UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

FACULTAD DE INGENIERÍA INDUSTRIAL

UNIDAD DE POSGRADO

Enrutamiento y secuenciación óptimos en un flexible Job Shop multiobjetivo mediante algoritmos genéticos

TESIS

Para optar el Grado Académico de Doctor en Ingeniería Industrial

AUTOR

Guillermo TEJADA MUÑOZ

ASESOR

Juan Manuel CEVALLOS AMPUERO

Lima – Perú

2017

UNIVERSIDAD NACIONAL
MAYOR DE SAN MARCOS
Universidad del Perú, DECANA DE AMERICA

UNIDAD DE POSGRADO

ACTA DE SUSTENTACIÓN Nº 20-UPG-FII-2017

SUSTENTACIÓN DE TESIS PARA OPTAR EL GRADO ACADÉMICO DE DOCTOR EN INGENIERIA INDUSTRIAL

En la ciudad de Lima, del día uno del mes de agosto del dos mil diecisiete, siendo las nueve horas, en acto público se instaló el Jurado Examinador para la Sustentación de la Tesis titulada: **"ENRUTAMIENTO Y SECUENCIACIÓN ÓPTIMOS EN UN FLEXIBLE JOB SHOP MULTIOBJETIVO MEDIANTE ALGORITMOS GENÉTICOS",** para optar el Grado Académico de Doctor en Ingeniería Industrial.

Luego de la exposición y absueltas las preguntas del Jurado Examinador se procedió a la calificación individual y secreta, habiendo sido.....APROBADO..... con la calificación de.....DIECISIETE (17).....

El Jurado recomienda que la Facultad acuerde el otorgamiento del Grado Académico de Doctor en Ingeniería Industrial, al **Mg. TEJADA MUÑOZ GUILLERMO.**

En señal de conformidad, siendo las.....10:20..... horas se suscribe la presente acta en cuatro ejemplares, dándose por concluido el acto.

Dr. INCHE MITMA, JORGE LUIS
Presidente

Dr. CHUNG PINZÁS, ALFONSO RAMÓN
Miembro

Dr. TEONILA DORIA , GARCÍA ZAPATA
Miembro

Dr. WONG CABANILLAS, FRANCISCO JAVIER
Miembro

Dr. CEVALLOS AMPUERO, JUAN MANUEL
Asesor

DEDICATORIA

Dedico este trabajo, con eterna e infinita gratitud, a la memoria de mis padres, Amanda Muñoz y Manuel Tejada.

AGRADECIMIENTO

A mi asesor, Dr. Juan Manuel Cevallos Ampuero, por su acertada orientación.

A mis amigos y familiares por su aliento constante.

Índice General

Lista de Tablas

Lista de Figuras

RESUMEN

El presente trabajo de tesis propone una solución óptima al problema de programar (*Scheduling*) el procesamiento de un conjunto de *Jobs* (Tareas) en un conjunto de máquinas de una manufactura tipo Flexible Job Shop (*FJS*-Taller Flexible). La solución minimiza tres criterios: El *Maximum Workload* (W_M), el *Total Workload* (W_T) y el *Makespan* (C_M), es decir, el problema es de Optimización Multiobjetivo.

El problema *FJS* es actualmente estudiado por muchos investigadores porque corresponde a uno de optimización combinatoria muy difícil de resolver (NP-Hard) y porque una solución óptima redunda en una producción eficiente de la manufactura. El problema también es conocido en la literatura como *Flexible Job Shop Scheduling* (*FJSS*) o *Flexible Job Shop Scheduling Problem* (*FJSSP*), cualquiera de estos términos son utilizados indistintamente en el presente trabajo.

Se ha solucionado el *FJSSP* desde un enfoque jerárquico que divide el problema en dos de menor complejidad: El Subproblema de Enrutamiento y el Subproblema de Secuenciación, utilizando en ambos subproblemas algoritmos genéticos.

El desempeño de los algoritmos ha sido demostrado solucionando los casos de *FJSS* planteados por Kacem, utilizados también por otros investigadores, por lo que es posible comparar los resultados. Adicionalmente, las soluciones son presentadas, para una validación objetiva, en Diagramas de Gantt y datos numéricos. El programa ha sido totalmente codificado en Lenguaje M (Matlab).

Palabras Claves: Manufactura Flexible, *Flexible Job Shop Scheduling*, *Flexible Job Shop Scheduling Problem*, Algoritmos Genéticos, Diagramas de Gantt, Metaheurística, Heurística, Enrutamiento, Secuenciación, *Makespan*, *workload*, *Maximum Workload*, *Total Workload*

ABSTRACT

This thesis proposes an optimal solution to the problem of programming (Scheduling) the processing of a set of Jobs in a set of machines of a Flexible Job Shop (FJS). The solution minimizes three performance criteria: Maximum Workload (WM), Total Workload (WT) and Makespan (CM), that is, the problem is multiobjective optimization.

Many researchers are currently studying the problem because it is a problem of combinatorial optimization very difficult to solve (NP-Hard) and because an optimal solution leads to an efficient production of the manufacture. The problem is also referenced in the literature as Flexible Job Shop Scheduling (FJSS) or Flexible Job Shop Scheduling Problem (*FJSSP*); any of these terms are used interchangeably in the present work.

The *FJSSP* has been solved from a hierarchical approach that divides the problem into two of less complexity: The Routing Subproblem and the Sequencing Subproblem, using in both subproblems genetic algorithms.

The performance of the algorithms has been demonstrated by solving the instances of FJSS of Kacem, also used by other researchers, so that the results can be compared. In addition, the solutions are presented, for objective validation, in Gantt Diagrams and numerical data. The program has been fully coded in M Language (Matlab).

Keywords: Flexible Manufacturing, Job Shop Scheduling Problem, Flexible Job Shop Scheduling Problem, Genetic Algorithms, Gantt Diagrams, Metaheuristics, Heuristics, Routing, Sequencing, Makespan, Workload, Maximum Workload, Total Workload.

CAPÍTULO 1: INTRODUCCIÓN

Uno de los problemas más difíciles de resolver, y que en los siguientes ítems será apropiadamente fundamentado y referenciado, es el encontrado en la programación de trabajos o tareas de un sistema de manufactura tipo taller flexible, cuya denominación en inglés se le conoce como *Flexible Job Shop Scheduling Problem* (*FJSSP*), nombre que en este trabajo de tesis se prefiere utilizar. También indistintamente se utilizarán lo términos de *Flexible Job Shop (FJS), Flexible Job Shop Scheduling (FJSS)* y cuando se trate de tarea(s) o trabajo(s) el término de *Job(s).*

El *FJSSP* está constituido por un conjunto de *Jobs* y un conjunto de máquinas. Cada *Job* se constituye con un número de operaciones que deben ser procesadas en las máquinas en un orden establecido (precedencia) y cada operación tiene la flexibilidad de ser procesada en cualquiera de las máquinas pero con costos de tiempos no necesariamente iguales.

En un *FJSSP* el desafío consiste encontrar las rutas de procesamiento de todos los *Jobs* y la secuenciación de las operaciones en cada máquina minimizando algunas funciones objetivos, con el consiguiente resultado de contribuir en incrementar la eficiencia de la producción en las industrias.

La ruta o camino de procesamiento de un *Job* está conformada por las máquinas elegidas para procesar sus operaciones. Así por ejemplo, si uno de los *Jobs* consiste de tres operaciones como por ejemplo: perforar, ranurar y pulir. Entonces, una entre muchas alternativas de rutas puede ser: La máquina 2 para perforar, luego la máquina 1 para ranurar y finalmente la máquina 3 para pulir. Podría suponerse rutas ideales con las máquinas más veloces. Sin embargo, si el proceso se concentra solamente en esas

máquinas generará sobrecarga de trabajo en ellas y un consiguiente retardo para el procesamiento de todas las operaciones de los *Jobs*. Por eso, en la propuesta de la tesis se entiende por "Enrutamiento Óptimo" la mejor manera de designar las "m" máquinas a las "p" operaciones del *FJSS,* luego de la cual, entonces, las "m" máquinas resulten con carga de trabajo mínimas y equilibradas, características que son estimadas mediante la minimización del *Maximum Workload* (W_M) y el *Total Workload* (W_T).

Después de solucionar el enrutamiento, son conocidas las máquinas que van a procesar las operaciones de los *Jobs*. Pero el orden o secuencia de procesamiento de las operaciones en cada máquina es desconocido. Así por ejemplo, si dos o más operaciones de diferentes *Jobs* demandan el servicio de una máquina, es necesario saber cuál operación debe ser procesada primero, cuál de ellas segunda, etc. Existen muchas combinaciones de secuenciación y ellas tienen impacto en el tiempo que tomará concluir el procesamiento de todos los *Jobs*, este tiempo es denominado *Makespan*. Por eso, en la propuesta de tesis se entiende por "Secuenciación Óptima" encontrar la secuencia u orden de operaciones en cada máquina minimizando el *Makespan* (C_M).

El número de posibles soluciones en un *FJSS* crece exponencialmente con el tamaño del problema. Así por ejemplo, en un *FJSS* de 4 *jobs*, 5 máquinas y 12 operaciones se puede estimar tener 10^9 posibles soluciones por analizar, mientras que en un *FJSS* de 15 *Jobs*, 10 máquinas, 56 operaciones el número de soluciones puede estar en el orden de 10^{84}, de todas estas soluciones hay que encontrar la óptima y la que cumpla con las restricciones del *FJSS*. Es por eso, que un algoritmo determinístico no puede encontrar una solución en tiempos polinómicos sino en tiempos exponenciales que para tamaños grandes de *FJSS* resultan ser tiempos totalmente prohibitivos.

Entonces, el *FJSS* es un problema intratable fuertemente NP-Hard, con soluciones solo aproximadas al óptimo mediante algoritmos no determinísticos que resuelven el problema en tiempos polinómicos.

Por esta razón, en esta investigación de tesis la solución planteada es mediante el diseño e implementación de Algoritmos Genéticos multiobjetivo, en donde los criterios a minimizar son: La carga de trabajo total de las máquinas (*Total Workload* - W_T), la carga de trabajo de la máquina más cargada (*Maximum Workload* - W_M) y el tiempo de terminación de procesamiento de todos los *Jobs* (*Makespan*-C_M). Reduciendo la complejidad del problema mediante un enfoque jerárquico.

A lo largo del presente capítulo, se ha descrito y formulado el problema de la investigación, la justificación y los objetivos.

El capítulo II, es el Marco Teórico, en donde se describen los fundamentos teóricos de la tesis, merece destacarse los antecedentes de la investigación que es abordado en el ítem 2.2 y las bases teóricas del ítem 2.3.

En el capítulo III, se describe la Metodología que se ha utilizado para cumplir con los objetivos. Es decir, se describe el diseño de los algoritmos genéticos y los algoritmos para la presentación gráfica de los resultados, así como se describe también el ensamblaje de todos los algoritmos.

En el Capítulo IV, se detallan los resultados de los algoritmos solucionando los casos de *FJSS* planteados por Kacem et al. (2002) y Kacem et al. (2002a). Se monitorea el funcionamiento del programa para análisis y verificación de las hipótesis planteadas. También se comparan los resultados con los de otros investigadores, quienes similarmente han solucionado los casos planteados por Kacem. Los resultados son presentados, para una validación objetiva en Diagramas de Gantt y los datos numéricos son exportados a una hoja de Excel.

En el capítulo V, se extiende el alcance de la tesis, con el propósito de ilustrar su aplicación a casos prácticos que han sido encontrados en la literatura.

El capítulo VI, corresponde a las conclusiones y sugerencias para trabajos futuros.

Finalmente, en el apéndice A se muestra el programa codificado en el lenguaje M de Matlab y en el Apéndice B se registran los datos numéricos de la solución de los cinco casos de *FJSS* planteados por Kacem, por cada caso se muestran resultados de veinte (20) corridas.

1.1 Situación Problemática

La programación adecuada de *Jobs* o trabajos en procesos de manufactura constituye un importante problema que se plantea dentro de la producción en muchas empresas. El orden en que los *Jobs* son procesados, es de suma importancia, pues determinará algún parámetro de interés que convendrá optimizar en la medida de lo posible. Así por ejemplo, puede verse afectado el coste total de ejecución de los trabajos, el tiempo necesario para concluirlos o el *stock* de productos en curso que será generado. Esto conduce de forma directa al problema de determinar cuál será el orden más adecuado para llevar a cabo los trabajos con vista a optimizar algunos de los anteriores parámetros u otros similares (Márquez et al., 2012).

Uno de los problemas más difíciles en la programación de *Jobs* se encuentra en la manufactura tipo taller (*Job Shop Scheduling Problem* - *JSSP*), donde un conjunto de *Jobs* que tienen una secuencia de operaciones consecutivas deben ser procesadas en un conjunto de máquinas. Cada operación requiere exactamente una máquina, las máquinas están siempre disponibles y pueden procesar una operación a la vez sin interrupción. El problema se genera al secuenciar las operaciones en las máquinas tratando de optimizar un indicador de rendimiento típico, como por ejemplo, el *Makespan*, es decir, el tiempo necesario para completar todos los trabajos (Pezzella et al., 2008).

Un proceso más complicado que el *JSSP* es el *Flexible Job Shop Scheduling Problem* (*FJSSP*), el cual es una generalización del *JSSP.* En el caso del *FJSSP* no solo hay que encontrar la secuencia de las operaciones en cada máquina como en el clásico *JSSP* sino que previamente a cada operación se le debe designar una máquina, entre un conjunto de máquinas disponibles, encargada para procesarla, es decir, encontrar las rutas de trabajo (Pezzella et al., 2008).

El problema *FJSS* se divide en dos subproblemas estrechamente relacionados. El primer subproblema (puede ser considerado como uno de máquinas paralelas) consiste en asignar, de un conjunto de máquinas candidatas, una máquina apropiada que ejecute cada operación; el segundo subproblema consiste en secuenciar el orden en que se ejecutaran las operaciones en cada una de las máquinas, lo que equivale a un clásico problema de programación del *Job Shop*. Ambos subproblemas se han demostrado ser de complejidad combinatoria NP-Hard (Li et al., 2010).

El *FJSSP* incrementa su complejidad al ser Multiobjetivo, es decir, al optimizar o minimizar varios criterios tales como: El *Makespan* (tiempo de finalización de todos los *Jobs*)*, Total Workload* (carga de trabajo total de las máquinas) y el *Maximum Workload* (carga de trabajo de la máquina más cargada) (Hsu et al, 2002). Razón por la cual se ha intensificado la necesidad de desarrollar nuevos enfoques en el ámbito Multiobjetivo (Wojakowski & Warzolek, 2013).

La dificultad de *FJSSP* sugiere adoptar métodos heurísticos para encontrar en tiempos razonables buenas soluciones, en lugar de buscar soluciones exactas, incluso para procesos pequeños. Las Heurísticas no necesariamente dan soluciones cuyo valor de la función objetivo es el óptimo, pero puede ser eficaz para la mayoría de los casos muy complicados. La adopción de Metaheurísticas tales como: Recocido Simulado, Búsqueda Tabú y Algoritmos Genéticos, han dado buenos resultados (Pezella, et al., 2008).

Las publicaciones, dedicadas a la optimización de *FJSSP* Multiobjetivo, siguen siendo reducidas en comparación con el número de obras relacionadas con otras clases de problemas de *Job* Schedulling. Sin embargo, en los últimos años existe una tendencia incremental por el estudio de estos problemas (Genova et al., 2015).

En cuanto a los datos, con los cuales son puestos a prueba los algoritmos, muchos científicos han presentado casos de *FJSSP* que han tenido amplia acogida entre los investigadores: Inicialmente Brandimarte (1993) presentó casos para un *FJSSP* con diversos grados de flexibilidad de producción de las máquinas. Más tarde, Hurink, Chambers and Barnes (1994) generaron casos con la muy especial propiedad que los tiempos de procesamiento de las operaciones son independientes de las máquinas asignadas. Últimamente, Kacem (2002) presentó casos con total flexibilidad, donde cada operación se puede procesar en cualquiera de las máquinas (Behnke & Geiger, 2012).

1.2 Formulación del Problema

1.2.1 Problema General

¿Cómo solucionar el problema de programar óptimamente, en las máquinas de un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), el procesamiento de las operaciones de los *Jobs*, dividiendo el problema, en un Subproblema de Enrutamiento y un Subproblema de Secuenciación, minimizando los objetivos del *Maximum Workload*, *Total Workload* y *Makespan* mediante la utilización de Algoritmos Genéticos?

1.2.2 Problemas Específicos

a) ¿Cómo solucionar el Subproblema de Enrutamiento óptimo, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en asignar a cada operación de los *Jobs*, una entre muchas máquinas candidatas para su procesamiento, minimizando los criterios de: *Total Workload* (W_T, suma de carga de trabajo de todas las máquinas) y *Maximum Workload* (W_M, máxima carga de trabajo entre todas las máquinas) mediante Algoritmos Genéticos?

b) ¿Cómo solucionar el Subproblema de Secuenciación óptima, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en secuenciar el orden en que se deben procesar las operaciones de los *Jobs* en las máquinas seleccionadas previamente, minimizando el objetivo del *Makespan* (C_M, tiempo de finalización de todos los *Jobs*) mediante Algoritmos Genéticos?

c) ¿Cómo probar la eficacia y eficiencia de los algoritmos propuestos considerando la existencia de resultados obtenidos por otros investigadores de la literatura que han probado sus algoritmos con los casos de sistemas de fabricación tipo *FJS* planteados por Kacem?

1.3 Justificación Teórica

El *Flexible Job Shop Scheduling* Problem (*FJSSP*) es uno de los problemas de optimización combinatoria más difíciles de resolver. Pertenece a la familia de NP-Hard (No Determinístico de Tiempo Polinómico Difícil). La solución de los problemas NP-Hard con un único objetivo es una tarea difícil. La adición de más de un objetivo (Multiobjetivo), obviamente, hace que este problema sea aún más difícil de resolver (Mekni & Chaâr, 2015).

Las publicaciones, dedicadas a la optimización de *FJSSP* Multiobjetivo, continúan siendo pequeñas en comparación con el número de obras relacionadas a otras clases de problemas de *Job* Schedulling. Pero existe una tendencia de incrementar el interés hacia el estudio de estos problemas en los últimos años (Genova et al., 2015).

1.4 Justificación Práctica

La optimización de la programación de tareas es muy importante en la industria moderna por permitir ahorrar recursos. Consecuentemente, la adecuada programación de trabajos en procesos de manufactura constituye un importante problema que se plantea dentro de la producción de muchas industrias (Márquez et al., 2012).

1.5. Objetivos

1.5.1 Objetivo General

Solucionar el problema de programar óptimamente, en las máquinas de un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), el procesamiento de las operaciones de los *Jobs*, dividiendo el problema, en un Subproblema de Enrutamiento y un Subproblema de Secuenciación, minimizando los objetivos del *Maximum Workload*, *Total Workload* y *Makespan* mediante la utilización de Algoritmos Genéticos.

1.5.2 Objetivos Específicos

a) Solucionar el Subproblema de Enrutamiento óptimo, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en asignar a cada operación de los *Jobs*, una entre muchas máquinas candidatas para su procesamiento, minimizando los criterios de: *Total Workload* (W_T, suma de carga de trabajo de todas las máquinas) y *Maximum Workload* (W_M, máxima carga de trabajo entre todas las máquinas) mediante la utilización de Algoritmos Genéticos.

b) Solucionar el Subproblema de Secuenciación óptima, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en secuenciar el orden en que se deben procesar las operaciones de los *Jobs* en las máquinas seleccionadas previamente, minimizando el objetivo del *Makespan* (C_M, tiempo de finalización de todos los *Jobs*) mediante Algoritmos Genéticos.

c) Probar la eficacia y eficiencia de los algoritmos propuestos considerando la existencia de resultados obtenidos por otros investigadores de la literatura que han probado sus algoritmos con los casos de sistemas de fabricación tipo *FJS* planteados por Kacem.

CAPITULO 2: MARCO TEÓRICO

2.1 Marco Filosófico o Epistemológico de la Investigación

Optimizar o minimizar las funciones objetivos, tales como *Maximum Workload*, *Total Workload* y *Makespan* de un *Flexible Job Shop Scheduling Problem (FSSP)*, tema de la investigación del presente trabajo de tesis, consiste en solucionar un problema combinacional con un grado de dificultad computacional NP-Hard. Por ello, los métodos deterministas no son utilizados como alternativa, optándose en cambio por los métodos heurísticos y metaheurísticos, en donde el abordaje ni las soluciones son únicas, razón por la que hasta el día de hoy, los investigadores siguen tratando al tema. Bajo esta perspectiva, es pertinente dar a conocer a los pensadores que han dado origen a los algoritmos heurísticos como respuesta viable y hasta única frente a la complejidad de un problema combinacional NP-Hard.

2.1.1 Heurística y Metaheurística

El término heurística aparece en el período clásico de la Grecia antigua. Etimológicamente consiste en la invención y el descubrimiento debido a la reflexión y no al azar. En consecuencia, todos los factores irracionales deben quedar fuera de la heurística. Desde el punto de vista heurístico, no existe diferencia entre invención y descubrimiento. La heurística forma parte, de los métodos del descubrimiento científico. Más ampliamente, la heurística forma parte del proceso mismo de descubrimiento y de investigación teórica (Maldonado, 2005).

Se debe distinguir un sentido *lato* (amplio) *y* un sentido *stricto* (estricto) del término 'heurística' y debe ser entendido en su sentido *stricto*. En sentido *lato* está más próxima de la psicología, intentando decir algo acerca de la naturaleza de los descubridores, pero en sentido *stricto está* más próxima de la lógica y metodología, coincide con la heurística 'genuina' de Lakatos, intenta decir algo acerca de la naturaleza del mundo a descubrir, o sobre la dinámica y estructura de las hipótesis, o sobre sus relaciones, es un auxilio, principio o elemento que pueden ser presentado metodológicamente (Menna, 2014).

El concepto básico de búsqueda heurística como una ayuda para la resolución de problemas se introdujo por primera vez por Polya en 1945. Una heurística es una técnica (que consiste en una regla o un conjunto de reglas) que busca (y se espera que encuentre) buenas soluciones a un coste computacional razonable. Una heurística es aproximado en el sentido de que ofrece (con suerte) una buena solución para un esfuerzo relativamente pequeño, pero no garantiza el óptimo (Voß, 2001).

El término metaheurísticas se obtiene de anteponer a heurística el sufijo meta que significa "más allá" o "a un nivel superior". Las metaheurísticas son estrategias inteligentes para diseñar o mejorar procedimientos heurísticos muy generales con un alto rendimiento. El término metaheurística apareció por primera vez en el artículo seminal sobre Búsqueda Tabú de Glover en 1986. A partir de entonces han surgido multitud de propuestas de pautas para diseñar buenos procedimientos para resolver ciertos problemas que, al ampliar su campo de aplicación, han adoptado la denominación de metaheurísticas (Melian et al., 2003).

El término heurístico, puede referirse a aquel procedimiento lógico por medio del cual se pretende solucionar un problema de manera eficiente contando con el conocimiento disponible acerca de él. Este mismo principio puede aplicarse a los algoritmos. De esta manera, en la investigación operativa se entiende por algoritmo heurístico aquel método donde se aplica este

principio a un procedimiento de solución de un problema de optimización, del cual se espera encontrar soluciones de alta calidad en un tiempo de cómputo razonable, aunque muchas veces no se puede estimar que tan cerca se encuentre de la solución óptima. Por otra parte, el razonamiento subyacente en los heurísticos o metaheurísticos es inspirado en algún proceso natural, artificial o físico (Morillo et al., 2014).

2.1.2 Problemas Científicos y Soluciones Heurísticas

Ciertamente la palabra *investigación* ya existía entre los griego y es solo en el siglo XX que surgió el concepto de *investigación*, con el aporte de Lakatos (1978). El núcleo de la investigación científica consiste en la formulación y búsqueda de solución de problemas, los problemas científicos y teóricos se definen en función de la complejidad combinatoria. Los de optimización combinatoria suelen pertenecer a la denominada NP (*nondeterministic polynomial time*). Es decir, que se pueden resolver en un tiempo polinómico no determinista, en la cual están incluidos aquellos problemas para los que no se conoce un algoritmo polinomial de resolución (Maldonado, 2005).

Son ejemplos de optimización combinatoria el problema del vendedor viajante (TSP), el problema de asignación cuadrática (QAP), el problema de horarios y el problema de programación de tareas. El uso de métodos aproximados de solución ha recibido cada vez más atención en los últimos treinta años. En los métodos aproximados se deja de lado la garantía de encontrar soluciones óptimas a cambio de obtener buenas soluciones en mucho menor de tiempo (Blum & Roli, 2003).

De acuerdo a Moreno et al. (2007), para los problemas de optimización NP-Hard se utilizan métodos aproximados mediante heurísticas que permiten aproximarse a una solución óptima, generando soluciones factibles al problema que resulten de utilidad práctica, existen muchos tipos de algoritmos metaheurísticos, pero en general se pueden clasificar en:

- Metaheurísticas de relajación: Procedimientos que relajan o modifican el problema de forma que el problema original sea más fácil de resolver.
- Metaheurísticas constructivas: Procedimientos que soluciona al problema a partir del análisis y selección paulatina de las componentes que la forman. Los algoritmos más utilizados son los algoritmos voraces (*greedy algorithms*) que construyen soluciones buscando lo mejor de partes de la misma sin buscar óptimos globales.
- Metaheurísticas de búsqueda: Procedimientos mediante transformaciones o movimientos para recorrer el espacio de soluciones y explotar las estructuras de entornos asociadas. Entre ellas se encuentran los algoritmos de *Hill climbing* de arranque múltiple (Martí, 2003), *Simulated Annealing* (Kirkpatrick, 1983), búsqueda tabú (Glover, 1997).
- Metaheurísticas evolutivas: Utilizan conjuntos de soluciones que evolucionan sobre el espacio de soluciones. Entre ellas se encuentran los algoritmos genéticos (Holland, 1975), meméticos (Moscato, 2003), estimación de distribuciones (Lozano, 2001), reencadenamiento de caminos (Glover, 2003), colonias de hormigas (Dorigo, 1996), etc.

2.1.3 Ciencia y Heurística

De acuerdo con el filósofo Morín, es indispensable un método cuyo mérito es el de superar las limitaciones y las trabas del reduccionismo. Para Morín, su método debe ser visto como el camino que ayude a pensar por uno mismo para responder al desafío de la complejidad de los problemas (Maldonado, 2005).

En la metodología anterior al siglo XVII, inferencias y heurísticas tenían igual dignidad, y las funciones heurísticas y evidenciales de las reglas eran igualmente valoradas. Esta relación sufrió grandes transformaciones, y el estudio de las heurísticas fue prácticamente abandonado. Este estudio está

comenzando a ser nuevamente valorizado, y podemos anunciar un 'giro heurístico' de la metodología o mejor, un 'retorno' de las reglas heurísticas a la metodología de la ciencia (Menna, 2014).

Las metodologías exactas y heurísticas no deben entrar en conflicto o competencia puesto que estas últimas se usan para resolver problemas con características particulares, las cuales podrían hacer imposible usar los métodos exactos o podrían ser muy costosos en relación al esfuerzo computacional (Morillo et al., 2014).

