single input and non-structure.
      error(message('globaloptim:ga:invalidStructInput'));
   end
end

% If fun is a cell array with additional arguments get the function handle
if iscell(fun)
   FitnessFcn = fun{1};
else
   FitnessFcn = fun;
end

% Only function handles or inlines are allowed for FitnessFcn
if isempty(FitnessFcn) ||  ~(isa(FitnessFcn,'inline') || isa(FitnessFcn,'function_handle'))
   error(message('globaloptim:ga:needFunctionHandle'));
end

% We need to check the nvars here before we call any solver
valid =  isnumeric(nvars) && isscalar(nvars)&& (nvars > 0) ...
   && (nvars == floor(nvars));
if ~valid
   error(message('globaloptim:ga:notValidNvars'));
end

% Specific checks and modification of options for mixed integer GA
if ~isempty(intcon)
   % Check whether the user has specified options that the mixed integer
   % solver will either ignore or error.
   gaminlpvalidateoptions(options);
   % If a user doesn't specify PopInitRange, we want to set it to the
   % bounds when we create the initial population. Need to store a flag
   % that indicates whether the user has specified PopInitRange so we can
   % do this in the creation function.
   UserSpecPopInitRange = isa(options, 'struct') && ...
      isfield(options, 'PopInitRange') && ~isempty(options.PopInitRange);
   % Change the default options for PopulationSize and EliteCount here.
   defaultopt.PopulationSize = max(min(10*nvars, 100), 40);
   defaultopt.EliteCount = floor(0.05*defaultopt.PopulationSize);
end

user_options = options;
% Use default options if empty
if ~isempty(options) && ~isa(options,'struct')
      error(message('globaloptim:ga:optionsNotAStruct'));
elseif isempty(options)
   options = defaultopt;
end
% Take defaults for parameters that are not in options structure
options = gaoptimset(defaultopt,options);

% All inputs should be double
try
```

```
    dataType = superiorfloat(nvars,Aineq,bineq,Aeq,beq,lb,ub);
    if ~isequal('double', dataType)
        error(message('globaloptim:ga:dataType'))
    end
catch
    error(message('globaloptim:ga:dataType'))
end

[x,fval,exitFlag,output,population,scores,FitnessFcn,nvars,Aineq,bineq,Aeq,beq,lb,ub, ...
    NonconFcn,options,Iterate,type] =
gacommon(nvars,fun,Aineq,bineq,Aeq,beq,lb,ub,nonlcon,intcon,options,user_options);

if exitFlag < 0
    return;
end

% Turn constraints into right size if they are empty.
if isempty(Aineq)
    Aineq = zeros(0,nvars);
end
if isempty(bineq)
    bineq = zeros(0,1);
end
if isempty(Aeq)
    Aeq = zeros(0,nvars);
end
if isempty(beq)
    beq = zeros(0,1);
end

