UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SECUENCIACIÓN DE ACTIVIDADES EN UN TALLER CON PRODUCCIÓN INTERMITENTE MEDIANTE ALGORITMOS GENÉTICOS

T E S I S

QUE PARA OBTENER EL TÍTULO DE INGENIERO INDUSTRIAL

P R E S E N T A:

ALBERTO CORTÉS GALINDO

DIRECTOR DE TESIS

DR.VÍCTOR HUGO JACOBO ARMENDÁRIZ

CD. UNIVERSITARIA ENERO 2011

AGRADECIMIENTOS

Primero y más importante que todo, agradecer tu apoyo incondicional, como nadie más me lo brindó y que sé me lo brindará, incondicional solo tú, a ti mamá preciosa. Gracias por tu cariño, este trabajo es para ti.

A tu apoyo, a tus consejos, a la figura que representas para mí, me acercaste de una forma muy peculiar a la lectura, y hoy es parte de mí para ser mejor como profesionista y como persona, gracias papá.

A mi hermano, eres en este mundo la persona que mejor me conoce, espero estar juntos siempre tanto en las buenas como es ésta, como también en las malas.

A mi familia, que siempre ha compartido de alguna forma mis alegrías y mis fracasos: Jorge, Tía Elsa, y claro; a esa persona que marcó mi carácter y que sé estaría feliz de compartir esto con nosotros.

A mis compadres, Aarón, Colín, Laura, Miguel y Olmo, en estricto orden alfabético por aquello de los reclamos. Gracias por tantos viajes, tantos buenos momentos. Ustedes conocen mejor que nadie los pormenores que son parte de este ciclo. Por el apoyo en esas ocasiones en las que inevitablemente suele uno culpar a todo el mundo menos a uno mismo por algún examen reprobado, no se diga un laboratorio.

A muchas más personas que estuvieron en distintas etapas de mi carrera, importantes todas; Caro, Miryam, Rebeca, Chelita, Sandra, Wendy.

A mi familia en Puerto Escondido. A todos ellos, porque aunque estén lejos siempre sentí su apoyo y cariño cerca; en especial a mi tío Ale y a mi tía Betina.

A mis sinodales, por su crítica y apoyo que hace posible que este trabajo finalmente esté listo. A mi tutor, el Doctor Víctor Hugo Jacobo, que siempre me hizo un espacio para asesorarme e impulsarme a terminar esta tesis.

A mi Universidad, en la que conocí a tantas personas que han hecho de estos años una experiencia inolvidable, y en la que tuve la oportunidad de formarme con grandes profesores. Por ese espíritu universitario que siempre me acompañará, y por poder pertenecer a esta comunidad que exige de mí lo mejor.

México DF Febrero 2011

INTRODUCCIÓN

El sector de empresas del ramo industrial metalmecánico, como muchos otros ámbitos industriales, se ha visto envuelto en una competencia cada vez mayor y que exige una mejora continua en sus sistemas de producción y planificación.

Necesidades surgidas en el entorno industrial, específicamente en el sector dedicado a la fabricación de troqueles, fueron planteadas a algunos miembros de la Facultad de Ingeniería buscando una solución que les permitiera llevar a cabo una mejor y más eficiente organización de las actividades que estas empresas desarrollan. En una primera etapa se llevó a cabo un proyecto que generó un sistema para la programación de actividades en forma de un paquete de software. Esta programación de actividades (*schedulling*), que se refiere a la asignación de las tareas o actividades a algún recurso específico como puede ser una máquina, en un tiempo específico, se llevó a cabo basándose en criterios de despacho tradicionales, criterios que dictan la secuencia en la que se programarán un cierto grupo de actividades. En este punto se sugirió utilizar alguna técnica que reemplazara estos criterios con el fin de hacer más eficiente la búsqueda de una secuenciación para las actividades. Por las características del problema y la explosión combinatoria de su espacio de soluciones, se decidió utilizar la técnica conocida como Algoritmos Genéticos. La técnica de los algoritmos genéticos ha sido aplicada a la resolución de problemas con este tipo de explosión combinatoria en su campo de soluciones, esta técnica, basada en los mecanismos de selección que utiliza la naturaleza, mediante los cuales los individuos más aptos de una población son los que sobreviven al adaptarse más fácilmente a los cambios que se producen en su entorno, es la técnica que se pretende utilizar para la solución del problema en este trabajo, buscando comprometer la calidad de las soluciones en el problema de secuenciación de actividades aspirando a soluciones cuasióptimas, con lo que se alcanzarían soluciones en intervalos de tiempo admisibles.

HIPÓTESIS

Mediante el uso de la técnica de Algoritmos Genéticos, en específico una adaptación propuesta por R. Cheng, M. Gen y Y. Tsujimura [6], *"A tutorial survey of job-shop schedulling problems using genetic algorithms, part II hybrid genetic search strategies"*1999, para resolver un problema clásico de asignación de actividades para un sistema de producción intermitente (*Job Shop schedulling Problem*), se generará un algoritmo genético para resolver una variante del mismo problema, en el cual se permite a una actividad ser procesada por cualquier máquina de un conjunto especifico, *(Flexible Job Shop Problem)*.

La secuencia estará basada en los trabajos pertenecientes a un taller de troqueles, el cual cumple con el sistema de producción establecido.

JUSTIFICACIÓN

El problema de Programación de actividades para un sistema de producción Intermitente (Job Shop Scheduling Problem) , está clasificado según la teoría de la complejidad computacional como un problema *NP-complejo,* es decir que el tiempo de cómputo que se requiere para resolver uno de estos problemas se incrementa conforme crece el tamaño del problema presentando una dependencia funcional tal que no admite ser acotada por un polinomio, o lo que es lo mismo, su tiempo de ejecución no está en función del número de valores de entrada (*Garey y Johnson (1979)*).

Esto hace de este problema intratable mediante técnicas de cálculo, y es aquí donde los algoritmos heurísticos (o de aproximación), se presentan como un método útil para su resolución. Los algoritmos de aproximación, dentro de los cuales existen los *de* búsqueda local (*local search*), utilizan el concepto de entorno (*neighborhood*), el cual es usado para guiar la búsqueda iterativa hacia buenas soluciones. Cuando se usan para problemas de optimización para maximizar una función objetivo resultan menos afectados por los máximos locales (falsas soluciones) que las técnicas tradicionales. Los algoritmos genéticos basan su búsqueda en dicho concepto, por lo cual es posible aplicarlos a este estudio.

CAPÍTULO 1

PROGRAMACIÓN DE ACTIVIDADES (Scheduling)

La *programación de actividades* en un sistema productivo se refiere al proceso de organizar, elegir y dar tiempos al uso de recursos para llevar a cabo todas las actividades necesarias, para producir las salidas deseadas en los tiempos deseados, satisfaciendo a la vez un gran número de restricciones de tiempo y relaciones entre las actividades y los recursos." (Morton y Pentico (1993). Un programa especifica el tiempo en el que comienza y termina cada trabajo en cada máquina. Una secuencia es un orden simple de trabajos, 3-1-2 indica que el trabajo 3 se hace primero, el 1 es el segundo y el 2 es el último. De esta forma la secuencia determina los tiempos de inicio y terminación y, por lo tanto, determina la programación.

Los datos que se utilizan para el estudio de la *programación de actividades*, son datos conocidos con certeza y con antelación, o lo que es igual se manejan datos determinísticos. Un trabajo consiste en varias etapas de proceso, llevadas a cabo en diferentes máquinas o centros de procesamiento, determinados por los requerimientos de cada etapa. Las decisiones en la *programación de actividades* deben resolver el *cuando,* y el *cual* en orden que las medidas de desempeño tales como la tardanza, inventario en proceso, y tardanza máxima (*makespan*) sean minimizados.

En la programación de actividades se identifican algunos tipos de problemas diferenciados entre sí por la naturaleza del flujo de materiales a través de los recursos disponibles (máquinas), estos se describen a continuación:

- **Problemas con una sola máquina**

Para los ***problemas en una sola máquina***, deben procesarse en ella todos los trabajos. La máquina puede procesar a lo más un trabajo a la vez. En este tipo de problema solo se busca una secuencia que dicta en qué orden se introducirán los trabajos a dicha máquina.

- **Problemas con Máquinas Paralelas**

Varias máquinas que pueden realizar el mismo tipo de procesamiento se llaman ***máquinas paralelas***. Un trabajo se puede procesar en cualquiera de las máquinas, y una vez procesado por cualquiera de ellas, queda terminado. A menos que se diga lo contrario, se supone que todas las máquinas son idénticas. El tiempo para procesar un trabajo en una de varias máquinas idénticas es independiente de qué máquina lo haga.

- **Problemas con un Sistema de Producción continua**
  (Flow Shop Problems)

Un taller de producción continua contiene máquinas diferentes. Cada trabajo debe procesarse en cada máquina exactamente una vez. Más aún, todos los trabajos siguen la misma ruta; esto es, deben visitar las máquinas en el mismo orden.

- **Problemas con un Sistema de Producción intermitente**
  (Job Shop Problems)

Un taller de producción intermitente es más general que el de producción continua:

Cada trabajo o pieza a realizar está formado por un conjunto de operaciones , cada una de las cuales deben ser procesadas en una determinada máquina durante un tiempo preestablecido de antemano. Cada trabajo tiene su propio flujo, es decir la secuencia en que visitan cada una de las máquinas no tiene que ser igual al de los otros.

- **Problemas con un sistema de Producción intermitente flexible**
  (Flexible Job Shop Problems)

Este tipo de problemas son una variante de los problemas para un sistema de producción intermitente. La diferencia se basa en que una actividad tiene asignado un grupo de máquinas en las cuales podrá ser procesada y no solo una máquina en específico. Por lo tanto, además de una secuenciación se tiene que realizar un proceso de asignación.

Este tipo de problema se ajusta de manera adecuada a los problemas reales de casi todos los tipos de manufactura y en todos los sectores, incluyendo el que concierne a este trabajo.

1.1. El problema de la programación de actividades para un sistema de producción intermitente flexible (FJSSP)

El problema de secuenciación de FJSS presenta una serie de variantes dependiendo de la naturaleza y el comportamiento; tanto de las operaciones como de las máquinas. Una de las variantes más difíciles de plantear, debido a su alta complejidad computacional, es aquella en donde las tareas son dependientes y las máquinas son diferentes. En esta variante cada trabajo presenta una lista de operaciones que lo preceden y para ser ejecutado debe de esperar el procesamiento de dicha lista en su totalidad. A esta situación hay que agregarle la característica de heterogeneidad de las máquinas: cada tarea demora tiempos distintos de ejecución en cada máquina. El objetivo más extensamente utilizado en la planeación de la producción es minimizar el tiempo acumulado de ejecución de las máquinas conocido en la literatura como *makespan*.

En un principio, existen un número infinito de programas posibles para un FJSSP debido a que un tiempo ocioso cualquiera puede ser insertado entre dos operaciones. Alternativamente se pueden acomodar las operaciones a la izquierda (en el eje del tiempo), para hacer el programa lo más compacto posible.

El problema de FJSS suele ser definido por las siguientes condiciones [3]:

- Hay *n* trabajos con subíndice *i*, y estos trabajos son independientes entre ellos.
- Cada trabajo *i* tiene una secuencia de operación, denotada por J_i
- Cada trabajo consiste de una o más operaciones $O_{i,j}$
- Cada secuencia de operación esta ordenada por un juego de operaciones $O_{i,j}$
- Hay *m* máquinas con subíndice *K* (La k-ésima máquina es denotada por m_k)
- Para cada operación $O_{i,j}$ hay un juego de máquinas capaces de cumplir con la función objetivo. Este juego de máquina es denotado por $U_{i,j}$
- El tiempo de procesamiento P_i de una operación $O_{i,j}$ en una máquina K es predefinido y mayor que cero.

Restricciones Generales.

- Cada operación no puede ser interrumpida durante el cumplimiento de la ejecución de la misma
- Todas las máquinas están disponibles en el tiempo t=0
- Cada máquina K, no puede procesar más de una operación simultáneamente.

1.2. *Técnicas utilizadas para la resolución del FJSSP*

Las técnicas para resolver un problema de programación de actividades para un sistema de producción intermitente flexible (FJSSP) pueden ser clasificadas en dos categorías: soluciones con planteamientos jerárquicos y soluciones con planteamientos integrados.

a) *Planteamiento jerárquico* trata de resolver el problema por descomposición de una secuencia en sub-problemas, y con ello reducir la dificultad. Esta descomposición consiste en realizar la asignación de cada actividad a la máquina en la cual se procesará y posteriormente realizar la secuenciación, como si se tratara de un ejercicio de JSS. Como ejemplo se puede realizar la asignación utilizando reglas de despacho, y luego resolver el resultado de JSSP usando algún algoritmo como la búsqueda Tabú.
b) Por otra parte el *planteamiento integrado*, consiste en realizar de forma integrada la asignación y la secuenciación, lo cual implica una mayor dificultad para su resolución, pero que también arroja mejores resultados.

La parte que corresponde a la asignación de las actividades para el FJSSP se puede atacar como se dijo antes con reglas de despacho, o con cualquier técnica propia de un problema con máquinas paralelas. En cuanto a la secuenciación, las técnicas no son muy diferentes de la forma en que se atacan los problemas de JSS, y las más comunes se describen a continuación.

Algoritmo Heurístico de Despacho

El enfoque más común para la producción intermitente es usar reglas de despacho con prioridades. La idea básica es programar una operación de un trabajo tan pronto como se pueda; si hay más de un trabajo que espera ser procesado por la misma máquina, se programa el que tiene la mayor prioridad.

La prioridad, con frecuencia expresada en forma numérica, se utiliza para determinar la secuencia en que deben procesarse las órdenes. Las reglas descritas a continuación son tal vez las más comunes, aunque hay muchas variaciones y combinaciones de éstos métodos.

La lista que aparece en el cuadro 1.1 proporciona una buena visión general de las reglas básicas y sus objetivos.

Existen otras prioridades y en la literatura se encuentran algunas pequeñas variaciones en las definiciones. La prioridad usada depende de lo que se quiere de la medida del programa.

Cuadro 1.1 Reglas de despacho tradicionales y su descripción

REGLA	OBJETIVO
PEPS (Primeras entradas primeras salidas)	Producir órdenes en la secuencia en la que llegan al centro de trabajo. Esta regla de "justicia" es especialmente apropiada en las organizaciones de servicio donde la mayor parte de los clientes con frecuencia necesita o desea la terminación del servicio tan pronto como sea posible.
MTP-MTO (El menor tiempo de proceso)	Producir órdenes en razón inversa al tiempo requerido para procesarlas en el departamento (primero el tiempo más breve). Esta regla da como resultado menor producción en proceso, menor promedio para terminación del trabajo y menor retraso promedio. A menos que esta regla se combine con la fecha de entrega o la regla de tiempo ocioso, los trabajos (ordenes) con tiempos largos de procesamiento pueden estar listos muy tarde.
MTPT, tiempo remanente del tiempo de procesamiento total mas corto.	Producir órdenes en razón inversa al tiempo remanente del procesamiento total; la lógica de esta regla es similar a la precedente. Cumple con objetivos similares cuando la mayor parte de los trabajos siguen un proceso común.
FEMP, fecha de entrega mas próxima	Producir órdenes con la primera fecha de entrega más inmediata. Esta regla trabaja bien cuando los tiempos de procesamiento son aproximadamente los mismos.
MO, Menos operaciones	Producir primero las órdenes con menos operaciones remanentes. La lógica de esta regla es que menos operaciones abarcan menor tiempo de cola y, como resultado, la regla reduce la producción promedio en proceso, el tiempo de obtención de la producción y el retraso promedio. No obstante los trabajos con un número relativamente grande de operaciones pueden abarcar tiempos excesivamente largos.
ST, Tiempo de holgura	Producir primero la orden con menor tiempo de holgura y continuar la secuencia en orden ascendente de tiempos de holgura. El tiempo de holgura es igual a la fecha de entrega menos el tiempo restante de procesamiento. Esta regla favorece el objetivo de fecha de entrega. El tiempo de holgura restante por operación es una variante de esta regla.
CR, Tasa Crítica	Para órdenes todavía no retrasadas (vencidas) producir primero las órdenes con menor tasa crítica. La tasa crítica es igual a la fecha de entrega menos la fecha actual, dividida entre el tiempo restante de

	conducción de la producción.

Algoritmo heurístico de Enfriamiento Simulado (Simulated annealing)

El *enfriamiento simulado* (Kirkpatrick, Gelatt y Vecchi, 1983; Cerny, 1985) se basa en una analogía con el proceso físico de recocido, en el cual se forma una estructura reticular en un sólido calentándolo por encima de su temperatura crítica seguido de un enfriamiento lento hasta llegar a un estado de baja energía.

Desde el punto de vista de la optimización combinatoria, el *enfriamiento simulado* es un algoritmo aleatorio de búsqueda local. Las soluciones vecinas de mejor costo son siempre aceptadas e incluso se aceptan las soluciones de peor costo, aunque con una probabilidad que decrece gradualmente durante el transcurso de la ejecución del algoritmo. Esta reducción de la tolerancia o probabilidad de aceptación es controlada por un conjunto de parámetros cuyos valores son determinados por un esquema de enfriamiento prescrito.

El *enfriamiento simulado* ha sido aplicado con considerable éxito. Su naturaleza aleatoria permite convergencia asintótica a la solución óptima bajo condiciones moderadas. Desafortunadamente, la convergencia requiere típicamente de tiempo exponencial, convirtiendo el *enfriamiento simulado* en impráctico como instrumento para procurarse soluciones óptimas.

Algoritmo Heurístico Búsqueda Tabú (Tabu Search)

La *Búsqueda Tabú* (Glover, 1990) combina un algoritmo determinístico de avance iterativo con la posibilidad de aceptar soluciones que incrementen el costo. De esta manera, la búsqueda es dirigida fuera de óptimos locales y otras partes del espacio de soluciones pueden ser exploradas.

Cuando es visitada una nueva solución, es considerada un vecino legítimo de la solución actual aún cuando empeore el costo.

El conjunto de vecinos legítimos queda representado por una lista tabú, cuyo objetivo es restringir la elección de soluciones para prevenir el regreso a puntos recientemente visitados. La lista tabú es actualizada dinámicamente durante la ejecución del algoritmo y define soluciones que no son aceptables en unas pocas siguientes iteraciones. Sin embargo, una solución de la lista tabú puede ser aceptada si su calidad, en algún sentido, elevada lo justifica. En este caso se establece que se ha alcanzado cierto nivel de aspiración *(aspiration level).*

La *Búsqueda tabú* ha sido aplicada en una gran variedad de problemas con considerable éxito ya que presenta un esquema de gran adaptabilidad que se ajusta a los detalles del problema considerado. Por otro lado, existen vagos conocimientos teóricos que guíen ese proceso de ajuste y cada usuario debe recurrir a la información práctica disponible y a su propia experiencia.

OPT/TOC

Un enfoque para atacar el JSSP está dado por una visión integral de la Planeación y Control de la Producción, la cual contiene una componente técnica y un concepto administrativo. La componente técnica de este enfoque es un programador de recursos restricciones **(cuello de botella)** conocido como *optimized production technology* (OPT) o *tecnología de producción optimizada*. El concepto administrativo se llama *teoría de restricciones* (TOC), y abarca no solo la programación de los trabajos en un taller, sino objetivos como lo son aumentar la producción, reducir el inventario y disminuir los gastos operativos.

El OPT es en esencia un sistema de retroalimentación en el que una vez identificados los cuellos de botella se realiza la programación de los trabajos asignando una secuencia para este recurso, en seguida se programan las operaciones hacia atrás, desde el recurso en cuestión hasta el punto de despacho de la materia prima y, después, hacia adelante hasta el envío.