2.2. Antecedentes de la Investigación

Brandimarte (1993) propone un enfoque jerárquico para encontrar una solución al problema del *Flexible Job Shop Scheduling,* y considera en un nivel de la jerarquía una solución al subproblema de asignación de máquinas para las operaciones y, en otro nivel, la solución del subproblema de secuencia de las operaciones, es decir, la solución del *Job Shop*. Utiliza Reglas de Despacho (*Dispatching Rules*) y Búsqueda Tabú (*Tabu Search*– *TS*) para solucionar ambos subproblemas, a diferencia de otros enfoques de jerarquía, en la cual la información fluye en un solo sentido de los niveles, en el enfoque del autor la información fluye en ambos sentidos. Considera la minimización de dos funciones objetivos: El *Makespan* (tiempo de terminación de todos los trabajos) y el *Total Weighted Tardiness* (resultado de sumar los productos entre la tardanza de producción de cada trabajo por el peso de prioridad asignado a cada trabajo). El autor crea quince casos prototipo de sistema de fabricación tipo *FJSS*, con las cuales prueba su propuesta.

Kacem et al. (2002a), proponen un enfoque de Pareto basado en la hibridación, es decir, la combinación de la Lógica Difusa (*Fuzzy Logic*) con los Algoritmos Evolutivos (*Evolutionary Algorithms*) para solucionar el problema del *Flexible Job Shop Scheduling*, aprovechando las capacidades

de representación del conocimiento por parte de la Lógica Difusa y la capacidad adaptativa de los Algoritmos Evolutivos. La función objetivo minimiza el *Makespan*, *Total Workload* y el *Maximum Workload*. Aplica Operadores de Mutación que emulan tener Organismos Modificados Genéticamente (OMG) los cuales crean cromosomas que aceleran la convergencia a la solución final, un sistema Multiobjetivo de Lógica Difusa ayuda a decidir si los nuevos individuos representan una interesante solución para las siguientes etapas del algoritmo. A pesar que no se garantiza una solución óptima, este enfoque proporciona soluciones de calidad, en un tiempo de ejecución del algoritmo razonable. Los autores son creadores de cinco (5) casos prototipo para simular sistemas de fabricación tipo *FJSS* que prueban su propuesta.

Xia & Wu (2005), aplican un enfoque jerárquico para minimizar tres criterios: el *Makespan*, el *Total Workload* y el *Maximum Workload*. La propuesta utiliza el Algoritmo de Optimización por Enjambre de Partículas (*Particle Swarm Optimization-PSO*) para asignar máquinas a las operaciones (Enrutamiento) y el algoritmo de Recocido Simulado (*Simulated Annealing- SA*) para programar las operaciones en cada máquina (Secuenciación). La función objetivo se define como la suma ponderada de los tres criterios. PSO utiliza las soluciones evaluadas por SA para continuar la evolución. Por medio de simulaciones computacionales se demuestra, que si bien este algoritmo híbrido no garantiza resultados óptimos, proporciona soluciones con buena calidad en un tiempo de ejecución del algoritmo razonable. Para ilustrar la efectividad y el desempeño del algoritmo, han sido solucionados tres, de los cinco, casos prototipo de sistema de fabricación tipo *FJSS* de Kacem et al. (2002), los cuales son los relativos al caso *FJSS* 8x8 (ocho *Jobs* y ocho máquinas), *FJSS* 10x10 (diez *Jobs* y diez máquinas) y el *FJSS* 15x10 (quince *Jobs* y quince máquinas).

Zhang et al. (2009), utilizan el algoritmo de Optimización de Enjamble de Partículas (*Particle Swarm Optimization-PSO*) conjuntamente con el algoritmo de Búsqueda Tabú (*Tabu Search-TS*), para solucionar el problema de *FJSS*. Como función objetivo utilizan una suma ponderada de tres

criterios: el *Makespan*; *Maximum Workload* y *Total Workload*. En su trabajo los autores codifican dos vectores: A-cadena (para las máquinas) y B-cadena (para las operaciones) las que describen respectivamente una asignación concreta de operaciones para cada máquina y la secuencia de las operaciones en cada máquina. El algoritmo fue implementado en C++ en una Computadora Personal Pentium IV de 1.8 GHz. y probado con cuatro casos prototipo de sistema de fabricación tipo *FJSS* de Kacem et al. (2002), que corresponden al caso FJSS 4x 5, caso FJSS 8x8, caso FJSS 10x10, y caso FJSS 15x10.

Xing et al. (2009 a), proponen un modelo de estructura de simulación para la solución del *FJSSP*. La estructura consta de 6 subsistemas: Subsistema de Entrada, Subsistema de Asignación de Operación, Subsistema de Secuencia de Operación, Subsistema de Evaluación del Objetivo, Subsistema de Control y Subsistema de Salida. El Subsistema de Asignación de Operación genera una asignación factible de operaciones para cada máquina y la optimiza teniendo en cuenta el *Total Workload* y el *Maximum Workload*. El algoritmo denominado Conocimiento de Asignación de Máquinas para la Operación (*Operation Assignment Machine Knowledge-OAMK*) es utilizado con la intención de mejorar el modelo de simulación en este Subsistema. En el Subsistema de Secuencia aleatoriamente se acomoda cada operación en cada máquina. Con el fin de mejorar el rendimiento de la secuencia, los autores aplican un algoritmo de Optimización de Colonia de Hormigas (*Ant Colony Optimization - ACO*). El Subsistema de Evaluación del Objetivo evalúa la programación por la suma ponderada del *Makespan*, *Total Workload* y *Maximum Workload*, los coeficientes de ponderación son obtenidas empíricamente. La función principal del Subsistema de Control es controlar el flujo computacional y la viabilidad de cálculo de las soluciones en curso. La función del Subsistema de Salida provee al usuario los resultados obtenidos de la optimización y el Diagrama de Gantt correspondiente.

El algoritmo ha sido codificado en Matlab, en una computadora Pentium IV, 2.4 Ghz y 512 MB RAM. Los autores prueban sus resultados mediante cinco

casos de sistema de fabricación tipo *FJSS* de Kacem et al. (2002). Para eliminar fluctuaciones aleatorias del proceso de optimización, los resultados experimentales fueron promediados sobre 10 corridas.

Xing et al. (2009b), con la finalidad de solucionar el problema del sistema de fabricación tipo FJSS proponen un método heurístico de búsqueda para la etapa de enrutamiento y seis reglas de despacho para la secuenciación de las operaciones. Minimiza las funciones objetivos: *Makespan*, *Total Workload* y *Maximum Workload* que son integrados a una sola función mediante la suma ponderada de las tres funciones. Utiliza diferentes coeficientes de ponderación para los tres funciones objetivos, independientemente para cada conjunto de pesos, muestra resultados. El algoritmo fue implementado en Matlab en una computadora *Workstation Dell Precision* 650 con *Pentium IV* de 2.4 Ghz y 1 GHz de RAM. Los casos prototipo de sistema de fabricación tipo *FJSS*, son tomados de Brandimarte (1993) y Kacem et al. (2002). Los resultados experimentales fueron promediados sobre 20 corridas.

Li et al. (2010), propone un Algoritmo Híbrido de Búsqueda Tabú (*Hybrid Tabu Search Algorithm-HTSA*) para solucionar el *FJSSP* Multiobjetivo. Los criterios por minimizar son: *Makespan*, *Total Workload* y *Maximum Workload*. Es utilizado un algoritmo de Búsqueda Tabú (*Tabu Search - TS*) para producir soluciones vecinas en el Módulo de Asignación de Máquinas y un algoritmo de Búsqueda Variable Vecina (*Variable Neighborhood Search-VNS*) realiza la búsqueda local en el Módulo de Secuenciación de Operaciones. Se aplican nuevas reglas de búsqueda de vecinos en ambos módulos. El algoritmo fue implementado en C++ en una Pentium IV de 1.7 GHz con 512 MB. El mejor resultado, promediando veinte ejecuciones independientes del algoritmo, fue registrado para comparación. El algoritmo fue probado con los caso de sistema de fabricación tipo *FJSS* planteados por Kacem et al. (2002) y Brandimarte (1993).

Motaghedi-Iarijani et al. (2010), desarrolla un Algoritmo Genético Multiobjetivo para resolver el *FJSSP*. Considera tres criterios: *Makespan*,

Total Workload y *Maximum Workload*. Es utilizado el Método de Criterio Global (también llamado Programación Compromiso) como un enfoque para la función objetivo, en este método es minimizada la distancia entre algún punto de referencia con la región del objetivo factible. Para codificar los cromosomas se utilizan dos representaciones vectoriales: A-Cadena y B-Cadena como en Zhang et al. (2009). El vector A-Cadena representa las máquinas asignadas a las operaciones y el vector B-cadena define la secuencia de las operaciones en las máquinas. El enfoque de Escalamiento de Colina (*Hill Climbing- HC*) ha sido utilizado para mejorar la solución de secuenciación derivadas del Algoritmo Genético. El algoritmo fue implementado en Java eclipse en un *Vostro* 1500 con 2.2 Ghz y 2 Gb de RAM. Prueba su propuesta con los cinco casos de sistema de fabricación tipo *FJSS* de Kacem et al. (2002).

Azardoost & Manipour (2011), en su estudio se presenta un Algoritmo Metaheurístico Híbrido basado en Búsqueda Tabú, Recocido Simulado y Algoritmos Genéticos. El enfoque desarrollado para encontrar la solución es jerárquico minimizando tres criterios: *Makespan*, *Maximum Workload* y *Total Workload*. Obtienen una ecuación lineal con el Software *SPSS* con la finalidad de obtener una óptima combinación de los sus tres operadores genéticos: Fusión (Porción de población que se fusiona con la siguiente generación), Cruce y Mutación. El algoritmo fue codificado en Matlab 7.4.0 y su eficacia y eficiencia es evaluada comparándolo con otros algoritmos heurísticos, para ello, las pruebas experimentales son realizadas con tres casos prototipo de sistema de fabricación tipo *FJSS* de Kacem et al. (2002) y diez de Brandimarte (1993).

Jiang et al. (2011), solucionan el *FJSSP* con un Algoritmo Genético con tasas de Mutación y Cruzamiento dinámicos y un Algoritmo Recocido Simulado. El enfoque utilizado es integrado, el Algoritmo Genético asigna máquinas a las operaciones y el Algoritmo Recocido Simulado secuencia las operaciones en las máquinas. Se minimiza dos criterios: *Makespan* y *Total Workload*, los resultados demuestran que este algoritmo híbrido puede proporcionar soluciones con buena calidad y su convergencia es rápida. Las

pruebas experimentales son realizadas con tres casos de sistema de fabricación tipo *FJSS* de Kacem et al. (2002).

Moslehi & Mahnam (2011), presentan un enfoque hibrido con un algoritmo de Enjambre de Partículas y otro de Búsqueda Local. Utilizan un enfoque integrado, es decir, tanto el enrutamiento como la secuenciación son solucionados paralelamente. Cada Partícula del Enjambre se compone de dos partes: una para definir la política de enrutamiento (asignación de la máquina para las operaciones) y la otra indicando la secuencia de las operaciones en cada máquina (prioridad de las operaciones). En el procedimiento de secuenciación se aplica un Algoritmo de Búsqueda Local a cada partícula del enjambre con la finalidad de volver asignar máquinas a las operaciones en la ruta crítica de la solución obtenida. Minimizan tres criterios: *Makespan*, *Maximum Workload* y *Total Workload*. El algoritmo fue implementado en Matlab 7 en una Pentium IV de 2 GHz con un sistema operativo Windows XP. Las pruebas experimentales son realizadas con tres casos de sistemas de fabricación tipo *FJSS* de Kacem et al. (2002).

Xiong et al. (2012), desarrollan un algoritmo hibrido, minimizando tres criterios: *Makespan*, *Total Workload* y el *Maximum Workload*. Además, incluyen una medida modificada para la Distancia de Crowding (Cuan próximo un individuo está de su vecino) para mantener la diversidad de los individuos e incorporan un Algoritmo de Búsqueda Local basado en la teoría de la ruta crítica que garantiza la convergencia de las soluciones óptimas de Pareto. Utilizan la representación de dos vectores para los cromosomas y la representación de permutaciones para la secuencia de operaciones. Las pruebas experimentales son realizadas con los casos de sistemas de fabricación tipo *FJSS* de Kacem et al. (2002) y Brandimarte (1993).

Chiang & Lin (2012), desarrollan un Algoritmo Memético Multiobjetivo (*Multiobjective Memetic Algorithm*) que resuelve el problema de *FJSSP* con tres criterios para minimizar: *Makespan*, *Maximum Workload* y *Total Workload*. Combinan un Algoritmo Genético con una técnica de búsqueda local. Se aplica una codificación del cromosoma, en donde cada gen es una

3-tupla (j, i, k), en la cual j, i, k representan los índices del *Job*, operación y máquina respectivamente. Las pruebas experimentales son realizadas con los cinco casos de sistemas de fabricación tipo *FJSS* de Kacem et al. (2002) y diez de Brandimarte (1993).

Chiang & Lin (2013), consideran la solución del *FJSSP* minimizando los criterios: *Makespan*, *Total Workload* y *Maximum Workload*. Proponen un Algoritmo Evolutivo Multiobjetivo (MOEA), utilizando operadores genéticos eficientes para la diversificación de la población. Para evaluar y comparar la solución utilizan la relación de dominancia de Pareto. Se aplica el esquema de codificación 3-tupla. El algoritmo encuentra 70% o más soluciones no dominadas, cuando se prueba con los casos de sistema de fabricación tipo *FJSS* de Kacem et al. (2002) y Brandimarte (1993).

Shao et al. (2013), proponen un algoritmo híbrido para encontrar soluciones óptimas de Pareto, se considera al *FJSSP* con tres criterios por minimizar al mismo tiempo: el *Makespan*, el *Maximum Workload* y el *Total Workload*. Para una búsqueda global es utilizada una variante de Optimización por Enjambre de Partículas (*Particle Swarm Optimization-PSO*), mientras que para búsqueda local, es utilizado un Algoritmo de Recocido Simulado (*Simulated Annealing- SA*). Para evaluar las Partículas del Enjambre, se aplica un método de clasificación de Pareto y la Distancia Crowding (medida que indica que tan próximo un individuo está de su vecino). El algoritmo ha sido implementado en C++, en una computadora personal de 2 GHz y 2Gb de RAM, y probado con los casos de sistema de fabricación tipo *FJSS* de 8x8 (8 *Jobs* × 8 máquinas), 10x10 (10 *Jobs* × 10 máquinas) y 15x10 (15 *Jobs*×10 máquinas), todos de Kacen et al. (2002).

Shahsavari-Poura & Ghasemishabankareh (2013), solucionan el *FJSSP* Multiobjetivo, minimizando los criterios: *Makespan*, *Total Workload* y el *Maximum Workload*, el algoritmo propuesto es una combinación del Algoritmo Genético Multiobjetivo (MOGA) y el Recocido Simulado (SA). La población inicial se genera por MOGA, y luego el número exacto de individuos en cada población se mejora por SA. Un enfoque de Solución

Óptima de Pareto se utiliza en MOGA y las soluciones son evaluadas para los tres criterios por minimizar durante todos los pasos. Utilizan dos vectores A-Cadena y B-Cadena para representar los cromosomas. El algoritmo es implementado en *Visual Basic Application* (VBA) corriendo en una PC de 2 GHz con 512 MB de RAM, se ha probado con los casos de sistema de fabricación tipo *FJSS* 4x5 (4 *Jobs* x 5máquinas) y 10x10 (10 *Jobs* x 10 máquinas) extraídos de Kacem et al. (2002).

Ziaee (2014), desarrolla un algoritmo heurístico basado en un procedimiento tipo constructivo para resolver el *FJSSP* minimizando tres funciones objetivos: *Makespan*, *Total Workload* y *Maximum Workload,* teniendo como restricciones el Mantenimiento Preventivo (PM). La suma ponderada de los tres objetivos anteriores convierte el problema en monoobjetivo, los períodos de mantenimiento, en los cuales las máquinas no están disponibles, también se incluyen en los tres criterios anteriores, ya que estos períodos son obligatorios por lo que se tienen que programar, junto con los *Jobs* a procesar. Mediante el establecimiento distintos pesos para las variables, se generan diferentes soluciones, las mismas que son evaluadas. El algoritmo fue codificado en Lenguaje C, corriendo en una Pentium IV de 2.2 GHz y 2 GB RAM. El autor hace referencia que los casos prototipo que han servido para probar el algoritmo son los presentados por Gao et al. (2006) y Rajkumar et al. (2010). Así mismo, estos casos se derivan de Kacem et al. (2002).

Sadaghiani et al. (2014), desarrollan un enfoque heurístico integrado para resolver el problema de *FJSSP* con tres objetivos por minimizar: *Makespan*, *Maximum Workload* y *Total Workload*. Integra los subproblemas de asignación y secuenciación. En primer lugar, la búsqueda se realiza en el Espacio de Asignación de Máquinas a las Operaciones, logrando una solución aceptable y luego la búsqueda prosigue en el Espacio de la Secuencia de Operaciones. En este trabajo los autores han utilizado un enfoque Multiobjetivo para producir y evaluar Soluciones de Pareto, para ello utilizan una versión adaptada del Algoritmo Genético *NSGA II*

(*Nondominated Sorting Genetic Algorithm II*). Prueban su algoritmo mediante los casos de sistema de fabricación tipo *FJSS* de Kacen et al. (2002).

Genova et al. (2015), documentan trabajos en donde se han aplicado modelos aproximados matemáticos y trabajos en donde se han aplicado soluciones heurísticas y metaheurísticas (algunos de los investigadores de ese estudio también han sido considerados en la presente investigación) para solucionar el *FJSSP* Multiobjetivo. Además, en la revisión que realizan se pueden notar investigadores que han minimizado otros criterios distintos al *Makespan* o *Workload*, estos criterios son por ejemplo: El Retardo Medio de los Jobs (*Mean Job Tardiness*), la Suma de los Tiempos de Ejecución (*Flow Time*), la Suma Promedio de los Tiempos de Ejecución (*Mean Flow Time*), el Tiempo Promedio de las Máquinas sin Utilizar (*Mean Machine Idle Time*), entre otros.

Finalmente, a la fecha de realizada esta investigación no se ha encontrado investigadores nacionales tratando el tema.

2.3 Bases Teóricas

2.3.1 Scheduling

Scheduling (Programar) es un proceso de toma de decisiones que se utiliza de forma regular en muchas industrias manufactureras y de servicios. Se ocupa de asignar recursos para los *Jobs* en determinados períodos de tiempo y su meta es optimizar uno o más objetivos. Los recursos pueden ser máquinas en un taller, pistas en un aeropuerto, equipos en un sitio de construcción, unidades de procesamiento en un entorno informático, etc. Los *Jobs* pueden ser operaciones en un proceso de producción, despegues y aterrizajes en un aeropuerto, etapas en un proyecto de construcción, ejecuciones de programas informáticos, etc. Cada *Job* puede tener un cierto

nivel de prioridad, un tiempo de inicio lo más anticipado posible y un tiempo de finalización. Los objetivos también pueden adoptar muchas formas diferentes. Un objetivo puede ser la minimización del tiempo de finalización del último *Job*, o la minimización del número de *Jobs* completados después de sus respectivos tiempos de finalización (Pinedo, 2016).

2.3.1.1 Producto-Proceso y Scheduling. En el ámbito de la fabricación se pueden encontrar casos simples como secuenciar un conjunto de tareas sobre una máquina determinada hasta situaciones más complejos como es el sistema de fabricación *Job Shop*. Para visualizar la complejidad, en la Figura 1 se muestra una Matriz Producto-Proceso, inicialmente desarrollada por Hayes & Wheelwright (1979) según se cita en Laviós (2013).

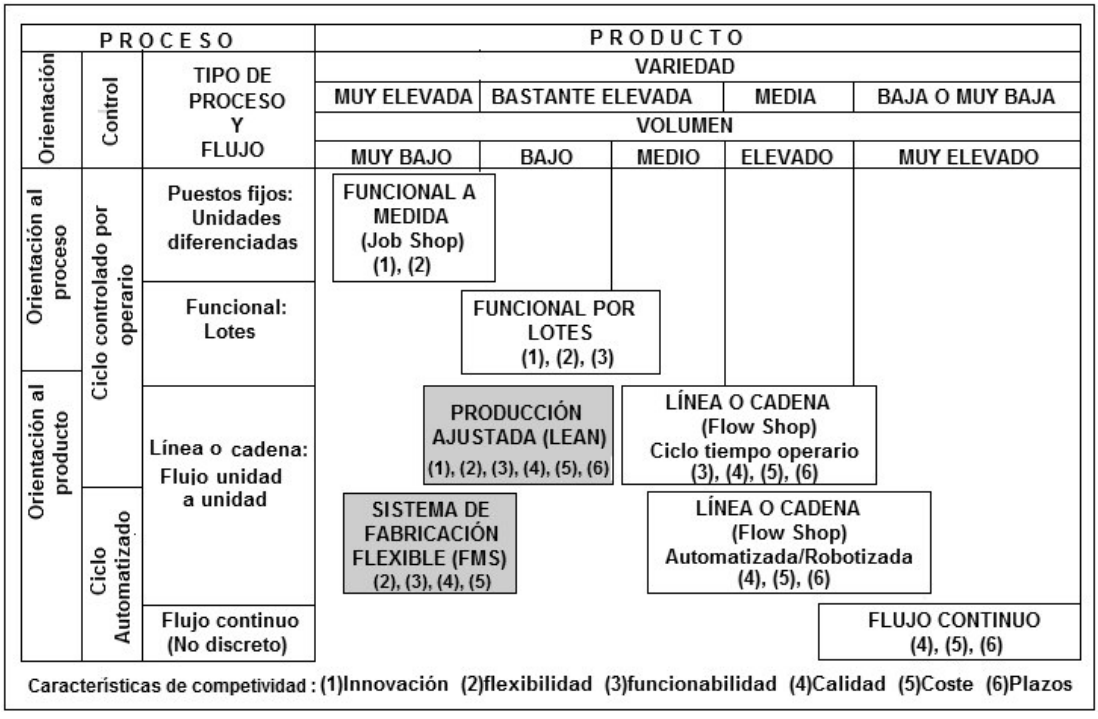


Figura 1. **Matriz Producto-Proceso**. *Fuente*. Laviós (2013)

Según Cuatrecasas (2009) citado por Laviós (2013), en la Matriz Producto-Proceso, se define los siguientes tipos de producción:

- Producción en Flujo Continuo: En esta modalidad productiva cada máquina y equipo realizan siempre la misma operación y el objeto en proceso es suministrado por la máquina precedente. Se caracteriza por tener un gran volumen de producción con calidad y costes muy bajos. Además, la variedad de los productos es pequeña y el diseño rigurosamente normalizado. En esta configuración, el *Scheduling* de operaciones se limita a determinar el tiempo de funcionamiento de la línea.
- Producción *Flow Shop*: Es un sistema de producción orientado a producto. Se fabrican lotes grandes de productos de poca variedad pero técnicamente homogéneos. Los productos requieren una secuencia similar de operaciones, aunque algunos de ellos puedan necesitar alguna operación diferente. Los puestos de trabajo con sus máquinas y equipos se disponen siguiendo el flujo de operaciones que sigue el producto. En esta configuración, el *Scheduling* de operaciones consiste en determinar la secuencia de entrada a la cadena de producción.
- Producción Funcional por Lotes: El producto no es a medida, los lotes son de gran volumen. La organización de la distribución del trabajo en la planta está orientada al proceso: recorridos largos y diversos para el producto, muchas actividades de manipulación y transporte, altas esperas en colas del producto y volúmenes importantes de stock. Consecuentemente, aparecen muchas actividades sin valor añadido (tiempos largos de proceso, stock, etc.). En este caso, el *Scheduling* de operaciones incluye la asignación de carga (si existe posibilidad de realizar el *Job* en más de una máquina), la secuenciación y la temporización.
- Producción Funcional a Medida (*Job Shop*): Se producen lotes más o menos pequeños de una amplia variedad de productos de poca o nula normalización. Generalmente, la distribución de planta está orientada al proceso, es decir, los equipos se agrupan en talleres o centros de trabajo según la función que desarrollan, son versátiles y permiten ejecutar operaciones diversas. Es habitual que exista pedidos pendientes (consecuentemente con prorroga en los plazos de entrega),

gran cantidad de stocks de materiales y *Jobs* en curso. Este tipo de producción acarrea cuellos de botella en determinados puestos de trabajo y se acumula mucho material en las colas internas. Al igual que en el caso anterior, el *Scheduling* de operaciones incluye la asignación, la secuenciación y la temporización.

Actualmente, las nuevas tendencias de gestión como *Lean Manufacturing* (Producción Ajustada) han hecho que se rompa el principio de la diagonal de la Figura 1. Los nuevos modelos, mostrados en color gris en la Figura 1, pretenden incluir características de competitividad como calidad, tiempo y coste, de los tipos de producción en línea; y la flexibilidad, funcionalidad y alta variación de producto, de los entornos enfocados a procesos. Brevemente se describen a continuación:

- La producción Lean, según Womack et al. (2007) citado por Laviós (2013), tiene por objeto optimizar los recursos y actividades necesarias para realizar la producción, utilizando procesos con orientación al producto. También, se busca lotes de producción pequeños y elevada variación de producto para alcanzar una mayor flexibilidad (zona izquierda de la matriz Producto-Proceso). El *Scheduling* de operaciones, en este caso, incluye la asignación, la secuenciación y la temporización.
- Los Sistemas de Fabricación Flexible (*FMS-Flexible Manufacturing Systems*), según Vollmann et al (1997) citado por Laviós (2013), son sistemas altamente automatizados y eficientes, con flexibilidad similar a los equipos poco especializados, que por sus bajos tiempos de configuración los hace adecuados para producciones de pequeños lotes, que permiten cambiar rápidamente a distintas variantes del producto. El *Scheduling* de operaciones*, en este caso, incluye* la asignación, la secuenciación y la temporización.

2.3.1.2 Representación de los Problemas de Scheduling Utilizando Triplete Estándar. En todos los problemas de *Scheduling* son finitos el número de *Jobs* y el número de máquinas, el número de *Jobs* se denota

por **n** y el número de máquinas por **m**. Por lo general, el subíndice j se refiere a un *Job* mientras que el subíndice i se refiere a una máquina. Si un *Job* requiere un número de pasos de procesamiento u operaciones, entonces el par (i, j) se refiere al paso de procesamiento u operación del *Job* j en la máquina i. Los datos asociados con el *Job* j son: Tiempo de Procesamiento, Fecha de Lanzamiento, Fecha de Vencimiento o Comprometida, Peso, como se describen líneas abajo (Pinedo, 2016).

- Tiempo de Procesamiento ($\mathbf{p_{ij}}$): Representa el tiempo de procesamiento del *Job* j en la máquina i. El subíndice i se omite si el tiempo de procesamiento del *Job* j no depende de la máquina o si el *Job* j se procesa en una sola máquina.
- Fecha de Lanzamiento ($\mathbf{r_j}$): Es la fecha en que el *Job* llega al sistema y puede iniciar su procesamiento.
- Fecha de Vencimiento o Comprometida ($\mathbf{d_j}$): Es la fecha comprometida con el cliente. Cuando debe ser cumplida es referida como fecha límite (*deadline*) y se denota por ($/\mathbf{d_j}$).
- Peso (**wj**): Es un factor de prioridad, denotando la prioridad del *Job* j en relación a los otros *Jobs* del sistema.

Según Pinedo (2016), un problema de *Scheduling* se puede describir mediante un triplete:

$$\boldsymbol{\alpha} \mid \boldsymbol{\beta} \mid \boldsymbol{\gamma}$$

$\boldsymbol{\alpha}$: Describe al entorno de la máquina y tiene una sola entrada. Puede tomar los siguientes valores:

- Una máquina ($\boldsymbol{\alpha} = \boldsymbol{1}$): Es el caso más simple de todos los entornos y es un caso especial.
- Máquinas en paralelo idénticas ($\boldsymbol{\alpha} = \boldsymbol{Pm}$): El *Job* j requiere una sola operación y puede ser procesada en cualquiera de **m** máquinas idénticas en paralelo.