% Call appropriate single objective optimization solver
if ~isempty(intcon)
    [x,fval,exitFlag,output,population,scores] = gaminlp(FitnessFcn,nvars, ...
        Aineq,bineq,Aeq,beq,lb,ub,NonconFcn,intcon,options,output,Iterate,...
        UserSpecPopInitRange);
else
    switch (output.problemtype)
        case 'unconstrained'
            [x,fval,exitFlag,output,population,scores] = gaunc(FitnessFcn,nvars, ...
                options,output,Iterate);
        case {'boundconstraints', 'linearconstraints'}
            [x,fval,exitFlag,output,population,scores] = galincon(FitnessFcn,nvars, ...
                Aineq,bineq,Aeq,beq,lb,ub,options,output,Iterate);
        case 'nonlinearconstr'
            [x,fval,exitFlag,output,population,scores] = gacon(FitnessFcn,nvars, ...
                Aineq,bineq,Aeq,beq,lb,ub,NonconFcn,options,output,Iterate,type);
    end
end
```

ANEXO B. MATRIZ GENERAL DE DATOS

componente	nombre	material	precedencia de procesos	lote	SEGUETAS																	
					SG-001						SG-002				SG-003				SG-004			
					CS			CX			CS		CX		CS		CX		CS		CX	
					tipo setup	setup	ciclo	tipo setup	setup	ciclo	setup	ciclo	setup	ciclo	setup	ciclo	setup	ciclo	setup	ciclo	setup	ciclo
CF40091303C	COMPLEMENTO DE TUBERIA INFERIOR	TUBING CAL. 18 DE 1/2 (SAEJ525) (4040-9100-18)	CS-DO-TR	0		1.5	0.3		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CF40091302C	TUBERIA INFERIOR	TUBING CAL. 18 DE 1/2 (SAEJ525) (4040-9100-18)	CS-DO-TR	0		1.5	0.3		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CF40091301C	TUBO PRINCIPAL	TUBING CAL. 18 DE 1/2 (SAEJ525) (4040-9100-18)	CS-DO-TR	0		1.5	0.3		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CF40111101C	TUBERIA SUPERIOR	TUBING CAL. 18 DE 1/2 (SAEJ525) (4040-9100-18)	CS-DO-TR	0		1.5	0.3		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CF40111102C	TUBERIA INFERIOR	TUBING CAL. 18 DE 1/2 (SAEJ525) (4040-9100-18)	CS-DO-TR	0		1.5	0.3		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CF40111103C	COMPLEMENTO DE TUBERIA INFERIOR	TUBING CAL. 18 DE 1/2 (SAEJ525) (4040-9100-18)	CS-DO-TR	0		0	0		0	0	0	0	0	0	0	0	0	0	1.5	0.3	0	0
CN08020215C	BUJE DEL PERNO PIVOTE	TUBO CEDULA 40 DE 1 1/2 (4040-1012-40)	CS-CA	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08020701C	TUBO DEL PARADOR	PTR DE 2 X 2 ROJO (4080-1008-02)	CS-BA	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08020802C	PERNO PARA FORMAR "T"	COLD ROLLED C-1018 DE 1/4 (4070-1006-18)	CS-CA	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08050601C	CARCASA HEMBRA	BARRA HUECA 40 MM X 56 MM (4050-1010-01)	CS-CA-BA	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08020009C	PERNO PIVOTE TERCER PUNTO	COLD ROLLED C-1045 DE 1 1/2 (4070-1046-45)	CS-CA-BA	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08020002C	PERNO INFERIOR UNION QUINTA-AVION	COLD ROLLED C-1045 DE 1 1/2 (4070-1046-45)	CS-CA-BA	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08050501C	EMBOLO MACHO	COLD ROLLED C-1045 DE 1 7/16 (4070-1044-45)	CS-CA-BA	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08020801C	PERNO QUINTA-AVION	COLD ROLLED C-1045 DE 3/4 (4070-1022-45)	CS-CA-CH-BA-EM-TR	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08020901C	PERNO QUINTA-CUCHILLA	COLD ROLLED C-1045 DE 1 1/2 (4070-1046-45)	CS-CN-DE-BA-TR	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08020007C	PERNO DEL PARADOR	COLD ROLLED C-1045 DE 3/4 (4070-1022-45)	CS-TO-DE-BA-TR	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08020001C	PERNO REY	COLD ROLLED C-1045 DE 3 (4070-1076-45)	CS-CA-CN-BA-AV	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08010011C	PERNO DEL PIE	COLD ROLLED C-1045 DE 1 1/2 (4070-1046-45)	CS-CN-DE-BA-TR	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08010010C	PERNO DEL PSE	COLD ROLLED C-1045 DE 1 1/2 (4070-1046-45)	CS-CN-DE-RA-TR	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08020006C	PERNO TENSOR MECANICO	COLD ROLLED C-1045 DE 1 1/8 (4070-1034-45)	CS-CA-CN-BA-TR	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN99170001C	BASTON	COLD ROLLED C-1045 DE 3/4 (4070-1022-45)	CS-CA-DO-BA-TR	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08020902C	ARGOLLA	COLD ROLLED C-1018 DE 1/4 (4070-1006-18)	CS-DO	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CN08031001C	VARILLA (SOPORTE PARA MANGUERA)	HOT ROLLED COM. DE 1/2 (4070-5014-00)	CS-DO	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA01110002C	PERNO SUPERIOR DE ENGANCHE	COLD ROLLED C-1045 DE 1	CS-CA-CN-TR	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA02030007C	PERNO DEL SEGURO	COLD ROLLED C-1045 DE 1 (4070-1030-45)	CS-CA-DE-BA-**TR**	