El problema para obtener la secuencia de actividades escogida para ser procesada en la máquina cuello de botella es un problema de optimización combinatoria del tipo NP-Complejo, en el cual se puede utilizar un heurístico de despacho con diferentes reglas de prioridades o algún otro tipo de algoritmo heurístico.

Algoritmos genéticos

La técnica de los Algoritmos genéticos, inicialmente desarrollada por John Holland en el año de 1975 cobró fuerza para la resolución de problemas del tipo NP-Complicados. La técnica solo fue utilizada para la resolución de problemas de programación de actividades hasta finales de la década de los ochentas y principios de los noventas, *Davis*, *L*. (*1985*). *Job shop scheduling with* genetic algorithms*,* Falkenauer E. y S Bouffouix (1991), *A genetic Algorithm For Job shop*. El buen desempeño de los algoritmos genéticos para este tipo de problemas permitió que se incursionara en un problema más complejo, que es una particularidad del problema de JSS. El problema para la programación de actividades para un sistema de producción intermitente flexible (FJSSP) ha sido tratado con distintas variantes de los algoritmos genéticos. Algunas de las más recientes la desarrollada por Ho, N. B. y Tay, J. C., 2008. *"Evolving dispatching rules using genetic programming for solving multiobjective flexible jobshop problems"*

CAPÍTULO 2

ALGORITMOS GENÉTICOS

2.1. *Imitación del proceso evolutivo*

En la naturaleza los individuos de una población compiten entre sí en la búsqueda de recursos tales como comida, agua y refugio. Incluso los miembros de una misma especie compiten a menudo en la búsqueda de un compañero. Aquellos individuos que tienen más éxito en sobrevivir y en atraer compañeros tienen mayor probabilidad de generar un gran número de descendientes. Por el contrario individuos poco dotados producirán un menor número de descendientes. Esto significa que los genes de los individuos mejor adaptados se propagarán en sucesivas generaciones hacia un número de individuos creciente. La combinación de buenas características provenientes de diferentes ancestros, puede a veces producir descendientes "superindividuos", cuya adaptación es mucho mayor que la de cualquiera de sus ancestros. De esta manera, las especies evolucionan logrando unas características cada vez mejor adaptadas al entorno en el que viven.

Los Algoritmos Genéticos (AG's) usan una analogía directa con el comportamiento natural. Trabajan con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado. A cada individuo se le asigna un valor ó puntuación, relacionado con la bondad de dicha solución. En la naturaleza esto equivaldría al grado de efectividad de un organismo para competir por unos determinados recursos. Cuanto mayor sea la adaptación de un individuo al problema, mayor será la probabilidad de que el mismo sea seleccionado para reproducirse, cruzando su material genético con otro individuo seleccionado de igual forma. Este cruce producirá nuevos individuos descendientes de los anteriores, los cuales comparten algunas de las características de sus padres. Cuanto menor sea la adaptación de un individuo, menor será la probabilidad de que dicho individuo sea seleccionado para la reproducción, y por tanto de que su material genético se propague en sucesivas generaciones.

De esta manera se produce una nueva población de posibles soluciones, la cual reemplaza a la anterior y verifica la interesante propiedad de que contiene una mayor proporción de buenas características en comparación con la población anterior. Así a lo largo de las generaciones las buenas características se propagan a través de la población. Favoreciendo el cruce de los individuos mejor adaptados, van siendo exploradas las áreas más prometedoras del espacio de búsqueda. Si el AG ha sido bien diseñado, la, población convergerá hacia una solución óptima del problema.

De manera más formal, los AG's son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza de acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin. Por imitación de este proceso, los AG's son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas.

Un AG consiste en una función matemática o una rutina de software que toma como entradas a los ejemplares y retorna como salidas cuales de ellos deben generar descendencia para la nueva generación. Versiones más complejas de AG's generan un ciclo iterativo que directamente toma a la especie (el total de los ejemplares) y crea una nueva generación que reemplaza a la antigua una cantidad de veces determinada por su propio diseño. Una de sus características principales es la de ir perfeccionando su propia heurística en el proceso de ejecución, por lo que no requiere largos períodos de entrenamiento especializado por parte del ser humano, principal defecto de otros métodos para solucionar problemas, como los Sistemas Expertos.

2.2. ***Limitaciones de los Algoritmos Genéticos***

El poder de los AG's proviene del hecho de que se trata de una técnica robusta, y pueden tratar con éxito una gran variedad de problemas provenientes de diferentes áreas, incluyendo aquellos en los que otros métodos encuentran dificultades. Si bien no se garantiza que el AG encuentre la solución óptima, del problema, existe evidencia empírica de que se encuentran soluciones de un nivel aceptable, en un tiempo competitivo con el resto de algoritmos de optimización combinatoria. En el caso de que existan técnicas especializadas para resolver un determinado problema, lo más probable es que superen al AG, tanto en rapidez como en eficacia. El gran campo de aplicación de los Algoritmos Genéticos se relaciona con aquellos problemas para los cuales no existen técnicas especializadas. Incluso en el caso en que dichas técnicas existan, y funcionen bien, pueden efectuarse mejoras de las mismas hibridándolas con los Algoritmos Genéticos.

2.3. ***Factibilidad de un Algoritmo Genético***

La aplicación más común de los algoritmos genéticos ha sido la solución de problemas de optimización, en donde han mostrado ser muy eficientes y confiables. Sin embargo, no todos los problemas pudieran ser apropiados para la técnica, y en general se deben tomar en cuenta las siguientes características del mismo antes de intentar usarla:

- *Su espacio de búsqueda (i.e., sus posibles soluciones) debe estar delimitado dentro de un cierto rango.*

  El primer punto es muy importante, y lo más recomendable es intentar resolver problemas que tengan espacios de búsqueda discretos aunque éstos sean muy grandes. Sin embargo, también podrá intentarse usar la técnica con espacios de búsqueda continuos, pero preferentemente cuando exista un rango de soluciones relativamente pequeño.

- *Debe poderse definir una función de aptitud que indique qué tan buena o mala es una cierta respuesta.*

La **función de aptitud** no es más que la función objetivo de un problema de optimización. El AG únicamente maximiza, pero la minimización puede realizarse fácilmente utilizando el recíproco de la función maximizante (debe cuidarse, por supuesto, que el recíproco de la función no genere una división por cero). Una característica que debe tener esta función es que tiene ser capaz de "castigar" a las malas soluciones, y de "premiar" a las buenas, de forma que sean estas últimas las que se propaguen con mayor rapidez.

- *Las soluciones deben codificarse de una forma que resulte relativamente fácil de implementar en la computadora.*

La **codificación** más común de las soluciones es a través de cadenas binarias, aunque se han utilizado también números reales y letras. El primero de estos esquemas ha gozado de mucha popularidad debido a que es el que propuso originalmente Holland, y además porque resulta muy sencillo de implementar.

2.4. *Variables y operadores de un Algoritmo Genético*

La búsqueda que realiza un AG está caracterizada por algunos parámetros que determinan que tan amplio será el espacio de soluciones que abarque, en cuanto tiempo encontrará una solución para el problema, que tan poderoso sea el algoritmo para no estancarse en óptimos locales, y en general que tan buen o mal desempeño tenga.

En la literatura se encuentran recomendaciones para ajustar estos parámetros, pero no existe un consenso que permita afirmar cuales son los mejores. En seguida se presenta una descripción de estos:

Tamaño de la población

Población es el conjunto de elementos de referencia sobre el que se realizan las observaciones. Parece intuitivo que las poblaciones pequeñas corren el riesgo de no cubrir adecuadamente el espacio de búsqueda, mientras que el trabajar con poblaciones de gran tamaño puede acarrear problemas relacionados con el excesivo costo computacional.

Población inicial

Si se conocen soluciones de las cuales se pretenda partir la búsqueda, se introduce desde un principio una *población inicial* conocida, de lo contrario, como se realiza habitualmente, la población inicial se escoge generando cadenas de genes al azar, pudiendo contener cada gen uno de los posibles valores del alfabeto con probabilidad uniforme. En algunas investigaciones donde los valores no son generados al azar se constata que se puede acelerar la convergencia del AG, sin embargo en algunos casos es una desventaja ya que se llega a una convergencia prematura del algoritmo.

Función de Evaluación

Un buen diseño de la función de evaluación (también conocida como función objetivo o función de adaptación) resulta extremadamente importante para el correcto funcionamiento de un AG. Esta función determina el grado de adaptación o aproximación de cada individuo al problema y por lo tanto permite distinguir a los mejores individuos de los peores.

Dos aspectos que resultan cruciales en el comportamiento de los Algoritmos Genéticos son la determinación de una adecuada función de adaptación o función objetivo, así como la codificación utilizada.

Por función se entiende la relación de una magnitud con una o otras variables de las cuales depende para tomar su valor, por ejemplo:

Función objetivo $f1(x1,x2,....x_n)$*, las variables estando sujetas a restricciones*

$f2(x1,x2,...x_n)= 0$ *o* $f2(x1,x2,...x_n) \geq 0$ *...)*

Selección

Existen distintos métodos para la selección de los padres que serán cruzados para la creación de nuevos individuos, uno de los más utilizados se denomina función de selección proporcional a la función de evaluación. Se basa en que la probabilidad de que un individuo sea seleccionado como padre es proporcional al valor de su función de evaluación. Esto hace que los mejores individuos sean los seleccionados para el proceso de reproducción. Esta técnica puede producir el inconveniente de que la población converja prematuramente hacia un resultado óptimo local. Esto significa que pueden aparecer “superindividuos” muy similares entre si, de forma que la diversidad genética sea bastante pobre y el algoritmo se estanque en una solución buena (óptimo local) pero no la mejor.

Otro método de selección es el de la Ruleta, consiste en asignar una porción de una “ruleta” a cada individuo de la población, de forma que el tamaño de cada porción sea proporcional a su desempeño. Los mejores individuos dispondrán de una porción mayor y por lo tanto de más posibilidades de ser seleccionados. El método de Torneo consiste en hacer competir a los

individuos en grupos aleatorios (normalmente parejas), el que tenga el mejor desempeño será el ganador. En el caso de competición por parejas se deben realizar dos torneos. Con este método de selección se asegura de que al menos dos copias del mejor individuo de la población actuarán como progenitores para la siguiente generación.

Cruzamiento

Consiste en el intercambio de material genético entre dos cromosomas (individuos). El

cruzamiento es el principal operador genético, hasta el punto que se puede decir que no es un AG si no tiene ***cruzamiento***, y, sin embargo, puede serlo perfectamente sin mutación, según descubrió Holland. El teorema de los esquemas confía en él para hallar la mejor solución a un problema, combinando soluciones parciales. Para aplicar el ***cruzamiento***, entrecruzamiento o recombinación, se escogen aleatoriamente dos miembros de la población. No pasa nada si se emparejan dos descendientes de los mismos padres; ello garantiza la perpetuación de un individuo con buena puntuación (y, además, algo parecido ocurre en la realidad; es una práctica utilizada, por ejemplo, en la cría de ganado, llamada *inbreeding*, y destinada a potenciar ciertas características frente a otras). Sin embargo, si esto sucede demasiado a menudo, puede crear problemas: toda la población puede aparecer dominada por los descendientes de algún gen, que, además, puede tener caracteres no deseados. Esto se suele denominar en otros métodos de optimización *atranque en un mínimo local*, y es uno de los principales problemas con los que se enfrentan los que aplican algoritmos genéticos. En cuanto al teorema de los esquemas, se basa en la noción de bloques de construcción (*Schematta*). Una buena solución a un problema está constituido por unos buenos bloques, igual que una buena máquina está hecha por buenas piezas. El ***cruzamiento*** es el encargado de mezclar bloques buenos que se encuentren en los diversos progenitores, y que serán los que den a los mismos una buena puntuación. La presión selectiva se encarga de que sólo los buenos bloques se perpetúen, y poco a poco vayan formando una buena solución. El teorema de los esquemas viene a decir que la cantidad de buenos bloques se va incrementando con el tiempo de ejecución de un AG, y es el resultado teórico más importante en algoritmos genéticos.

El intercambio genético se puede llevar a cabo de muchas formas, pero hay dos grupos

Principales:

Cruzamiento por N-puntos:

Los dos cromosomas (individuos) se cortan por n puntos, y el material genético situado entre ellos se intercambia. Lo más habitual es un ***cruzamiento*** de un punto o de dos puntos. En el operador de cruce basado en dos puntos, los cromosomas (individuos) pueden contemplarse como un circuito en el cual se efectúa la selección aleatoria de dos puntos, tal y como se indica en la figura 2.1.

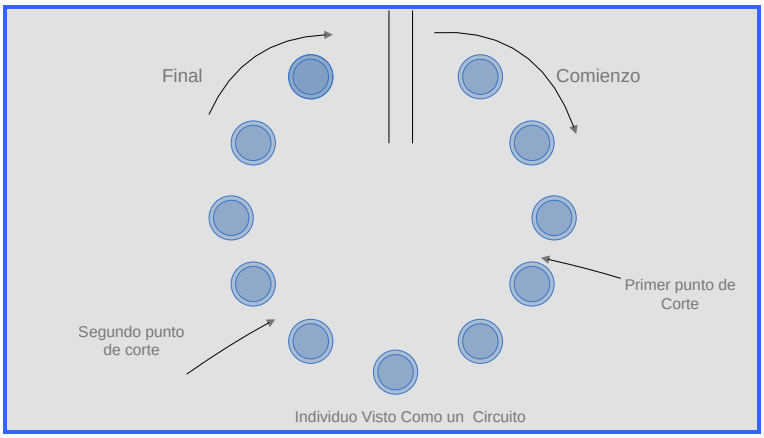


Figura 2.1 Cruzamiento por *n* puntos

En la figura 2.2 se muestra un ejemplo de este tipo de ***cruzamiento***.

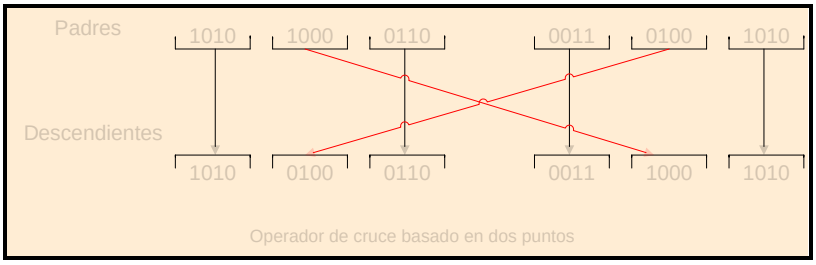


Figura 2.2 Cruce basado en dos puntos

El cruce basado en dos puntos, representa una mejora mientras que añadir más puntos de cruce no beneficia el comportamiento del algoritmo. La ventaja de tener más de un punto de cruce radica en que el espacio de búsqueda puede ser explorado más fácilmente, siendo la principal desventaja el hecho de aumentar la probabilidad de ruptura de buenos esquemas.

Cruzamiento uniforme:

Se genera un patrón aleatorio de unos y ceros, y se intercambian los bits de los dos cromosomas (individuos) que coincidan donde hay un 1 en el patrón. O bien se genera un número aleatorio para cada bit, y si supera una determinada probabilidad se intercambia ese bit entre los dos cromosomas. Este tipo de ***cruzamiento*** de muestra en la figura 2.3.

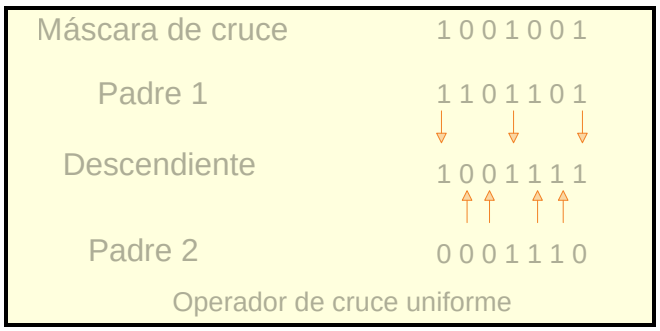


Figura 2.3 Cruce uniforme

Si tuviésemos en cuenta el valor de la función de adaptación de cada padre en el momento de generar la "máscara de cruce", de tal manera que cuanto mayor sea la función de adaptación de un individuo, más probable sea heredar sus características, se podría definir un operador de cruce basado en la función objetivo, en el cual la "máscara de cruce" se interpreta como una muestra aleatoria de tamaño *l* proveniente de una distribución de Bernoulli de parámetro P obtenido de:

$$p = g\left(I_t^j\right) / \left(g\left(I_t^j\right) + g\left(I_t^i\right)\right)$$

Ecuación 2.1

Función

Donde (I superíndice j subíndice t) y I(superíndice i subíndice t) denotan los padres seleccionados para ser cruzados.

Mutación

La mutación se considera un operador básico, que proporciona un pequeño elemento de aleatoriedad en la vecindad (entorno) de los individuos de la población. Si bien se admite que el operador de cruce es el responsable de efectuar la búsqueda a lo largo del espacio de posibles soluciones, también parece desprenderse de los experimentos efectuados por varios investigadores que el operador de mutación va ganando en importancia a medida que la población de individuos va convergiendo.

La búsqueda del valor óptimo para la probabilidad de mutación, es una cuestión que ha sido motivo de varios trabajos. Así, De Jong recomienda la utilización de una probabilidad de mutación del bit de (l^{-1}), siendo l la longitud de la cadena. Schaffer y Col. Utilizan resultados experimentales para estimar la tasa óptima proporcional a l /($l^{-0.9318}$),($l^{-0.4535}$), donde λ denota el número de individuos en la población.

Una vez establecida la frecuencia de mutación, por ejemplo, uno por mil, se examina cada bit de cada cadena cuando se vaya a crear la nueva criatura a partir de sus padres (normalmente se hace de forma simultánea al ***cruzamiento***). Si un número generado aleatoriamente está por debajo de esa probabilidad, se cambiará el bit (es decir, de 0 a 1 o de 1 a 0). Si no, se dejará como está. Dependiendo del número de individuos que haya y del número de bits por individuo, puede resultar que las mutaciones sean extremadamente raras en una sola generación.

No hace falta decir que no conviene abusar de la mutación. Es cierto que es un mecanismo generador de diversidad, y, por tanto, la solución cuando un AG está estancado, pero también es cierto que reduce el AG a una búsqueda aleatoria. Siempre es más conveniente usar otros mecanismos de generación de diversidad, como aumentar el tamaño de la población, o garantizar la aleatoriedad de la población inicial. Si bien en la mayoría de las implementaciones de Algoritmos Genéticos se asume que tanto la probabilidad de cruce como la de mutación permanecen constantes, algunos autores han obtenido mejores resultados experimentales modificando la probabilidad de mutación a medida que aumenta el número de iteraciones.

Este operador, junto con la anterior y el método de selección de ruleta, constituyen un AG simple, SGA, introducido por Goldberg en su libro.

Reducción

Una vez obtenidos los individuos descendientes de una determinada población en el tiempo t, el proceso de reducción al tamaño original, consiste en escoger λ individuos, (donde λ (lambda) denota el número de individuos en la población) de entre los λ individuos que forman parte de la población en el tiempo t, y los λ individuos descendientes de los mismos. Dicho proceso se suele hacer fundamentalmente de dos formas distintas. O bien los λ individuos descendientes son los que forman parte de la población en el tiempo *t + 1*, es lo que se denomina reducción simple, o bien se escogen de entre los 2 λ individuos, los λ individuos más adaptados al problema, siguiendo lo que podemos denominar un criterio de reducción elitista de grado λ.

El concepto de reducción está ligado con el de tasa de reemplazamiento generacional, (t_{gr}) es decir en el porcentaje de hijos generados con respecto del tamaño de la, población.