- Máquinas en paralelo con diferentes velocidades ($\alpha = \boldsymbol{Qm}$): hay **m** máquinas en paralelo con diferentes velocidades. La velocidad de la máquina i es denotada por vi. El tiempo pij que el *Job* j emplea en la máquina i es pj/vi (asumiendo que el *Job* j se procesa toda en la máquina i). Este entorno es referido como máquinas uniformes).
- Máquinas en paralelo no relacionadas ($\alpha = \boldsymbol{Rm}$): hay diferentes **m** máquinas en paralelo. La máquina i puede procesar el *Job* j a la velocidad vij. El tiempo pij que el *Job* j emplea en la máquina i es pj/vij (asumiendo que el *Job* j recibe todo su tiempo de procesamiento de la máquina i).
- *Flow Shop* ($\alpha = \boldsymbol{Fm}$): Hay **m** máquinas en serie, cada *Job* debe ser procesada en cada una de las **m** máquinas. Todos os *Jobs* siguen la misma ruta. Por ejemplo, los *Jobs* deben ser procesados primero en la máquina 1, luego en la máquina 2, etc. Los *Jobs* deben esperar que el *Job* que llegó primero a la máquina sea procesada, esta característica es referida como *Permutation Flow Shop*, por lo tanto el campo $\boldsymbol{\beta}$ tomará el valor ***prmu***.
- *Flexible Flow Shop* ($\alpha = \boldsymbol{FFc}$): Es la generalización del *Flow Shop* con máquinas paralelas. En lugar de haber **m** máquinas en serie, hay **c** etapas en serie, cada etapa agrupando un número de máquinas en paralelo idénticas. Cada *Job* debe ser primero procesada en la etapa 1, luego en la etapa 2, etc. En cada etapa cualquiera de las máquinas puede procesar al *Job*. También se le conoce como *Hibrid Flow Shop* o *Multiprocessor Flow Shop*.
- *Job Shop* ($\alpha = \boldsymbol{Jm}$): En un *Job Shop* con **m** máquinas, cada *Job* tienen su propia predeterminada ruta a seguir. Se deben distinguir *Jobs* que solo pasan una vez por cada máquina y aquellos *Jobs* que puede pasar más de una vez por una determinada máquina. En este último caso el campo $\boldsymbol{\beta}$ debe contener la entrada ***rcrc*** indicando que se trata de un problema con recirculación.
- *Flexible Job Shop* ($\alpha = \boldsymbol{FJc}$): Es una generalización del entorno *Job Shop* y el entorno de máquinas paralelas. En lugar de disponer de **m** máquinas se cuenta con **c** centros de trabajo y en cada centro existen

máquinas en paralelo. Si el *Job* en su ruta a través de la planta puede visitar un centro de trabajo más de una vez el campo $\boldsymbol{\beta}$ contiene la entrada ***rcrc***.

- *Open Shop* ($\boldsymbol{\alpha}$ =***Om***): Se dispone de **m** máquinas, cada *Job* debe ser procesada en cada una de las **m** máquinas. Sin embargo, algunos tiempos de procesamiento pueden ser nulos. No hay restricciones con respecto a la ruta de cada *Job*.

$\boldsymbol{\beta}$: Describe características del procesamiento y restricciones, puede no tener o tener múltiples entradas. Puede tomar los siguientes valores:

- Fechas de Lanzamiento ($\boldsymbol{\beta}$ = ***rj***): Si este símbolo aparece en el campo $\boldsymbol{\beta}$, entonces un *Job* j no puede iniciar su procesamiento antes de su fecha de lanzamiento $\boldsymbol{r_j}$.
- Restricciones de No Continuidad ($\boldsymbol{\beta}$ =***prmp***): Este símbolo indica que no es necesario completar una tarea en una determinada máquina una vez que ha comenzado, el proceso puede ser interrumpido para dar pase al procesamiento de otro *Job*. Cuando ***prmp*** no está incluido los *Jobs* no pueden ser interrumpidos hasta su finalización.
- Restricciones de Precedencia ($\boldsymbol{\beta}$ =***prec***): Pueden aparecer en entornos de una máquina o en máquinas paralelas, requiere que uno o más *Jobs* completen su proceso antes que se inicie otro *Job*. Hay muchas formas especiales de restricciones de precedencia: cuando un *Job* puede tener como máximo una *Job* precedente y como máximo un *Job* sucesor, las restricciones son referidas como ***cadenas;*** Si cada *Job* tiene como máximo un sucesor, las restricciones son referidas como ***intree***; si cada *Job* tiene como máximo un predecesor, las restricciones son referidas como ***outtree***.
- Secuencia Dependiente del Tiempo de Preparación ($\boldsymbol{\beta}$ =$\boldsymbol{S_{jk}}$): Representa la secuencia dependiente del tiempo de preparación de las máquinas que es incurrido entre el procesamiento de los *Jobs* j y k. Si el tiempo de configuración depende de cada máquina, entoces debe ser incluido el subíndice i, es decir $\boldsymbol{S_{ijk}}$. Si ningún $\boldsymbol{S_{jk}}$ aparecen en el

campo β, entonces todos los tiempos de preparación son nulos (secuencia independiente), en tal caso los tiempos son incluidos en los tiempos de procesamiento.

- Familias de *Jobs* (β = ***fmls***): Los **n** *Jobs* pertenecen a F diferentes familias. *Jobs* de la misma familia pueden tener diferentes tiempos de procesamiento, sin ningún tiempo de preparación. Sin embargo, si la máquina cambia de una familia a otra (de la familia ***g*** a la ***h***), entonces se requiere un tiempo de preparación. Si este tiempo de preparación depende de las familias ***g*** y ***h*** y es dependiente de la secuencia, entonces es denotado por s_{gh}. Si este tiempo de preparación depende solamente de la familia que está a punto de comenzar, es decir, la familia ***h***, entonces se denotará por s_h. Si no depende de ninguna de las familias, entonces se denomina ***s***.
- Procesamiento por Lotes (β = ***batch (b)***): Una máquina puede ser capaz de procesar un lote de hasta b *Jobs*, simultáneamente. Los tiempos de procesamiento de los *Jobs* en un lote pueden no ser todos iguales y el lote completo sólo se termina cuando el último *Job* del lote se ha completado, lo que implica que el tiempo de finalización de todo el lote está determinado por el Job con el tiempo más largo de procesamiento. Si ***b*** = 1, entonces el problema se reduce a un entorno de *Scheduling* convencional. Otro caso especial que es de interés es ***b*** = ∞, es decir, no hay límite en el número de *Jobs* que la máquina puede manejar en cualquier tiempo.
- Paradas (β = ***brkdwn***): Implica que una máquina puede no estar continuamente disponible. El periodo que una máquina no está disponible son asumidas fijas (por ejemplo por mantenimiento programado). Las paradas también son conocidas como restricciones de disponibilidad.
- Restricciones de Elegibilidad de Máquina (β = ***Mj***): Significa que del conjunto de máquinas **m** en paralelo (***Pm***) solo un conjunto de máquinas ***Mj*** es capaz de procesar el *Job* j.
- Permutación (β = ***prmu***): Es una restricción que puede aparecer en el entorno *Flow Shop*. Implica que el orden (o permutación) en la cual los

Jobs van a través de la primera máquina se debe mantener durante todo el proceso.

- Bloqueo ($\boldsymbol{\beta}$ =***block***): Es un fenómeno que puede ocurrir en *Flow Shops*, cuando se tiene un buffer que limita entre dos máquinas. Al llenarse el buffer se impide que un *Job* pueda procesarse en la máquina precedente.
- Ninguna Espera ($\boldsymbol{\beta}$ =***nwt***): El requisito de ninguna espera es otro fenómeno que puede ocurrir en *Flow Shop*. Los *Jobs* no se permiten esperar entre dos máquinas sucesivas. Esto implica que el tiempo de inicio de un *Job* en la primera máquina tiene que ser retrasado para asegurarse que el *Job* puede pasar por el *Flow Shop* sin esperar a ninguna máquina.
- Recirculación ($\boldsymbol{\beta}$ =***rcrc***): Recirculación puede ocurrir en un *Job Shop* o *Flexible Job Shop* cuando un *Job* puede visitar una máquina o centro de trabajo más de una vez.

$\boldsymbol{\gamma}$: Describe el objetivo a minimizar y generalmente posee una sola entrada. El objetivo a ser minimizado es siempre una función del tiempo de finalización de los *Jobs*. El tiempo de finalización de la operación del *Job* j en la máquina i se denomina C_{ij}. El tiempo del *Job* j saliendo del sistema (es decir, el tiempo de finalización en la última máquina en la cual requiere procesamiento) se denomina Cj.

$$C_j = \max (C_{ij}) \tag{2.1}$$

- *Lateness* es definido como:

$$L_j = C_j - d_j \tag{2.2}$$

- *Tardiness* del *Job* es definida como:

$$T_j = \max (C_j - d_j, 0) = \max (L_j, 0) \tag{2.3}$$

La diferencia entre *Lateness* y *Tardiness* se basa en el hecho que el *Tardiness* nunca es negativo.

- *Earliness* (diferencia positiva entre la fecha de entrega comprometida y el tiempo de finalización del *Job*), ecuación 2.4.

$$E_i = \max (0; d_i - C_i) \qquad (2.4)$$

Las funciones objetivo se definen en base a las anteriores variables. Ejemplos de $\boldsymbol{\gamma}$, es decir de funciones objetivos a minimizar son: *Makespan*, *Maximum Lateness*, *Total Weighted Completion Time*, *Discounted Total Weighted Completion Time,* etc. (Pinedo, 2016).

2.3.2 Job Shop Scheduling (JSS)

Un sistema de fabricación tipo *Job Shop* (*JS*) se caracteriza por una alta variedad de productos y bajo volumen de producción, la programación de operaciones en ella (*Job Shop Scheduling,* JSS) puede plantearse como un problema de ***n*** *Jobs y* ***m*** *máquinas, en donde* cada *Job* está conformada de una secuencia ordenada de operaciones que deben ser procesadas, sin interrupción, en una de las máquinas. Se considera que en un mismo *Job* cada una de sus operaciones se debe procesar en máquinas diferentes, los tiempos de procesamiento están en unidades de tiempo. La minimización del *Makespan*, es el objetivo más utilizado encontrado en la literatura, se logra de una secuencia de procesamiento con menor tiempo posible. Teniendo en cuenta este criterio de optimización el *JSS* está clasificado como NP-Hard (Garey et al. 1976 citado por De Melo, 2014).

El problema, entonces, consiste en encontrar una secuencia de operaciones en las máquinas, optimizando un indicador de rendimiento típico, como por ejemplo, el *Makespan*, es decir, el tiempo necesario para completar todos los *Jobs* (Pezzella et al., 2008).

Las restricciones presentes en un Problema *JSS* (*Job Shop Scheduling Problem, JSSP*) son las siguientes (Freitas, 2007):

- Todas las operaciones de un *Job* ocurren secuencialmente.
- No es permitida la interrupción de un *Job*.
- Ningun *Job* es procesado dos veces en la misma máquina.
- Cada *Job* és ejecutado hasta su finalización, aunque pueda haber esperas o atrasos durante la ejecución de las operaciones.
- Los *Jobs* pueden ser iniciados en cualquier momento, desde que no existan, en el problema, requisitos de fecha de liberación.
- Los *Jobs* deben esperar que la próxima máquina esté disponible para su ejecución.
- Ninguna máquina puede ejecutar más de una operación al mismo tiempo.
- Los tiempos de configuración son independientes de la secuencia.
- Existe siempre sólo una máquina de cada tipo.
- Las restricciones de precedencia (restricciones tecnológicas) son conocidas e inmutables.

2.3.3 Flexible Job Shop Scheduling (FJSS)

El *Flexible Job Shop Scheduling* (*FJSS*), es una generalización del *Job Shop Scheduling* (*JSS*). En el *FJSS* se permite que cada operación pueda ser procesada en cualquiera de un conjunto de máquinas disponibles (Demir & Isleyen, 2013). Esa flexibilidad de selección convierte en posible que un *Job* pueda ser procesado a través de varios caminos alternativos a través de las máquinas, lo que aumenta significativamente el número de soluciones factibles. Por esta razón, para la solución del *FJSS* son utilizados diferentes abordajes y métodos, especialmente heurísticos (De Melo, 2014).

El Problema del *FJSS* (*Flexible Job Shop Scheduling Problem, FJSSP*) puede ser clasificado de acuerdo al grado de flexibilidad permitida, así tenemos (Kacem, et al., 2002b):

- *Total FJSSP* (*TF-JSSP*): Cada operación puede ser procesada en cualquiera de las máquinas.
- *Partial FJSSP* (*PF-JSSP*): Al menos es considerada una operación que no puede procesarse en cualquiera de las máquinas.

De acuerdo con esta clasificación, el *PF-JSSP* es más difícil de solucionar que el primero.

Los abordajes de solución del *FJSSP* se distinguen por dividir o no el problema en subproblemas con la intención que sean menos complejos y puedan ser abordados de forma aislada. El *FJSS* puede ser visto como un problema que tiene dos fases. La Primera Fase consiste en asignar a cada una de las operaciones una máquina, entre un conjunto de máquinas disponibles, para su procesamiento. Esta Fase, es llamada de Enrutamiento, ya que las asignaciones especifican caminos que los *Jobs* recorren hacia las máquinas. Luego, de las asignaciones viene la Segunda Fase, que consiste en dar un orden o secuencia a las operaciones que van a ser procesadas en cada máquina. Dividir el problema, como se ha descrito, se denomina un Enfoque Jerárquico. En contraste, con este enfoque, se encuentra el Enfoque Integrado, método de solución que busca obtener soluciones para el *FJSS* tratando al mismo tiempo, asignaciones y secuencias. Al igual que en *JS*, el *FJSS* puede trabajar con distintos criterios de evaluación (De Melo 2014).

El *FJSSP* incrementa su complejidad al combinar varios criterios de optimización, tales como reducir al mínimo el Tiempo de Finalización de todos los *Jobs* (*Makespan*-C_M), la Carga de Trabajo Total de las Máquinas (*Total Workload* - W_T) y la Carga de Trabajo de la Máquina más Cargada (*Maximum Workload* - W_M) (Hsu et al., 2002).

Dada la complejidad del problema, se ha intensificado la necesidad de desarrollar nuevos enfoques en el ámbito de la optimización *FJSS* Multiobjetivo (Wojakowski & Warzolek, 2013).

El *JSSP* es un problema combinatorio NP-Hard, el *FJSSP* es una versión más compleja que el *JSSP*, por lo tanto, son fuertemente NP-Hard. Las dificultades computacionales aumentan excesivamente cuando la solución de estos problemas en encarado en un contexto de objetivos múltiples. Estos problemas representan un desafío para los investigadores en diferentes organizaciones internacionales de investigación científica (Genova et al., 2015).

La dificultad del *FJSSP* sugiere adoptar para su solución, incluso aun cuando se trate de *FJSSP* de un número reducido de *Jobs* y máquinas, métodos heurísticos que encuentren en tiempos razonables buenos resultados, en lugar de buscar los resultados exactos. Las Heurísticas no necesariamente dan soluciones óptimas, pero pueden ser eficaces para la mayoría de los sistemas de fabricación tipo *FJSS* complejos. En los últimos años, la aplicación de Metaheurísticas tales como el Recocido Simulado, Búsqueda Tabú y los Algoritmos Genéticos han dado buenos resultados (Pezella et al., 2008).

2.3.3.1 Formulación Matemática del FJSSP Multiobjetivo. El *Flexible Job Shop Scheduling Problem (FJSSP)* consiste en organizar la ejecución de **n** *Jobs* en **m** máquinas. El conjunto J = {J_1, J_2,..., J_n} de *Jobs* son independientes y Ji está formado por una secuencia O_{i1}, O_{i2},..., O_{ini} de operaciones que se procesan una tras otra de acuerdo con una secuencia especifica.

Dado un conjunto U = {M_1, M_2,..., M_m} de máquinas. Cada operación O_{ij} puede ser ejecutada en cualquiera de entre un subconjunto Uij ⊆ U de máquinas compatibles. El tiempo de procesamiento de cada operación depende de la máquina. Se denota con $d_{i,j,k}$ el tiempo de procesamiento de la operación O_{ij} cuando se ejecuta en la máquina M_k (Pezella et al., 2007).

En el caso de la Flexibilidad Total, $U_{i,j} = U$ para cada operación $O_{i,j}$. En el caso de Flexibilidad Parcial, se observa la existencia de al menos una operación $O_{i0,j0}$ tal que $U_{i0,j0} \subset U$ (Kacem et al., 2002).

A cada *FJSP*, se le asocia una tabla de tiempos de procesamiento como las mostradas en Tabla 1 y Tabla 2 respectivamente. En ellas se describen los *Jobs*, las operaciones de cada *Job* y los tiempos de procesamiento. La Tabla 1, corresponde a un *Total Flexible Job Shop Problem (TF-JSSP)*, en cada celda aparecen los tiempos de procesamiento de las operaciones que demoraría en cada máquina. La Tabla 2, corresponde a un *Partial Flexible Job Shop Problem (PF-JSSP)*, en donde las líneas, en algunas celdas, significan que la máquina no está disponible para ejecutar la operación correspondiente (Kacen et al, 2002a).

Al igual que Kacem (2002), Shao et al. (2013); Roshanaei (2012) y otros investigadores, en esta tesis se considera:

- El plan de proceso es determinado *a priori* para cada tipo de parte.
- Existe flexibilidad de ruta.
- Los *Jobs* son independientes y no es asignada ninguna prioridad.
- Para cada *Job* el orden de las operaciones es fija y no puede modificarse (restricción de precedencia).
- No se considera la suspensión o cancelación de *Jobs*.
- Una máquina puede procesar solo una operación a la vez.
- El tiempo de procesamiento de cada máquina es determinista e incluye el tiempo de configuración, transporte e inspección (aprobación).
- Todos los *Job*s se inspeccionan *a priori*, es decir, no se considera ninguna pieza defectuosa.
- Todas las máquinas están disponibles en el momento 0;
- Todos los *Job*s son liberados en el tiempo 0;
- Una operación no puede ser interrumpida una vez que se procesa.

Tabla 1
Tiempos de Procesamiento para un TF-JSSP

	$O_{i,j}$	M_1	M_2	M_3	M_4
J_1	$O_{1,1}$	1	3	4	1
	$O_{1,2}$	3	8	2	1
	$O_{1,3}$	3	5	4	7
J_2	$O_{2,1}$	4	1	1	4
	$O_{2,2}$	2	3	9	3
	$O_{2,3}$	9	1	2	2
J_4	$O_{4,1}$	8	6	3	5
	$O_{4,2}$	4	5	8	1

Fuente. Kacen et al. (2002)

Tabla 2
Tiempos de Procesamiento para un PF-JSSP

	$O_{i,j}$	M_1	M_2	M_3
J_1	$O_{1,1}$	6	-	4
	$O_{1,2}$	4	5	6
	$O_{1,3}$	5	-	6
J_2	$O_{2,1}$	3	2	1
	$O_{2,2}$	-	-	4
J_3	$O_{3,1}$	-	6	3
	$O_{3,2}$	9	8	-
	$O_{3,3}$	3	3	2
	$O_{3,4}$	2	2	-

Fuente. Shao, et al. 2013

El *FJSSP* sobre la base de estos supuestos está orientado a minimizar simultáneamente las siguientes funciones objetivos:

- *Makespan* (C_M), es equivalente al tiempo de finalización del último *Job* que deja el sistema. Un mínimo *Makespan,* generalmente, implica una buena utilización de las máquinas (Pinedo, 2016).

  $C_M = \max (C1, \ldots, Cj); \quad j = 1,\ldots, n$ (2.5)

  Donde:
  Cj es el tiempo que le toma salir del sistema al *Job* j (esto es, su tiempo de finalización en la última máquina en la que se requiere procesamiento)

- *Maximum Workload* (W_M), es el máximo tiempo de procesamiento utilizado en cualquier máquina. Este objetivo es importante para mantener el equilibrio de carga de trabajo entre diferentes máquinas y puede ayudar a evitar el exceso de programación de trabajo en una determinada máquina (Shao et al., 2013). La expresión para calcularlo es (Gen & Lin, 2012):

  $W_M = \max \{ W_j \}; \quad j = 1, \ldots, m$ (2.6)

  Donde:
  W_j es el tiempo total de proceso de la máquina Mj

- *Total Workload* (W_T), es el tiempo total de procesamiento de todas las máquinas. Este objetivo es significativo cuando alguna máquina no es eficiente para la realización de alguna operación (Shao et al., 2013). La expresión para calcularlo es (Gen & Lin, 2012):

  $W_T = \sum_1^m . \{Wj\}; \quad j = 1,\ldots, m$ (2.7)

En esta investigación se minimiza los tres objetivos anteriores. Es decir, se busca: min (C_M), min (W_M), min (W_T).

De acuerdo al ítem 2.3.1.2, los problemas de *Scheduling* pueden ser definidos mediante una tripleta, que a continuación se repite:

$$\boldsymbol{\alpha} \mid \boldsymbol{\beta} \mid \boldsymbol{\gamma}$$

Donde:

$\boldsymbol{\alpha}$: Describe al entorno de la máquina y tiene una sola entrada.
$\boldsymbol{\beta}$: Describe características del procesamiento y restricciones.
$\boldsymbol{\gamma}$: Describe el objetivo a minimizar.

Entonces, de acuerdo a esta tripleta y la nomenclatura descrita en el ítem 2.3.1.2, la presente investigación se denota de la siguiente manera:

Para el caso de Flexibilidad Total:

$$\boldsymbol{\alpha|\beta|\gamma} = \boldsymbol{FJc \mid rcrc \mid C_M, W_M, W_T} \quad (2.8)$$

Para el caso de Flexibilidad Parcial:

$$\boldsymbol{\alpha|\beta|\gamma} = \boldsymbol{FJc \mid M_j, rcrc \mid C_M, W_M, W_T} \quad (2.9)$$

2.3.4 Complejidad de los JSSP y FJSSP

Uno de los más difíciles problemas de *Scheduling* es el *Job Shop Scheduling Problem-JSSP*, que es conocido, por ser un problema NP-Hard. El *Flexible Job Shop Scheduling Problem* (*FJSSP*) es una generalización del *JSSP*, por consiguiente es un problema NP-Hard aún más complejo. En el *FJSSP* al reducirse las restricciones de máquinas se amplía el rango de búsqueda de soluciones factibles, aumentando las dificultades del problema. Es por eso,

que en el *FJSSP* no solo se debe asignar a cada operación una máquina entre muchas disponibles, sino además resolver el problema de secuencias como en el caso del *JSSP*. El *FJSSP* no tiene ningún algoritmo determinista que lo solucione en tiempo polinomial, sino más bien, los algoritmos deterministas solucionan el problema en un tiempo exponencial. Por consiguiente, se han propuesto soluciones Heurísticas como Reglas de Despacho, Búsqueda Local y Metaheurísticas. Entre estas últimas, se pueden citar a: Búsqueda Tabú, Enjambre de Partículas, Recocido Simulado y Algoritmos Genéticos. Además, para la solución del problema se puede recurrir a un Enfoque Jerárquico, en donde el problema se descompone en una secuencia de subproblemas de menor dificultad, no siendo así para el caso de un Enfoque Integrado (Sun W., et al., 2010).

La dificultad de encontrar la solución óptima del *FJSSP* puede entenderse a partir de la dificultad de solucionar el *JSSP,* reconocidamente uno de los problemas combinatorios más complejos. En el *JSSP*, en el que cada *Job* (de un total de **n** *Jobs*) pasa una vez por cada máquina, existen por cada máquina **n!** posibilidades de secuenciación. Ahora bien, si el problema tiene **m** máquinas, entonces el *JSSP* tiene un total de $(\mathbf{n!})^{\mathbf{m}}$ combinaciones posibles de soluciones. En un *FJSS* con Total Flexibilidad, si cada *Job* tiene **m** operaciones, entonces un solo *Job* tiene $\mathbf{m}^{\mathbf{m}}$ posibilidades para el enrutamiento en **m** máquinas. Teniendo en cuenta los **n** *Jobs*, las posibilidades de enrutamientos son: $\mathbf{m}^{\mathbf{m.n}}$. Donde cada posibilidad de enrutamiento representa un *JSSP* por resolver. Ahora, supongamos, que en cada *JSSP* por resolver, las **m** máquinas, en promedio, deben procesar, cada una, **n** operaciones, entonces, como se calculó antes se tiene $(\mathbf{n!})^{\mathbf{m}}$ combinaciones. Entonces por por todos los *JSSP* por resolver en el *FJSSP* con Total Flexibilidad se tiene: $(\mathbf{n!})^{\mathbf{m}}.\mathbf{m}^{\mathbf{m.n}}$ combinaciones posibles de solución. Por supuesto, estas cantidades del *JSSP* y del *FJSSP* engloban soluciones factibles y no factibles. Con esta explosión de combinaciones se puede señalar que, dependiendo del nivel de flexibilidad y del número de operaciones por *Job*, un *FJSSP* con 5 *Jobs* y 5 máquinas puede tener aproximadamente $7{,}41 \times 10^{27}$ combinaciones, mientras que un *JSSP* con el

mismo número de *Jobs* y de máquinas tiene un orden mucho menor de combinaciones, cerca de $2{,}48x10^{10}$ (De Melo, 2014).

2.3.5 *Ejemplos Demostrativos de la Complejidad del JSSP y FJSSP*

A continuación, se describirá dos ejemplos, uno sobre el *JSSP* y el otro sobre el *FJSSP.* Ambos ejemplos están fundamentados con la teoría precedente. Los dos ejemplos son simples con el objeto de ilustrar la complejidad combinacional cuando pretendemos encontrar un Programa de Operaciones (*Scheduling*) en los sistemas de fabricación tipo *Job Shop* y *Flexible Job Shop*.

2.3.5.1 Ejemplo A: JSSP*.* La Tabla 3 muestra un *JSSP* compuesto de tres *Jobs*, con tres operaciones por cada *Job* y tres máquinas. Las celdas de cada fila representan las operaciones de cada *Job*, dentro de las celdas se especifica a las máquinas que ejecutan la operación y el tiempo de procesamiento para su ejecución. Así por ejemplo, el *Job* 3, está constituido por 3 operaciones (representadas por las tres columnas siguientes al *Job*), en donde la operación 1 se procesa en la máquina 2, en 3 segundos, 2 (3). La operación 2 se procesa en la máquina 1 en 2 segundos, 1(2). La operación 3 se procesa en la máquina 3 en 1 segundo, 3(1).

Tabla 3
JSSP 3x3

JOB	Máquina (Tiempo)		
1	1 (3)	2 (3)	3 (3)
2	1 (2)	3 (3)	2 (4)
3	2 (3)	1 (2)	3 (1)

Fuente. Freitas (2007)

Los datos de la Tabla 3 se pueden representar ordenados de otra forma, como se muestra en la Tabla 4, donde $O_{i,j}$ representa la operación del *Job*

i y su operación j. Cada celda de la tabla es el tiempo de procesamiento. Observar, que las operaciones del *Job* 3 se lee, en esta tabla, como lo hemos hecho para el ejemplo anterior.

De la Tabla 4, de cada columna, se puede deducir las operaciones que tienen designadas las máquinas M1, M2 y M3.