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA02030002C	PERNO DE LA HORQUILLA	COLD ROLLED C-1045 DE 1 1/2 (4070-1046-45)	CS-CN-DE-BA-**TR**	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA02030107C	BUJE PARA CAJA DE TIRON	COLD ROLLED C-1045 DE 1 1/2 (4070-1046-45)	CS-BA-CA-BA-MA	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA02031001C	VARILLA ROSCADA	COLD ROLLED C-1045 DE 1 1/4 (4070-1038-45)	CS-TO-CU	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA01110001C	PERNO INFERIOR DE ENGANCHE CAT II	COLD ROLLED C-1045 DE 1 1/8 (4070-1034-45)	CS-CA-BA-**TR**	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA02033101C	PERNO ENGANCHE	COLD ROLLED C-1045 DE 13/16 (4070-1036-45)	CS-CA-BA	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA02031102C	BUJE ROSCADO	COLD ROLLED C-1045 DE 15/8 (4070-1050-45)	CS-BA-CA-BA-MA	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA02031002C	BUJE SEPARADOR	COLD ROLLED C-1045 DE 15/8 (4070-1050-45)	CS-BA-CA	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
CF40090304C	ESPIGA DE EJE	COLD ROLLED C-1045 DE 2 (4070-1060-45)	CS-CA-CE-CN-BA-AV	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA02030008C	BUJE DE SUJECCION	COLD ROLLED C-1045 DE 2 (4070-1060-45)	CS-BA-CA	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA01010503C	TAPA DE DISCOS CONCAVOS	HIERRO GRIS	CA-DE	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA01010504C	TAPA DE DISCOS CONVEXOS	HIERRO GRIS	CA-DE	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA01110101C	LARGUERO DERECHO	PTR DE 3 X 4 ROJO (4080-1640-02)	CS	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA01110102C	LARGUERO IZQUIERDO	PTR DE 3 X 4 ROJO (4080-1640-02)	CS	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA01110104C	TRAVESAÑO POSTERIOR	PTR DE 3 X 4 ROJO (4080-1640-02)	CS	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
RA01110103C	TRAVESAÑO	PTR DE 3 X 4 ROJO (4080-1640-02)	CS	0		0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0

ANEXO C. DESCRIPCIÓN DE TOOLBOX UTILIZADOS

TORSCHE SCHEDULING TOOLBOX.

Es un *Toolbox* de uso libre dedicado a la utilización y desarrollo de algoritmos de secuenciación. Este software se ha desarrollado en la Universidad Tecnológica de Praga en la República Checa.

La secuenciación es una disciplina muy popular que ha crecido en importancia en años recientes. A pesar de que no existe una herramienta que resuelva los problemas determinísticos, TORSCHE es un software que puede resolver muchos problemas en un ambiente de programación con MATLAB.

La última versión de este software cubre las siguientes áreas de secuenciación:

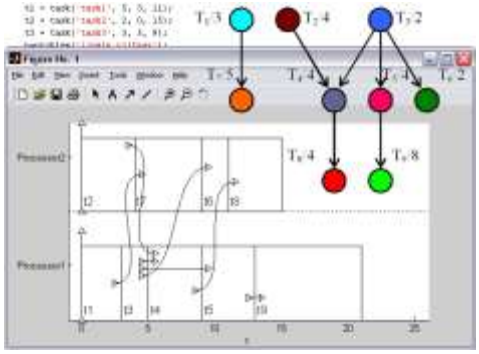
- Programación de procesadores únicos
- Programación de procesadores dedicados.
- Programación de procesadores paralelos.
- Programación cíclica
- Programación en tiempo real.

El software dedica recursos a algoritmos gráficos enlazados con la teoría de la secuenciación y optimización combinatoria. Se pueden realizar editar gráficos de algoritmos y se cuenta con una interface de programación lineal.

TORSCHE es utilizado como herramienta de investigación en las siguientes universidades:

- Georgia Institute of Technology
- Faculty of comptures & information Cairo University
- Dept Informatique Scientifique, IFP, CEDI René Navarre, Vernaison - France
- Department of Management Science, Seoul National University, Seoul, Korea

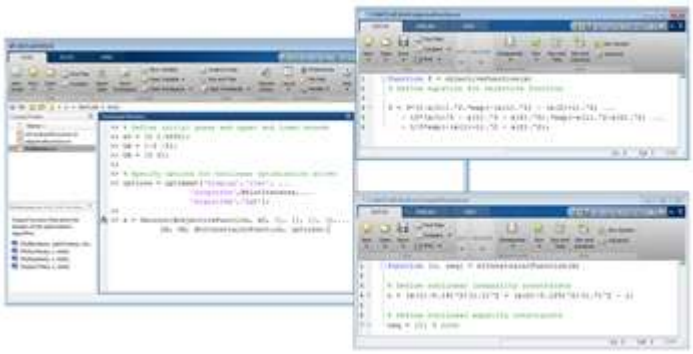
- College of Technology, University of Houston
- Gdansk University of Technology
- Faculty of Engineering, Mutah University, Jordan



MATLAB OPTIMIZATION TOOLBOX

Este *Toolbox* provee una gran cantidad de algoritmos utilizados en el ámbito de la optimización. Permite resolver problemas continuos y discretos, incluye funciones de programación lineal, programación cuadrática, programación lineal binaria, optimización no lineal, sistemas de ecuaciones no lineales y optimización multi-objetivo. Se pueden encontrar soluciones óptimas, análisis comparativos o incorporar nuevos métodos en algoritmos o modelos ya definidos.

CARACTERÍSTICAS PRINCIPALES:

- Herramientas interactivas para definir y resolver problemas de optimización y monitorear el progreso de la solución.
- Solucionadores de optimizaciones no lineales multi-objetivo.
- Métodos de solución de problemas cuadráticos y de programación lineal.
- Métodos para solución de programación binaria entera.

Incluye una amplia gama de métodos utilizados para minimización y maximización. El *Toolbox* implementa algoritmos de gran escala y estándar, permitiendo resolver problemas conociendo su estructura.