Si bien en la idea primitiva de Holland dicho reemplazamiento se efectuaba, de l en 1, es decir $(t_{gr}) = (l^{-1})$, habitualmente dicho reemplazamiento se efectúa en bloque, $(t_{gr}) = 1$. De Jong introdujo el concepto de tasa de reemplazamiento generacional con el objetivo de efectuar un solapamiento controlado entre padres e hijos. En su trabajo, en cada paso una proporción de la población es seleccionada para ser cruzada. Los hijos resultantes podrán reemplazar a miembros de la población anterior.

El Doctor Zbigniew Michalewicz (Genetic Algorithms +Data Structures= Evolution Programs), introduce un algoritmo que denomina AG Modificado, (MOD sub GA), en el cual para llevar a cabo el reemplazamiento generacional, selecciona al azar r1 individuos para la reproducción, así como r2 individuos (distintos de los anteriores) destinados a morir. Estas selecciones aleatorias tienen en consideración el valor de la función objetivo de cada individuo, de tal manera que cuanto mayor es

la función objetivo, mayor es la probabilidad de que sea seleccionado para la reproducción, y menor es la probabilidad de que dicho individuo fallezca. El resto de las λ's. (r1 + r2) individuos son considerados como neutros y pasan directamente a formar parte de la población en la siguiente generación.

CAPÍTULO 3

CARACTERIZACIÓN DEL ALGORITMO GENÉTICO PARA LA SIMULACIÓN DEL FJSSP

3.1. ***Codificación de las soluciones***

La codificación de las soluciones para un *problema de programación de actividades para un sistema de Producción intermitente* o (*JSSP)*, es la base para la programación del algoritmo. De la codificación de las soluciones dependerán los principales operadores que lo componen. Una cadena de caracteres que define una solución puede tener mucha o poca información, en general, entre más sea la información contenida en una solución será más fácil la interpretación de ésta para transformarla en una secuencia de actividades. Por otro lado, la simplicidad en la codificación de las soluciones se reflejará en la simplicidad de la programación para el cruzamiento de los individuos, de la mutación de un individuo, o más aun, en la forma en que se evalúe su desempeño. Dicho de otra forma, una vez que se conoce la codificación que se utilizará, se puede pensar en cómo se cruzaran los individuos, como mutaran y como se evaluará su desempeño. Existen en general 2 formas de codificación para las soluciones, la forma Directa y la Forma indirecta. Cuando se codifica de forma directa, la secuencia de actividades se encuentra codificada en el cromosoma que forma a nuestro individuo. Cuando se trabaja con una codificación de tipo indirecta, los genes que componen el cromosoma o individuo pueden representar no las operaciones a programar, si no una lista de reglas de despacho que se busca acomodar de manera que aplicadas estas en un orden específico, se minimice un parámetro en especial.

Para la programación del Algoritmo Genético (AG), en este trabajo se utilizará la codificación desarrollada por Gen, Tsujimura y Kubota (1994). Esta codificación establece que el cromosoma de las soluciones represente en cada uno de sus genes una operación, se trata pues, de una codificación directa. Gen y sus colaboradores nombraron a todas las operaciones de cada trabajo con un símbolo idéntico y se interpretan de acuerdo al orden de aparición en el cromosoma que representa el programa de actividades. Para un trabajo con *m* máquinas y *n* trabajos el cromosoma tendrá *nxm* genes. El símbolo de cada trabajo aparecerá exactamente *m* veces. El ejemplo del cromosoma se muestra en la siguiente figura:

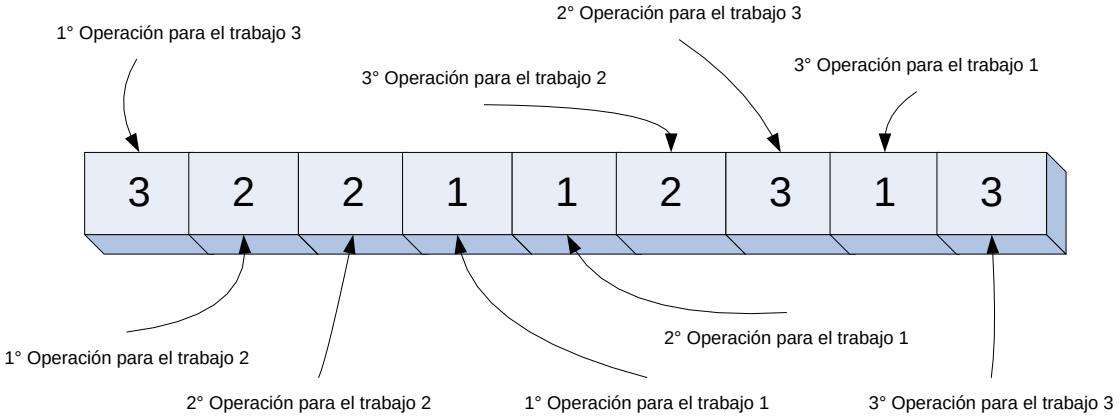


FIGURA 3.1 CROMOSOMA PARA UN ALGORITMO GENÉTICO, REPRESENTACIÓN BASADA EN OPERACIONES PARA EL JSP

Cualquier permutación de genes en esta codificación asegura que siempre se obtenga una secuenciación de actividades válida. Esta es la ventaja operacional que muestra este tipo de codificación basada en operaciones. La secuenciación obtenida para la codificación anterior se muestra en la figura no **3.2**.

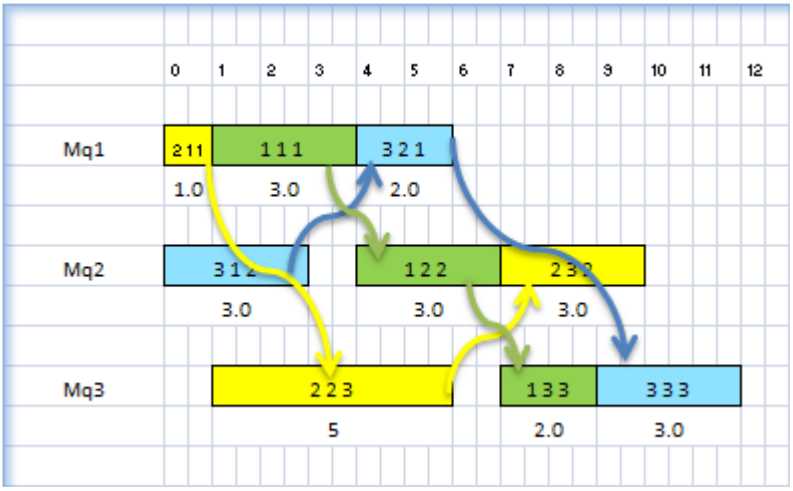


Figura 3.2 Secuencia obtenida para la solución [3 2 2 1 1 2 3 1 3]

3.2. Funciones y parámetros del algoritmo genético

La programación del AG se realizará utilizando la herramienta *Optimtool* de MATLAB versión R2009a la cual proporciona una interfaz con la capacidad de graficar el comportamiento del algoritmo y que permite probar con diferentes parámetros y escoger los que mejor rendimiento aporten al algoritmo genético. Esta herramienta permite programar las funciones de *cruzamiento*, *mutación*, la función de evaluación del *Intervalo Operativo* para las soluciones y la función para generar la población inicial, las cuales son particulares de la codificación utilizada.

Operador *Cruzamiento*

Este operador cruzamiento realiza el cruce de secuencias parciales por dos puntos, como se describió en el capítulo 2, a semejanza de los *bloques de construcción* (*building blocks)* utilizados por Holland (1975). Estos bloques son secuencias parciales dentro de las soluciones, que suponen buenas características de una solución, el buen funcionamiento de los algoritmos genéticos se basa en que conforme realizan la búsqueda de una solución óptima, los individuos con este tipo de bloques incrementan su número en cada generación de forma exponencial.

Las figuras 3.3 a 3.6 muestran los pasos para realizar el cruce de dichos cromosomas y la legalización de la progenie generada a partir de la operación *cruzamiento*, esta se encuentra descrita detalladamente a continuación.

Paso 1

En la figura no. 3.3 se muestran dos soluciones que representan dos secuencias validas para un JSSP con 4 máquinas y 4 trabajos, estos serán los padres **p1** y **p2** sometidos a cruzamiento.

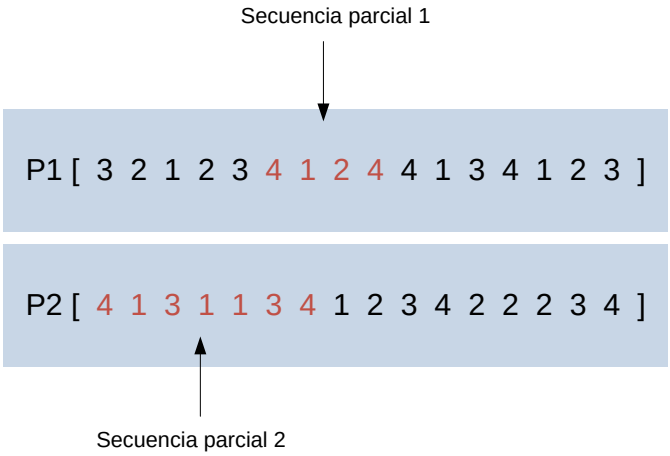


Figura 3.3 Paso 1,2 y 3: Selección de secuencias parciales

Se escoge una secuencia parcial en el primer padre de forma aleatoria. La primera posición en la sub-secuencia es la sexta. En esta posición se encuentra la indicación de una operación del trabajo 4.

Paso 2

Se encuentra el siguiente gen que represente al trabajo número 4 ,el cual se encuentra en la posición no. 9. Así, la sub-secuencia se compone de (4 1 2 4). Una secuencia parcial está identificada con el mismo trabajo en la primera y última posición del mismo segmento.

Paso 3

Se localiza la subsecuencia ahora para el padre **p2** como sigue. Esta será la primera cadena de caracteres localizada en el cromosoma **p2** que empiece y termine con el trabajo no. 4. Esta secuencia parcial 2 estaría compuesta por lo tanto de (4 1 3 1 1 3 4). Esta subsecuencia se muestra en la figura 3.3.

Paso no 4.

Realizar el intercambio de las dos secuencias parciales (4 1 2 4) y (4 1 3 1 1 3 4) para generar los nuevos individuos **o1** y **o2** como se muestra en la figura 3.4.

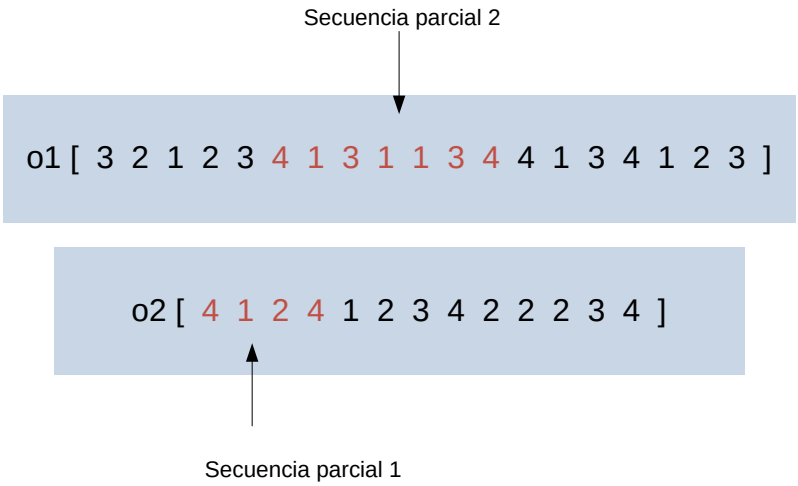


Figura 3.4 Paso 4: Intercambio de secuencias parciales

Paso no 5.

Generalmente las dos secuencias parciales encontradas tendrán un número diferente de genes, por lo tanto, los individuos generados inmediatamente después del cruce serán soluciones ilegales. Un individuo ilegal puede tener genes en exceso o carecer de algunos. Los genes faltantes y excedentes se muestran en la figura no. 3.5. El siguiente paso por tanto será reparar cada individuo borrando los genes en exceso o añadiendo los faltantes para así validar las nuevas soluciones.

En el ejemplo, a causa del *cruzamiento*, el nuevo individuo o1 ganó los genes extra 3, 1, 1 y 3, mientras ha perdido un gen 2. Por lo tanto los genes (3 1 1 y 3) tendrán que ser borrados y el gen faltante 2 tendrá que ser insertado en o1 (en la posición inmediatamente posterior a la nueva secuencia parcial) para así legalizar el nuevo individuo. Esta reparación se repite entonces con o2 para también legalizarlo.

Secuencia parcial 1 | 4 1 2 4

Reemplazado en p1 por | 4 1 3 1 1 3 4

Gen “2” faltante en o1, genes excedentes son “3”, “1”,”1" y “3”.

Secuencia parcial 2 | 4 1 3 1 1 3 4

Reemplazado en p2 por | 4 1 2 4

Genes “3”, “1”,”1" y “3” faltantes en o2 y gen excedente “2”.

Figura 3.5 Paso 5: Identificación de genes excedentes y faltantes

Borrado aleatorio de los genes excedentes “3”,”1",”1" y “3” sin disturbar la secuencia parcial insertada.

o1 [**3** 2 **1** 2 **3** 4 1 3 1 1 3 4 4 **1** 3 4 1 2 3]

o1 [2 2 4 1 3 1 1 3 4 4 3 4 1 2 3]

Inserción de el gen faltante “2” al término de la secuencia parcial.

o1 [2 2 4 1 3 1 1 3 4 **2** 4 3 4 1 2 3]

Figura 3.6 Paso 5: Validación del nuevo individuo **o1**

Este operador se programó en MATHLAB, realizando las operaciones que anteriormente se describen. En la interfaz de MATHLAB especializada en Algoritmos Genéticos, se hace referencia a esta función que se creó como un fichero M o M-File, el cual es un bloque que contiene alguna función programada en MATHLAB por es usuario y puede ser llamada por la herramienta de *Optimtool.*

Operador *Mutación*

Una vez que son seleccionados los individuos que han de someterse a este operador, se genera un número aleatorio entre 0 y 1, el cual decidirá si es que el individuo mutará alguno de sus elementos cuando el número es menor a 0.1 o pasará idéntico a la siguiente generación en el caso contrario. Esto se debe a que el operador mutación no debe de alterar de manera definitiva a un porcentaje de la población en todas las ocasiones. Es solo en un bajo porcentaje que ocasionalmente algún individuo es sujeto al intercambio de dos de sus elementos para generar un nuevo individuo que puede explorar un nuevo rango en el espacio solución del problema.

El operador para la mutación utilizado para este tipo de codificación es del tipo *intercambio de un par de trabajos (job pair exchange)* o *inversión de un bit,* esto es que dos trabajos son escogidos aleatoriamente para después cambiarlos de posición como se muestra en la figura 3.7. El resultado de esta operación siempre resultará en un nuevo individuo válido, por lo tanto no se requiere de alguna reparación después de la mutación.

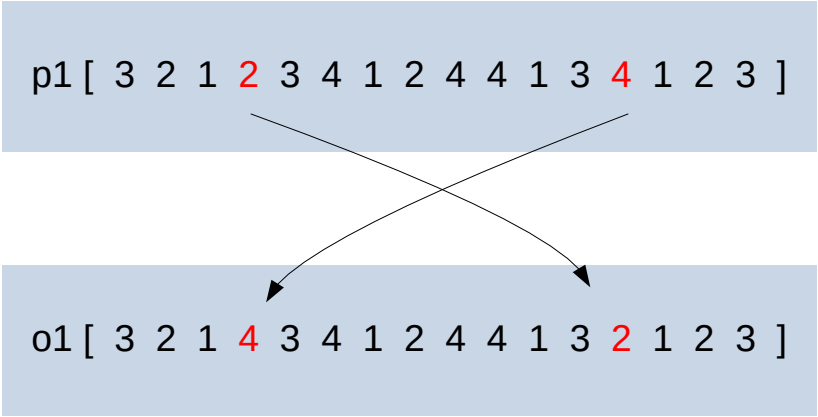


Figura 3.7 Mutación por intercambio de un par de trabajos.

Esta función también se creó como un fichero M y se muestra a continuación la programación, a diferencia del Fichero M de la función cruzamiento o el de la función de evaluación, es pequeño, por eso se muestra a continuación.

FICHERO M CON LA FUNCIÓN MUTACIÓN

```
function mutationChildren =
mymutfuncb1(parents,options,GenomeLength,FitnessFcn,state,thisScore,thisPopulation,mutation
Rate)
 if(strcmpi(options.PopulationType,'doubleVector'))
  mutationChildren = zeros(length(parents),GenomeLength);
  parents
      for i=1:length(parents)
       child = thisPopulation(parents(i),:);
```

```
      t=rand
      if t>.8

      mutationPoints=[0 0];
            while mutationPoints(1,1)==mutationPoints(1,2)
                  for j=1:2
                  m=rand(1,GenomeLength);
                  mutationPoints(1,j)=find(m==max(m));
                  end
            end

     a=child(1,mutationPoints(1,1));
     b=child(1,mutationPoints(1,2));
     child(1,mutationPoints(1,1))=b;
     child(1,mutationPoints(1,2))=a;

      mutationChildren(i,:) = child;
       else
mutationChildren(i,:) = child;
thisPopulation;
       end
      end
     end
 end
```

Función Generación de población inicial

La generación de la población inicial se tiene que programar debido a la particularidad de los vectores utilizados.

La herramienta Optimtool proporciona un campo para que el usuario pueda especificar el número de población para cada generación, así como otro campo para introducir una población inicial si se tiene. Este proceso solo se realiza para la primera generación, ya que para la segunda, tercera y *n* generaciones en adelante, la población se compondrá de los individuos seleccionados para

sobrevivir de una generación, de los individuos resultado del cruzamiento y de los individuos generados por la mutación de algún elemento.

Para generar los primeros individuos se introdujo un vector con *n "1"s, n "2"s, n "3"s, n "4"s… m "no. de trabajos", donde n es igual al número de máquinas.* Este vector se reacomodó de forma aleatoria generando una solución factible para cada iteración hasta completar la población inicial, de la cual se partirá para formar las siguientes generaciones.

Función Evaluación

De calificar que tan buena o mala es una solución se encarga la *función Evaluación.* La probabilidad de pasar a la siguiente generación intacta como parte de una población "Elite", de generar nuevas soluciones mediante el operador cruzamiento o del operador mutación, es directamente proporcional a la calificación asignada por la función evaluación a cada individuo. Esto no quiere decir que un individuo con una buena calificación no pueda de una generación a otra ser eliminado o no ser considerado para generar nuevos individuos.

La *función evaluación* será programada para evaluar qué individuo genera la secuencia de actividades con el menor *tiempo acumulado de ejecución de las máquinas o intervalo operativo* (*Makespan*), el cual se define como:

$$Cmáx = máx\{C_1, C_2, C_3, \cdots C_{n-1}, C_n\} \text{ Donde:}$$

C_i= tiempo de terminación del trabajo i

$i=1,2,3\ldots n-1,n$

n= número de trabajos que serán procesados

Esta medida o parámetro es comúnmente monitoreado por su implicación en los costos de producción de casi cualquier industria, es por esto que se buscará minimizarlo.

La forma en que se programará la evaluación del *intervalo operativo* para una solución se describe a continuación:

Al programa se han introducido de inicio una matriz de *nXm*, llamada L, donde *n* representa el número de tareas que será variable para las diferentes simulaciones y *m* es igual a 4. De esta forma, un renglón o individuo de esta matriz, se compone de *m* columnas, las cuales indican en forma consecutiva, el número de trabajo al que pertenece la tarea, el orden en el que tienen que ser procesadas indicado por una secuencia de números enteros que van desde el 1 hasta el 7 en el caso de una operación con el número máximo de tareas, la indicación de la máquina donde será procesada, y en la cuarta y última columna el tiempo que se supone tomara el llevar a cabo la tarea.