- M1: $O_{1,1}$, $O_{2,1}$, $O_{3,2}$
- M2: $O_{1,2}$, $O_{2,3}$, $O_{3,1}$
- M3: $O_{1,3}$, $O_{2,2}$, $O_{3,3}$

Tabla 4
Problema Job Shop 3x3 - Modificada

JOB	$O_{i,j}$	M_1	M_2	M_3
J_1	$O_{1,1}$	3	-	-
	$O_{1,2}$	-	3	-
	$O_{1,3}$	-	-	3
J_2	$O_{2,1}$	2	-	-
	$O_{2,2}$	-	-	3
	$O_{2,3}$	-	4	-
J_3	$O_{3,1}$	-	3	-
	$O_{3,2}$	2	-	-
	$O_{3,3}$	-	-	1

Fuente. Elaboración propia

Existen varias posibilidades en que estas operaciones se pueden ordenar o secuenciar en cada máquina. Así por ejemplo, en la máquina M1 las tres (3) operaciones pueden ejecutarse en cualquiera de las seis (6) secuencias siguientes (3!):

- $O_{11} \rightarrow O_{21} \rightarrow O_{32}$ (Primero ejecutar $O_{1,1}$, luego $O_{2,1}$ por último $O_{3,2}$)
- $O_{11} \rightarrow O_{32} \rightarrow O_{21}$

- $O_{21} \rightarrow O_{32} \rightarrow O_{11}$
- $O_{21} \rightarrow O_{11} \rightarrow O_{32}$
- $O_{32} \rightarrow O_{21} \rightarrow O_{11}$
- $O_{32} \rightarrow O_{11} \rightarrow O_{21}$

Igualmente el número de secuencias posibles para las operaciones en M2 y M3 son: M2 = 3! = 6 y M2 = 3! = 6. Por lo tanto, el número de secuencias posibles de operaciones en las tres máquinas es:

- En las tres máquinas = 6x6x6 = $(3!)^3$ = 216 posibles secuencias

También hay que tener en cuenta que las operaciones del mismo *Job* (ejecutadas en cualquier máquina) tienen que tener un orden de precedencia (orden de ejecución) entre ellas (restricciones tecnológicas), las cuales se indican a continuación:

- J1 : $O_{1,1} \rightarrow O_{1,2} \rightarrow O_{1,3}$ (ejecutadas en las máquinas: M1, M2 y M3)
- J2 : $O_{2,1} \rightarrow O_{2,2} \rightarrow O_{2,3}$ (ejecutadas en las máquinas: M1, M3 y M2)
- J3: $O_{3,1} \rightarrow O_{3,2} \rightarrow O_{3,3}$ (ejecutadas en las máquinas: M2, M1 y M3)

Consecuentemente, de las doscientos dieciséis posibilidades se deben descartar aquellas que no cumplen con las restricciones tecnológicas.

Por cada una de las 216 posibilidades de secuencias en las tres máquinas se puede hacer uso de los Diagramas de Gantt. Estos diagramas grafican las operaciones en cada máquina registrando, en cada paso, el tiempo del proceso.

En las Figuras del 2 al 5 se muestran sólo cuatro (4) de los doscientos dieciséis (216) Diagramas de Gantt posibles para este problema. Las operaciones se muestran como segmentos de rectas cuya longitud coincide con su tiempo de ejecución en el eje de las abscisas. Los números que aparecen entre paréntesis indican el número de *Job* y su

correspondiente operación. Así, por ejemplo el par ordenado (2,3) corresponde al *Job* 2 operación 3. En todas las figuras se cumple con la restricción de precedencia entre operaciones.

Las figuras se caracterizan por tener diferentes valores de *Makespan* (C_M, *Makespan*). Un criterio de seleccionar a la mejor o más óptima es la de menor C_M. De las posibilidades mostradas, el Diagrama de Gantt de la Figura 5 es la de menor C_M, con un valor igual a 12, pero no necesariamente es la mejor, de un universo de 216 Diagramas de Gantt posibles para este problema.

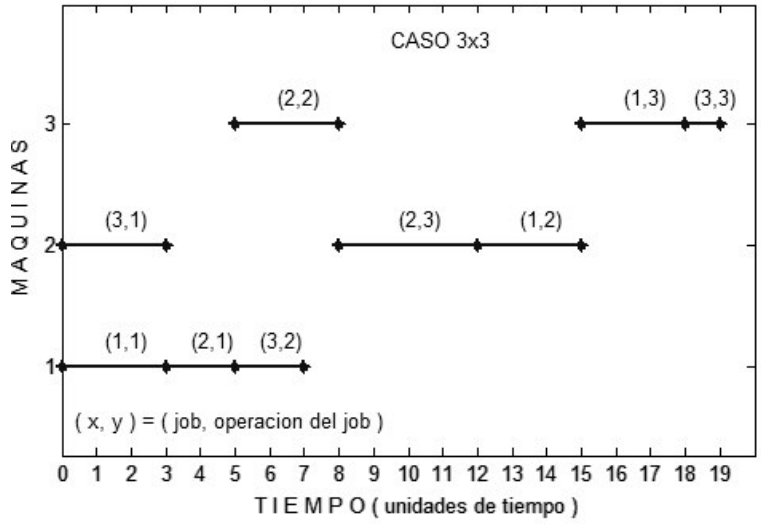


Figura 2. **Solución del *JSSP* con *Makespan* 19**. *Fuente*: Elaboración propia

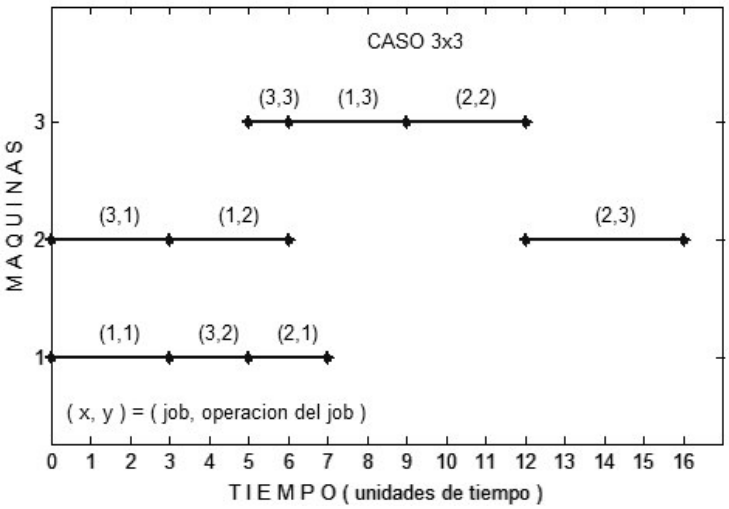


Figura 3. **Solución del *JSSP* con *Makespan* 16.** *Fuente*: Elaboración propia

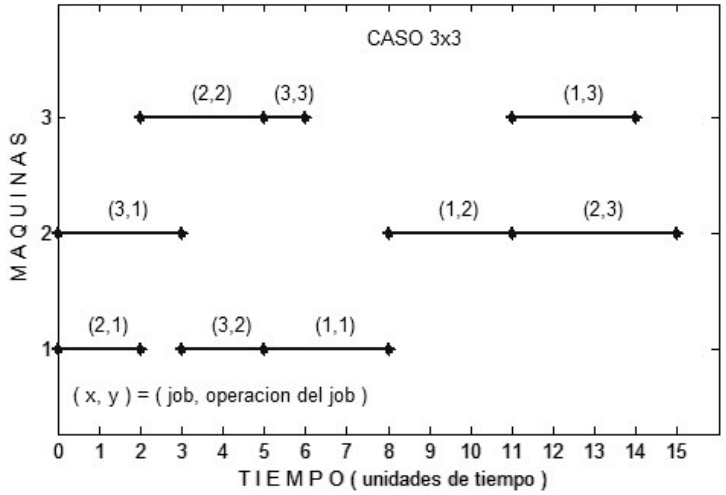


Figura 4. **Solución del *JSSP* con *Makespan* 15**. *Fuente*. Elaboración propia

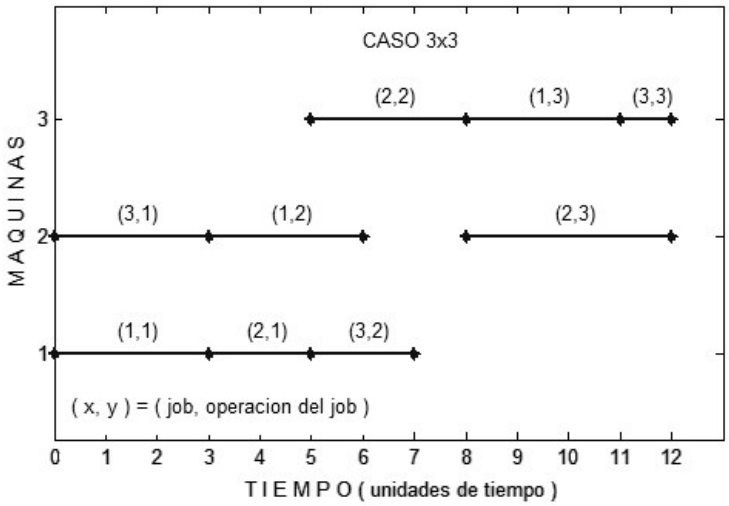


Figura 5. **Solución del *JSSP* con *Makespan* 12**. *Fuente*. Elaboración propia

2.3.5.2 Ejemplo B: FJSSP. La Tabla 5 muestra un *FJSSP* de tres Jobs, con tres operaciones por cada *Job* y tres máquinas, de flexibilidad total, donde cada valor de la celda corresponde al tiempo de procesamiento de las operaciones en cada máquina.

Tabla 5
FJSSP 3x3

JOB	$O_{i,j}$	M_1	M_2	M_3
J_1	$O_{1,1}$	3	1	2
	$O_{1,2}$	2	3	5
	$O_{1,3}$	4	1	3
J_2	$O_{2,1}$	2	3	5
	$O_{2,2}$	1	4	3
	$O_{2,3}$	5	4	7
J_3	$O_{3,1}$	5	3	2
	$O_{3,2}$	2	5	3
	$O_{3,3}$	3	5	1

Fuente. Elaboración propia

En base a la tabla anterior, el problema planteado puede dividirse en dos subproblemas que se pueden denominar como: el Subproblema de Enrutamiento y el Subproblema de Secuenciación, este último visto para el caso del *Job Shop*.

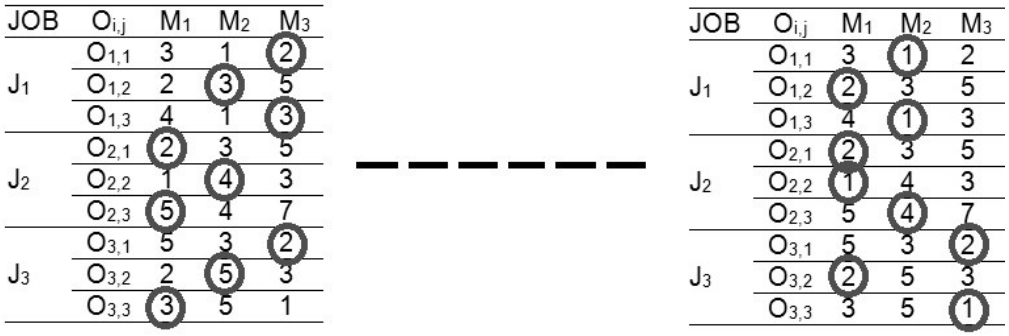
El Subproblema de Enrutamiento consiste en designar a cada operación ($O_{i,j}$) una de las tres máquinas (M1, M2 y M3) para su ejecución. Por lo que, el número de rutas posibles para cada operación es:

- Por cada operación ($O_{i,j}$) = 3 rutas posibles

Entonces, para las nueve (9) operaciones de todo el problema se tiene:

- Para todas las operaciones = $(3)^9$ = 19683 rutas posibles

En la Figura 6, se muestra sólo dos (2) tablas de enrutamiento de las 19683 posibles tablas que se pueden generar. Un círculo indica las máquinas que se han designado para cada operación.

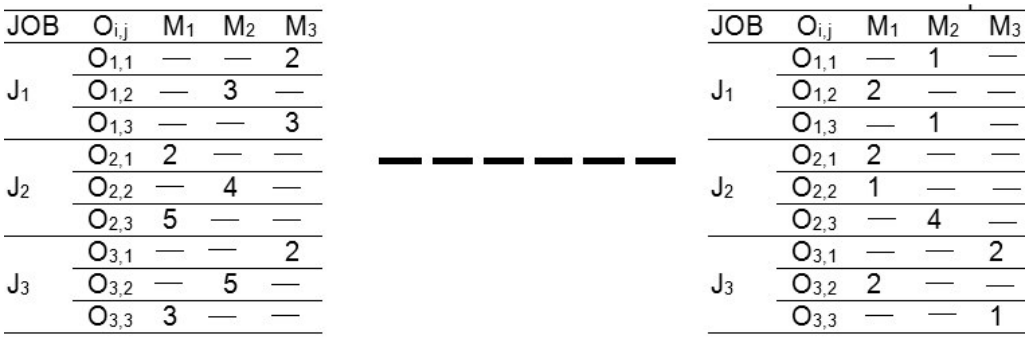


JOB	$O_{i,j}$	M_1	M_2	M_3
J_1	$O_{1,1}$	3	1	**2**
	$O_{1,2}$	2	**3**	5
	$O_{1,3}$	4	1	**3**
J_2	$O_{2,1}$	**2**	3	5
	$O_{2,2}$	1	**4**	3
	$O_{2,3}$	**5**	4	7
J_3	$O_{3,1}$	5	3	**2**
	$O_{3,2}$	2	**5**	3
	$O_{3,3}$	**3**	5	1

JOB	$O_{i,j}$	M_1	M_2	M_3
J_1	$O_{1,1}$	3	**1**	2
	$O_{1,2}$	**2**	3	5
	$O_{1,3}$	4	**1**	3
J_2	$O_{2,1}$	**2**	3	5
	$O_{2,2}$	**1**	4	3
	$O_{2,3}$	5	**4**	7
J_3	$O_{3,1}$	5	3	**2**
	$O_{3,2}$	**2**	5	3
	$O_{3,3}$	3	5	**1**

Figura 6. **Dos de las Soluciones al Subproblema de Enrutamiento**. *Fuente*. Elaboración propia

Para una mejor visualización, la Figura 7, muestra solo a las máquinas seleccionadas en cada operación. Estas tablas ponen en evidencia que el problema ahora se reduce a solucionar varios problemas equivalentes al de *Jobs Shop*s (véase la similitud que existen entre las tablas de la Figura 7 con la Tabla 4).



JOB	$O_{i,j}$	M_1	M_2	M_3
J_1	$O_{1,1}$	—	—	2
	$O_{1,2}$	—	3	—
	$O_{1,3}$	—	—	3
J_2	$O_{2,1}$	2	—	—
	$O_{2,2}$	—	4	—
	$O_{2,3}$	5	—	—
J_3	$O_{3,1}$	—	—	2
	$O_{3,2}$	—	5	—
	$O_{3,3}$	3	—	—

(a)

JOB	$O_{i,j}$	M_1	M_2	M_3
J_1	$O_{1,1}$	—	1	—
	$O_{1,2}$	2	—	—
	$O_{1,3}$	—	1	—
J_2	$O_{2,1}$	2	—	—
	$O_{2,2}$	1	—	—
	$O_{2,3}$	—	4	—
J_3	$O_{3,1}$	—	—	2
	$O_{3,2}$	2	—	—
	$O_{3,3}$	—	—	1

(b)

Figura 7. **Las Soluciones se reducen a *Job Shops***. *Fuente*: Elaboración propia

Un criterio para encontrar el enrutamiento óptimo consiste en minimizar el *Total Workload* y *Maximum Workload* de las máquinas. Estos parámetros se miden de la siguiente forma:

En la Figura 7a, la Carga de Trabajo (*Workload*) de cada máquina se encuentra sumando los tiempos de ejecución de todos las operaciones destinadas a ellas. Así se tiene:

- M1: $O_{2,1} + O_{2,3} + O_{3,3} = 2+5+3 = 10$
- M2: $O_{1,2} + O_{2,2} + O_{3,2} = 3+4+5 = 12$
- M3: $O_{1,1} + O_{1,3} + O_{3,1} = 2+3+2 = 7$

Por tanto, la Máxima Carga de Trabajo (W_M) entre todas las máquinas y Carga Total de Trabajo o suma de carga de trabajo de todas las máquinas (W_T), son respectivamente:

W_M = M2 = 12
$W_T = \sum_1^3 \mathrm{Mi}$ = M1+M2+M3 = 10+12+7 = 29

En el segundo caso, Figura 7b, se tiene:

- M1: $O_{1,2} + O_{2,1} + O_{2,2} + O_{3,2} = 2+2+1+2 = 7$
- M2: $O_{1,1} + O_{1,3} + O_{2,3} = 1+1+4 = 6$
- M3: $O_{3,1} + O_{3,3} = 2+1 = 3$

Por tanto:

W_M = M1 = 7
$W_T = \sum_1^3 \mathrm{Mi}$ = M1+M2+M3 = 7+ 6 + 3 = 16

Por tanto, de estas dos 2 posibilidades la segunda es mejor que la primera (W_M y W_T son menores). Pero no necesariamente es la mejor de las 19683 tablas posibles para este problema.

Luego de seleccionar las máquinas más eficientes para cada operación, como se ha descrito, termina el Subproblema de Enrutamiento y se pasa al Subproblema de Secuenciación. En este Subproblema, se encuestran tablas equivalente al de la Figura 7, que como se ha indicado corresponde a una de *Job Shop*. Por tanto, el Subproblema de Secuenciación ha sido desarrollado en el ítem anterior.

La complejidad total de un problema *FJSS* se puede cuantificar de la siguiente manera: Para este problema, existen 19683 enrutamientos o rutas posibles. Eso significaría solucionar 19683 problemas de *Job Shop* o problemas de secuenciación. Supongamos que en cada uno de los problemas de *Job Shop* las máquinas M1, M2 y M3 tienen a su cargo, en promedio, tres operaciones. Entonces, por cada *Job Shop* el número de secuencias posibles es $(3!)^3$ o 216. Por lo que, el número de secuencias posibles para todo el problema es de 19,683x216 o 4'251,528. Evidentemente, para sistemas de fabricación con mayor número de *Jobs* y máquinas, el número de posibles resultados serían extremadamente grandes, tal como fue analizado en el ítem 2.3.4.

Finalmente, la complejidad del problema viabiliza una solución Metaheurística, método desarrollado en esta tesis, que tiene por objeto encontrar el enrutamiento óptimo o aproximado al óptimo, entre todas las posibles rutas; minimizando W_M y W_T. Para luego, de realizado lo anterior, se pase a encontrar la secuenciación óptima o aproximadamente óptima, entre todas las secuencias posibles; minimizando el C_M. Para ello, en ambos subproblemas se han diseñado e implementado Algoritmos Genéticos.

2.3.6 Algoritmos Genéticos

En el libro *Adaptation in Natural and Artificial Systems* de Holland, 1975 se presenta el GA (Genetic Algorithm - GA) como una abstracción de la evolución biológica, dando un marco teórico para la adaptación del GA. El algoritmo de Holland, es un método para mover una población de "cromosomas" (cadenas de "bits" que representan candidatas de la solución a un problema) a una nueva población, utilizando la "selección", junto con operadores genéticos de cruce, mutación e inversión. Cada cromosoma se compone de "genes" (por ejemplo, bits), con cada gen siendo una instancia de un particular "allele" (por ejemplo, 0 o 1). El operador de selección elige a los cromosomas más aptos a los cuales se les permitirá reproducir. Los cromosomas más aptos producen en promedio más hijos que los menos aptos. El operador de cruzamiento intercambia subpartes de dos cromosomas, más o menos imitando la recombinación de dos organismos biológicos (“haploides”); el operador mutación cambia aleatoriamente los valores “allele” de algunas posiciones en el cromosoma; y el operador inversión invierte el orden de una sección contigua del cromosoma. Así se reordena la disposición de los genes. El cuarto elemento de GAs de Inversión de Holland es rara vez utilizado en implementaciones actuales y sus ventajas, si las hay, no están bien establecidas (Mitchell, 1999).

Un GA normalmente se itera de 50 a 500 o más generaciones, la que se llama una "corrida". Al final de una corrida, a menudo hay uno o más cromosomas altamente ajustados a la población. Dado que el azar juega un papel importante en cada corrida, dos corridas con diferentes poblaciones aleatorias producirán generalmente diferente comportamientos. Los investigadores en GA informan a menudo estadísticas tales como el mejor ajuste encontrado y la generación en la que se encontró, promediando para muchas corridas del GA los resultados del mismo problema (Mitchell, 1999).

2.3.6.1 Definición. Un Algoritmo Genético (GA) es un algoritmo de búsqueda local que sigue el paradigma de la evolución. A partir de una

población inicial, el algoritmo aplica a los operadores genéticos para producir descendientes (en la terminología de búsqueda local, corresponde a explorar el vecindario), que son presumiblemente más aptos que sus antepasados. En cada generación (iteración), cada nuevo individuo (cromosoma) corresponde a una solución, por ejemplo, un cromosoma es un *Schedule* en un *FJSP*. La fortaleza de GA se debe a que se pueden adoptar más estrategias que otros algoritmos de búsqueda local para encontrar nuevos individuos desde la fase inicial de la población y en la fase de la generación. Por lo que puede ser explorado un espacio más variable de búsqueda en cada etapa del algoritmo (Pezzella et al., 2008).

2.3.6.2 Codificación de los Cromosomas. Cualquier solución potencial a un problema puede representarse dando valores a una serie de parámetros (*genes* en la terminología de algoritmos genéticos) que se codifica en una cadena de valores denominada cromosoma. El conjunto de los parámetros representado por un cromosoma particular recibe el nombre de *genotipo*, que contiene la información necesaria para la construcción del organismo, es decir, la solución real al problema, denominada *fenotipo*. Por ejemplo, en términos biológicos, la información genética contenida en el ADN de un individuo sería el genotipo, mientras que la expresión de ese ADN (el propio individuo) sería el fenotipo (Aguiar, 2014).

Las técnicas de codificación en Algoritmos Genéticos son específicas de la naturaleza de cada problema, las cuales transforman las soluciones del problema en cromosomas. Varias técnicas de codificación se utilizan en algoritmos genéticos, así se tiene por ejemplo la codificación: binaria, permutación, valores y de árboles (Malhotra et al., 2011).

Para los problemas combinatorios (Vendedor Viajante o el propio problema de *Scheduling*) una representación clásica binária de algoritmos genéticos se ha considerado poco apropiado (Beck, 2000).

La codificación de permutación es la más adecuada para ordenar o para los problemas de colas. El vendedor viajante es un problema desafiante de optimización, donde se utiliza la codificación de permutación. En la codificación de permutación, cada cromosoma es una cadena de números en una secuencia no repetida (Malhotra et al., 2011). En el problema como del Vendedor Viajante, un cromosoma representa una ruta, y un gen puede representar una ciudad (Sastry et al., 2005)

Dado la importancia de la codificación, como ejemplos, se examinan brevemente la manera como distintos investigadores han codificado el cromosoma en problemas relacionados a *Job Shop*s. En Xiong et al. (2012), dado que el problema de *FJSS* tiene dos tareas independientes: la asignación de máquinas y secuenciación de las operaciones, la representación del cromosoma contiene la información de estas dos partes. Es por eso que representan al cromosoma en un enfoque de dos vectores, denominado vector de asignación de máquina y vector de secuencia de operación. El cromosoma que representa la secuencia de operaciones la representa en una forma de permutación. La Figura 8 muestra la representación de cromosomas de dos vectores de un *FJSS* con 4 *Job*s, 12 operaciones y 4 máquinas. La longitud de cada cromosoma es igual al número total de operaciones, denominado O. Para cada posición k en el cromosoma, el vector de asignación de máquina v_1 (k) representa la máquina seleccionada para la operación en la posición k, el vector de secuencia de operación v_2(k) indica el índice ID de la operación.

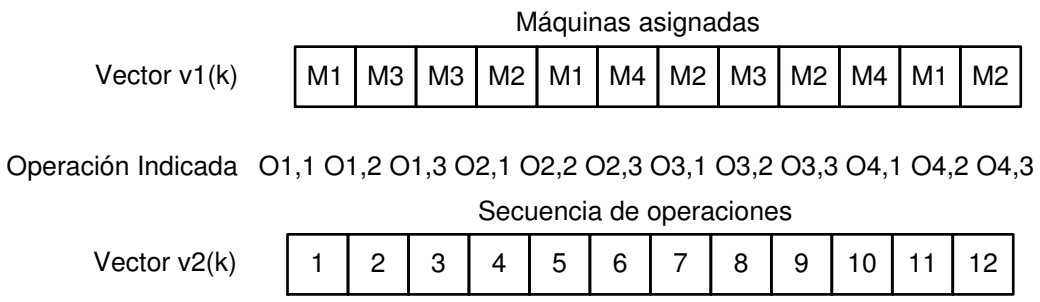


Figura 8. **Ejemplo 1 - Representación de Cromosoma**. *Fuente*. Xion et al.(2012).

En Shahsavari et al (2013), se representa el problema de *Flexible Job Shop*, en dos partes. La parte A, representa las máquinas que procesan las operaciones y la parte B, ilustra una secuencia de las operaciones. El número de genes en cada parte es igual a la suma de las operaciones en todos los *Job*s. Para la parte A, cada valor del gen muestra el número de la máquina y en la parte B muestra el número del *Job*. Debe considerarse que el número de cada *Job* que se repite en la parte B debe ser igual al número de las operaciones de ese *Job*. La Figura 9 ilustra el cromosoma para un caso de tener 4 *Jobs* con 5 máquinas.

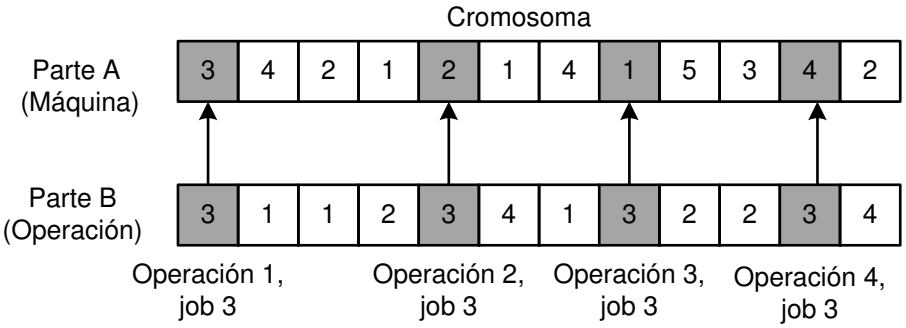


Figura 9. **Ejemplo 2 - Representación Cromosoma.** *Fuente*. Shahsavari et al. (2013)

En Ranjini & Zoraida (2013), representan al cromosoma de un *Job Shop* constituido por 3 *Jobs* con 5 operaciones por cada *Job* como en la Figura 10. Cada gen representa una operación. Todas las operaciones del mismo *Job* están representadas por el mismo número y se interpretan de acuerdo con el orden de su aparición en la secuencia cromosómica.

Como cada *Job* consta de cinco operaciones, el número de cada *Job* se repite exactamente cinco veces en el cromosoma. Así por ejemplo, el cuarto gen representa la primera operación del *Job* 3 porque el número aparece la primera vez. Del mismo modo, el sexto gen en el cromosoma implica la tercera operación del *Job* 2.

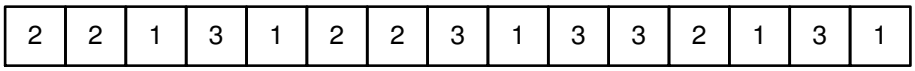


Figura 10. **Ejemplo 3 - Representación de Cromosoma**. *Fuente*: Ranjini & Zoraida (2013)

2.3.6.3 Selección. Los métodos para seleccionar los individuos que se van a reproducir dan preferencia aquellos con mayores valores de aptitud, a continuación se presentan los principales métodos.