Con el vector solución en turno se realiza una búsqueda en cada uno de sus elementos, a la vez que se busca en la matriz que contiene la información de todas las operaciones (en qué centro de trabajo se realizaran y el tiempo de operación). Una vez identificada la operación, esta se traslada a una nueva matriz que le denominamos **L**, esta matriz contendrá la misma información que la matriz **X** pero en el orden que dicta el vector solución. Una vez teniendo esta matriz, se realiza una iteración en cada uno de sus elementos. En esta búsqueda se extraerá la información de a que máquina se asignará la operación correspondiente y la información del tiempo de operación. Tomando en cuenta las precedencias entre las operaciones, y la disponibilidad en las máquinas a las que se han asignado los trabajos, se asigna un tiempo de inicio para cada operación y se suma este tiempo a la carga de trabajo de cada máquina. Esto representado en un vector que contiene en cada uno de sus elementos el tiempo de terminación de trabajo en las máquinas, este es el vector $M=[mq_1, mq_2, mq_3 \ldots mq_n]$.

Finalmente el intervalo operativo será igual al valor más alto de entre los elementos del vector M, y el número de renglón de este elemento indicará en que máquina se realizará dicho trabajo. Este proceso se muestra en forma gráfica en la figura no. 3.8.

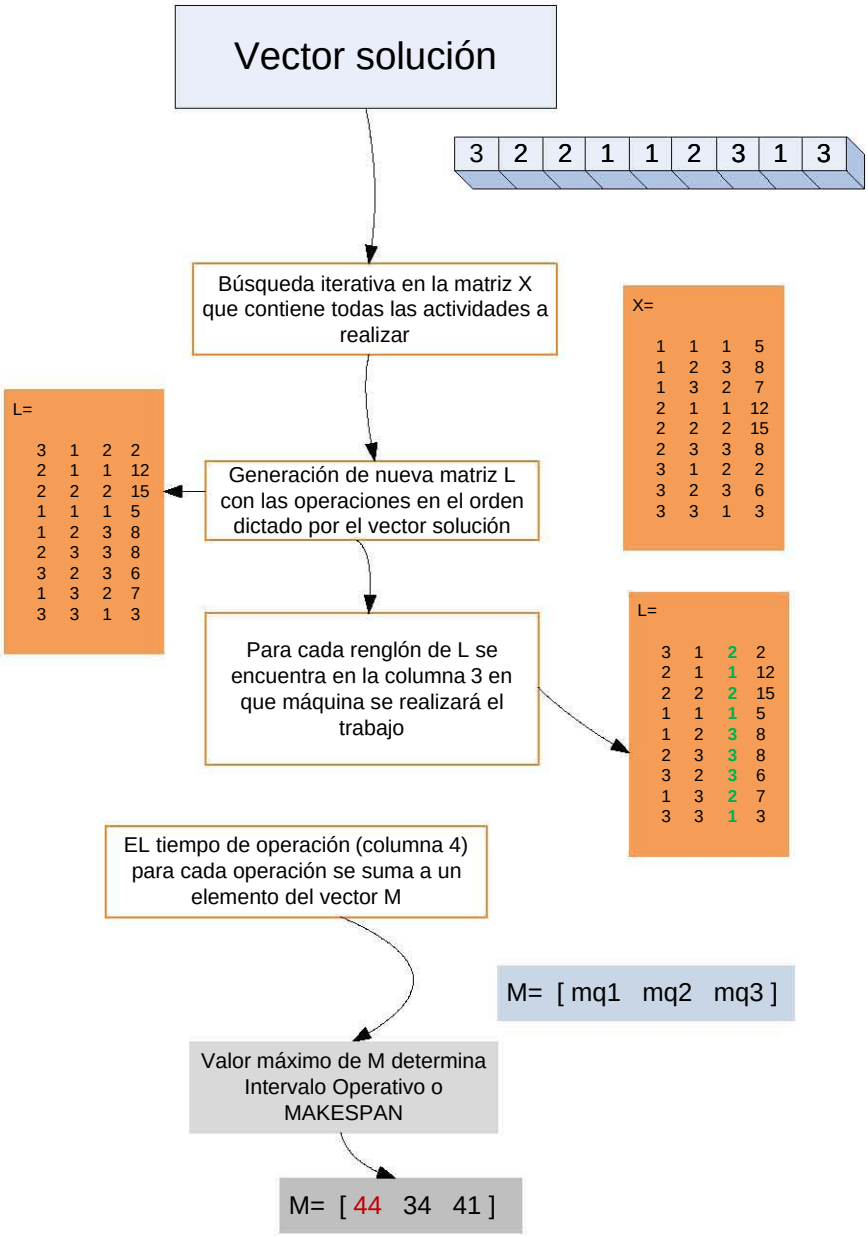


FIG 3.8 Representación de la función para evaluar el MAKESPAN

Función escalamiento

En seguida se define la función que acomodará los individuos en una escala apta para que la función de selección escoja a los que formaran la siguiente generación.

Existen opciones para esta función.

La función *proporcional* asigna una expectativa a cada uno de los individuos en proporción a su valor arrojado por la función evaluación. Esta técnica tiene debilidades si es que los valores de los individuos no se encuentran en un rango de valores amplio, de esta forma los ejemplares más aptos pueden fácilmente ser desechados y contrariamente algunos de los menos aptos, ser escogidos para formar la siguiente generación. No siendo el caso el amplio rango de valores que arroja la función evaluación que se introdujo en el algoritmo, se escogió esta función.

Selección de individuos para producir la nueva generación

Los individuos para una nueva generación serán resultado de tres tipos de procesos. El primero y más sencillo, pero no menos importante, es la selección de los individuos mejor adaptados y que sobrevivirán a la generación anterior para pasar a la nueva generación sin alguna alteración, al número de individuos de este tipo se le denomina como el numero de *Población Elite*, y es un parámetro que es posible modificar en el *Optimtool* para evaluar qué repercusiones tiene en el comportamiento general del algoritmo.

El segundo tipo de población para la nueva generación es la población resultado del proceso de *cruzamiento*. Este parámetro también influirá de forma importante sobre el rendimiento del algoritmo, y debido a que es el operador más importante para el AG se definirá de forma práctica

efectuando ensayos con diferentes porcentajes. El porcentaje restante definirá el tamaño de la población que se someterá a la mutación con alguna probabilidad de no ser alterado en ninguna forma.

La selección de los individuos que se someterán al cruzamiento y a la mutación, puede ser realizada mediante el proceso que Goldberg utilizó para su AG simple o SGA, que es el del método de selección de ruleta, también se puede utilizar el método de Torneo, o el método de selección Proporcional.

Esta función utiliza los valores escalados que arroja la anterior función de escalamiento. Existen 5 opciones prediseñadas para este propósito, selección estocástica uniforme, selección por remanente, selección uniforme, selección por ruleta, y selección por torneo. Para la simulación se decidió utilizar la selección por ruleta. Esta favorece la selección de los individuos más fuertes, sin dejar de lado la posibilidad de que se escojan algunos del menor rango, permitiendo así la diversidad a la población de soluciones.

Reproducción

Los parámetros para la reproducción permiten decidir si es que existirá una población “elite”, que sobreviva a la anterior y si es que la habrá, determinar su tamaño. También se determina el porcentaje de la población que se generará por medio del operador cruzamiento. Aunque no se especifica en la ventana, el resto de la población automáticamente se generará por medio de la función mutación. En la figura 3.9 se muestra el ejemplo de una población elite de 2 individuos, que es el valor preestablecido que maneja *Mathlab*, y un porcentaje de cruzamiento de 80%.

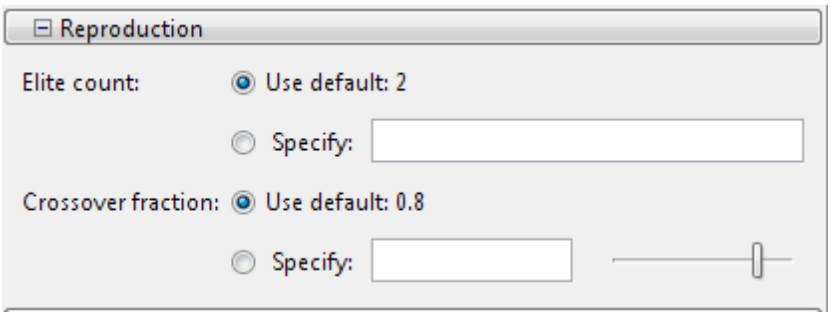


Figura 3.9 Ventana Mathlab Reproducción

Migración

La opción que permite la migración entre individuos de varios subgrupos dentro de la población permite evitar que el algoritmo se aloje en un óptimo local y no continúe la búsqueda de una mejor solución. Esta herramienta maneja una migración entre sub-poblaciones del tipo pasarela (*Stepping Stone*), para la cual se intercambian individuos solo con subgrupos vecinos, en nuestro caso en ambas direcciones , es decir, de la población *n* se migra en ambos sentidos, tanto a la sub-población *(n–1)* como a la (*n*+1). En la ventana se especifica que porcentaje de los mejores resultados migran y con qué frecuencia.

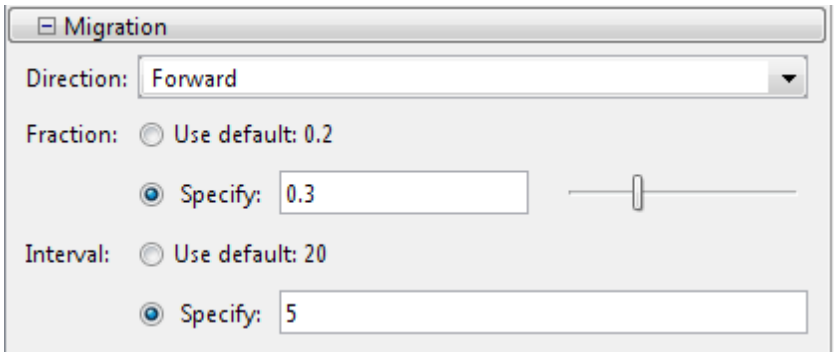


Figura 3.10, Ventana Mathlab Migración

Criterios de parada para el algoritmo

Algunos de los criterios de paro para el algoritmo que permite utilizar la herramienta *Optimtool*, se basan en el número de generaciones a crear, el tiempo de corrida del algoritmo, un límite si se desea para el parámetro que se busca optimizar, un número de generaciones o un tiempo límite en que el parámetro permanezca sin mejoras. Para la simulación actual se utilizaron como criterios de paro solo el número de generaciones a crear, siendo éste variable en diferentes situaciones dependiendo del tiempo de simulación deseado.

CAPÍTULO 4

SIMULACIÓN DEL ALGORITMO GENÉTICO

4.1. *Metodología para la simulación*

Con esta simulación del algoritmo genético en una fase anterior a la definitiva donde no se permitirá la asignación de actividades a máquinas paralelas, servirá para ejemplificar cómo funciona el algoritmo y qué información arroja para poder obtener una secuencia factible. También se pretende ajustar de forma experimental los parámetros que se mencionaron en el capítulo anterior.

La siguiente simulación de actividades para un taller con un sistema de producción intermitente se elabora con base en el AG desarrollado en este trabajo, basado en la codificación propuesta por Gen, Tsujimura y Kubota, y con la modificación a esta en su función evaluación propuesta por Gen, Cheng y Tsujimura.

Introducción de la base de datos con tiempos de las operaciones y restricciones

Para extraer la información necesaria para la simulación se parte de las órdenes de trabajo de cada una de las piezas o productos a fabricar de un supuesto lote, estas órdenes son un documento auxiliar que sirve de base para planear la asignación de las actividades con los recursos disponibles en un taller o empresa. Este documento describe las actividades a realizar para los productos, desde el orden de sus operaciones como el tiempo estimado para cada una de ellas. Contiene además otros campos que son indispensables para la retroalimentación que permite

estimar mejor los tiempos de operación, y por lo tanto, mejorar la planeación de futuras actividades.

	TALLER	NOMBRE PIEZA:	FORMADOR DE PUNZON			
		DIBUJO No.	KL458FF	OT	10633	
		CANTIDAD	1	PZA		
P.C.P		FECHA ALMACEN	23-sep-10	FECHA PISO		
SECUENCIA	DESCRIPCION DE LA OPERACIÓN	TIEMPO ESTIMADO	TIEMPO REAL	NOMBRE DEL OPERADOR	FIRMA DEL OPERADOR	FECHA DE REALIZACION
T.CNC	CAREAR CILINDRAR, VACIAR, DEJAR EXCESO DE MATERIAL PARA RECTIFICAR	4.5				
F.CNC	BARRENAR,	2.5				
BANCOS	REBABEAR, ENVIAR AL ALMACEN	2				
ALMACEN	ENVIAR A TRATAMIENTO TERMICO **53-55 HRC D2**	5				
RECTIF	RECTIFICAR DIAMETROS PARA DAR MEDIDAS FINALES	4				
BANCOS	REBABEAR, AJUSTAR PISTA PARA BALERO, EMBALAR	2				

Figura 4.1 Orden de trabajo del trabajo número 3 en la lista de actividades a simular

La información que se tiene en las órdenes de trabajo, se extrae para construir una base de datos en forma de matriz, la cual es necesaria para que el algoritmo en Mathlab encuentre una secuencia de actividades factible. La tercera columna indica el número de máquina en la cual se llevará a cabo la tarea. La máquina correspondiente a cada número se indica en la tabla no. 4.1, así por ejemplo, el número 1 indica que la operación se llevará a cabo en un torno CNC, el número 2 corresponde a un centro de maquinado CNC, y así sucesivamente.

Tabla 4.1

T. CNC.	1
F. CNC	2
BANCOS	3
HORNOS	4
RECTIFICADORA	5
T CONV	6
F CONV	7

La cuarta columna en la matriz indica el tiempo en que se estima la duración de la actividad, plasmado en horas. En este tiempo se incluye el de preparación del material a trabajar y del equipo a utilizar, suponiendo que es independiente de la secuencia en que se realicen las actividades.

X =[

1.0000	1.0000	2.0000	7.0000	8.0000	1.0000	1.0000	4.0000
1.0000	2.0000	3.0000	2.0000	8.0000	2.0000	3.0000	2.0000
1.0000	3.0000	5.0000	2.000	8.0000	3.0000	4.0000	4.0000
1.0000	4.0000	3.0000	1.0000	8.0000	4.0000	5.0000	3.0000
2.0000	1.0000	6.0000	4.50000	9.0000	1.0000	3.0000	2.0000
2.0000	2.0000	3.0000	1.000	9.0000	2.0000	6.0000	3.0000
2.0000	3.0000	4.0000	5.0000	9.0000	3.0000	4.0000	2.0000
2.0000	4.0000	5.0000	3.0000	9.0000	4.0000	5.0000	3.0000
2.0000	5.0000	3.0000	7.0000	9.0000	5.0000	3.0000	7.0000
3.0000	1.0000	1.0000	4.50000	10.0000	1.0000	1.0000	4.0000
3.0000	2.0000	2.0000	2.5000	10.0000	2.0000	6.0000	2.000
3.0000	3.0000	3.0000	2.0000	10.0000	3.0000	2.0000	2.0000
3.0000	4.0000	4.0000	5.0000	10.0000	4.0000	5.0000	3.000
3.0000	5.0000	5.0000	4.0000	10.0000	5.0000	3.0000	1.000
3.0000	6.0000	3.0000	2.0000				
				11.0000	1.0000	6.0000	3.0000
4.0000	1.0000	6.0000	2.0000	11.0000	2.0000	3.0000	3.0000
4.0000	2.0000	1.0000	5.0000	11.0000	3.0000	4.0000	6.0000
4.0000	3.0000	2.0000	4.000	11.0000	4.0000	5.0000	4.0000
4.0000	4.0000	3.0000	5.000				
				12.0000	1.0000	1.0000	3.0000
5.0000	1.0000	1.0000	4.50000	12.0000	2.0000	3.0000	1.0000
5.0000	2.0000	5.0000	5.0000	12.0000	3.0000	4.0000	4.000
5.0000	3.0000	2.0000	2.000	12.0000	4.0000	5.0000	3.0000
5.0000	4.0000	3.0000	2.0000	12.0000	5.0000	7.0000	2.000
5.0000	5.0000	4.0000	5.0000				
5.0000	6.0000	1.0000	2.50000	13.0000	1.0000	6.0000	3.50000
5.0000	7.0000	3.0000	2.0000	13.0000	2.0000	7.0000	5.000
				13.0000	3.0000	3.0000	3.000
6.0000	1.0000	3.0000	4.0000				
6.0000	2.0000	1.0000	3.0000	14.0000	1.0000	3.0000	10.0000
6.0000	3.0000	2.0000	3.000	14.0000	2.0000	6.0000	6.000
6.0000	4.0000	3.0000	1.0000	14.0000	3.0000	2.0000	6.0000
				14.0000	4.0000	3.0000	4.0000
7.0000	1.0000	1.0000	9.0000	15.0000	1.0000	1.0000	5.0000
7.0000	2.0000	3.0000	3.000	15.0000	2.0000	5.0000	5.000
7.0000	3.0000	4.0000	5.000	15.0000	3.0000	2.0000	2.0000
7.0000	4.0000	5.0000	9.0000	15.0000	4.0000	3.0000	1.000
7.0000	5.0000	1.0000	2.000	15.0000	5.0000	4.0000	3.0000
7.0000	6.0000	3.0000	2.000	15.0000	6.0000	1.0000	1.5000
				15.0000	7.0000	3.0000	1.000];

También se requiere para evaluar el tiempo de terminación de las actividades, una matriz para cada uno de los trabajos en la cual se plasmen los tiempos en que se determina el comienzo de cada una de las tareas, se sumen los tiempos de operación conforme se vayan asignando las

actividades y darle un seguimiento a cada uno de los trabajos. En este ejemplo, se muestra la matriz para el trabajo número 1, la cual se compone de 6 columnas, las cuatro primeras con los mismos datos que en la matriz principal, la columna número 5 se declara de principio con ceros, y es en esta columna que el algoritmo irá determinando el tiempo de comienzo de cada actividad dictados por la secuencia obtenida en cada iteración. Por último, en la columna 6 se irán determinando los tiempos de término para cada operación, que no es otra cosa que la suma de la columna 4 y 5. Al final de la corrida del programa estas matrices son las que permitirán establecer en un diagrama de Gantt de manera gráfica la secuencia de actividades, arrojando el tiempo de comienzo y el de término. Para el algoritmo final, que se diseñó para un problema del tipo FJSSP o problema para un sistema de producción intermitente flexible, existirá una 5 columna en la matriz principal que establezca la máquina alternativa en la que se podrá programar la actividad en caso de estar ocupada la máquina principal. También incluirán las matrices de cada trabajo al final una séptima columna que indicará en qué máquina se llevó a cabo el trabajo.

Matriz X_1 que contiene el orden de las actividades para el trabajo número 1.

```
x1 =[
    1.0000    1.0000    2.0000    7.0000         0         0;
    1.0000    2.0000    3.0000    2.0000         0         0;
    1.0000    3.0000    5.0000    2.000          0         0;
    1.0000    4.0000    3.0000    1.0000         0         0;
    1.0000    5.0000    1.0000         0         0         0;
    1.0000    6.0000    1.0000         0         0         0;
    1.0000    7.0000    1.0000         0         0         0];
```

Matriz X_1 al final de la corrida del AG, esta dicta la programación de cada actividad en una máquina y tiempo determinado.

```
x1 =

    1.0000    1.0000    2.0000    7.0000         0    7.0000
    1.0000    2.0000    3.0000    2.0000   20.0000   22.0000
    1.0000    3.0000    5.0000    2.0000   23.5000   25.5000
    1.0000    4.0000    3.0000    1.0000   41.0000   42.0000
    1.0000    5.0000    1.0000         0         0         0
    1.0000    6.0000    1.0000         0         0         0
    1.0000    7.0000    1.0000         0         0         0
```

SIMULACIÓN DE PROBLEMA JSS DE 15 TRABAJOS EN 7 MÁQUINAS

La herramienta MATHLAB, Optimtool, presenta campos al usuario en los cuales es necesario escoger algún tipo de parámetro de los establecidos previamente, o llamar a alguna función previamente programada. La figura número 4.1 presenta la interfaz del programa.

En el campo número 1 como se indica, se hace referencia a la *Función Evaluación,* que en este caso es la *makespan15,* todas las funciones son llamadas con un símbolo de @.

Inmediatamente en el número 2 se encuentra el campo para introducir el número de variables, que en este ejercicio es la longitud de los vectores soluciones o individuos, que para 15 actividades y 7 máquinas será de 105 elementos.

Del lado derecho de la pantalla, en el número 3, se encuentran los campos de los parámetros que caracterizarán el algoritmo como son Población, Función Escalamiento, Función Selección, Reproducción, Función Mutación, Función cruzamiento y Función Migración .Por último, en el número 4 también se encuentran las opciones para graficar el comportamiento del algoritmo y los criterios de paro.

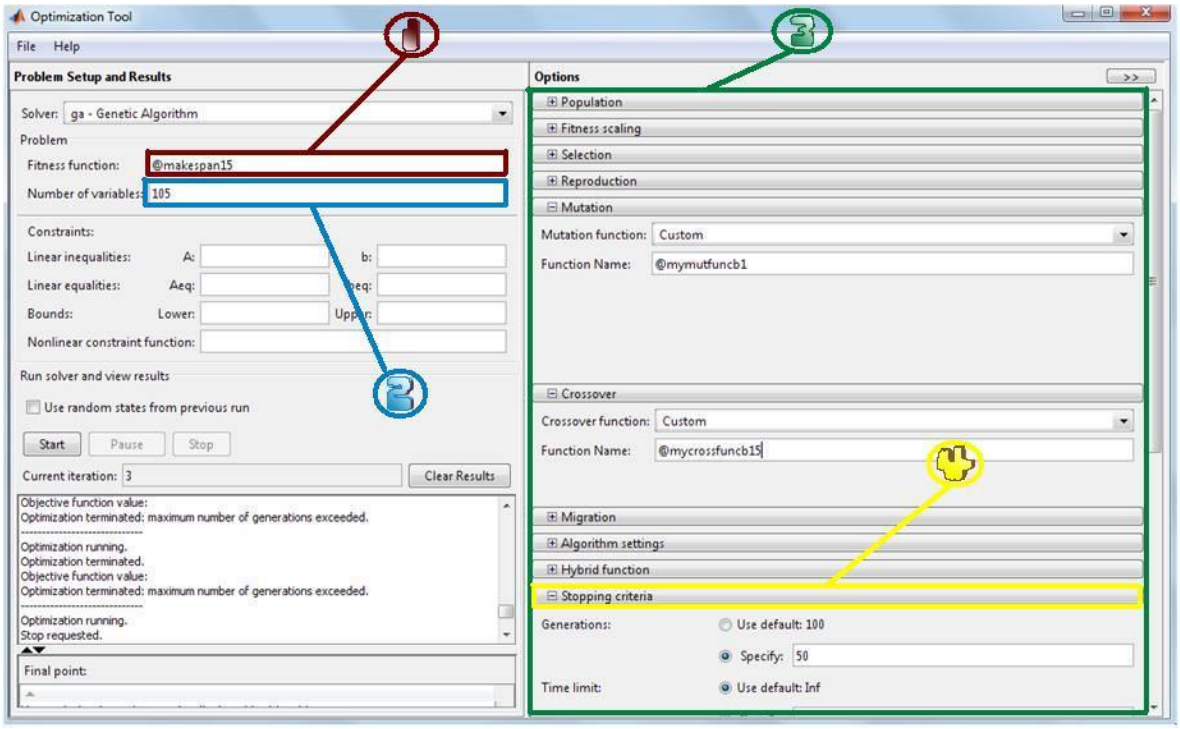


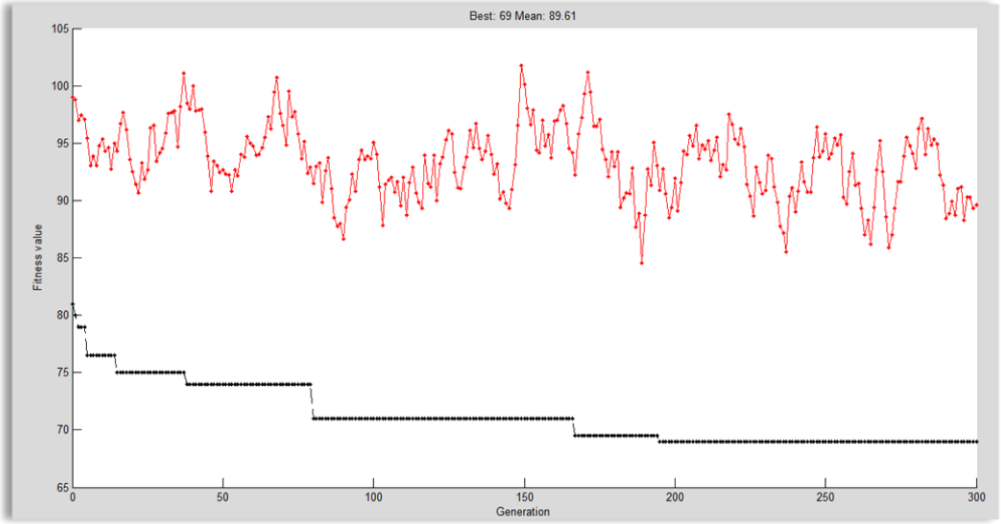
Figura 4.1 Interfaz de Mathlab Optimtool

Parámetros de la simulación

Tabla 4.2

15 Actividades X7 máquinas				
Criterio de paro 300 generaciones		Población-50		
f(x) Evaluación-Makespan15		Población inicial - Generada Aleatoriamente		
f(x) Escalamiento-Proporcional		Porcentaje de población para emigrar:20%		
f(x)Selección-Ruleta		Intervalo de Migracion-5		
f(x) Cruzamiento -90% de población				
f(x) Mutación - 10% población t>0.8				
Población Elite-2 individuos				

En la gráfica 4.1 se muestra con línea roja la evolución del parámetro que se busca optimizar representado por la media de la población en cada generación, también se muestra en cada generación el valor del mejor individuo obtenido para cada iteración. El intervalo operativo que arrojo este ensayo fue de 69.



Gráfica 4.1 Desempeño del Algoritmo

Una vez que se ha dado por terminada la simulación se exportan los resultados al *espacio trabajo* de MATHLAB, creando el vector solución ***optimresults.x*** *(figura 4.2)***,** ésta es la solución que indica la secuencia con el mejor intervalo operativo. El vector se forma con un valor en cada uno de sus 105 elementos que van desde el número 1 hasta el número 15.

```
>> optimresults.x

ans =

 Columns 1 through 13

  14   2   2   2  10  15  11   3   1  13  13  11  10

 Columns 14 through 26

  15   5   3   2  10  10   6  14   6  13   2  14   4

 Columns 27 through 39

   3  13  11   9   3   1   6   5   9   7  10   7   1

 Columns 40 through 52

   9   5  13   8  12  15   4  12   8  15  13  11   2

 Columns 53 through 65

  10   6  10   6   4  14   6  13   9   8   3   7   1

 Columns 66 through 78

   4   3  14  12   5   9   5   8   8   7  11   3  11

 Columns 79 through 91

  12   7   2   7   5   5   6  11  15  15   8  12  15

 Columns 92 through 104

   4   4  12   9  12   9   1  14   1   8   7   4   1

 Column 105

  14
```

1 **Figura 4.2 Vector Solución**

Esta secuencia en cada iteración le dicta a la función evaluación la tarea a agendar, y es esta función que se encarga de ir asignando los tiempos de inicio y de término a la matriz que le corresponda a cada actividad, estas matrices que van desde la x1 a la x15 se muestran a continuación:

x1 =

1	1	2	7	0	7
1	2	3	2	32	34
1	3	5	2	34	36
1	4	3	1	47	48
1	5	1	0	0	0
1	6	1	0	0	0
1	7	1	0	0	0

x2 =

2	1	6	4.5	0	4.5
2	2	3	1.0	10	11.0
2	3	4	5.0	11	16.0
2	4	5	3.0	16	19.0
2	5	3	7.0	21	28.0
2	6	1	0	0	0
2	7	1	0	0	0

x3 =

3	1	1	4.5	9.0	13.5
3	2	2	2.5	13.5	16.0
3	3	3	2.0	28.0	30.0
3	4	4	5.0	30.0	35.0
3	5	5	4.0	43.0	47.0
3	6	3	2.0	53.0	55.0
3	7	1	0	0	0

x4 =

4	1	6	2	19	21
4	2	1	5	37	42
4	3	2	4	42	46
4	4	3	5	48	53
4	5	1	0	0	0
4	6	1	0	0	0
4	7	1	0	0	0

x5 =

5	1	1	4.5	13.5	18.0
5	2	5	5.0	22.0	27.0
5	3	2	2.0	28.0	30.0
5	4	3	2.0	55.0	57.0
5	5	4	5.0	57.0	62.0
5	6	1	2.5	62.0	64.5
5	7	3	2.0	66.0	68.0

x6 =

6	1	3	4	14	18
6	2	1	3	18	21
6	3	2	3	25	28
6	4	3	1	42	43
6	5	1	0	0	0
6	6	1	0	0	0
6	7	1	0	0	0

x7 =

7	1	1	9	21	30
7	2	3	3	35	38
7	3	4	5	45	50
7	4	5	9	50	59
7	5	1	2	59	61
7	6	3	2	64	66
7	7	1	0	0	0

x8 =

8	1	1	4	30	34
8	2	3	2	39	41
8	3	4	4	41	45
8	4	5	3	47	50
8	5	1	0	0	0
8	6	1	0	0	0
8	7	1	0	0	0

x9 =

9	1	3	2	30	32
9	2	6	3	32	35
9	3	4	2	35	37
9	4	5	3	40	43
9	5	3	7	57	64
9	6	1	0	0	0
9	7	1	0	0	0

x10 =

10	1	1	4	0	4
10	2	6	2	11	13
10	3	2	2	16	18
10	4	5	3	19	22
10	5	3	1	34	35
10	6	1	0	0	0
10	7	1	0	0	0

x11 =

11	1	6	3	4.5	7.5
11	2	3	3	11.0	14.0
11	3	4	6	16.0	22.0
11	4	5	4	36.0	40.0
11	5	2	0	0	0
11	6	2	0	0	0
11	7	2	0	0	0

x12 =

12	1	1	3	34	37
12	2	3	1	38	39
12	3	4	4	50	54
12	4	5	3	59	62
12	5	7	2	62	64
12	6	3	0	0	0
12	7	3	0	0	0

x13 =

13	1	6	3.5	7.5	11
13	2	7	5.0	11.0	16
13	3	3	3.0	18.0	21
13	4	1	0	0	0
13	5	1	0	0	0
13	6	1	0	0	0
13	7	1	0	0	0

x14 =

14	1	3	10	0	10
14	2	6	6	13	19
14	3	2	6	19	25
14	4	3	4	43	47
14	5	1	0	0	0
14	6	1	0	0	0
14	7	1	0	0	0

x15 =

15	1	1	5.0	4	9.0
15	2	5	5.0	9	14.0
15	3	2	2.0	30	32.0
15	4	3	1.0	41	42.0
15	5	4	3.0	62	65.0
15	6	1	1.5	65	66.5
15	7	3	1.0	68	69.0

El la figura 4.2 se muestra la secuencia obtenida en forma gráfica, dictada por los tiempos de inicio y termino de cada una de las matrices anteriores.

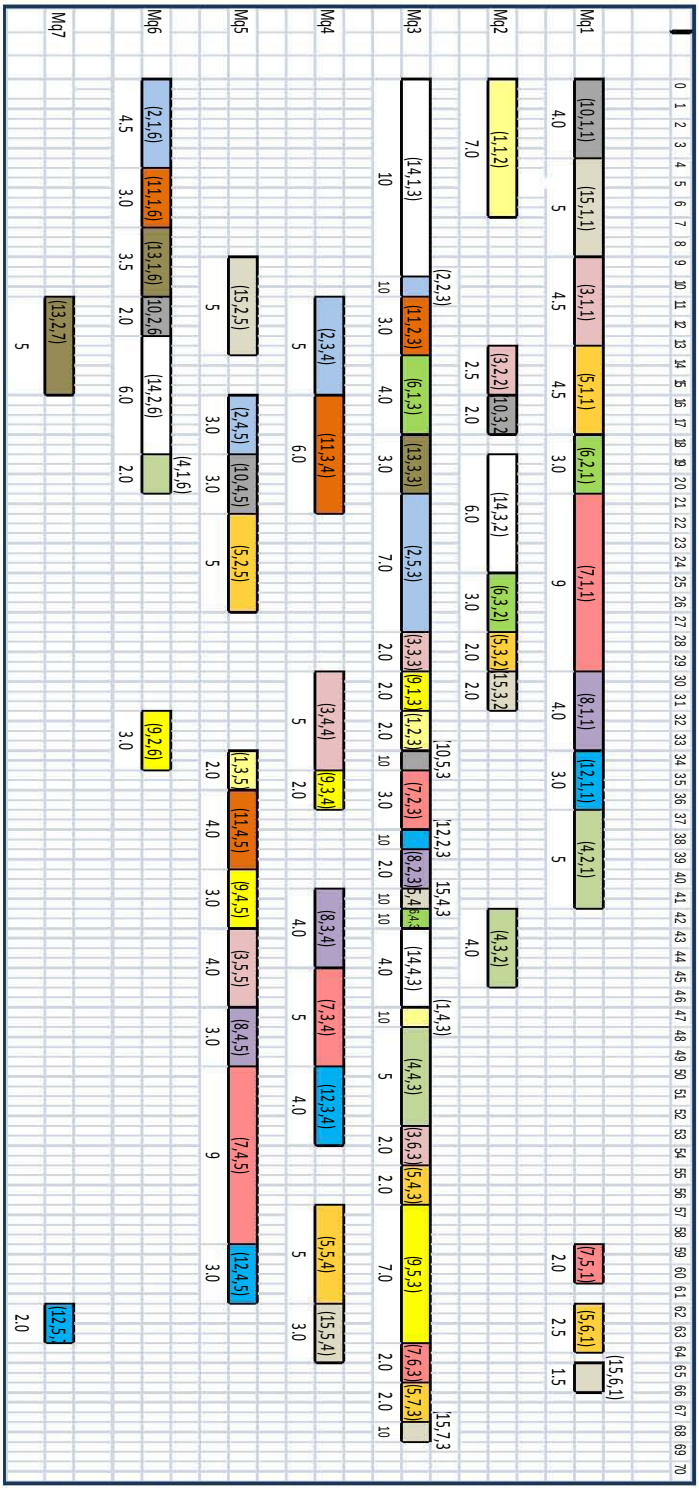


Figura 4.2 Diagrama de Gantt de la secuencia Obtenida

4.2.*Ajuste de parámetros*

Para realizar el ajuste de los parámetros, se realizarán distintos ensayos que permitan visualizar el efecto que éstos tienen en el algoritmo a medida que crecen o disminuyen.

Determinación del parámetro Población

El tamaño de población implica algunos efectos en el algoritmo como son:

Afectan su velocidad; Si el algoritmo realiza en cada iteración la generación de una población en extremo extensa, el algoritmo tendera a ser lento y será menos factible el realizar una búsqueda a lo largo de varias generaciones. Por otro lado, la generación de grandes poblaciones permite el abarcar más espacio en una sola generación, y claro permite la diversidad de las soluciones. Es por esto que se buscará un balance entre una población que permita una búsqueda generalizada, y que no entorpezca la generación de poblaciones nuevas de manera dinámica.

Tabla 4.3 Parámetros AG

20 Actividades X7 máquinas		
Criterio de paro 100 Generaciones		
f(x) Evaluación-Makespan20as		
f(x) Escalamiento-shiftLinear 10		
f(x)Selección-Ruleta		
f(x) Cruzamiento 60% de población		
f(x) Mutación - 40% población t>0.1		
f(x) Migración - 20% Cada 5 generaciones		
Población Elite-variable		

En la tabla 4.3 se establecen las funciones y parámetros utilizados en la simulación con la base de datos con 20 trabajos y 7 máquinas. En la tabla 4.4 se muestran los resultados de 5 ensayos, en los que se asignaron tamaños distintos de población.

Tabla 4.4 Resultados para la simulación de un problema con 20 actividades y 7 máquinas

	NÚMERO DE POBLACIONES											
	20		40		60		100		140		160	
20X7 JSSP	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo
1	66	01:09	65.5	01:55	67	02:10	64.5	03:23	65	05:05	65.5	07:20
2	65.5	01:09	67.5	01:55	65.5	02:10	66	03:23	65	05:05	66	07:20
3	67.5	01:09	65	01:55	67	02:10	65	03:23	66	05:05	65.5	07:20
4	66	01:09	67	01:55	65	02:10	65	03:23	66	05:05	64.5	07:20
5	66	01:09	67	01:55	66.5	02:10	64	03:23	63.5	05:05	65.5	07:20
	Elite 2		Elite 4		Elite 6		Elite 10		Elite 14		Elite 16	
PROMEDIO	**66.2**		**66.4**		**66.2**		**64.9**		**65.1**		**65.4**	

Tabla 4.5 Parámetros del AG

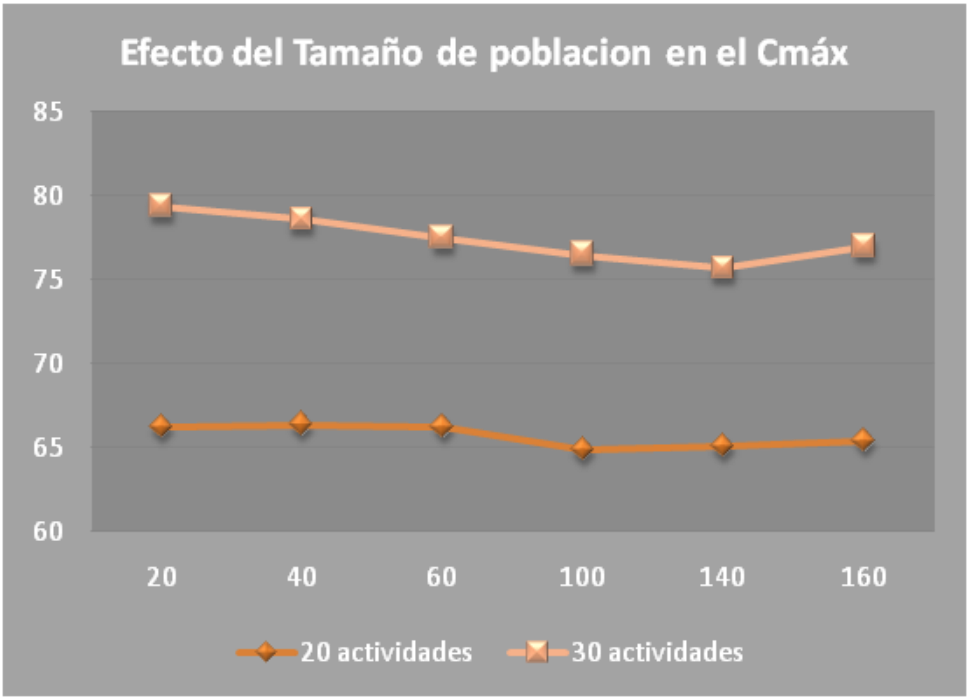
30 Actividades X7 máquinas	
Criterio de paro 50 generaciones	
f(x) Evaluación-Makespan30as	
f(x) Escalamiento-Shift Linear 4	
f(x)Selección-Torneo 20 individuos	
f(x) Cruzamiento 60% de población	
f(x) Mutación - 40% población t>0.1	
f(x) Migración - 20% Cada 5 generaciones	
Población Elite-4 individuos	

En la tabla 4.5 se muestran los parámetros y funciones utilizados en la simulación con la base de datos de 30 trabajos y 7 máquinas. En la tabla 4.6 se muestran los resultados de la simulación para distintos tamaños de población.