Selección por Ruleta. Cada individuo de la población es representado en una ruleta proporcionalmente a su índice de aptitud. Así a los individuos con más alta aptitud se les da porciones mayores en la ruleta. Para seleccionar un individuo se genera un número aleatorio en el intervalo [0...1] seleccionando el individuo situado en esa posición de la ruleta (Gestal et al., 2010).

Selección por Torneo. Existen dos versiones, el torneo determinístico y el torneo probabilístico. En la versión determinística se selecciona al azar un número p de individuos (generalmente se escoge p=2) y entre los dos se selecciona el más apto. En la versión probabilística en vez de escoger siempre el más apto, se genera un número aleatorio del intervalo [0...1], si es mayor que un parámetro fijado se escoge el individuo más apto y en caso contrario el menos apto (Gestal et al., 2010).

Ranking Lineal: Los individuos son clasificados de acuerdo a su valor de ajuste y un rango $r_i \in \{1,..., N\}$ se asigna a cada uno, donde N es el tamaño de la población. El mejor individuo obtiene el rango N mientras que el peor obtiene el rango 1. Luego: $Pi = (2r_i)/N(N+1)$. Es la probabilidad de elegir el i-enésimo individuo en el Ranking Lineal (Pezzela et al., 2007).

2.3.6.4 Cruce o Recombinación. Una vez seleccionados los individuos, éstos son recombinados para producir la descendencia que se insertará en la siguiente generación. Si se opta por una estrategia destructiva los

descendientes se insertarán en la población temporal aunque sus padres tengan mejor ajuste. Por el contrario, utilizando una estrategia no destructiva la descendencia pasará a la siguiente generación únicamente si supera la bondad del ajuste de los padres, o de los individuos a reemplazar (Gestal et al., 2010).

***Cruce de 1 punto*.** Se seleccionan dos individuos y se cortan sus cromosomas por un punto aleatorio para generar dos segmentos diferenciados en cada uno de ellos: la cabeza y la cola. Se intercambian las colas entre los dos individuos para generar los nuevos descendientes. En la Figura 11, se puede ver con claridad el proceso descrito anteriormente. En la bibliografía a este tipo de cruce se le conoce con el nombre de *SPX* (*Single Point Exchange*).

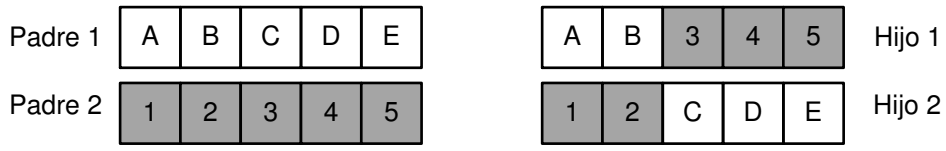


Figura 11. **Cruce de un Punto.** *Fuente*. Gestal, et al.(2010)

***Cruce de 2 puntos*.** En vez de cortar por un único punto los cromosomas de los padres, como en el caso anterior, se realizan dos cortes. Deberá tenerse en cuenta que ninguno de estos puntos de corte coincida con el extremo de los cromosomas. Para generar la descendencia se escoge el segmento central de uno de los padres y los segmentos laterales del otro padre. Generalmente, es habitual referirse a este tipo de cruce con las siglas *DPX* (*Double Point Crossover*). En la Figura 12, se muestra un ejemplo de cruce por dos puntos.

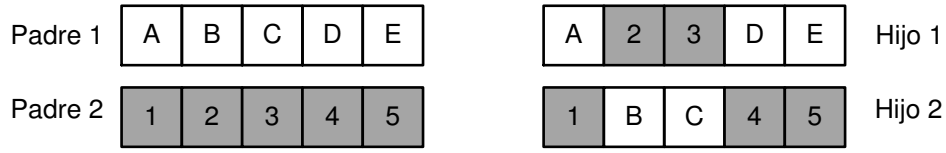


Figura 12. **Cruce de dos Puntos**. *Fuente*. Gestal et al. (2010)

Generalizando, se pueden añadir más puntos de cruce dando lugar a algoritmos de cruce multipunto. Sin embargo existen estudios que desaprueban esta técnica (Gestal et al., 2010).

***Cruce en cromosomas de secuencias permutadas*.** En problemas que consisten en ordenar una lista o poner las cosas en el orden correcto, como en el problema del agente viajero, los operadores de cruce y mutación estándar no son adecuados. Por ejemplo, considere dos cromosomas de padres de longitud seis (6). Un cruce simple de los elementos que están más adelante del segundo elemento de los cromosomas produce dos hijos, ver Figura 13. Obviamente, esto no resulta adecuado, ya que en el $hijo_1$ se repite el número 3 y está ausente el número 1, mientras que en el $hijo_2$ el 1 está repetido y el 3 está ausente (Haupt & Haupt, 2004).

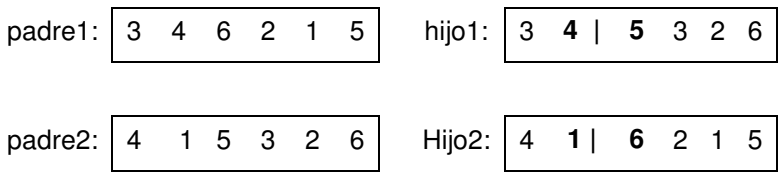


Figura 13. **Cruce Simple en Secuencias Permutadas**. *Fuente*. Haupt & Haupt (2004)

Goldberg (1989), citado por Haupt & Haupt (2004), discute varias soluciones posibles para este problema, que se resumen brevemente. El primero es conocido como *PMX* (*Partially Matched Crossover*), en este método se eligen dos puntos de cruce, se intercambian los valores de los padres entre estos puntos. Por ejemplo, en la Figura 14, si los puntos de cruce están entre los elementos 1, 2, y 3, 4 de los padres, resultan los $hijo_{1A}$ y $hijo_{2A}$ (Etapa A).

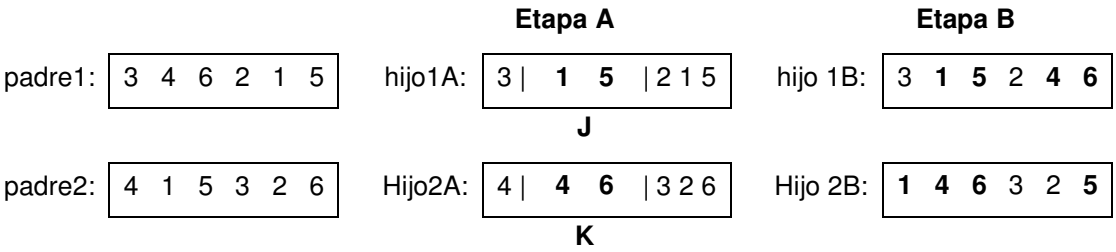


Figura 14. **Cruce *PMX***. *Fuente*. Haupt & Haupt (2004)

Hasta ahora todavía tenemos el problema de tener números duplicados y otros ausentes. Las cadenas conmutadas, J y K, permanecen intactas durante todo el resto del procedimiento. Los duplicados del $hijo_{2A}$ son intercambiados con los duplicados del $hijo_{1A}$. Es decir, el 4 del $hijo_{2A}$ se intercambia con el 1 del $hijo_{1A}$, y el 6 del $hijo_{2A}$ se intercambia con el 5 del $hijo_{1A}$ para obtener la solución fina (Etapa B). Cada hijo contiene parte del padre inicial en la misma posición (números sin enfatizar) e incluye cada entero una vez y solo una vez.

El segundo, se le conoce como *OX* (*Order Crossover*), en donde se trata de mantener el orden de los números enteros como si el vector cromosoma estuviera dispuesto en un círculo. Comienza, como *PMX*, eligiendo dos puntos de cruce e intercambiando los números. Esta vez, se dejan agujeros (marcados con X) en los espacios donde se repiten los números. Así por ejemplo, si los puntos de cruce de los padres, están después del segundo y cuarto enteros, la Primera Etapa se muestra en la Figura 15. En este punto los huecos son empujados hacia el inicio de las posiciones de los hijos desplazando todos los números circularmente al otro extremo del cromosoma. Al mismo tiempo, las cadenas de J y K que fueron intercambiados mantienen sus posiciones (Segunda Etapa). Finalmente, las X's son remplazados con las cadenas J and K (Etapa Final).

Figura 15. **Cruce *OX***. *Fuente*: Haupt & Haupt (2004)

El último método discutido, ver Figura 16, es el *Cycle Crossover* (*CX*). En este método, el intercambio de información comienza en la izquierda de la cadena intercambiando los dos primeros dígitos de los padres (Primera Etapa). Ahora, el $hijo_{1W}$ tiene repetido el número 4, avanzamos al otro 4 y lo intercambiamos con el número del $hijo_{2W}$ en esa posición (Segunda Etapa). Al repetirse en el $hijo_{1X}$ el número 1, intercambiamos el otro 1, con el número de esa posición del $hijo_{2X}$ (Tercera Etapa). De la misma manera, el número 2 de la posición 4 del $hijo_{1Y}$, lo intercambiamos con el número del $hijo_{2Z}$ de esa posición (Cuarta Etapa).

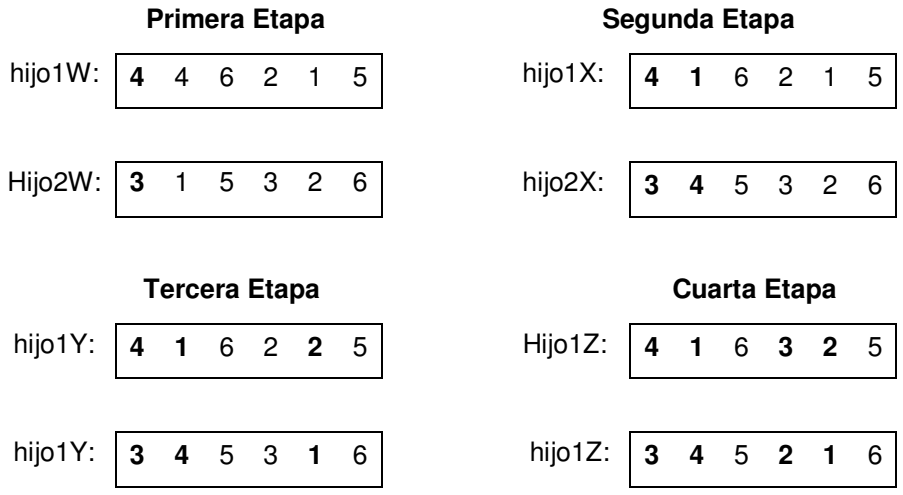


Figura 16. **Cruce *CX.*** *Fuente*. Haupt & Haupt (2004)

En este punto hemos intercambiado el 2 con el 3 y no hay números enteros repetidos en ninguno de hijos, por lo que la transición está completa (Haupt & Haupt, 2004).

2.3.6.5 Mutación. Mientras que el cruzamiento tiende a que la población converja (preferiblemente a valores óptimos), la mutación tiene como objetivo mantener un cierto nivel de diversidad en la población, es decir, evitar que la población converja rápidamente. Como el cruzamiento generalmente mezcla características de ambos padres, una cantidad de información se puede perder. La mutación puede, en este sentido, restablecer o recuperar material genético perdido (o todavía no explotada), evitando así la población converja a un valor inferior al óptimo. Incluso en algunos casos, la mutación por sí solo puede proporcionar una búsqueda mejor que los operadores de cruzamiento. No existen muchas reglas para los operadores de mutación (Srinivas & Patnaik, 1994) y (Schaffer & Eshelman, 1991) citados por (Beck, 2000).

La mutación de un cromosoma más usual es aquel en que uno de sus genes varia aleatoriamente (Gestal et al., 2010). Pero, en el caso de secuencias permutadas, al variar aleatoriamente uno de sus genes pueden ocurrir genes duplicados y otros perdidos. La solución más sencilla, en este caso, consiste en elegir al azar un cromosoma para mutar, y luego elegimos al azar dos posiciones dentro de ese cromosoma para intercambiarlos. Como un ejemplo, si en un cromosoma las posiciones segunda y quinta son elegidos al azar, el cromosoma mutado se transforma como en la Figura 17 (Haupt & Haupt, 2004).

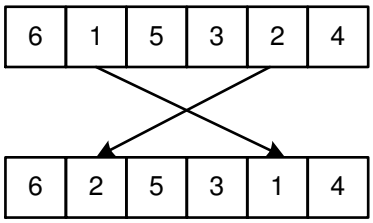


Figura 17. **Mutación.** *Fuente*: Haupt & Haupt (2004)

2.3.6.6 Evaluación. Después de elegir la forma de representar las soluciones, debe definirse cómo se evaluará a cada individuo de la población. Es decir, es necesario definir a los mejores cromosomas que puedan ser utilizados para reproducciones futuras. Está bastante clara la importancia de la evaluación, después de todo, junto con la decodificación de los cromosomas para las soluciones, la evaluación de cada una de ellos es el principal vínculo entre el espacio utilizado en la búsqueda (el espacio de representación) y el espacio de soluciones. A menudo, los términos de la función objetivo y la función de aptitud (*fitness function*) se utilizan con el mismo significado. El valor de la función objetivo a menudo se utiliza directamente como el valor de función de aptitud (Beasley, 1993) citado por (Beck, 2000).

2.3.6.7 *Criterio de Parada.* Hay tres categorías de criterios de parada que son utilizadas con frecuencia por los investigadores (Yu & Gen, 2010):

1. Parar con el valor *fitness*. Si se conoce el valor *fitness* de la solución óptima, éste puede ser el umbral para que el algoritmo se detenga. Otra forma es la de verificar la varianza de los valores *fitness* la población, si la varianza coincide con la escogida, situación de convergencia, el algoritmo se detiene. Otra manera, es calcular el error relativo con el *fitness* máximo y mínimo de la población, cuanto más cerca está la relación a 1 (o cuanto menor es el error relativo) más convergente es la población y el algoritmo se detiene.

2. Parar con el cambio de *fitness*. Se puede registrar los mejores valores *fitness* obtenidos en cada generación. Si el cambio está por debajo de umbral predefinido el algoritmo se detiene. También se podría utilizar los cambios de valor medio o la tasa de cambios.

3. Parar con tiempo. Se podría parar si se alcanza un número de generaciones predefinido o un número de mejores *fitness* que alcanzan un umbral predefinido. En una implementación real, se podrían combinar varios criterios para lograr una regla flexible y efectiva de finalización.

2.4 Optimización Mutiobjetivo

La Optimización Multiobjetivo tiene por objeto encontrar un vector de variables de decisión que satisfaga un conjunto de restricciones y optimice una función vectorial cuyos elementos representan a las funciones objetivo. Estas funciones describen matemáticamente criterios de desempeño que están en conflicto entre sí. En este sentido, el término optimizar significa encontrar aquella solución que proporcione valores para todas las funciones objetivo aceptables para el diseñador. Matemáticamente se formula como sigue (López, 2005):

$$\text{Optimizar: } F(\mathbf{X}) = (F1(\mathbf{X}), F_2(\mathbf{X}), \ldots, F_k(\mathbf{X}))^T \quad (2.6)$$

$$\text{Sujeto a: } g_i(\mathbf{X}) \leq 0 \quad i = 1,\ldots, m \quad (2.7)$$

$$Hi(\mathbf{X}) = 0 \quad i = 1, p \quad (2.8)$$

Donde:

X, es un vector $\mathbf{X} = (x_1, x_2,\ldots, xn)^T$ conformado por variables de decisión xi que buscan resolver el problema, denota a las soluciones óptimas (usualmente más de una).

$g_i(\mathbf{X})$ y $h_i(\mathbf{X})$, son las restricciones del mundo real dadas por las condiciones del ambiente o los recursos disponibles. Son necesarias que sean cumplidas para validar la solución, se representan por ecuaciones e inecuaciones. Las restricciones definen una región factible Ω dentro del espacio de las variables de decisión; cualquier solución **X** dentro de esta región es considerada una región factible.

F(**X**), es la función vectorial objetivo conformada por las funciones objetivos del problema. Proyecta el conjunto Ω al conjunto λ el cual representa el conjunto de todos los valores posibles de la función objetivo.

Intervienen dos espacios euclidianos, el espacio n-dimensional de las variables de decisión - llamado también espacio de decisión, o espacio genotípico - en el cual cada eje coordenado corresponde con cada componente del vector **X**. El espacio k-dimensional de las funciones objetivo (llamado también espacio de criterios, o espacio fenotípico), en donde cada eje coordenado se corresponde con cada componente del vector F(**X**).

En la Figura 18, se muestra, a manera de ejemplo, una función vectorial F(**X**) para el caso de 2 variables de decisión y 3 funciones objetivos.

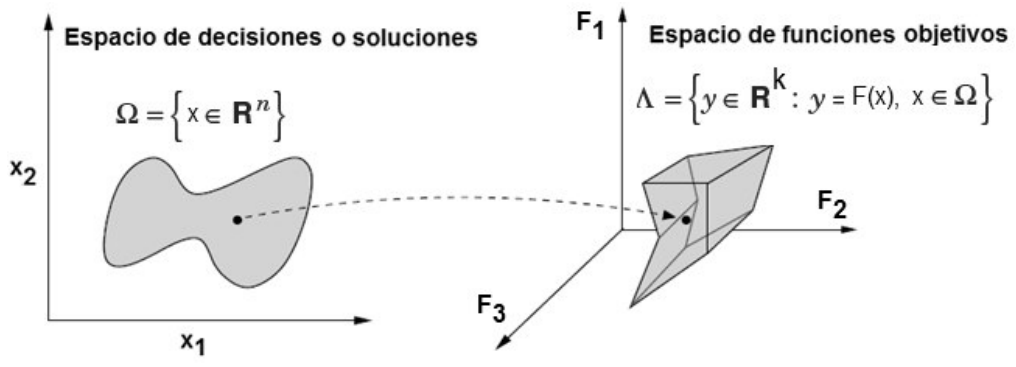


Figura 18. **Función de Proyección**. *Fuente*. López (2005)

Para una Optimización Multiobjetivo, es necesario introducir el concepto de Dominación, No Dominación, Solución Pareto y Conjunto de Pareto. Consideremos que en R^n (Espacio de Solución) y en R^k (Espacio del Objetivo), los puntos de soluciones en el Espacio de Solución y sus correspondientes puntos objetivos son los que se muestran en la Figura 19 (Shao, 2013).

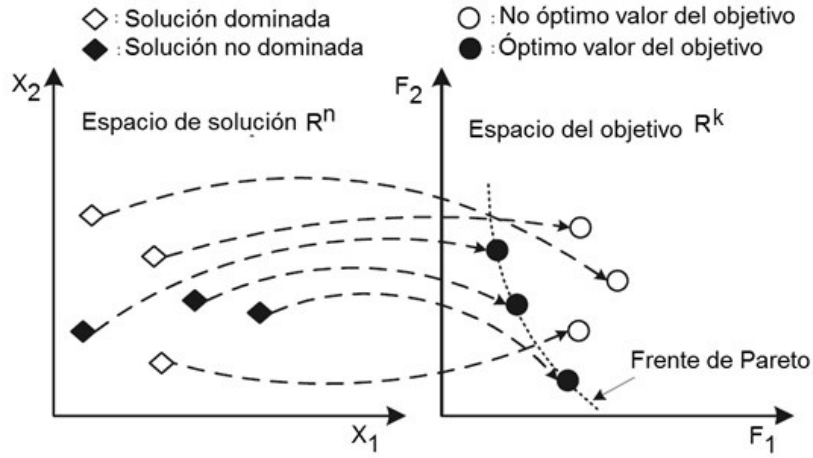


Figura 19. **Puntos de Solución en R^n y sus Puntos Objetivos en R^k**. *Fuente*. Shao (2013).

Dominación, en un problema de Optimización Multiobjetivo una solución **X** domina otra solución **Y**, si para:

F (**X**) = (f1 (x), f2 (x), f3 (x),..., fn (x)), y (2.9)

F (**Y**) = (f1 (y), f2 (y), f3 (y),..., fn (y)). (2.10)

Se cumplen las dos condiciones siguientes:

- Para cualquier índice i, el valor fi (x) es menor o igual al valor fi (y).
- La solución **X** es estrictamente menor que **Y** en al menos un índice j

Solución no dominada y conjunto de Pareto, si una solución no está dominada por ninguna otra solución, se la considera una solución de Pareto.

Generalmente, un problema de Optimización Multiobjetivo tiene más de una solución de Pareto, que forman un conjunto de Pareto (Shao, 2013).

2.4.1 Optimización Multiobjetivo con Algoritmos Genéticos

Muchos enfoques se han desarrollado para resolver la dificultad frente a un problema de Optimización Multiobjetivo. Estos enfoques se pueden clasificar según Hsu, et al. (2002) citado por Li et al. (2010), en tres categorías:

1) Transformar el problema Multiobjetivo en un problema mono-objetivo asignando diferentes coeficientes de peso para cada objetivo.
2) El enfoque no-Pareto, que trata cada objetivo de forma separada.
3) El enfoque de Pareto, que se basa en el concepto de optimización de Pareto. Es decir, satisfacer dos objetivos: converger hacia el Frente de Pareto y obtener soluciones diversificadas dispersas por todo el Frente de Pareto.

En cuanto al segundo enfoque, el no-Pareto, son aquellas propuestas que obtienen individuos no dominados, como por ejemplo, la propuesta de Schaffer con su algoritmo VEGA, pero por el procedimiento utilizado sus resultados fallan con superficies no cóncavas, más detalles se puede obtener de Fonseca & Fleming, 1995.

Con relación al tercer enfoque, el de Pareto, en esta aproximación se garantiza con igual probabilidad la reproducción a todos los individuos “no dominados” de la población, los problemas en la superficie no convexas que se pueden presentar en otras aproximaciones no se plantean en este caso (Fonseca & Fleming, 1995). La desventaja del enfoque de Pareto radica en su dificultad para seleccionar la mejor solución no dominada del conjunto de soluciones no dominadas ya que en la práctica el usuario utilizará solo una única solución (Dehuri, et al. 2008).

Con respecto al primer enfoque, lo hemos dejado al final porque se desea ampliarlo. El método de transformar el problema Multiobjetivo a un solo objetivo es, probablemente, el más simple de los métodos clásicos, en este caso, se forma una función f(x) que es la combinación lineal de los objetivos. El problema escalar resultante según Chankong & Haimes (1983) citado por Claudio (2002) es:

$$\text{Minimizar } f(x) = \sum_{i=1}^{r} w_i f_i(x) \quad (2.11)$$

$$\text{Sujeto a } x \in X^* \quad (2.12)$$

Donde $w_i \geq 0$ es el peso que representa la importancia de f_i con respecto a los otros. Estos pesos son generalmente normalizados, tal que:

$$\sum_{i=1}^{r} w_i = 1 \quad (2.13)$$

Considerando que $W = (w_1, w_2, ..., w_r)$ es un vector de pesos, una solución x^* del problema será Pareto óptima si:

$$x^* \text{ es una solución única y } w_i > 0 \ \forall \ i = 1,\ldots,r \quad (2\text{-}14)$$

Para alcanzar Soluciones Paretos Óptimas, el problema debe ser resuelto iterativamente considerando diferentes vectores de peso positivos W. El optimizador es el encargado por la definición de los pesos apropiados de acuerdo con la importancia de los objetivos. Para que los pesos w_i reflejen aproximadamente la importancia de los objetivos los objetivos deben estar en el mismo orden de magnitud. La principal desventaja del método es que no consigue generar todas las Soluciones Pareto Óptimas cuando el Espacio del Objetivo no es convexo. Esto se ilustra en la Figura 20 para el caso de dos objetivos. Considerar los pesos w_1 y w_2 para minimizar la siguiente función:

$$y = w_1 f_1(x) + w_2 f_2(x), \ x \in X^* \quad (2.15)$$

Esta ecuación puede ser escrita como:

$$f_2(x) = -(w_1/w_2)\, f_1(x) + y/w_2 \qquad (2.16)$$

Esta última ecuación define una recta L cuya inclinación es $-w_1/w_2$ que intersecta el eje f_2 en y/f_2. Esta recta es tangente (o soporte) al Espacio del Objetivo factible Z* en un punto Pareto óptimo. Es decir el método de suma ponderada genera diferentes rectas tangentes, definidas por los valores de w_1 y w_2. En general, no todos los puntos Pareto óptimos admiten rectas tangentes. En la Figura 20, los puntos C y D no poseen rectas tangentes, es decir, no pueden ser encontrados por este método de minimización.

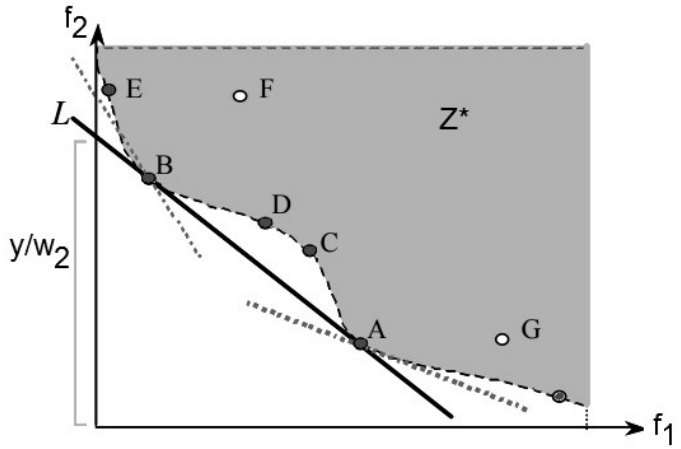


Figura 20. **Interpretación Gráfica de Suma Ponderada.** *Fuente*. Claudio (2002)

Afortunadamente, no hay muchos problemas de Optimización Multiobjetivo del mundo real en donde la región óptima de Pareto es no convexa. Esta es la razón que explica porque el método de Suma Ponderada todavía tiene mucha aceptación, razón por la cual es utilizado en la práctica (Deb, 2005).

La aceptación que tiene el método de Suma Ponderada en el mundo real y concretamente en el ámbito industrial, se verifica al revisar la literatura y comprobar que muchos investigadores han utilizado para solucionar el problema de *Flexible Job Shop Scheduling* Multiobjetivo convirtiendo el problema a mono objetivo mediante la suma ponderada de cada uno de sus objetivos (*Makespan*, *Maximum Workload*, *Total Workload*) se pueden citar

por ejemplo a: Xia & Wu (2005), Xing et al. (2009b), Zhang et al.(2009), Xing et al. (2009 a), Li et al. (2010), Xiong (2012), Ziaee (2014), etc.

En esta tesis, en la solución del Subproblema de Enrutamiento, al tener como criterios de parada del algoritmo a W_M y W_T. Se minimiza a la suma simple de estos dos criterios (sin ponderar) y luego se busca en esa suma mínima si los sumandos cumplen con los criterios de parada, si no fuera así se reinicia la búsqueda (ver detalles en la Metodología).

CAPÍTULO 3: METODOLOGÍA

En este capítulo previamente a la descripción de los diseños de los algoritmos, que se realiza a partir del ítem 3.7, se describirán las hipótesis y variables involucradas en el *Flexible Job Shop Scheduling Problem* (*FJSSP*) a resolver.

También, es pertinente esclarecer que se utilizará, en donde sea pertinente, las denominaciones de *Flexible Job Shop* (*FJS*), *Flexible Job Shop Scheduling* (*FJSS*) o *Flexible Job Shop Scheduling Problem* (*FJSSP*). Siendo la última denominación la más general porque se refiere al problema de programar el procesamiento de *Jobs* en un sistema de fabricación tipo *FJS*.

3.1 Hipótesis

3.1.1 Hipótesis General

Se solucionará el problema de programar óptimamente, en las máquinas de un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), el procesamiento de las operaciones de los *Jobs*, dividiendo el problema en un Subproblema de Enrutamiento y un Subproblema de Secuenciación, minimizando los objetivos del *Maximum Workload*, *Total Workload* y *Makespan* mediante la utilización de Algoritmos Genéticos.