Tabla 4.6 Resultado de la simulación para un problema de 30 actividades y 7 máquinas

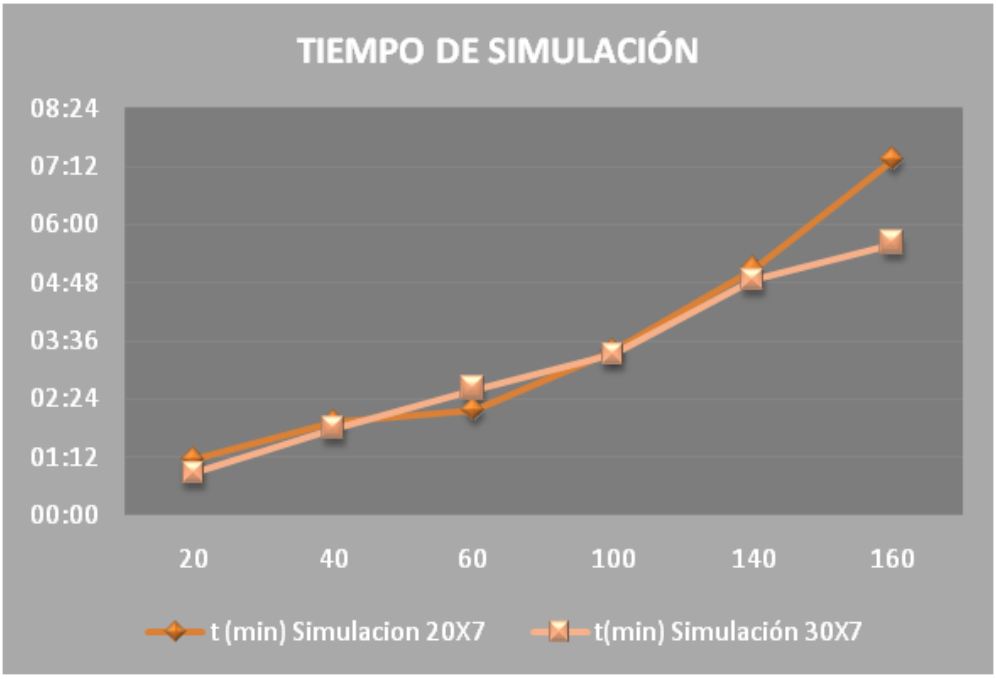
	NÚMERO DE POBLACIONES											
	20		40		60		100		140		160	
30X7 JSSP	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo
1	78.5	00:52	83.5	01:47	79	02:36	78	03:19	77.5	04:51	79.5	05:37
2	79.5	00:52	77	01:47	80	02:36	78	03:19	79.5	04:51	78	05:37
3	79.5	00:52	83	01:47	80	02:36	78	03:19	77.5	04:51	77.5	05:37
4	81.5	00:52	80	01:47	75.5	02:36	75.5	03:19	79.5	04:51	78	05:37
5	81	00:52	78	01:47	78.5	02:36	80	03:19	77	04:51	80	05:37
	Elite 2		Elite 4		Elite 6		Elite 10		Elite 14		Elite 16	
PROMEDIO	**80**		**80.3**		**78.6**		**77.9**		**78.2**		**78.6**	

En la gráfica 4.2 se muestra el comportamiento del Cmáx dependiente del tamaño de población a utilizar para las dos simulaciones de 20X7 y 30X7.



Gráfica 4.2 Cmáx VS No. Población

En la gráfica 4.3 se muestra el tiempo que se utilizo para cada uno de los ensayos de la tabla 4.4 y 4.6.



Gráfica 4.3 tiempo de simulación VS Población

Se puede apreciar que el desempeño en el parámetro Cmáx no mejora proporcionalmente con el número de población que se establece para la simulación. En efecto, el espacio solución es muy extenso, por lo que aumentar el tamaño de la población en algunos cientos, no reflejaría ninguna mejoría en el algoritmo. La mejoría del parámetro para la población de 160 contra el resultado que arrojó la simulación con 20 ejemplares, es solo de 1.78% , cuando se incrementó el tiempo en un 548%.

Determinación del criterio de paro; "Generaciones"

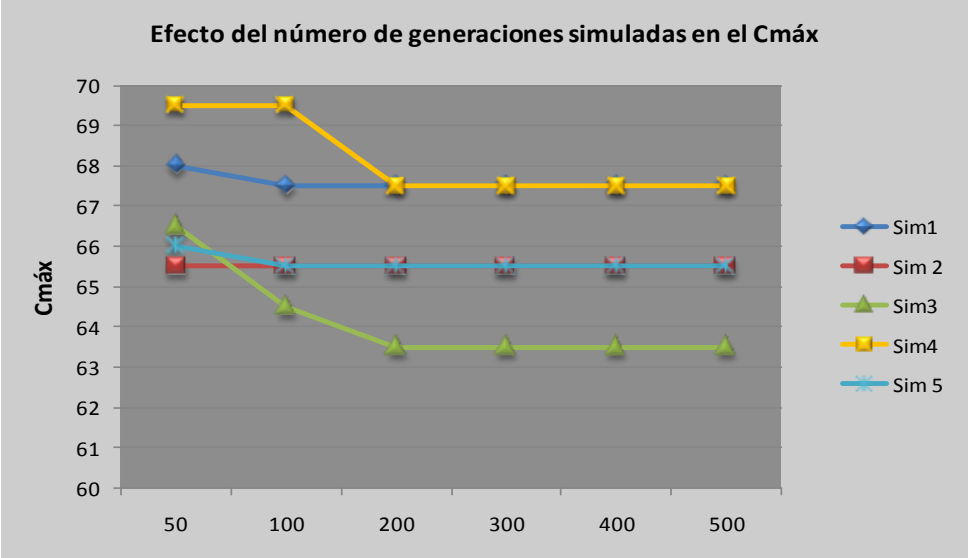
De los criterios de paro que permite elegir la herramienta de *Optimtool*, se escogió el de número de generaciones a simular. Se realizarán los ensayos para dos bases de datos, con el algoritmo programado para observar el desempeño del algoritmo en función del número de generaciones.

Los datos se observan en las tablas número 3 y número 4.

Tabla 4.7 Resultado de la simulación para 20 actividades

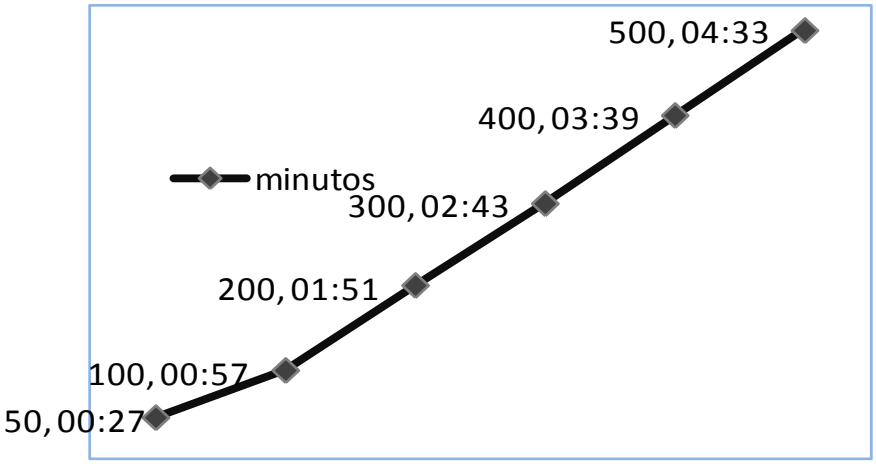
	NÚMERO DE GENERACIONES											
	50		100		200		300		400		500	
20X7 JSSP	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo
1	68	00:27	67.5	00:57	67.5	01:51	67.5	02:43	67.5	03:39	67.5	04:33
2	65.5	00:27	65.5	00:57	65.5	01:51	65.5	03:43	65.5	04:39	65.5	04:33
3	66.5	00:27	64.5	00:57	63.5	01:51	63.5	04:43	63.5	05:39	63.5	04:33
4	69.5	00:27	69.5	00:57	67.5	01:51	67.5	05:43	67.5	06:39	67.5	04:33
5	66	00:27	65.5	00:57	65.5	01:51	65.5	06:43	65.5	07:39	65.5	04:33
	Elite 2		Elite 4		Elite 6		Elite 10		Elite 14		Elite 16	
PROMEDIO	**67.1**		**66.5**		**65.9**		**65.9**		**65.9**		**65.9**	

En la gráfica 4.5 se muestran los valores del Cmáx para cada uno de los ensayos de la tabla 4.7 conforme se avanza en el número de generaciones simuladas.



Gráfica 4.5 Cmáx VS No. de Generaciones simuladas

La gráfica 4.6 muestra el tiempo transcurrido en la simulación del AG para cada uno de los ensayos de la tabla 4.7. El tiempo para cada ensayo es el mismo, por lo que se muestra solo una gráfica, en cada uno de los puntos se indica en número de generaciones simuladas y el tiempo en el que se alcanzó dicha generación.



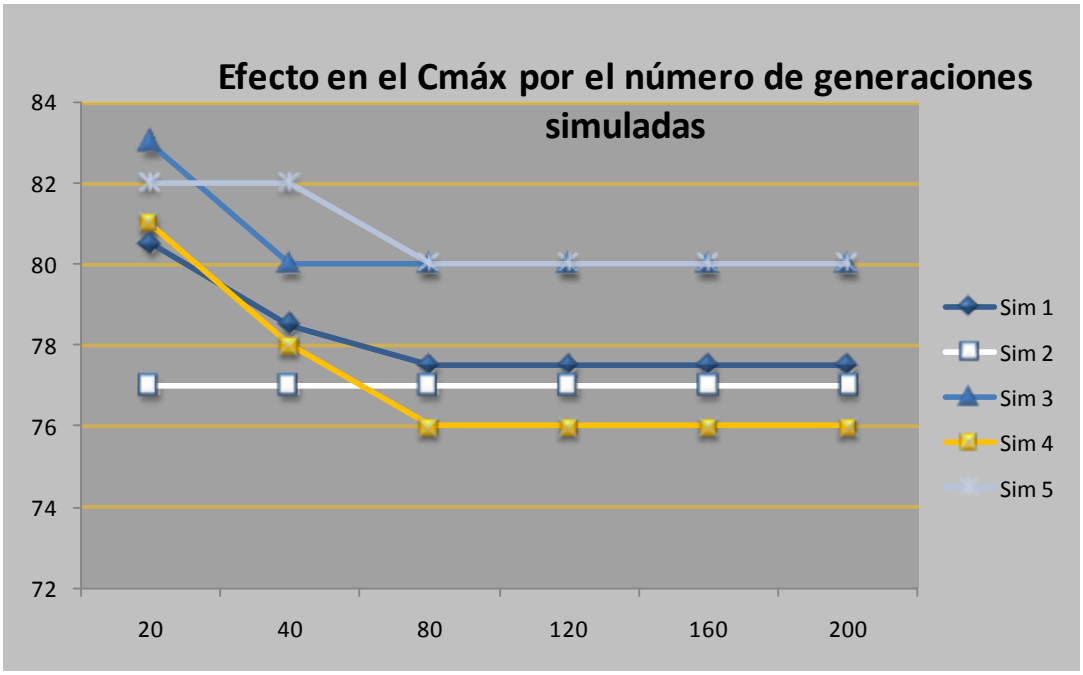
Gráfica 4.6 tiempo VS Generaciones simuladas

En la tabla 4.8 se muestran los resultados de la simulación realizada con el AG para 5 ensayos, los valores de Cmáx alcanzados en las generaciones 20, 40, 80, 120, 160 y 200.

Tabla 4.8 Resultado de la simulación para 30 actividades.

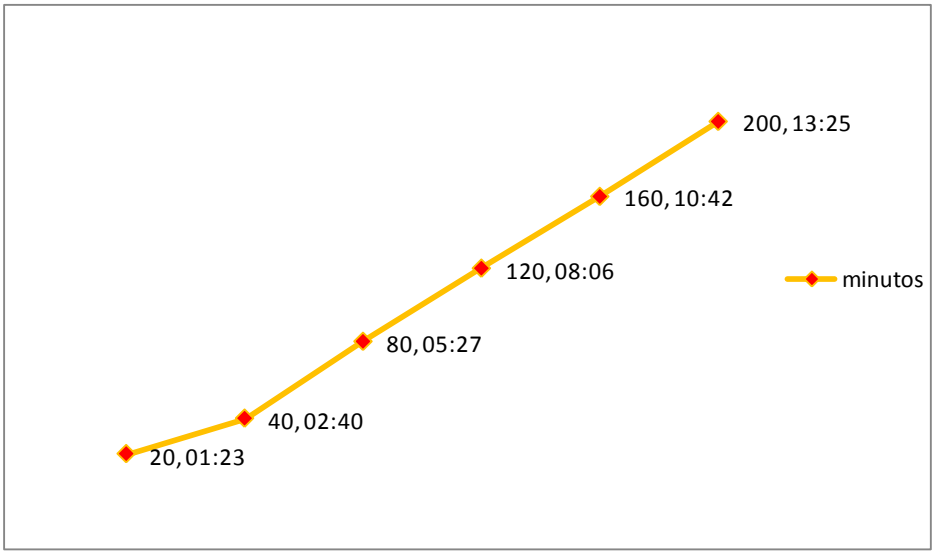
	NÚMERO DE GENERACIONES											
	20		40		80		120		160		200	
30X7 JSSP	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo	Cmáx	Tiempo
1	80.5	01:23	78.5	02:40	77.5	05:27	77.5	08:06	77.5	10:42	77.5	13:25
2	77	01:23	77	02:40	77	05:27	77	08:06	77	10:42	77	13:25
3	83	01:23	80	02:40	80	05:27	80	08:06	80	10:42	80	13:25
4	81	01:23	78	02:40	76	05:27	76	08:06	76	10:42	76	13:25
5	82	01:23	82	02:40	80	05:27	80	08:06	80	10:42	80	13:25
	Elite 2		Elite 4		Elite 6		Elite 10		Elite 14		Elite 16	
PROMEDIO	**80.7**		**79.1**		**78.1**		**78.1**		**78.1**		**78.1**	

En la gráfica 4.7 se muestra la evolución de los resultados obtenidos con el AG conforme se incrementa el número de generaciones simuladas.



Gráfica 4.7 Cmáx VS no. de Generaciones simuladas

La gráfica 4.8 muestra el tiempo transcurrido en la simulación del AG para cada uno de los ensayos de la tabla 4.8. El tiempo para cada ensayo es el mismo, por lo que se muestra solo una gráfica, en cada uno de los puntos se indica en número de generaciones simuladas y el tiempo en el que se alcanzó dicha generación.



Gráfica 4.8 Cmáx VS no. de Generaciones simuladas

La gráfica 4.5 indica que después de la población número 200 no se registra ningún cambio aunque el algoritmo continúe con su búsqueda desde el minuto 1:51 hasta en minuto 4:33 como lo indica la gráfica 4.6. Para el segundo grupo de datos, en la gráfica 4.7, esto sucede mucho más temprano. Hasta la generación número 80 se aprecia una mejoría en el parámetro y en adelante, desde el minuto 5:27 hasta el minuto 13:25 según la gráfica 4.8 no se registra ningún cambio. Esta diferencia se debe a que la primera de estas simulaciones se corrió con una población de 20 individuos y la segunda con 60 individuos.

Queda claro que la búsqueda de una buena solución depende de los dos parámetros anteriormente graficados. Las poblaciones pequeñas pueden aportar una solución buena en muy poco tiempo, que es lo deseable para cualquier problema. El conflicto surge en cuanto la

búsqueda con poblaciones pequeñas se estanca en un óptimo local. De estos ensayos que no muestran cambio en ninguna de sus 6 mediciones se encuentra un ejemplo en cada una de las gráficas 4.5 y 4.7.

Para los ensayos finales se simulará considerando una población de 100 como base. No se pretenderá correr el algoritmo por grandes periodos de tiempo puesto que, como muestran los anteriores ensayos, la mejoría del parámetro a la larga representa un esfuerzo computacional muy grande.

Si en alguna de las simulaciones se disminuye este parámetro será debido a que el esfuerzo computacional es muy grande para dicha simulación y no permite el alcanzar un número de generaciones deseado. Se contemplará alcanzar un mínimo de 200 generaciones con poblaciones de 100 individuos.

CAPÍTULO 5

RESULTADOS DE LAS SIMULACIONES

5.1. *Resultados de la simulación para el algoritmo genético con codificación para generar secuencias activas*

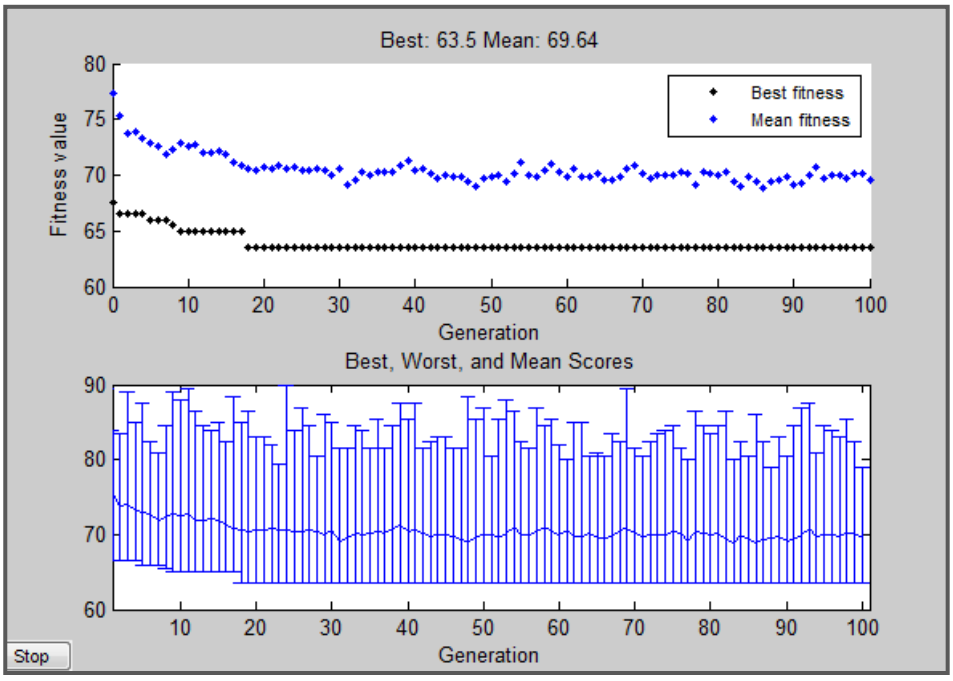
En la tabla 5.1 se muestran los valores arrojados para la simulación de 5 ensayos para 15 actividades y 7 máquinas, los tiempos en que se alcanzó cada uno de los valores del Cmáx, y en negritas el mejor valor alcanzado por el AG. El parámetro de 63.5 se conoce como óptimo por no haber espacios en la línea del tiempo posibles para reacomodar alguna tarea en el centro de trabajo donde se estableció este tiempo, es decir, todas las tareas asignadas en dicho centro de trabajo, se comienzan desde el tiempo 0 y se preceden sin dejar tiempo ocioso alguno entre ellas.