3.1.2 Hipótesis Específicas

H1: Se solucionará el Subproblema de Enrutamiento óptimo, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en asignar a cada operación de los *Jobs*, una entre muchas máquinas candidatas para su procesamiento, minimizando los criterios de: *Total Workload* (W_T, suma de carga de trabajo de todas las máquinas) y *Maximum Workload* (W_M, máxima carga de trabajo entre todas las máquinas) mediante la utilización de Algoritmos Genéticos.

H2: Se solucionará el Subproblema de Secuenciación óptima, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en secuenciar el orden en que se deben procesar las operaciones de los *Jobs* en las máquinas seleccionadas previamente, minimizando el objetivo del *Makespan* (C_M, tiempo de finalización de todos los *Jobs*) mediante la utilización de Algoritmos Genéticos.

H3: Se probará la eficacia y eficiencia de los algoritmos propuestos considerando la existencia de resultados obtenidos por otros investigadores de la literatura que han probado sus algoritmos con los casos de sistemas de fabricación tipo *FJS* planteados por Kacem.

3.2 Identificación de Variables

H1: Se solucionará el Subproblema de Enrutamiento óptimo, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en asignar a cada operación de los *Jobs*, una entre muchas máquinas candidatas para su procesamiento, minimizando los criterios de: *Total Workload* (W_T, suma de carga de trabajo de todas las máquinas) y *Maximum Workload* (W_M, máxima carga de trabajo entre todas las máquinas) mediante la utilización de Algoritmos Genéticos.

Variables dependientes: Vector con óptima distribución de máquinas para cada operación, mínimo (*Maximum Workload*, W_M) y mínimo (*Total Workload* W_T).

Variables independientes: Datos de casos de *FJSS* planteados por Kacem (Conjunto de máquinas disponibles, conjunto de *Jobs*, operaciones por cada *Job*, tiempo de procesamiento de las operaciones en cada máquina), parámetros del Algoritmo Genético de Rutas.

H2: Se solucionará el Subproblema de Secuenciación óptima, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en secuenciar el orden en que se deben procesar las operaciones de los *Jobs* en las máquinas seleccionadas previamente, minimizando el objetivo del *Makespan* (C_M, tiempo de finalización de todos los *Jobs*) mediante la utilización de Algoritmo Genéticos.

Variables dependientes: Vector con óptima secuencia de operaciones, mínimo *Makespan*.

Variables independientes: Vector con óptima distribución de máquinas (del proceso anterior), tiempos de procesamiento de las máquinas, parámetros del Algoritmo Genéticos de Secuencias.

H3: Se probará la eficacia y eficiencia de los algoritmos propuestos considerando la existencia de resultados obtenidos por otros investigadores de la literatura que han probado sus algoritmos con los casos de sistemas de fabricación tipo *FJS* planteados por Kacem.

Variables dependientes: mínimo *Makespan*, mínimo *Maximum Workload* y mínimo *Total Workload*, Tiempo de Ejecución de los algoritmos.

Variables independientes Datos de los casos de *FJSS* planteados por Kacem, parámetros del Algoritmo Genético de Rutas y del Algoritmo Genético de Secuencias.

3.3 Operacionalización de las Variables

Tabla 6
Operacionalización de las Variables

H1: Se solucionará el Subproblema de Enrutamiento óptimo, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en asignar a cada operación de los *Jobs*, una entre muchas máquinas candidatas para su procesamiento, minimizando los criterios de: *Total Workload* (W_T, suma de carga de trabajo de todas las máquinas) y *Maximum Workload* (W_M, máxima carga de trabajo entre todas las máquinas) mediante la utilización de Algoritmos Genéticos.		
VARIABLES DEPENDIENTES	**CONCEPTO**	**UNIDADES DE MEDIDA**
Vector con óptima distribución de máquinas para cada operación	Matemáticamente es un vector cuyos elementos son las máquinas seleccionadas para cada operación. En términos de algoritmos genéticos es el mejor Cromosoma de Máquinas encontrado.	Adimensional (Cada elemento del vector es una máquina asignada a la operación).
mínimo (*Maximum Workload*, W_M)	El *Maximun Workload* se calcula en cualquier Cromosoma de Máquinas e indica la máxima carga de trabajo que tiene una de las máquinas del cromosoma y el mínimo (*Maximum Workload*) es encontrado en el cromosoma optimizado como resultado final del Algoritmo Genético de Rutas.	Tiempo (ms, seg, minutos, horas, etc.)
mínimo (*Total Workload*, W_T)	El *Total Workload* se calcula en cualquier Cromosoma de Máquinas e indica la suma total de la carga de trabajo de todas las máquinas del cromosoma y el mínimo (*Total Workload*) es encontrado en el cromosoma optimizado como resultado final del Algoritmo Genético de Rutas.	Tiempo (ms, seg, minutos, horas, etc.)
VARIABLES INDEPENDIENTES	**CONCEPTO**	**UNIDADES DE MEDIDA**
Conjunto de máquinas disponibles	Constituye parte de la matriz de datos del problema *FJSS*. Son las máquinas posibles para ser designadas a cada operación.	Adimensional
Conjunto de *Jobs*	Constituye parte de la matriz de datos del problema *FJSS*. Son los Jobs del sistema de fabricación.	Adimensional
Operaciones por cada *Job*	Constituye parte de la matriz de datos del problema *FJSS*. Son las operaciones en que se divide cada Job.	Adimensional
Tiempo de procesamiento de las operaciones en cada máquina	Constituye parte de la matriz de datos del problema *FJSS*. Es el tiempo que le corresponde a cada máquina para procesar una determinada operación.	Tiempo (ms, seg, minutos, horas, etc.)
Parámetros del Algoritmo Genéticos de Rutas	Son los datos para los cuales el Algoritmo Genético de Rutas opera, principalmente son: umbrales de parada, tamaño de población de cromosomas, porcentaje de sobrevivientes, porcentaje de mutación.	Adimensional

Fuente. Elaboración propia

Tabla 6 (continuación…)
Operacionalización de las Variables

H2: Se solucionará el Subproblema de Secuenciación óptima, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en secuenciar el orden en que se deben procesar las operaciones de los *Jobs* en las máquinas seleccionadas previamente, minimizando el objetivo del *Makespan* (C_M, tiempo de terminación de todos los *Jobs*) mediante la utilización de Algoritmos Genéticos.		
VARIABLES DEPENDIENTES	**CONCEPTO**	**UNIDADES DE MEDIDA**
Vector con óptima secuencia de operaciones	Matemáticamente es un vector cuyos elementos son operaciones y cuyo orden es la secuencia de operaciones que debe tener el proceso. En términos de algoritmos genéticos es el mejor Cromosoma de Secuencias que es encontrado.	Adimensional (Cada elemento del vector es una operación cuya secuencia óptima es el orden el orden de la fila)
mínimo (*Makespan*)	El *Makespan* se calcula en cualquier Cromosoma de Secuencias e indica el tiempo que se emplea para que todos los *Jobs* sean procesados. El mínimo (*Makespan*) se encuentra en el cromosoma optimizado como resultado final del Algoritmo Genético de Secuencias.	Tiempo (ms, seg, minutos, horas, etc.)
VARIABLES INDEPENDIENTES	**CONCEPTO**	**UNIDADES DE MEDIDA**
Vector con óptima distribución de máquinas (del proceso anterior).	Fue descrito	Adimensional
Tiempo de procesamiento de las operaciones en cada máquina	Es un vector cuyos elementos contienen los tiempos de proceso de cada máquina seleccionada.	Tiempo (ms, seg, minutos, horas, etc.)
Parámetros del Algoritmo Genéticos de Secuencias	Son los datos para los cuales el Algoritmo Genético de Secuencias opera, principalmente son: umbrales de parada, tamaño de población de cromosomas, porcentaje de sobrevivientes, porcentaje de mutación	Adimensional

Fuente. Elaboración propia

Tabla 6 (continuación…)
Operacionalización de las Variables

H3: Se probará la eficacia y eficiencia de los algoritmos propuestos considerando la existencia de resultados obtenidos por otros investigadores de la literatura que han probado sus algoritmos con los casos de sistemas de fabricación tipo *FJS* planteados por Kacem.		
VARIABLES DEPENDIENTES	CONCEPTO	UNIDADES DE MEDIDA
Mínimo *Makespan*	Fue descrito	Tiempo (ms, seg, minutos, horas, etc.)
Mínimo *Maximum Workload*	Fue descrito	Tiempo (ms, seg, minutos, horas, etc.)
Mínimo *Total Workload*	Fue descrito	Tiempo (ms, seg, minutos, horas, etc.)
Tiempo de Ejecución de los algoritmos	Tiempo promedio sobre veinte corridas que le tomará a los algoritmos solucionar cada uno de los casos de *FJSS* planteados por Kacem.	Tiempo (ms, seg, minutos, horas, etc.)
VARIABLES INDEPENDIENTES	CONCEPTO	UNIDADES DE MEDIDA
Datos de los casos de *FJSS* planteados por Kacem	Son matrices con datos de sistemas de fabricación tipo FJSS que el investigador Kacem (2002, 2002a) creó para probar sus algoritmos y que han sido adoptados por otros investigadores para probar también sus propuestas.	Multidimensional (adimensional y tiempos)
Parámetros del algoritmo Genético de Rutas y del Algoritmo Genético de Secuencias	Fue descrito	Adimensional

Fuente. Elaboración propia

Lo anterior se articula dentro de las Matrices de Consistencia mostradas en la Tabla 7.

Tabla 7
Matriz de Consistencia

Problema General	Objetivo General	Hipótesis General
¿Cómo solucionar el problema de programar óptimamente, en las máquinas de un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), el procesamiento de las operaciones de los *Jobs*, dividiendo el problema, en un Subproblema de Enrutamiento y un Subproblema de Secuenciación, minimizando los objetivos del *Maximum Workload*, *Total Workload* y *Makespan* mediante la utilización de Algoritmos Genéticos?	Solucionar el problema de programar óptimamente, en las máquinas de un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), el procesamiento de las operaciones de los *Jobs*, dividiendo el problema, en unl Subproblema de Enrutamiento y un Subproblema de Secuenciación, minimizando los objetivos del *Maximum Workload*, *Total Workload* y *Makespan* mediante la utilización de Algoritmos Genéticos.	Se solucionará el problema de programar óptimamente, en las máquinas de un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), el procesamiento de las operaciones de los *Jobs*, dividiendo el problema, en el Subproblema de Enrutamiento y el Subproblema de Secuenciación, minimizando los objetivos del *Maximum Workload*, *Total Workload* y *Makespan* mediante la utilización de Algoritmos Genéticos.

Fuente. Elaboración propia

Tabla 7 (continuación…)
Matriz de Consistencia

Problemas Específicos	Objetivos Específicos	Hipótesis Específica	Variables	Técnica
¿Cómo solucionar el Subproblema de Enrutamiento óptimo, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en asignar a cada operación de los *Jobs*, una entre muchas máquinas candidatas para su procesamiento, minimizando los criterios de: *Total Workload* (W_T, suma de carga de trabajo de todas las máquinas) y *Maximum Workload* (W_M, máxima carga de trabajo entre todas las máquinas) mediante Algoritmos Genéticos?	Solucionar el Subproblema de Enrutamiento óptimo, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en asignar a cada operación de los *Jobs*, una entre muchas máquinas candidatas para su procesamiento, minimizando los criterios de: *Total Workload* (W_T, suma de carga de trabajo de todas las máquinas) y *Maximum Workload* (W_M, máxima carga de trabajo entre todas las máquinas) mediante la utilización de Algoritmos Genéticos.	Se solucionará el Subproblema de Enrutamiento óptimo, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en asignar a cada operación de los *Jobs*, una entre muchas máquinas candidatas para su procesamiento, minimizando los criterios de: *Total Workload* (W_T, suma de carga de trabajo de todas las máquinas) y *Maximum Workload* (W_M, máxima carga de trabajo entre todas las máquinas) mediante la utilización de Algoritmos Genéticos.	<u>Variables dependientes</u>. Vector con óptima distribución de máquinas. Mínimo *Maximum Workload*. Mínimo *Total Workload*. <u>Variables independientes.</u> Datos de casos de *FJSS* planteados por Kacem: Conjunto de máquinas disponibles, conjunto de *Jobs*, operaciones por cada Job, tiempo de ejecución de las operaciones en cada máquina. Parámetros del Algoritmos Genético de Rutas.	Algoritmos Genéticos
¿Cómo solucionar el Subproblema de Secuenciación óptima, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en secuenciar el orden en que se deben procesar las operaciones de los *Jobs* en las máquinas seleccionadas previamente, minimizando el objetivo del *Makespan* (C_M, tiempo de finalización de todos los *Jobs*) mediante Algoritmos Genéticos?	Solucionar el Subproblema de Secuenciación óptima, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en secuenciar el orden en que se deben procesar las operaciones de los *Jobs* en las máquinas seleccionadas previamente, minimizando el objetivo del *Makespan* (C_M, tiempo de finalización de todos los *Jobs*) mediante Algoritmos Genéticos.	Se solucionará el Subproblema de Secuenciación óptima, en un sistema de fabricación tipo *Flexible Job Shop* (*FJS*), consistente en secuenciar el orden en que se deben procesar las operaciones de los *Jobs* en las máquinas seleccionadas previamente, minimizando el objetivo del *Makespan* (C_M, tiempo de terminación de todos los *Jobs*) mediante la utilización de Algoritmos Genéticos.	<u>Variables dependientes</u>. Vector óptima secuencia de operaciones. Mínimo *Makespan*. <u>Variables independientes</u>. Vector con óptima distribución de máquinas (Resultado anterior). Tiempo de ejecución de las operaciones en cada máquina. Parámetros del Algoritmos Genético de Secuencias.	Algoritmos Genéticos
¿Cómo probar la eficacia y eficiencia de los algoritmos propuestos considerando la existencia de resultados obtenidos por otros investigadores de la literatura que han probado sus algoritmos con los casos de sistemas de fabricación tipo *FJS* planteados por Kacem?	Probar la eficacia y eficiencia de los algoritmos propuestos considerando la existencia de resultados obtenidos por otros investigadores de la literatura que han probado sus algoritmos con los casos de sistemas de fabricación tipo *FJS* planteados por Kacem.	Se probará la eficacia y eficiencia de los algoritmos propuestos considerando la existencia de resultados obtenidos por otros investigadores de la literatura que han probado sus algoritmos con los casos de sistemas de fabricación tipo *FJS* planteados por Kacem.	<u>Variables dependientes</u> Mínimo *Makespan*. Minimo *Maximum Workload*. Mínimo *Total Workload*. Tiempo de Ejecución de los algoritmos. <u>Variables independientes</u> Datos de casos de *FJSS* planteados por Kacem. Parámetros del Algoritmo Genético de Rutas y del Algoritmo Genéticos de Secuencias.	Algoritmos Genéticos

Fuente. Elaboración propia

3.4 Unidad de Análisis

El objeto de estudio o unidad de análisis es un sistema de fabricación tipo Flexible Job Shop, en donde hay que encontrar las máquinas más idóneas que ejecutaran las operaciones de cada Job y posteriormente encontrar en qué orden o secuencia las operaciones destinadas a cada máquina se deben procesar en las máquinas, respetando el orden de precedencia que tienen las operaciones en cada Job. Por la complejidad, lo convierte en un problema combinatorio NP-Hard.

3.5 Población de Estudio

Los resultados obtenidos mediante la implementación de los Algoritmos Genéticos propuestos que resuelva el *FJSSP*, beneficiará aquellas industrias manufactureras mecanizadas que funcionan a pedido y en donde la aplicación de algoritmos eficientes permite obtener productos en forma más rápida y eficaz, y que puedan tener un impacto importante sobre la productividad de un proceso. Sin la existencia de la programación de tareas no es posible obtener eficiencia, competitividad y rentabilidad en las industrias.

3.6 Muestra

Se ha probado la propuesta con datos que simulan cinco (5) casos de *FJSSP* de tamaños diferentes (número variables de *Jobs* y máquinas) de Total Flexibilidad y Parcial Flexibilidad. De Kacem et al. (2002) y Kacem et

al. (2002 a), se han extraído los cinco casos de *FJSSP* que prueban el funcionamiento de los algoritmos propuestos. Se han seleccionado estos casos, porque los investigadores de la literatura consultada se han apoyado de ellos para medir y probar el rendimiento de sus algoritmos lo que garantiza la calidad de la muestra. Además, de esta forma se facilitará la comparación de resultados con los otros investigadores.

3.7 Diseño y Tipo de Investigación

Debido al gran número de posibles soluciones que tiene el *FJSSP*, los algoritmos determinísticos o métodos exactos que resuelven sus modelos matemáticos solo encuentran soluciones deterministas en tiempos razonables para tamaños pequeños del *FJSSP*. Sin embargo, para tamaños grandes, en donde el número de sus variables aumenta (número de máquinas disponibles, número de *Jobs*, etc.) los tiempos de ejecución de los algoritmos determinísticos crecen exponencialmente con las variables. De ahí su clasificación como un problema intratable fuertemente NP-Hard, en donde no se pueden plantear métodos tradicionales de solución.

NP-Hard significa que sólo se pueden encontrar soluciones óptimas aproximadas mediante algoritmos no determinísticos que resuelven el problema en tiempos polinómicos (no en tiempos exponenciales como fue descrito en el párrafo precedente). Por esta razón, en esta investigación de tesis se ha aplicado técnicas metaheurísticas diseñando algoritmos genéticos que consigan una buena solución en un tiempo razonable para minimizar las variables dependientes del problema *FJSSP*.

Dentro de este contexto las variables independientes son manipuladas por los algoritmos genéticos que se han diseñado. Para comprobar las tres hipótesis se han solucionado los cinco casos de sistemas de fabricación tipo *FJSSP* planteados por Kacem (2002) y Kacem (2002 a). Los casos de Kacem han sido utilizados como *Benchmarking* por muchos investigadores

razón por la cual han logrados minimizar, con sus propias técnicas metaheurísticas, las variables dependientes: *Maximum Workload* (W_M), *Total Workload* (W_T) y *Makespan* (C_M). Por lo tanto, los valores obtenidos por estos investigadores serán los valores fijos (umbrales) para comparación en cada hipótesis.

Para verificar la propuesta de la primera hipótesis, durante la ejecución del Algoritmo Genético de Rutas, se ha analizado la progresión (en cada generación) de la minimización de las variables dependientes W_M *y* W_T al tratar de alcanzar o superar sus valores fijos umbrales. Mientras que para verificar la propuesta de la segunda hipótesis, durante la ejecución del Algoritmo Genético de Secuencias, se ha analizado la progresión de la variable dependiente C_M al tratar de alcanzar o superar su valor fijo umbral. En la tercera hipótesis se ha verificado si las minimizaciones alcanzadas coinciden con la mayoría de los resultados de los investigadores (eficacia) y si los tiempos de ejecución de los algoritmos superan o son iguales al de la mayoría de los investigadores (eficiencia).

En la Figura 21, se muestra un esquema de representación gráfica, en donde se puede observar cómo se relaciona el subproblemas de Enrutamiento y el Subproblema de Secuenciación con las variables involucradas del problema.

El Algoritmo Genético de Rutas, el Algoritmo Genético de Secuencias y el Algoritmo de Presentación de Resultados en Diagrmas de Gantt, se han ensamblado en un programa denominado *Multiobjective Flexible Job Shop Scheduling,* su diagrama de bloques se muestra en la Figura 22, el cual es coherente con la Figura 21. Los procedimientos del diseño y las características de los datos involucrados y otros se describirán en el resto del capítulo.

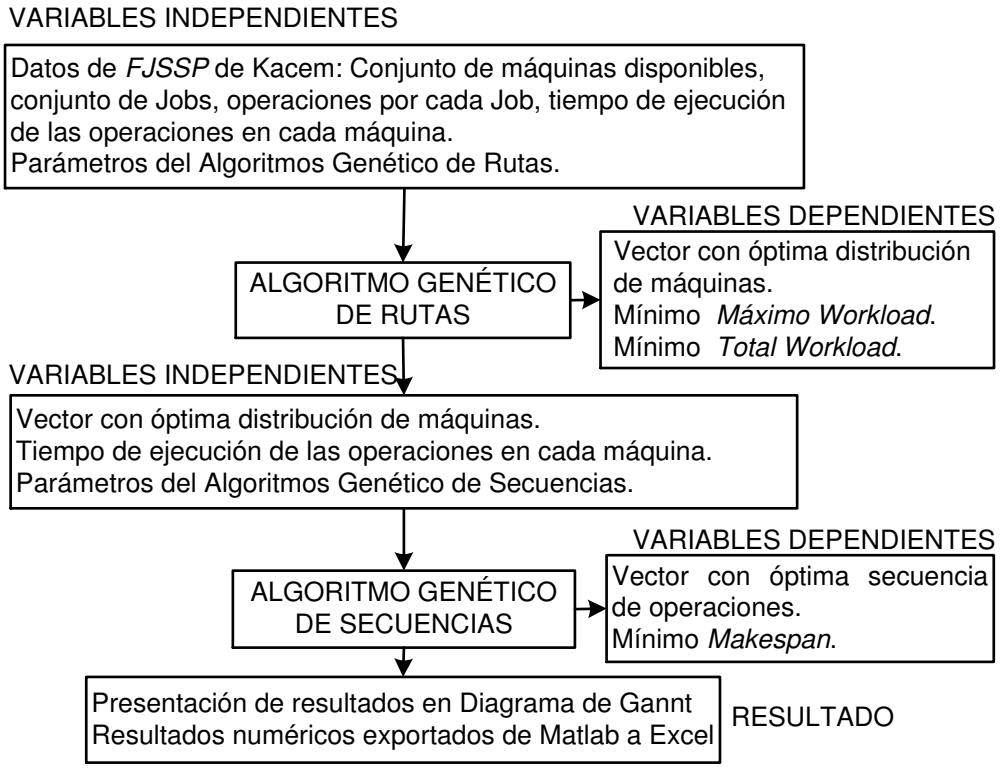


Figura 21. **Esquema Gráfico de Relación de Variables**. *Fuente.* Elaboración propia

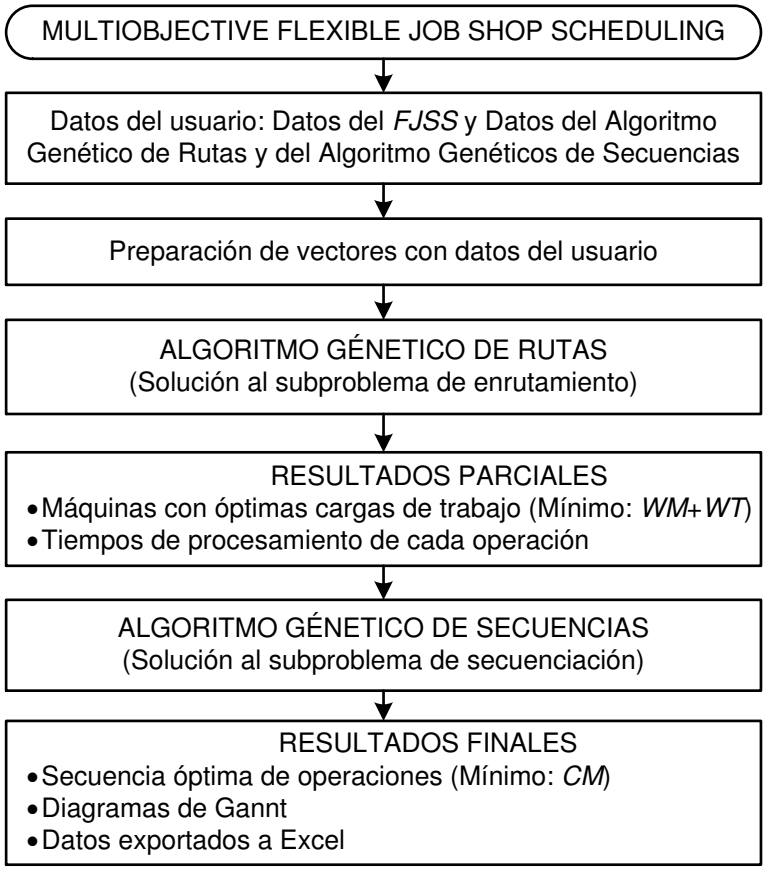


Figura 22. **Secuencia General del Programa**. *Fuente*. Elaboración propia

Los datos del usuario (ver Figura 22) son los relativos al *FJSSP* y a los Algoritmos Genéticos. Entre los datos del *FJSSP*, se tienen por ejemplo: Conjunto de máquinas disponibles, conjunto de *Jobs*, operaciones por cada *Job*, tiempo de ejecución de las operaciones en cada máquina, etc. Para lo cual, se utilizaran los casos de *FJSSP* planteados en los trabajos de Kacem et al. (2002) y Kacem et al. (2002a). Mientras que los datos relativos a los Algoritmos Genéticos tanto de Rutas como de Secuencias corresponden a los parámetros para los cuales se ejecutarán los algoritmos, son por ejemplo: Tamaño de la población de cromosomas, número de generaciones, etc. (la información solicitada lo veremos con mayor detalle en el capítulo 4). Luego que se ingresan los datos, estos son convertidos en vectores que van a facilitar los procesos de cálculo de los algoritmos.

Los datos relativos al *FJSSP* se especifican en tablas, como aquellas presentadas y definidas en Kacem et al (2002) y Kacem et al (2002 a) y que han servido a este trabajo y a los demás investigadores como *Benchmarking* para probar sus algoritmos (ver ítem 3.8 para mayores detalles). Las tablas deben ser ingresadas al programa como matrices.

En la Figura 23, se muestra una tabla de un *FJSSP*, se ha realizado algunas anotaciones dentro de ella. La tabla corresponde a un sistema de fabricación de cuatro *Jobs* (J1, J2, J3 y J4). Donde, el *Job* 1 tiene 3 operaciones, el *Job* 2 también tiene 3 operaciones, el *Job* 3 tiene 4 operaciones y finalmente el *Job* 4 tiene 2 operaciones; sumando un total de 12 operaciones. Las operaciones pueden ser denotadas como números consecutivos desde 1 hasta la última operación (12 en el caso mostrado), o como un número relativo al *Job* al que pertenece la operación. Así por ejemplo, la operación 7, es la primera operación del *Job* 3.

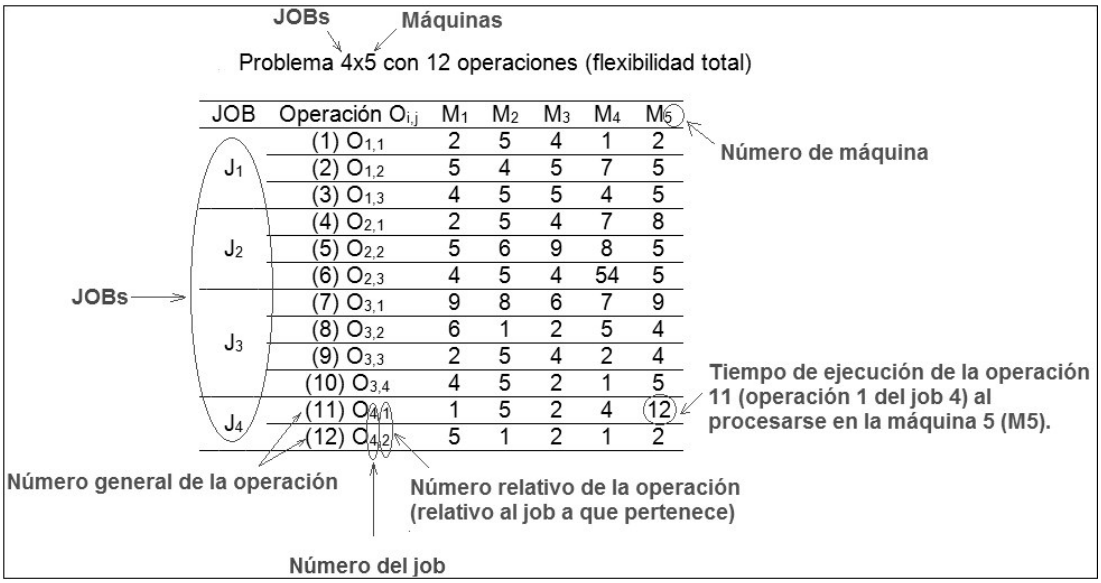


Figura 23. **Datos *FJSSP***. *Fuente*. Elaboración propia

El proceso mostrado es de Flexibilidad Total (como lo define Kacem) porque cualquiera de las operaciones tiene la posibilidad de ser procesada en cualquiera de cinco máquinas (M1, M2, M3, M4 y M5) con tiempos que no necesariamente son los mismos. En cambio, en un problema de Flexibilidad Parcial bastaría que uno de las operaciones no tenga la posibilidad de ser procesada en una de las máquinas.

La tabla de la Figura 23, en la investigación de la tesis, es interpretada como un arreglo matricial tridimensional de 60 elementos, cada elemento representa un tiempo de proceso. Si la tabla es llamada T, entonces, cada elemento es especificado por T(i, j, k) donde i representa un *Job*, j la operación dentro del *Job* y k una máquina. Así por ejemplo, T(4,1, 5) corresponde al elemento 12, lo que significa que si la operación 1 del *Job* 4 es procesada en la máquina 5, este proceso toma 12 unidades de tiempo.

La tabla, es escrita en un archivo de texto con la extensión **m** (de Matlab). En la Figura 24, se muestra, el archivo que contiene los datos de la Figura 23, es una de las maneras en que puede ser escrita esa matriz. Observar que al tener el *Job* 3, 4 operaciones (4 filas), entonces, a todos los demás *Jobs* se le deben aumentar filas ficticias con elementos nulos hasta

completar cuatro filas. En el ejemplo de la Figura 24, el nombre del archivo que lo identifica es: DATOS_4X5.

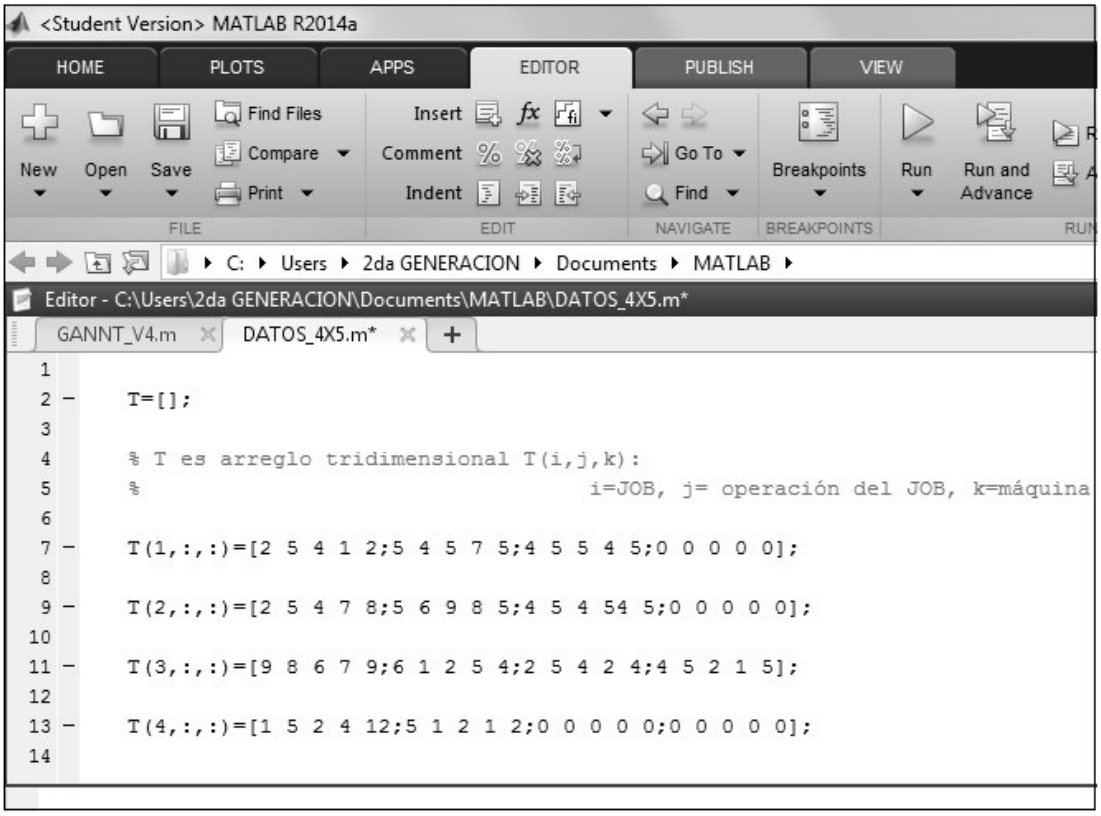


Figura 24. **Matriz de Datos en Matlab**. *Fuente.* Elaboración propia

Como se ha señalado, para solucionar el *FJSSP,* se ha dividido el problema en dos, este enfoque también ha sido realizado por otros investigadores, el primero de los cuales es Brandimarte en 1993. Como resultado de la división, se ha generado dos problemas, a los cuales se les ha denominado: Subproblema de Enrutamiento y Subproblema de Secuenciación.

Para solucionar el Subproblema de Enrutamiento se ha diseñado e implementado un Algoritmo Genético denominado Algoritmo Genético de Rutas (descrito en detalle en el ítem 3.8), el cual designa una máquina a cada operación del sistema de fabricación para su procesamiento. Se debe tener en consideración que las máquinas no resulten con sobrecarga de trabajo, por ello, una designación óptima implica minimizar los criterios de

Maximum Workload (W_M) y el *Total Workload* (W_T) de las máquinas. Se genera como resultado un vector que contiene las máquinas que se han designado a cada operación, el cual es de una longitud tan grande como el número de operaciones del sistema de fabricación. También, se genera otro vector que contiene el tiempo que le toma a cada máquina procesar individualmente a cada operación. Ambos vectores, están correlacionados.

Para solucionar el Subproblema de Secuenciación se ha diseñado e implementado un Algoritmo Genético denominado Algoritmo Genético de Secuencias (descrito en detalle en el ítem 3.9), el cual encuentra el orden óptimo en que se deben procesar las operaciones destinadas a cada máquina. Este orden redunda en el tiempo del sistema de fabricación, por lo que es vital para la producción ordenar o secuenciar óptimamente la ejecución de cada operación. Además, el algoritmo debe tener en cuenta el mantener la precedencia (uno a continuación del otro) que deben tener las operaciones en un mismo *Job*. El criterio a minimizar, es el tiempo de finalización de todas las operaciones de los *Jobs*.

Por consiguiente, el resultado final es la obtención de una secuencia óptima de operaciones con un mínimo C_M, y cargas de trabajo equilibradas con un mínimo W_M y W_T.

Para visualizar y validar de forma objetiva los resultados, éstos son presentados mediante Diagramas de Gantt y los datos numéricos exportados desde Matlab a archivo en Excel, el diseño de esta parte del programa es descrito en el ítem 3.10.

Es relevante mencionar que los algoritmos han sido codificados totalmente en el Lenguaje **m** de Matlab, para ello se ha utilizado la versión estudiantil R214A de Matlab, el cual ha corrido en una PC con procesador i5 de 2.9 GHz. y 4 G.B de RAM.

3.8 Diseño del Algoritmo Genético de Rutas

El Algoritmo Genético de Rutas, soluciona el Subproblema de Enrutamiento, es el encargado de designar máquinas a cada operación de los *Jobs*, por este motivo, luego de la designación, la ruta o camino que deben seguir las operaciones para su procesamiento es la establecida para llegar a las máquinas elegidas. En el proceso de designar máquinas a las operaciones, el Algoritmo Genético de Rutas debe evitar sobrecargarlas, por este motivo debe minimizar al *Maximun Workload* (W_M) y el *Total Workload* (W_T).

En la Figura 25, se muestra el Diagrama de Flujo del Algoritmo Genético de Rutas, genera una población de cromosomas, en donde se designa a cada operación una máquina aleatoriamente (véase ítem 3.81). Luego, se calcula el valor de W_M y el valor de W_T de cada cromosoma de la población, se suman ambos resultados y se guarda la suma en un arreglo: costo suma = $W_M + W_T$.

Luego se clasifica de menor a mayor el resultado de la suma, de acuerdo a ello, se ordenan a los cromosomas, así como también se ordenan sus valores correspondientes de W_M y W_T. El resultado ordenado actualiza al arreglo “costo suma”. Se toma como resultado preliminar al primero de la población y se guarda el dato en un vector: mejor suma = costosuma (1, :).

Estos pasos incluso se realizan, antes de ingresar al proceso de evolución propiamente dicho. Es decir, antes de seleccionar, cruzar y mutar a los cromosomas. El objeto es poder detectar si en la selección aleatoria de la población, el mejor cromosoma resultante ya pueda estar cumpliendo con el umbral de parada del usuario. Es probable que esto ocurra especialmente para tamaños de pequeños de *FJSSP*, de esta manera se consigue rapidez en obtener los resultados.

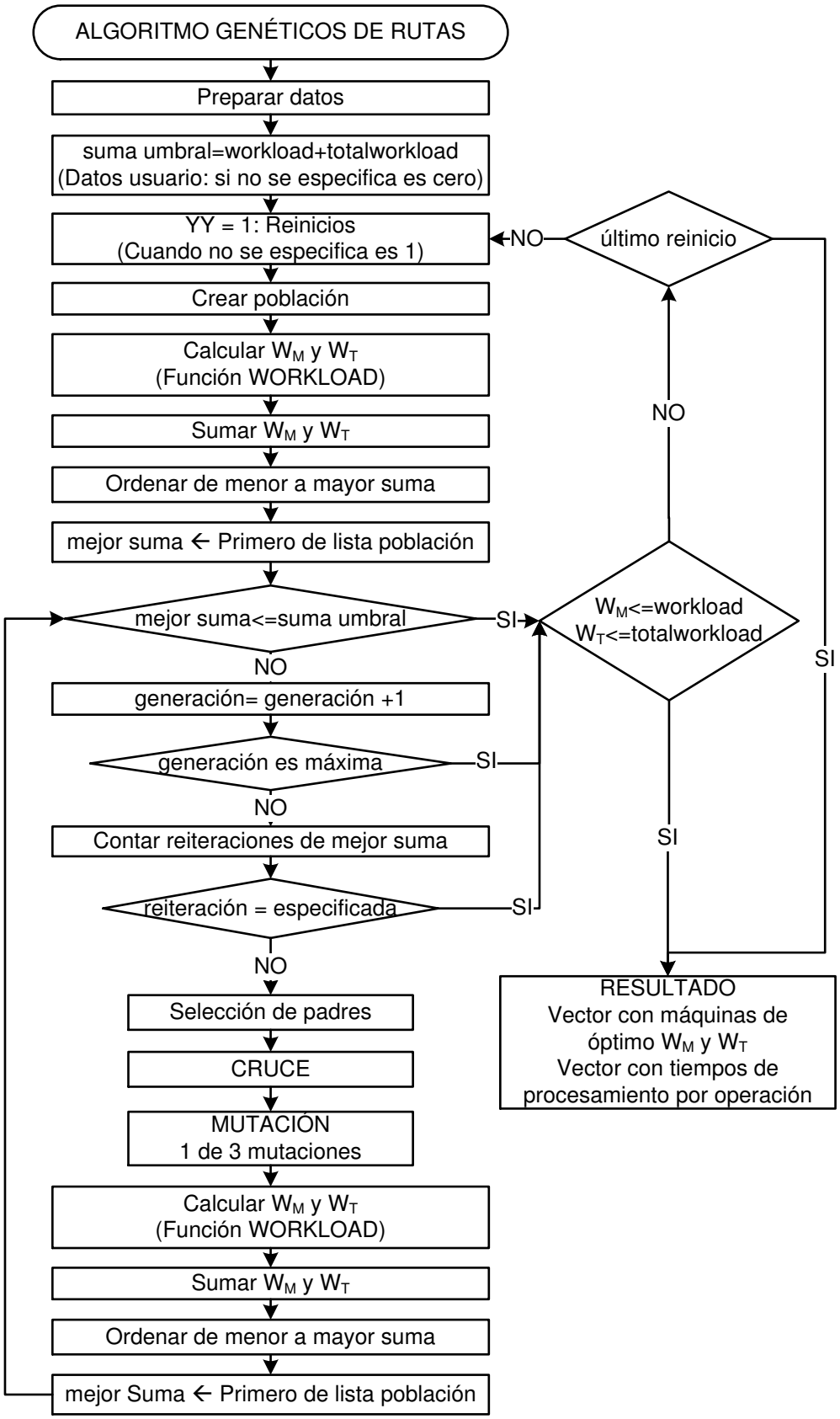


Figura 25. **Algoritmo Genético de Rutas**. *Fuente*. Elaboración propia

Cuando el usuario especifica umbrales para W_M y W_T el algoritmo actúa de un modo particular. Como se observa en la Figura 25, a los umbrales se les ha denominado: “workload” y “totalworkload”. Estos valores se suman y se almacenan como “suma umbral”. Entonces, en cada generación se compara la “mejor suma” (que se ha obtenido de la población) con la “suma umbral”. Si la “mejor suma” de la población es menor o igual que “suma umbral”, entonces se podría suponer que el algoritmo debe concluir. Sin embargo, los sumandos de la “mejor suma” no necesariamente van a ser menores o iguales a los umbrales de cada sumando. Por lo que, se realiza una segunda verificación, la cual consiste en comparar los sumandos de la “mejor suma” con los valores umbrales de: “workload” y “totalworkload”. Si no se cumple, se reinicia a la población, significa entonces que nuevamente se inicia con el proceso. El Algoritmo Génetico de Rutas, en este caso, tiene dos criterios de parada prioritarios que son la “suma umbral” y “Reinicios”, los otros criterios de parada como son: el número de generaciones y las reiteraciones del mejor valor obtenido, son de segunda prioridad, porque se les asignan valores altos.

Al reinicializar la población del Algoritmo Genético, se emula a aquellas herramientas heurísticas que se reinician como parte de su proceso de búsqueda. Se experimentó, también, en no reinicializar a la población y continuar con la misma población hasta satisfacer la condición de los umbrales o alcanzar el máximo número de generaciones. Para ello, se modificó al Algoritmo Genético de Rutas como se muestra en la Figura 26. En este caso, el funcionamiento del Algoritmo fue más convencional ya que con la evolución de la misma población se llegó a encontrar los resultados. Sin embargo, el tiempo de ejecución del Algoritmo fue comparativamente excesivo (especialmente en el caso de *FJSSP* de 8 *Jobs* y 8 máquinas). Por consiguiente, el primer método es el que se propone en la tesis.

Es necesario anotar, que los umbrales del usuario no deben ser arbitrarios. Por eso, para un problema desconocido, se puede ejecutar el algoritmo, sin dar valores umbrales y luego si se desea afinar la búsqueda proporcionar umbrales aproximados.

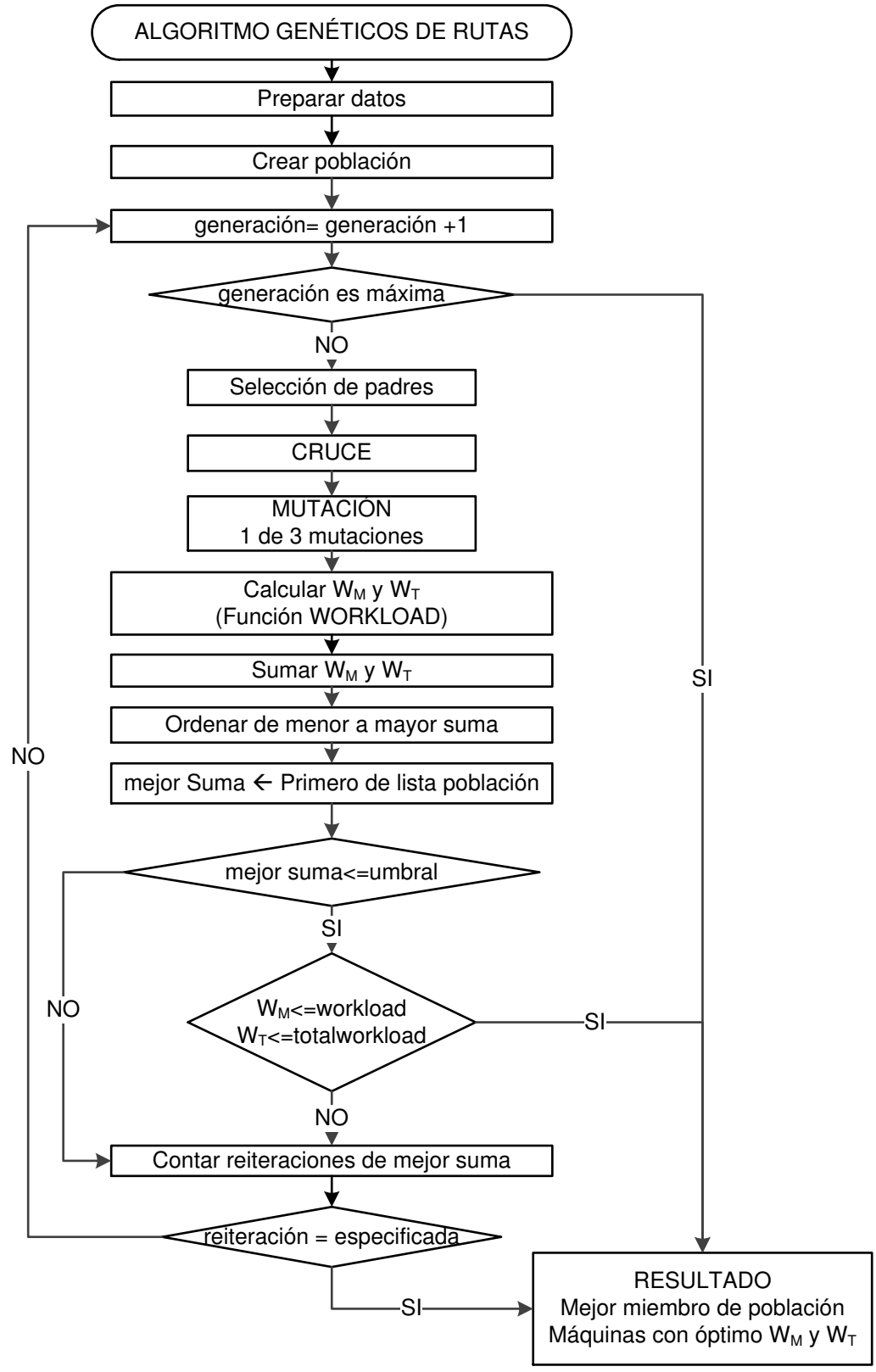


Figura 26. **Algoritmo Genético de Rutas Alterno**. *Fuente*. Elaboración propia

Cuando el Algoritmo Genético de Rutas se ejecuta sin umbrales para el *Total Workload* y *Maximum* Workload. Es decir, cuando "corre libremente", internamente a la "suma umbral" y a los umbrales de los sumandos se les asignan valores nulos, al número de reinicios de la población se le asigna la unidad, este último valor en el bloque "yy = 1: reinicios" de la Figura 25, implicará que no habrá reinicios. En este caso, entonces, el algoritmo no se detiene por alcanzar el valor de la suma umbral ya que la "mejor suma" nunca va ser menor que la "suma umbral", que es nula. El Algoritmo se detendrá con el máximo número de reiteraciones del mejor resultado o con el número máximo de generaciones.

Observe que en la Figura 25, en todos los casos hay una evaluación que converge a la bifurcación "W_M<= workload, W_T<= totalworkloadl", esto es correcto porque en el caso de "correr libremente" al llegar a esa bifurcación la repuesta será NO y luego al derivarse a la segunda bifurcación "último reinicio" la respuesta será SI (ya que en este caso, como se dijo el reinicio vale 1) con lo cual se forzará a finalizar la búsqueda.

No se debe entender a todas las bifurcaciones siempre como un "IF", en el código Matlab en realidad se está utilizando "IF", "While" y "For" en las diferentes bifurcaciones.

Es muy importante los operadores de cruce y la mutación, sin ellos no habría diversidad y no se encontraría de manera eficaz los resultados. Más adelante se tratarán estos temas y veremos los cruces y mutaciones propuestos.

El resultado final del Algoritmo Genético de Rutas, es la entrega de dos vectores, uno conteniendo las máquinas que han sido designadas a cada una de las operaciones y el otro conteniendo los tiempos de procesos de las operaciones en las máquinas.

3.8.1 Cromosoma en el Algoritmo Genético de Rutas

La población del Algoritmo Genético de Rutas, es un arreglo constituido por filas y columnas. Las filas son los cromosomas, miembros o individuos mientras que a las columnas son los genes. A los cromosomas del Algoritmo Genético de Rutas se les ha denominado en la tesis, indistintamente, como Cromosomas de Máquinas o Cromosomas de Rutas.

Cada Cromosoma de Máquinas tiene una longitud, o número de columnas, igual al número de operaciones totales del *FJSSP,* es decir, la primera operación es representada por la ubicación de la primera columna, la segunda operación por la segunda columna y asi sucesivamente. En cada columna se ingresa un número entero que representa al número de máquina designada para procesar la operación. A continuación, se explica la codificación del Cromosoma de Máquinas, colocando como ejemplo los datos del *FJSSP* de 4Jobs, 5 máquinas de 12 operaciones de la Tabla 8,

Tabla 8
FJSSP con 12 Operaciones

JOB	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5
J_1	(1) $O_{1,1}$	2	5	4	1	2
	(2) $O_{1,2}$	5	4	5	7	5
	(3) $O_{1,3}$	4	5	5	4	5
J_2	(4) $O_{2,1}$	2	5	4	7	8
	(5) $O_{2,2}$	5	6	9	8	5
	(6) $O_{2,3}$	4	5	4	54	5
J_3	(7) $O_{3,1}$	9	8	6	7	9
	(8) $O_{3,2}$	6	1	2	5	4
	(9) $O_{3,3}$	2	5	4	2	4
	(10) $O_{3,4}$	4	5	2	1	5
J_4	(11) $O_{4,1}$	1	5	2	4	12
	(12) $O_{4,2}$	5	1	2	1	2

Fuente. Kacen et al. (2002)

En la Figura 27, a manera de ejemplo, se muestra a uno de los Cromosomas de Máquinas al cual se le ha designado aleatoriamente las máquinas que se indican. Para propósitos didácticos las máquinas son representadas con la letra M, por ejemplo M1. Sin embargo, la asignación real no considera la letra M, es decir solo debe figurar el número entero 1.

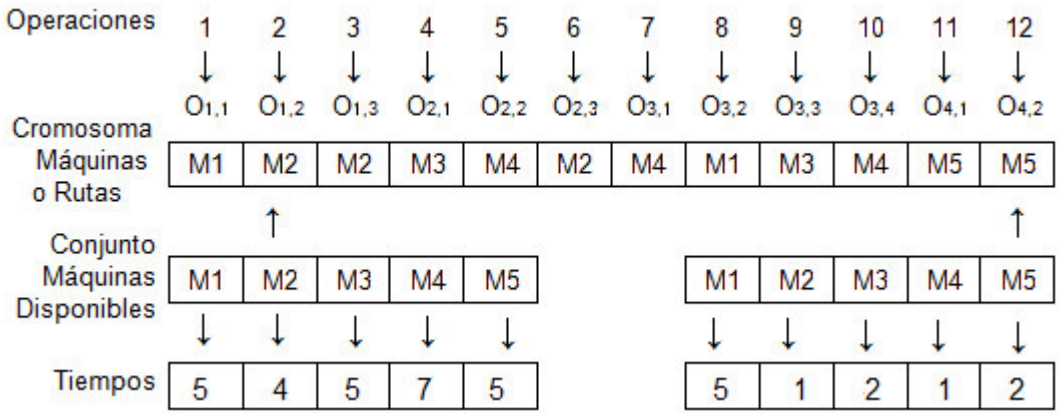


Figura 27. **Cromosoma de Máquinas con 12 Genes Arbitrarios**. *Fuente*. Elaboración propia

En cada columna existe una máquina asignada a la operación $O_{i,\ j}$ (*Job* i operación j del *Job*). Para el ejemplo mostrado, se tiene un total de doce (12) operaciones y a cada operación se le puede asignar una entre cinco máquinas (ver Tabla 8). Es decir, cada columna del Cromosoma de Máquinas puede tener a cualquiera de un conjunto de cinco máquinas disponibles y cada máquina se distingue de otras porque ejecuta la operación correspondiente en un tiempo T_i. Así, por ejemplo, a la operación 12 ($O_{4,2}$) se le ha designado la máquina 5 (M5) que a su vez le corresponde un tiempo de proceso (para esa operación) de 2 unidades de tiempo.

También, en el Cromosoma de Máquinas es posible observar la ruta o camino, a través de las máquinas, que debe seguir cada *Job* para su procesamiento. Por ejemplo, el *Job* 2 constituido por las operaciones $O_{2,1}$, $O_{2,2}$ y $O_{2,3}$, tiene que seguir la ruta de M3 a M4 y finalmente a M2. Por este

motivo, al cromosoma se le ha dado la denominación de Cromosoma de Máquinas o Cromosoma de Rutas.

3.8.2 Selección y Cruce en el Algoritmo Genético de Rutas

La Figura 28, muestra el proceso de selección y cruce. Los primeros cálculos que se realizan son para encontrar el número de sobrevivientes de la población (los cuales se cruzaran) y el número de cruces (apareamientos) entre los sobrevivientes. En cada cruce se generan dos descendientes o hijos. Así, por ejemplo, para una tasa de sobrevivientes de 20% de una población de 20 Cromosomas de Máquinas, los 4 más aptos sobreviven para cruzarse 8 veces y generar 16 descendientes que remplazan a la población.

Luego de haberse escogido a los Cromosomas de Máquinas más aptos, la técnica para seleccionarlos para cruce es de Ranking Lineal. Por ejemplo, para los cuatro Cromosomas de Máquinas sobrevivientes, se genera una suerte de “cupones” en un vector: cuponesxpadremq = [4 3 3 2 2 2 1 1 1 1], el Cromosoma de Máquinas 1 tiene la mayor probabilidad de ser elegido (4/10=40%), seguido del 2 (3/10=30%), el 3 (2/10=20%) y finalmente el 4(1/10=10%).

Luego, se generan dos números aleatorios que indican las posiciones, en el vector cuponesxpadre, de los padres seleccionados. Por ejemplo, aleatoriamente se selecionan las posiciones 5 y 2, que corresponde a los Cromosomas de Máquinas: 2 y 3 respectivamente. Los mismos padres pueden cruzarse varias veces generando siempre hijos diferentes. Puede ocurrir, que tanto el padre como la madre sean los mismos cromosomas, en lugar de evitarlo, se ha preferido no hacerlo para ahorrar tiempo de computación no afectando el desempeño del algoritmo.