Tabla 5.1 Resultados de la simulación, parámetros y Funciones del AG

RESULTADOS PARA LA SIMULACIÓN DE 15 ACTIVIDADES Y 7 MÁQUINAS EN UN PROBLEMA DE JSSP			
Ensayo Número	Parámetro alcanzado en generación no.	tiempo corrida	parámetro alcanzado Cmáx
1	11	9 seg	**63.5**
2	18	16 seg	**63.5**
3	180	2 min 36 seg	65
4	10	9 seg	67
5	107	1 min 33 seg	63.5

Parámetro óptimo	**63.5**
Función Evaluación	Con codificación basada en operaciones y generación de secuencias activas
Función Cruzamiento	Cruzamiento en dos puntos , genera el 60% de la población
Función Mutación	Tipo inversión de un bit
Función Escalamiento	Escalamiento Lineal
Función Selección	Selección por método de Ruleta
Función Migración	Migración tipo pasarela en ambas direcciones
Población	100 individuos
Criterio de paro	200 Generaciones

La gráfica 5.1 muestra en cada generación el valor del Cmáx alcanzado por la mejor secuencia hasta dicha generación, también muestra en azul el promedio del Cmáx de los individuos de la generación y el rango de los valores alcanzados.



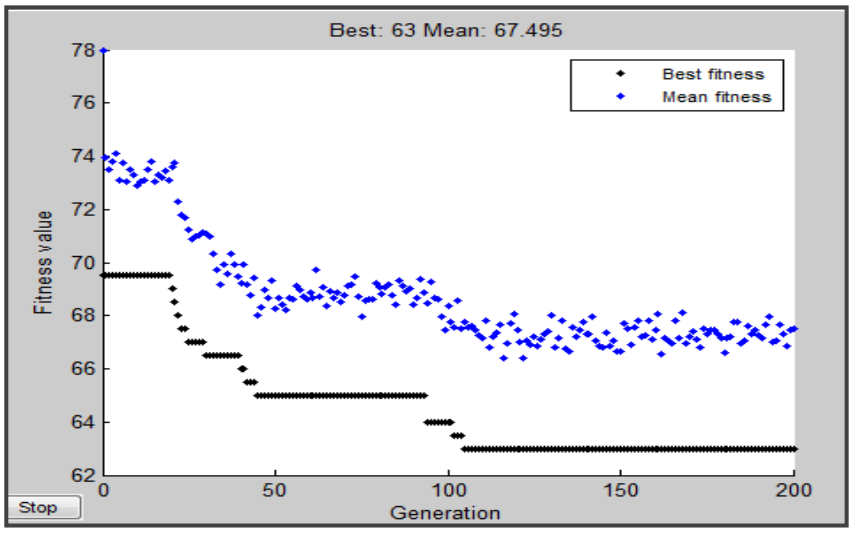
Gráfica 5.1 Cmáx promedio y Mejor Individuo en Cada Generación

En la tabla 5.2 se muestran los valores arrojados para la simulación de 5 ensayos para 20 actividades y 7 máquinas, los tiempos en que se alcanzó cada uno de los valores del Cmáx, y en negritas el mejor valor alcanzado por el AG.

Tabla 5.2 Resultados de la simulación, parámetros y Funciones del AG

RESULTADOS PARA LA SIMULACIÓN DE 20 ACTIVIDADES Y 7 MÁQUINAS EN UN PROBLEMA DE JSSP			
Ensayo Número	Parámetro alcanzado en generación no.	tiempo corrida	parámetro alcanzado Cmáx
1	21	35 seg	66
2	180	5 min 7 seg	63.5
3	104	2 min 54 seg	**63**
4	12	20 seg	68
5	60	1 min 42 seg	64
Parámetro óptimo	**63**		
Función Evaluación	Con codificación basada en operaciones, generación de secuencias activas.		
Función Cruzamiento	Cruzamiento en dos puntos , genera el 60% de la población		
Función Mutación	Tipo inversión de un bit		
Función Escalamiento	Escalamiento Lineal		
Función Selección	Selección por método de Ruleta		
Función Migración	Migración tipo pasarela en ambas direcciones		
Población	100 individuos		
Criterio de paro	200 Generaciones		

La gráfica 5.2, perteneciente al ensayo no. 3, muestra en cada generación el valor del Cmáx alcanzado por la mejor secuencia hasta dicha generación, muestra además en azul el promedio del Cmáx de los individuos de la generación.



Gráfica 5.2 Cmáx VS Generaciones

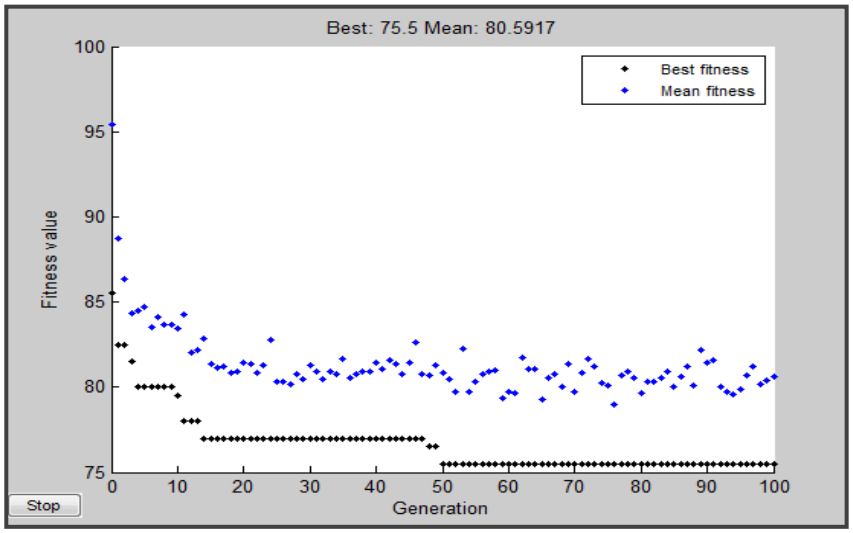
En la tabla 5.3 se muestran los valores arrojados para la simulación de 5 ensayos para 30 actividades y 7 máquinas, los tiempos en que se alcanzó cada uno de los valores del Cmáx, y en negritas el mejor valor alcanzado por el AG.

Tabla 5.3 Resultados de la simulación, parámetros y Funciones del AG

RESULTADOS PARA LA SIMULACIÓN DE 30 ACTIVIDADES Y 7 MÁQUINAS EN UN PROBLEMA DE JSSP			
Ensayo Número	Parámetro alcanzado en generación no.	tiempo corrida	parámetro alcanzado Cmáx
1	90	8 min 54 seg	76.5
2	50	4 min 57 seg	**75.5**
3	25	2 min 27 seg	77
4	60	5 min 56 seg	78
5	50	4 min 57 seg	77.5

Parámetro óptimo	No conocido
Función Evaluación	Con codificación basada en operaciones, generación de secuencias activas.
Función Cruzamiento	Cruzamiento en dos puntos , genera el 60% de la población
Función Mutación	Tipo inversión de un bit
Función Escalamiento	Escalamiento Lineal
Función Selección	Selección por método de Ruleta
Función Migración	Migración tipo pasarela en ambas direcciones
Población	100 individuos
Criterio de paro	200 Generaciones

La gráfica 5.3, perteneciente al ensayo no. 2 de la tabla 5.3 y muestra en cada generación el valor del Cmáx alcanzado por la mejor secuencia hasta dicha generación, aparece también en azul el promedio del Cmáx de los individuos de la generación. En la generación 50 se observa que se alcanzó el valor de 75.5, no mejorando desde esta generación hasta el final de la simulación.



Gráfica 5.3 Cmáx VS Generaciones

5.2. *Comparación de resultados con WinQSB para el JSSP*

A continuación se muestra la comparación de los resultados obtenidos anteriormente con el AG desarrollado en este trabajo y los métodos heurísticos utilizados por el programa Win QSB. En la tabla 5.4 se establecen las abreviaturas que se utilizarán para la comparación de los métodos y el AG en la tabla 5.5

Tabla 5.4 Abreviaturas de Métodos

WinQSB	*Abreviatura*
Heurístico de despacho basado en FCFS (primeras entradas, primeras salidas)	FCFS
Heurístico de despacho basado en MWKR (mas trabajo remanente)	MWKR
Heurístico de despacho CON Cmáx mínimo como objetivo, ALL HDR (todos los criterios de despacho)	ALL HDR
Generación de *n* secuencias aleatorias	RAND *n*
ALGORITMO GÉNETICO	AG

Tabla 5.5 Resultados 3 Bases de datos JSSP

	PROBLEMA JSSP 15X7	PROBLEMA JSSP 20X7	PROBLEMA JSSP 30X7
FCFS	78	72	95.5
MWKR	68	68	83.5
ALL HDR	68	66	79
RAND 100	66	67	82.5
RAND 500	66	67	79.5
RAND 1000	65	67	79.5
RAND 10000	64	66	78
RAND 50000	64	66	77.5
RAND 100000	**63.5**	66	77.5
RAND 200000	**63.5**	66	77.5
AG	**63.5**	**63**	**75.5**

Los resultados del AG superan los resultados que arrojan los métodos heurísticos tradicionalmente utilizados. La secuenciación por la regla de despacho Primeras entradas primeras Salidas fue la que tuvo el peor desempeño.

De los métodos que utiliza *Win QSB* para resolver los problemas de secuenciación para un JSSP, el que arroja en todos los casos mejores resultados, es el método basado en la generación de secuencias aleatorias. En este caso se generaron como tope 200,000 secuencias aleatorias.

El algoritmo solo fue empatado en la primera simulación, siendo este el problema con menor número de posibles soluciones.

5.3. Resultados de la simulación para el algoritmo genético con codificación para generar secuencias activas para un sistema de producción intermitente flexible

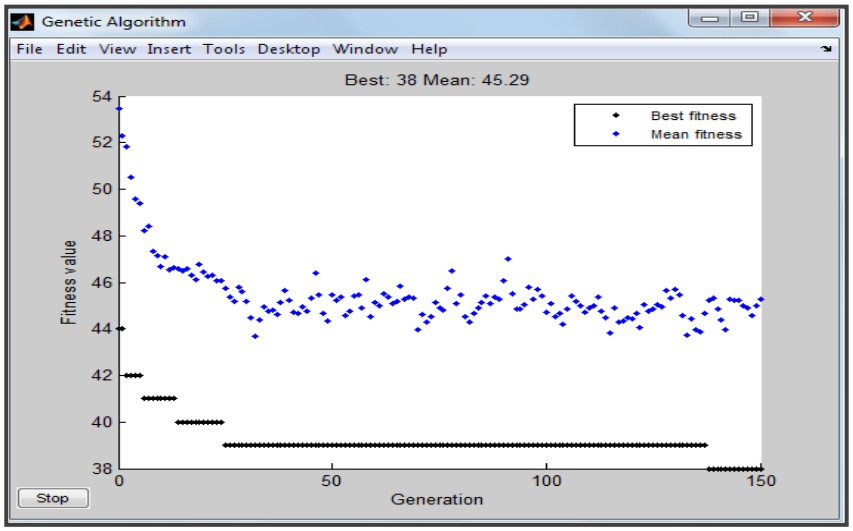
En la tabla 5.6 se muestran los valores arrojados para la simulación de 5 ensayos para un problema de *JSS* con la asignación de cada tarea a una máquina de entre 2 posibles, es decir, con máquinas paralelas. Estos primeros ensayos para la base el problema con 15 actividades y 7 máquinas. Se muestran los tiempos en que se alcanzó cada uno de los valores del Cmáx, y en negritas el mejor valor alcanzado por el AG.

Tabla 5.6 Resultados de la simulación, parámetros y Funciones del AG para el FJSSP

RESULTADOS PARA LA SIMULACIÓN DE 15 ACTIVIDADES Y 7 MÁQUINAS EN UN PROBLEMA DE JSSP FLEXIBLE			
Ensayo Número	Parámetro alcanzado en generación no.	tiempo corrida	parámetro alcanzado Cmáx
1	20	56 seg	41
2	58	2 min 42 seg	41
3	25	1 min 10 seg	40
4	138	6 min 26 seg	**38**
5	102	4 min 45 seg	41

Parámetro óptimo	Desconocido
Función Evaluación	Con codificación basada en operaciones, generación de secuencias activas, con **máquinas paralelas**
Función Cruzamiento	Cruzamiento en dos puntos , genera el 60% de la población
Función Mutación	Tipo inversión de un bit
Función Escalamiento	Escalamiento Lineal
Función Selección	Selección por método de Ruleta
Función Migración	Migración tipo pasarela en ambas direcciones
Población	100 individuos
Criterio de paro	150 Generaciones

La gráfica 5.4, perteneciente al ensayo no. 4 de la tabla 5.6 y muestra en cada generación el valor del Cmáx alcanzado por la mejor secuencia hasta dicha generación, aparece también en azul el promedio del Cmáx de los individuos de la generación. En la generación 138 se observa que alcanzó el valor de 38.



Gráfica 5.4 Cmáx VS Generaciones

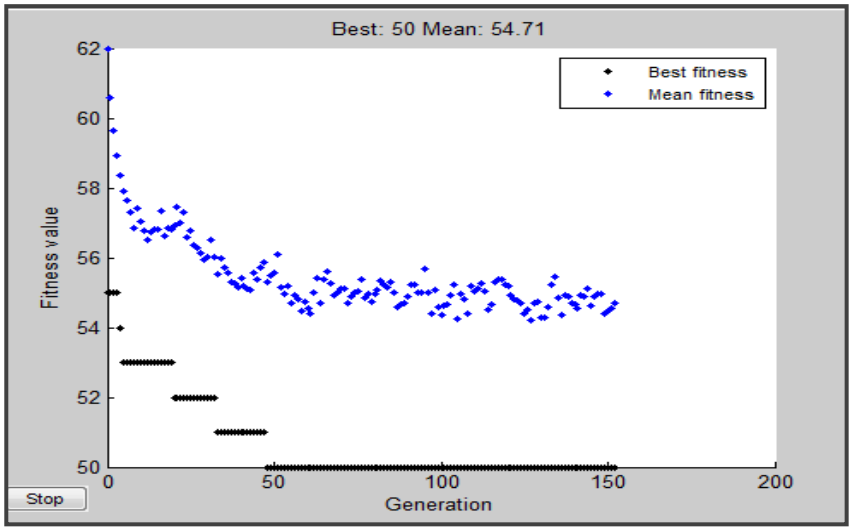
En la tabla 5.7 se muestran los resultados para la simulación del JSSP flexible con 20 actividades y 7 máquinas.

Tabla 5.7 Resultados de la simulación, parámetros y Funciones del AG para el FJSSP

RESULTADOS PARA LA SIMULACIÓN DE 20 ACTIVIDADES Y 7 MÁQUINAS EN UN PROBLEMA DE JSSP FLEXIBLE			
Ensayo Número	Parámetro alcanzado en generación no.	tiempo de corrida	parámetro alcanzado Cmáx
1	104	2 min 49 seg	52
2	98	2 min 36 seg	51
3	1	10 seg	52
4	50	4 min 9 seg	**50**
5	1	10 seg	52

Parámetro óptimo	Desconocido
Función Evaluación	Con codificación basada en operaciones, generación de secuencias activas, con **máquinas paralelas**
Función Cruzamiento	Cruzamiento en dos puntos , genera el 60% de la población
Función Mutación	Tipo inversión de un bit
Función Escalamiento	Escalamiento Lineal
Función Selección	Selección por método de Ruleta
Función Migración	Migración tipo pasarela en ambas direcciones
Población	Ensayo 3, 4 y 5 con 100 individuos , ensayo 1 y 2 con 30 individuos
Criterio de paro	150 Generaciones

La gráfica 5.5, obtenida del ensayo no. 4 de la tabla 5.7 muestra el comportamiento del AG. El mínimo Cmáx y el Cmáx promedio obtenido en cada generación.



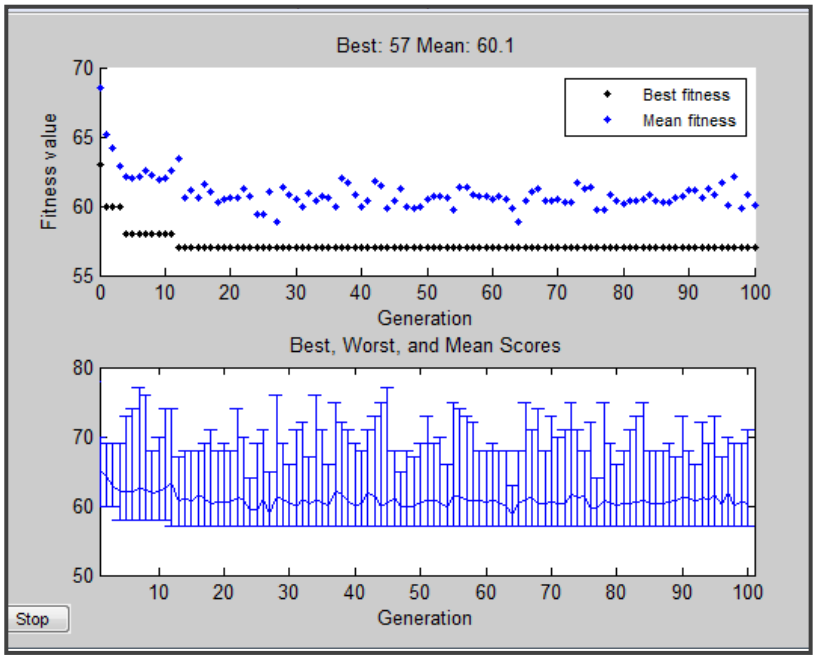
Gráfica 5.5 Cmáx VS Generaciones

Tabla 5.8 Resultados de la simulación, parámetros y Funciones del AG para el FJSSP

RESULTADOS PARA LA SIMULACIÓN DE 30 ACTIVIDADES Y 7 MÁQUINAS EN UN PROBLEMA DE JSSP FLEXIBLE			
Ensayo Número	Parámetro alcanzado en generación no.	tiempo corrida	parámetro alcanzado Cmáx
1	120	7 min 48 seg	58
2	117	7 min 34 seg	58
3	12	46 seg	**57**
4	92	6 min	58
5	63	4 min 5 seg	59

Parámetro óptimo	Desconocido
Función Evaluación	Con codificación basada en operaciones, generación de secuencias activas, con **máquinas paralelas**
Función Cruzamiento	Cruzamiento en dos puntos , genera el 60% de la población
Función Mutación	Tipo inversión de un bit
Función Escalamiento	Escalamiento Lineal
Función Selección	Selección por método de Ruleta
Función Migración	Migración tipo pasarela en ambas direcciones
Población	30 individuos
Criterio de paro	200 Generaciones

En la tabla 5.8 se muestran los resultados para la simulación del JSSP flexible con 30 actividades y 7 máquinas. La gráfica 5.6, perteneciente al ensayo no. 3 de la tabla 5.8 y muestra en cada generación el valor del Cmáx alcanzado por la mejor secuencia hasta dicha generación, aparece también en azul el promedio del Cmáx de los individuos de la generación. En la generación numero 12 se observa que alcanzó el valor de 57.



Gráfica 5.6 Cmáx VS Generaciones

5.4. Comparación de resultados con WinQSB para el FJSSP

En la tabla 5.9 se muestra la comparación de los resultados obtenidos anteriormente con el segundo AG desarrollado en este trabajo y los métodos heurísticos utilizados por el programa WinQSB. Las abreviaturas son las mismas que en la tabla 5.4.

Tabla 5.9 Resultados 3 Bases de datos FJSSP

	PROBLEMA FJSSP 15X7	PROBLEMA FJSSP 20X7	PROBLEMA FJSSP 30X7
FCFS	45	54	70
MWKR	41	52	57
ALL HDR	40	51	58
RAND 100	42	53	60
RAND 500	41	52	59
RAND 1000	41	51	59
RAND 10000	40	51	56
RAND 50000	39	50	56
RAND 100000	**39**	50	56
RAND 200000	**39**	50	56
AG	**38**	**50**	**57**

El algoritmo para estas tres simulaciones con la posibilidad de asignación a máquinas paralelas no se desempeñó como el anterior.

Para la primera base de datos fue donde mostró su mejor desempeño en comparación con el mejor resultado obtenido por medio de *WinQSB*.

Para la segunda base de datos solo alcanzó para empatar el mejor de los resultados de *WinQSB*, que lo obtuvo como en las anteriores situaciones mediante la búsqueda de secuencias aleatorias. Es en la tercera base de datos donde se obtuvo un mejor resultado con la generación aleatoria de secuencias y no gracias al AG.

6. CONCLUSIONES

Los resultados para el primero de los dos algoritmos genéticos, el desarrollado para la resolución de problemas del tipo JSS, arrojó mejores resultados para dos de los tres problemas, para el JSSP con 15 actividades, solo empató el parámetro de *WinQSB* (63.5), siendo este resultado el mejor tiempo posible para ese conjunto de actividades. Los mejores resultados de *WinQSB* son arrojados al generar un gran número de secuencias aleatorias, los métodos heurísticos tradicionales no arrojaron ningún resultado superior a los obtenidos con el AG.

El segundo algoritmo resultó más complejo en su programación por encargarse no solo de buscar soluciones que representen secuencias activas, sino que además, se encarga de decidir en qué máquina habrá de procesarse una tarea, ya que en este tipo de problema se permite la asignación a más de una máquina disponible. Esto repercutió en el tiempo de cómputo, incrementado en un 200 % a comparación del algoritmo anterior. Aun así, arrojó para la primera base de datos el mejor resultado sobre cualquiera obtenido por *WinQSB*, en la segunda base de datos, obtuvo un resultado igual que el obtenido por la generación de secuencias aleatorias y para la tercera fue el único ensayo en donde *Win QSB* superó el parámetro obtenido por el AG.

La búsqueda en el espacio de soluciones realizada por el AG está orientada por varios parámetros y funciones que se buscaron optimizar para su buen funcionamiento. Aun así, es deseable mejorar el desempeño del algoritmo para disminuir su tiempo de cómputo. Se obtuvieron resultados en lapsos no mayores que los que ocupa *WinQSB* para generar un gran número de secuencias aleatorias.

Una desventaja de este algoritmo es que no siempre se obtiene la misma solución corriendo la simulación con los mismos parámetros, en el mismo tiempo y claro con la misma base de datos. En ocasiones se encontró una muy buena solución en un lapso muy corto de tiempo, y en otras se dio

el caso de obtener resultados con un Cmáx más grande, invirtiendo lapsos de cómputo mayores. Esto se debe a que la búsqueda del algoritmo no siempre apunta al mismo espacio de soluciones, aunque en teoría, al dejar correr el algoritmo de forma indeterminada, este debe encontrar las mismas soluciones. No se realizó de esta manera por la inconveniencia de prolongar por mucho tiempo cada uno de los ensayos.

En una de las etapas para la simulación del algoritmo en *MATLAB,* se tomaron en cuenta como soluciones iniciales las soluciones con el Cmáx arrojadas por *WinQSB*, partiendo de estas que se suponen como buenas soluciones, se logró disminuir el parámetro en algunos casos. Si bien fue en un porcentaje mínimo esta mejora, esto muestra que el algoritmo desarrollado de inicio puede utilizarse para mejorar alguna secuenciación previamente obtenida por algún otro método.

Otra ventaja del algoritmo es que su estructura permite que al modificar solo la función de evaluación del desempeño, que en este caso evaluó el *Intervalo Operativo*, se pueda utilizar para buscar soluciones optimas basadas en otros parámetros utilizados en la ingeniería industrial para mejorar el desempeño de la secuenciación de actividades.

El sistema de producción intermitente se ha implementado en muchas empresas o talleres por las necesidades del mercado en el que compiten. Las herramientas para este tipo de empresas suelen ser menos, y la utilización de una herramienta pensada para servir de apoyo a sistemas de producción continua puede resultarles contraproducente. El algoritmo desarrollado en este trabajo puede ser utilizado como base para el desarrollo de herramientas computacionales que cumplan con necesidades específicas para estas empresas. A diferencia de las técnicas ya establecidas, este algoritmo puede ser moldeado mediante los parámetros y funciones que modifican su desempeño para cumplir con necesidades específicas de un sistema de producción intermitente.

BIBLIOGRAFÍA

1 Daniel Sipper , Robert L. Bulfin Jr., *Planeación y Control de la Producción,* Mc Graw-Hill education (1998)

2 Bagchi, Tapan P. *Multiobjective Scheduling by Genetic Algorithm .* Kluwer Academic Publishers , 1999

3 Alexander Alberto Correa, Ph.d, Elkin Rodríguez Velázquez , Msc, María Isabel Londoño Restrepo, Ing . *Secuenciación de Operaciones para Configuraciones de tipo planta tipo flexible Job Shop: estado del arte,* Escuela de Ingeniería de la organización , Universidad Nacional de Colombia, 2008

4 Dr. Fabio Vicentini, Dra. Susana Puddu, *Algoritmos Heurísticos y el problema de Job Shop Scheduling.* Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, 2003

5 David E. Golberg, *Genetic Algorithms in search, Optimizations & Machine Learning.* Addison-wesley publishing company,1989.

6 Mitsuo Gen, Runwei Cheng, Lin Lin. *Networks Models and Optimizations, Multiobjetive Genetic Algorithm Approach.* Editorial Springer 2008

7 Efredy delgado A., Carlos julio cortés, Oscar duarte V., *Aplicación de Algoritmos Genéticos para la Programación de Tareas en una celda de manufactura,* Revista Ingeniería e Investigación, vol. 25, no 2 2005

8 Mc. Martín Martinez Rangel, *El problema de calendarizar el trabajo en un taller,* Centro de Investigaciones en Ingeniería y Ciencias Aplicadas. Universidad autónoma del Estado de Morelos, 2006

9 Juan Carlos Osorio Gomez, Tulio Gerardo Garavito, *modelo jerárquico para el job shop flexible,* Asociación Brasileña de ingeniería de producción Abepro, 2007

10 Ravindra k. Ahuja, Özlem Ergun, james b. Orlin, Abraham p. Punnen, *Estudio de Técnicas de búsqueda por vecindad a muy gran escala*, 1999

11 Ho, N.B y Tay, J C, *Envolving distpatching rules using genetic programming for solving multi-objective flexible job shop problems,* Revista Computers & Industrial Engineering vol. 54

APÉNDICE

Solución para el ensayo 1 de la tabla 5.1 , 15 actividades X 7 máquinas

Optimresults.x =

[2 6 12 14 11 12 7 6 3 5 7 13 5 6 11 4 6 2 8 8 14 2 3 1 9 8 9
1 10 13 5 3 3 3 12 3 11 15 14 15 14 2 1 12 5 12 12 2 15 13 14 14 7
10 1 9 8 9 5 13 10 9 15 1 4 10 12 13 11 10 4 9 1 7 13 5 7 11 1
6 6 15 9 8 2 10 11 5 13 15 4 8 10 15 14 6 11 4 4 8 7 4 7 2 3]

Solución para el ensayo 3 de la tabla 5.2, 20 actividades X 7 máquinas

Optimresults.x=[

15 5 15 3 2 19 15 8 6 1 7 11 3 11 8 13 3 6 2 19 18 17 13 5 7 16 11 10
3 14 3 20 17 12 5 1 16 19 9 19 7 16 14 1 1 1 8 4 14 9 7 6 7 17 2 13
20 6 2 19 18 4 13 1 20 18 9 1 12 5 15 3 11 8 13 3 5 4 12 18 12 6 18 19
12 6 8 17 8 2 4 15 16 14 20 10 11 12 7 9 19 18 10 4 14 15 7 16 17 10 5 12
9 11 20 13 11 13 4 6 9 14 17 10 2 14 4 10 16 18 20 17 2 9 10 8 16 20 15 5]

Solución para el ensayo 2 de la tabla 5.3, 30 actividades X 7 máquinas

Optimresults.x=[

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

Solución para el ensayo 4 de la tabla 5.6, 15 actividades X 7 máquinas

Optimresults.x =

[1 14 6 3 6 5 6 6 14 13 1 3 3 2 5 7 6 14 7 13 7 1 7 11 8 14 9
15 10 1 3 13 5 4 12 1 13 4 2 5 11 8 8 9 15 15 10 9 3 2 10 13 5
4 4 2 12 11 8 4 4 8 8 14 15 13 13 3 9 5 15 12 6 2 15 15 12 10 12
2 4 9 10 12 10 12 10 8 14 5 1 9 14 7 2 7 7 9 3 1 6 11 11 11 11]

Solución para el ensayo 4 de la tabla 5.7, 20 actividades X 7 máquinas

Optimresults.x =

[1 17 8 9 18 19 19 8 11 1 5 5 6 14 1 4 12 4 6 13 20 12 14 16 11 16 17 2
10 7 16 7 18 7 16 3 12 12 10 13 11 20 2 15 3 10 13 12 20 11 19 9 11 5 6 1
14 4 14 1 4 19 12 4 15 3 10 7 3 18 6 9 9 13 20 12 14 19 13 1 17 11 20 18
15 8 16 2 2 15 5 10 17 15 4 15 6 18 2 10 7 15 7 17 8 9 18 19 19 8 5 2
10 3 4 9 3 20 13 9 14 7 6 20 17 16 8 13 11 3 1 5 5 2 8 6 17 16 14 18]

Solución para el ensayo 3 de la tabla 5.8, 30 actividades X 7 máquinas

Optimresults.x =[

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

Datos del problema JSS 15 actividades X 7 máquinas, resultados en la tabla 5.1

Trabajo 1 4/1 - 1.5/2 - 1.5/3 - 6/4 - 4/5 - 2/1 - 3/3
Trabajo 2 4/1 - 1.5/3 - 5/4 - 4/5 - 4/3
Trabajo 3 2/6 - 2.5/7 - 2/3
Trabajo 4 7/6 - 7/2 - 2.5/3
Trabajo 5 4/1 - 5/2 - 1.5/3 - 4/4 - 4/5 - 2/1 - 3/3
Trabajo 6 10/1 - 3/2 - 1.5/3 - 4/4 - 10/5 - 4/3
Trabajo 7 4/6 - 2.5/7 - 1.5/3
Trabajo 8 6/2 - 2/3 - 3/4 - 3/5 - 2/3
Trabajo 9 6/2 - 2/3 - 3/4 - 3/5 - 2/3
Trabajo 10 5/2 - 2.5/3
Trabajo 11 6/6 - 4/3 - 6/7
Trabajo 12 4/1 - 2/2 - 1.5/3
Trabajo 13 8/1 - 1.5/3 - 3.5/4 - 7/5 - 4/1 - 2/3
Trabajo 14 10/1 - 1.5/3 - 4/4 - 10/5 - 4/1 - 3/3
Trabajo 15 4/1 - 2.5/2 - 1.5/3

Problema JSS 20 actividades X 7 máquinas, resultados en la tabla 5.2

Trabajo 1 6/1 - 5/2 - 5/1 - 3/5 - 4/4 - 5/3 - 5/7
Trabajo 2 3/1 - 3/7 - 4/4 - 3/5 - 2/3
Trabajo 3 1/6 - 2/7 - 1/3
Trabajo 4 6/6 - 6/2 - 2/3
Trabajo 5 3/1 - 4/2 - 4/7 - 3/4 - 3/5 - 2/1 - 2/3
Trabajo 6 6/6 - 2/2 - 1/3 - 5/4 - 5/7 - 3/2

Trabajo 7 3/6 - 4/7 - 5/3
Trabajo 8 5/2 - 3/1 - 2/4 - 2/5 - 1/3
Trabajo 9 6/5 - 5/1 - 2/3 - 6/2 - 4/5
Trabajo 10 4/2 - 4/4 - 6/6
Trabajo 11 6/4 - 3/3 - 5/1
Trabajo 12 4/2 - 2/1 - 3/3
Trabajo 13 7/4 - 2/3 - 3/4 - 6/5 - 3/1 - 4/5
Trabajo 14 9/2 - 3/6 - 3/4 - 6/7 - 2/3 - 4/1
Trabajo 15 3/1 - 2/2 - 3/3
Trabajo 16 6/6 - 6/2 - 2/4
Trabajo 17 3/4 - 1/2 - 4/5 - 5/1 - 1/3
Trabajo 18 3/1 - 1/3 - 6/4 - 3/5
Trabajo 19 3/2 - 6/1 - 2/6 - 3/7 - 6/3
Trabajo 20 3/1 - 6/4 - 6/7 - 5/1 - 3/5

Problema JSS 30 actividades X 7 máquinas, resultados en la tabla 5.3

Trabajo 1 3/1 - 0.5/2 - 0.5/3 - 5/4 - 3/5 - 1/1 - 2/3
Trabajo 2 3/1 - 0.5/3 - 4/4 - 3/5 - 2/3
Trabajo 3 1/6 - 1.5/7 - 1/3
Trabajo 4 6/6 - 6/2 - 1.5/3
Trabajo 5 3/1 - 4/2 - 0.5/3 - 3/4 - 3/5 - 1/1 - 2/3
Trabajo 6 9/1 - 2/2 - 0.5/3 - 3/4 - 9/5 - 3/3
Trabajo 7 3/6 - 1.5/7 - 0.5/3
Trabajo 8 5/2 - 1/3 - 2/4 - 2/5 - 1/3
Trabajo 9 5/2 - 1/3 - 2/4 - 2/5 - 1/3
Trabajo 10 4/2 - 1.5/3
Trabajo 11 5/5 - 3/3 - 5/6
Trabajo 12 3/1 - 1/2 - 0.5/3
Trabajo 13 7/1 - 0.5/3 - 2.5/4 - 6/5 - 3/1 - 1/3
Trabajo 14 9/1 - 0.5/3 - 3/4 - 9/5 - 3/1 - 2/3
Trabajo 15 3/1 - 1.5/2 - 0.5/3
Trabajo 16 6/6 - 6/2 - 1.5/3
Trabajo 17 3/1 - 1/2 - 4/4 - 1.5/5 - 1/3
Trabajo 18 2.5/1 - 1/3 - 5/4 - 3/5
Trabajo 19 2.5/1 - 3/2 - 0.5/3 - 2/5 - 6/3
Trabajo 20 2.5/1 - 6/4 - 2/5 - 1/1 - 3/3
Trabajo 21 2.5/2 - 2/3 - 1.5/5 - 1.5/3
Trabajo 22 2.5/6 - 2/3 - 5/4 - 3/5 - 2/3
Trabajo 23 4/6 - 1.5/2 - 2/3
Trabajo 24 6/2 - 1/3 - 3/5 - 1/3
Trabajo 25 8/2 - 1/3 - 4/4 - 1/3
Trabajo 26 2.5/1 - 1.5/2 - 0.5/3 - 2/4 - 3/5 - 5/7
Trabajo 27 4/6 - 4/7 - 2/3
Trabajo 28 5/1 - 1/2 - 2/3 - 6/5 - 1/1 - 1/3
Trabajo 29 4/2 - 0.5/3 - 4/4 - 3.5/5 - 2/3
Trabajo 30 1.5/6 - 2/7 - 1.5/3

Problema JSS Flexible 15 actividades X 7 máquinas, resultados en la tabla 5.6

Trabajo 1 4/1 4/2-1/2 1/3- 1/6 1/4- 6/4 6/5 - 4/5 4/6 -2/1 2/7 -3/3 3/1
Trabajo 2 4/1 4/2 - 1/3 - 5/4 - 4/5 - 4/7 4/6
Tranajo 3 2/6 2/7 - 2/7 2/1 - 2/3 2/2
Trabajo 4 7/6 - 7/2 7/4 - 2/3 2/5
Trabajo 5 4/1 4/6 - 5/2 5/7 - 1/5 1/1 - 4/4 4/2 - 4/5 4/3 - 2/1 2/4 - 3/3 3/5
Trabajo 6 10/1 10/6 -3/2 3/7 - 1/3 1/1 - 4/4 4/2 - 10/5 10/3 - 4/3 4/4
Trabajo 7 4/6 4/5 - 2/7 2/6 - 1/2 1/7
Trabajo 8 6/2 6/1 - 2/3 2/2 - 3/4 3/3 - 3/5 3/4 - 2/3 2/5
Trabajo 9 6/2 6/6 - 2/3 2/7 - 3/4 3/1 - 3/5 3/2 - 2/7 2/3
Trabajo 10 5/2 5/4 - 2/3 2/5
Trabajo 11 6/6 - 4/3 4/7 - 6/7 6/1
Trabajo 12 4/1 4/2 - 2/2 2/7 - 1/5 1/4
Trabajo 13 8/1 8/5 - 1/3 1/6 - 3/4 3/7 - 7/5 7/1 - 4/1 4/2 - 2/3
Trabajo 14 10/1 10/4 - 1/3 1/5 - 4/4 4/6 - 10/5 10/7 - 4/1 - 3/3 3/2
Trabajo 15 4/1 4/3 - 2/2 2/4 - 1/3 1/5

Problema JSS Flexible 20 actividades X 7 máquinas, resultados en la tabla 5.7

Trabajo 1 6/1 6/2 - 5/2 5/3 - 5/1 5/4 - 3/5 - 4/4 4/6 - 5/3 5/7 - 5/7 5/1
Trabajo 2 3/1 3/2 - 3/3 3/4 - 4/4 4/5 - 3/5 3/6 -2/3 2/7
Trabajo 3 1/6 1/7 - 2/7 2/6 - 1/3 1/5
Trabajo 4 6/6 6/4 - 6/2 6/3 - 2/3 2/2
Trabajo 5 3/1 - 4/2 4/7 - 4/3 4/6 - 3/4 3/5 - 3/5 3/4 - 2/1 2/3 - 2/3 2/2
Trabajo 6 6/6 6/1 - 2/2 2/7 - 1/3 1/6 - 5/4 5/7 - 5/7 5/6 - 3/3 3/5
Trabajo 7 3/6 3/4 - 4/7 4/3 - 5/3 5/2
Trabajo 8 5/2 5/1 - 1/3 1/7 - 2/4 2/6 - 2/5 - 1/3 1/4
Trabajo 9 6/5 6/3 - 5/1 5/2 - 2/3 2/1 - 6/2 6/7 - 4/5 4/6
Trabajo 10 4/2 4/5 - 2/3 2/4 - 6/6 6/3
Trabajo 11 6/4 6/2 - 3/3 3/2 - 5/1
Trabajo 12 4/2 4/7 - 2/1 2/6 - 3/3 3/5
Trabajo 13 7/4 - 2/3 2/5 - 3/4 - 6/5 6/3 - 3/1 3/2 - 1/3 1/1
Trabajo 14 9/2 9/7 - 3/3 3/6 - 3/4 3/5 - 6/7 6/4 - 2/3 - 4/1 4/2
Trabajo 15 3/1 - 2/2 2/7 - 3/3 3/6
Trabajo 16 6/6 6/5 - 6/2 6/4 - 2/3
Trabajo 17 3/4 3/2 - 1/2 1/1 - 4/5 4/7 - 5/1 5/6 - 1/3 1/5
Trabajo 18 3/1 3/4 - 1/3 - 6/4 6/2 - 3/5 3/1
Trabajo 19 3/2 3/7 - 6/1 6/6 - 2/6 2/5 - 3/7 3/4 - 6/3
Trabajo 20 3/1 3/2 - 6/4 6/1 - 6/7 - 5/1 5/6 - 3/3 3/5

Problema JSS Flexible 30 actividades X 7 máquinas, resultados en la tabla 5.8

Trabajo 1 3/1 3/6 - 1/2 1/7 - 1/3 1/1 - 5/4 5/2 - 3/5 3/2 - 1/1 1/4 - 2/3 2/5
Trabajo 2 3/1 3/6 - 1/3 1/7 - 4/4 4/1 - 3/5 3/2 - 2/3 2/5
Trabajo 3 1/6 1/4 - 2/7 2/5 - 1/3 1/6

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