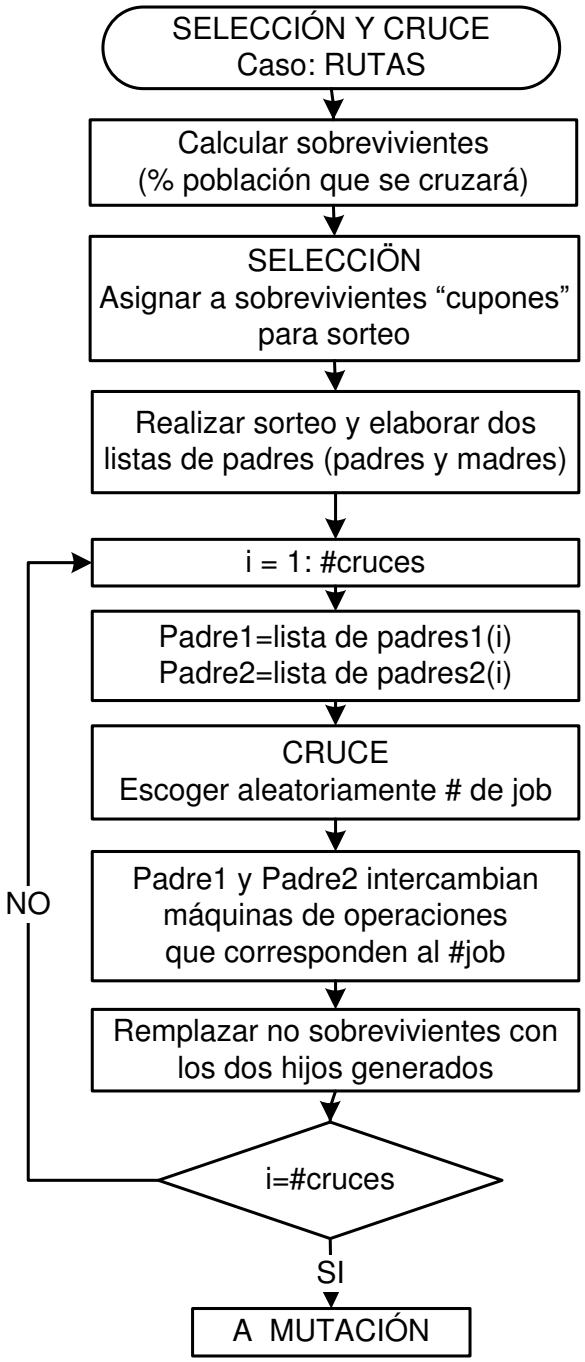


Figura 28. **Selección y Cruce en el Algoritmo Genético de Rutas**. *Fuente*. Elaboración propia

Para el caso del cruce, se ha optado por el cruce de dos puntos, pero a diferencia de tomar dos puntos aleatorios, en la propuesta, se encuentra aleatoriamente el punto de inicio de un *Job* y se ubica el punto donde finaliza. Luego, son intercambiadas las máquinas a que pertenecen a ese *Job*, generándose dos hijos por cada cruce. Este método propuesto ha dado mejores resultados que los dos puntos aleatorios para intercambio. En el ejemplo de la Figura 29, se observa el procedimiento, luego, de generarse aleatoriamente el número 2, entonces en ambos Cromosomas de Máquinas se intercambian los genes que pertenecen al *Job* 2.

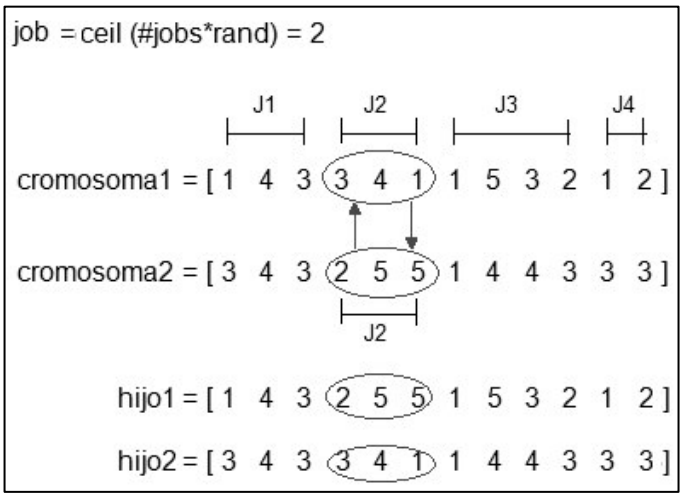


Figura 29. **Cruce en el Algoritmo Genético de Rutas**. *Fuente*. Elaboración propia

3.8.3 Mutación en el Algoritmo Genético de Rutas

Se propone tres tipos de mutación, ver Figura 30, una de las cuales se ejecuta aleatoriamente generando un número entre 0 y 1. La mutación, de mayor probabilidad de ejecutarse, remplaza una máquina de una operación (seleccionada aleatoriamente) por una de las máquinas más rápidas disponibles para esa operación. Las otras mutaciones, en cambio, en un caso reemplazan la máquina por una máquina de la operación seleccionada aleatoriamente y en el otro caso, por una máquina menos frecuente entre

todas las máquinas del Cromosoma de Máquinas. El número de mutaciones, que ha dado buenos resultados, es del 1% del tamaño de la población.

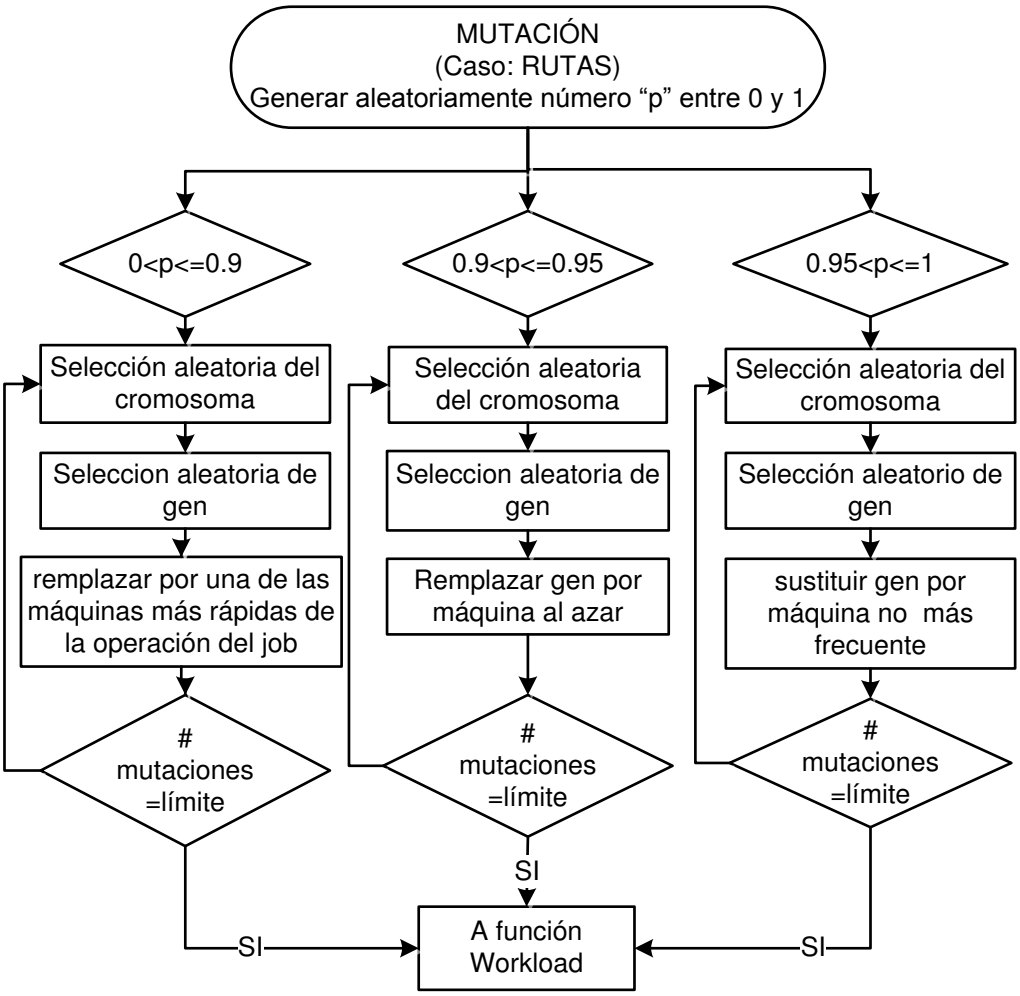


Figura 30. **Mutación en el Algoritmo Genético de Rutas**. *Fuente*. Elaboración propia

3.8.4 Evaluación en el Algoritmo Genético de Rutas

Se ha diseñado la Función Workload que evalúa en cada Cromosoma de Máquinas el *Maximum Workload* (W_M) y el *Total Workload* (W_T). El procedimiento se describe mediante la Figura 31, en donde el Cromosoma de Máquinas puede ser uno generado como para el sistema de fabricación representado en la Tabla 8. Para cada máquina se encuentra el *Job* y la operación a la que pertenece. Luego, con los datos del *Job*, la operación y la

máquina, se busca, en la Tabla 8, el tiempo que le toma a la máquina procesar la operación generando el vector de Tiempos.

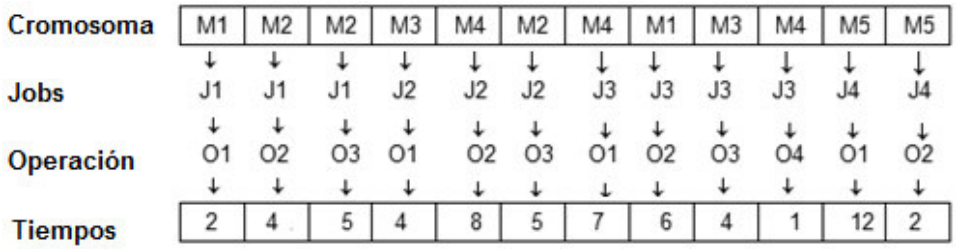


Figura 31. **Identificando Tiempos de Proceso para Calcular *Workload***. *Fuente*. Elaboración propia

Luego, con los vectores CROMOSOMA y TIEMPOS se calcula:

- Workload de M1 = 2 + 6 = 8
- Workload de M2 = 4 + 5 + 5 =14
- Workload de M3 = 4 + 4 = 8
- Workload de M4 = 8 + 7 + 1 = 16
- Workload de M5 = 12 + 2 = 14

Siendo, por consiguiente, el *Maximum Workload* (W_M) de 16 y el *Total Workload* (W_T) de 60 (igual a la suma). En base a lo explicado, la Figura 32 muestra el algoritmo de la Función *Workload.*

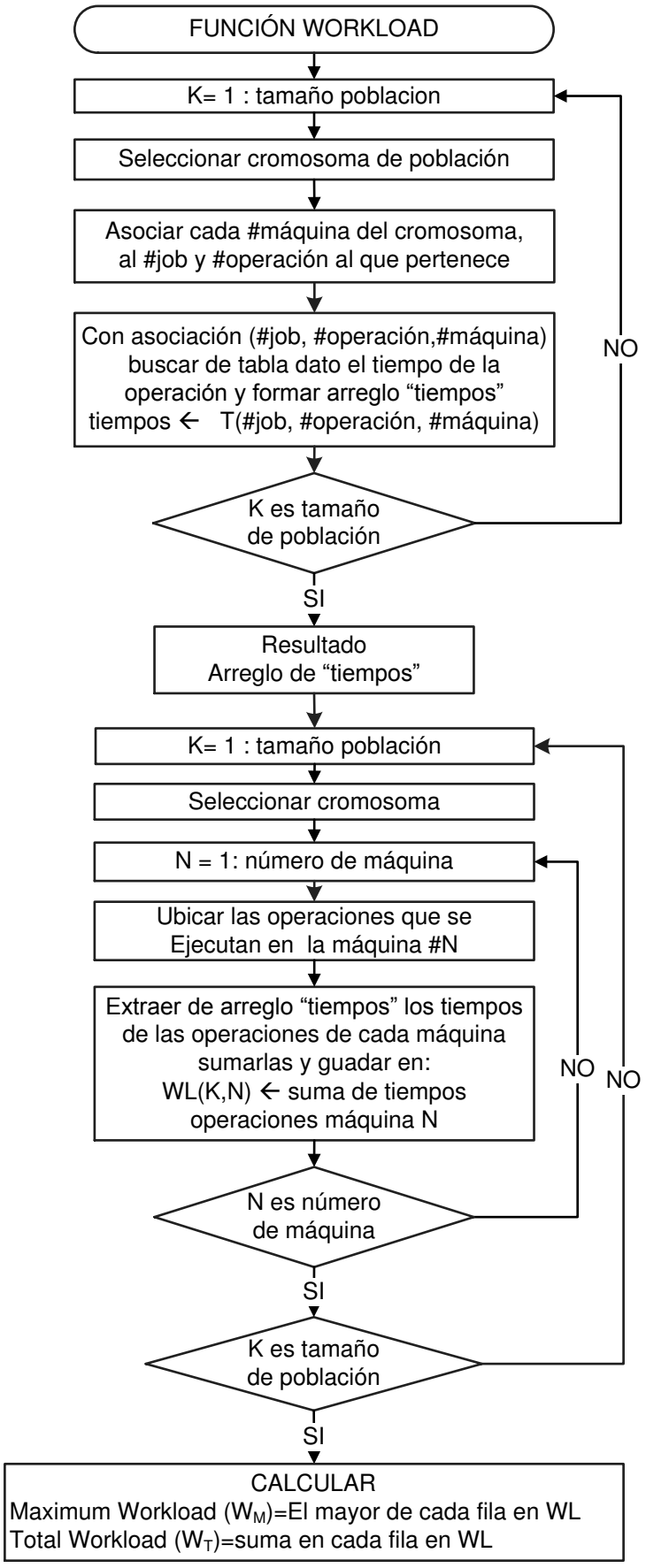


Figura 32. **Función *Workload***. *Fuente*. Elaboración propia

3.9 Diseño del Algoritmo Genético de Secuencias

El Algoritmo Genético de Secuencias, soluciona el Subproblema de Secuenciación, es el encargado de secuenciar u ordenar las operaciones que concurren a las máquinas para su procesamiento. Esto quiere decir, por ejemplo, que cuando dos o más operaciones están destinadas a concurrir a la misma máquina, la secuencia establece cuál de ellas se procesa primero, cuál segunda, etc. Una óptima secuencia de operaciones tiene impacto en el tiempo en que se debe concluir el procesamiento de todas las operaciones, por este motivo, el algoritmo tiene que encontrar la secuencia que tenga un mínimo *Makespan* (C_M).

En la Figura 33, se muestra el Diagrama de Flujo del Algoritmo Genético de Secuencias, se ejecuta inmediatamente después del Algoritmo Genético de Rutas, quien designó para cada operación una máquina minimizando W_M y W_T.

El Algoritmo Genético de Secuencias genera una población inicial de individuos a los cuales se les ha denominado Cromosomas de Secuencia de Operaciones o simplemente Cromosomas de Secuencias, que contienen números aleatorios y permutados representando operaciones. Luego, se corrige la precedencia de las operaciones, se calcula el Makespan y se ordena la población de menor a mayor, estos dos últimos pasos no se acostumbran hacer antes de cruzar y mutar, pero en la propuesta, solo se realiza al inicio con la posibilidad de identificar si el mejor Cromosoma de Secuencias pueda satisfacer el umbral de C_M, siempre y cuando éste sea un criterio de parada. El Algoritmo Genético de Secuencias puede detenerse cuando: C_M es igual o menor que su umbral, se alcanza el máximo número de generaciones o el máximo número de reiteraciones consecutivas del mejor C_M. La secuencia óptima generada, luego, es convertida a un Diagrama de Gantt en donde se pueden validar objetivamente los resultados, también los datos son exportados a un archivo de Excel.

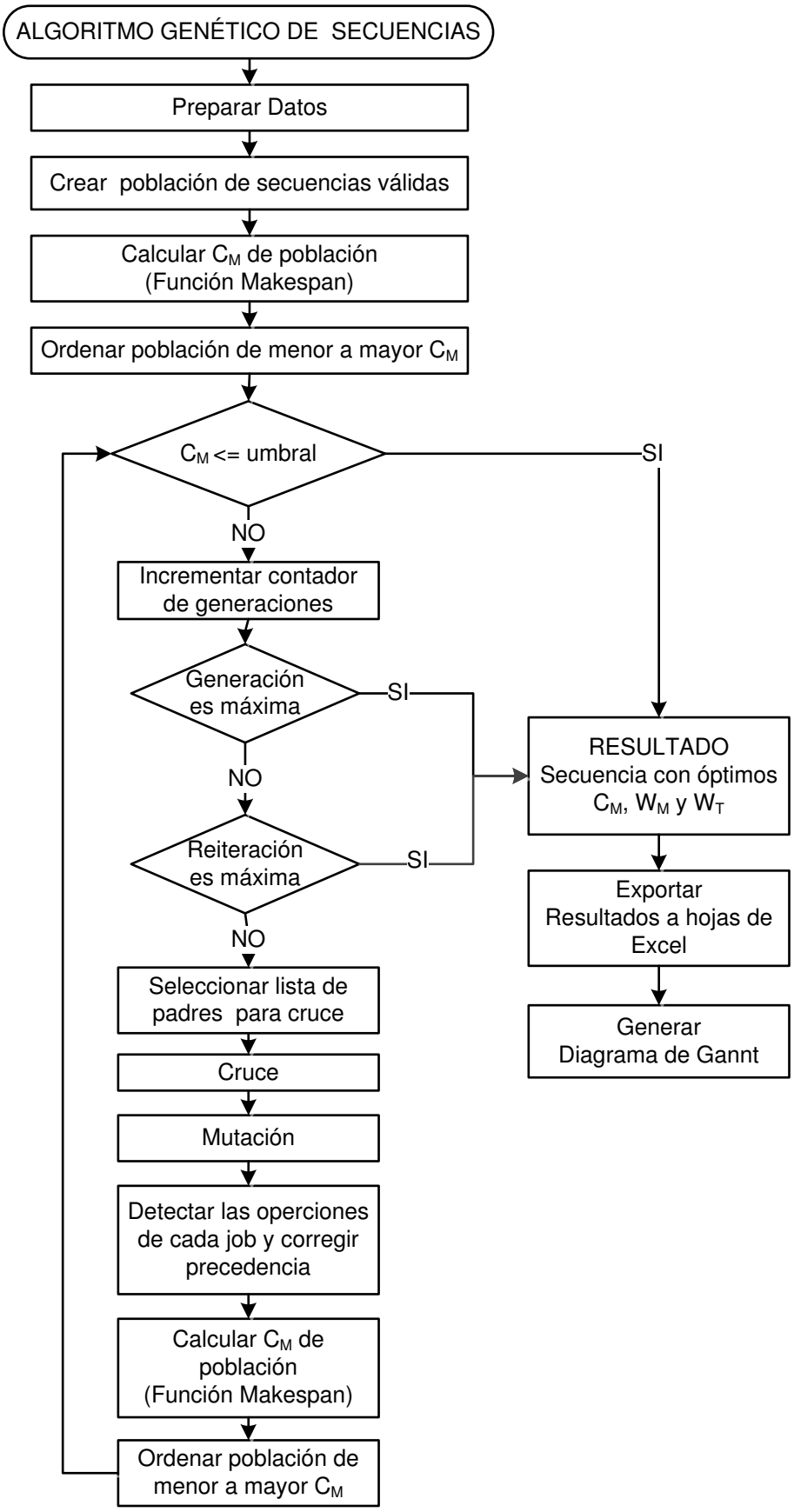


Figura 33. **Algoritmo Genético de Secuencias**. *Fuente*. Elaboración propia.

3.9.1 Cromosoma en el Algoritmo Genético de Secuencias

En la Figura 34, se muestra un Cromosoma de Secuencias de Operaciones o simplemente denominado Cromosoma de Secuencias, en donde cada número representa a una operación del *FJSSP*, como el de la Tabla 8. Cada operación se relaciona con un *Job* (Ji), una máquina (Mi) que la ejecuta y un tiempo de ejecución (Ti). Se muestra un total de doce operaciones, tres operaciones (1, 2 ,3) del *Job*1 (J1), tres (4, 5, 6) del *Job*2 (J2), cuatro (7, 8, 9, 10) del *Job*3 (J3) y dos (11, 12) del *Job*4 (J4). Como se observa el orden de precedencia de las operaciones de cada *Job* es respetado.

Además, se puede observar de la Figura 34, que la secuencia mostrada obliga a respetar un orden o secuencia para el procesamiento de las operaciones. Por ejemplo, las operaciones 5, 7 y 10 que deben procesarse en la máquina 4 (M4). Lo deben hacer en el siguiente orden: primero debe procesarse la operación 7, luego a la operación 5 y finalmente la operación 10. Claro está, que no necesariamente se procesará una inmediatamente después de terminada la otra, porque puede ocurrir que después de la operación 5, M4 tenga que esperar, sin hacer nada, hasta la llegada de la operación 10; esto porque pudo ser que la operación 9 en M3 no haya concluido antes.

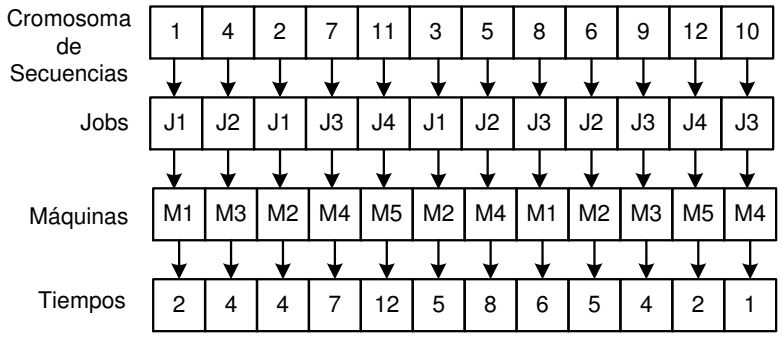


Figura 34. **Cromosoma de Secuencias y Variables Relacionadas**. *Fuente*. Elaboración propia

Todos los cromosomas de la población son como el mostrado en la Figura 34. Por otra parte, para respetar el orden de precedencia de las operaciones de en cada *Job*, esta secuencia debe ser corregida previamente a su evaluación.

En otras propuestas de la literatura no se utiliza números permutados para el Cromosoma de Secuencias y todas las operaciones del mismo *Job* son identificadas con el mismo número. En este trabajo se prefiere utilizar la secuencia permutada porque facilita realizar los cálculos del *Makespan*.

3.9.2 Selección y Cruce en el Algoritmo Genético de Secuencias

La Figura 35, muestra el proceso de selección y cruce, la selección se realiza del mismo modo como en el caso del Algoritmo Genético de Rutas. En el caso de cruce, se realiza como lo explicado en ítem 2.3.6.4 para el caso del cruce denominado Ciclo de Cruce (CX). Pero, en este caso los genes iniciales de cruce que se intercambian entre los padres son seleccionados aleatoriamente. Luego, el gen colisionante se intercambia con el del otro padre. El proceso se repite hasta que en los cromosomas no existan genes repetidos, de esta manera se generan dos hijos por cada par de padres.

3.9.3 Mutación y Corrección en el Algoritmo Genético de Secuencias

La mutación se realiza tal como fue expuesto en ítem 2.3.6.5. Es decir, se selecciona al azar un Cromosoma de Secuencias. Luego también al azar se seleccionan dos de sus posiciones y estos se intercambian. El número de mutaciones para la mayoría de casos experimentados, dando buenos resultados, ha sido de 0.1% del tamaño de la población.

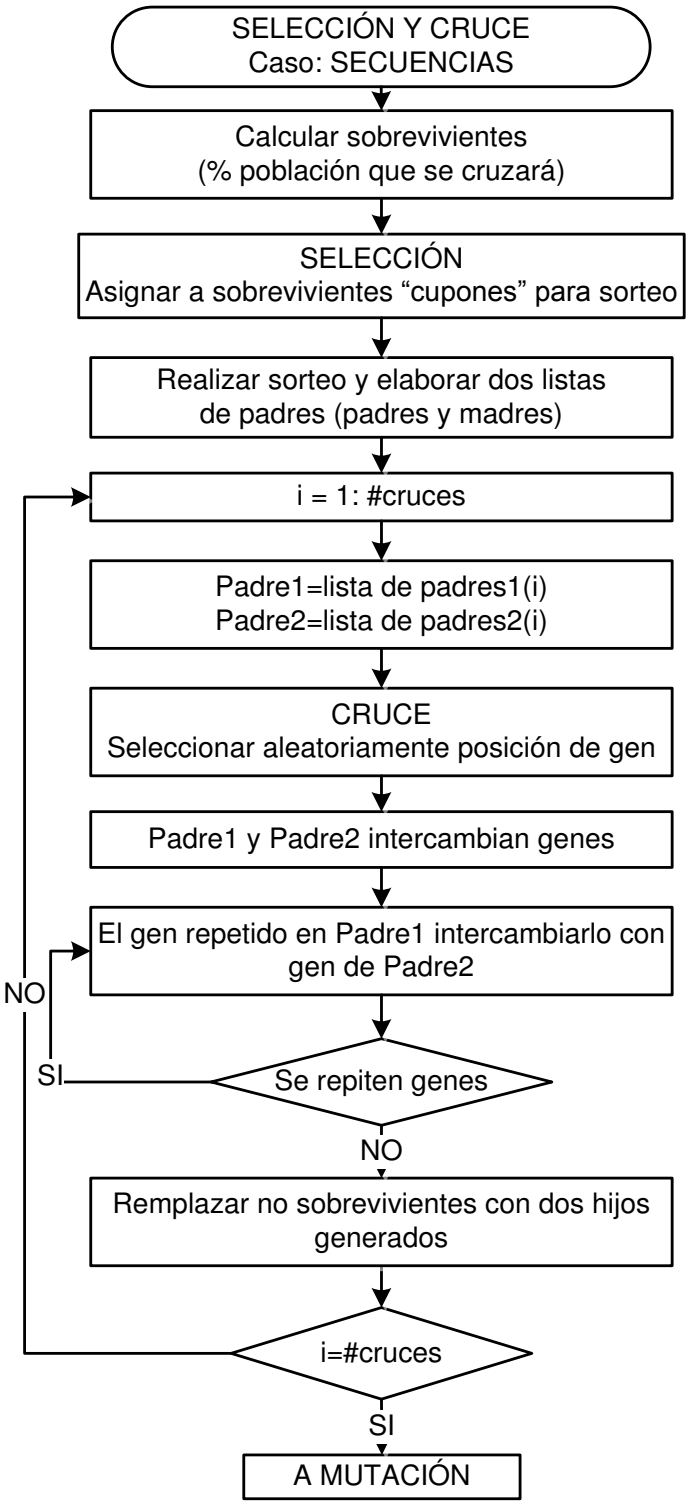


Figura 35 **Selección y Cruce en el Algoritmo Genético de Secuencias**. *Fuente*. Elaboración propia.

Luego, de pasar por la mutación se deben corregir la secuencia de las operaciones para respetar la precedencia de las operaciones. Suponiendo que los *Jobs* y operaciones son como los de la Tabla 8, el ejemplo de la Figura 36, muestra un Cromosoma de Secuencias antes y después de la corrección. Cada operación en la secuencia está asociada con el *Job* a que pertenece. Antes de la corrección se puede observar que las operaciones del *Job* 1 están en el orden: 2→3→ 1 y las operaciones del *Job* 4 están en el orden: 12→11. Después de la corrección, manteniendo sus posiciones relativas dentro del Cromosoma de Secuencias, el orden que tienen es el correcto. Es decir, para el *Job*1 se obtiene: 1→2→3 y para el *Job* 4: 11→12.

Por otro lugar, considerando alta la probabilidad, que después del cruce y mutación, la secuencia no respete la precedencia, entonces, en lugar de detectar solo a los afectados para corregirlos, se aplicó la corrección a todos los miembros de la población. La Figura 37 muestra el algoritmo explicado.

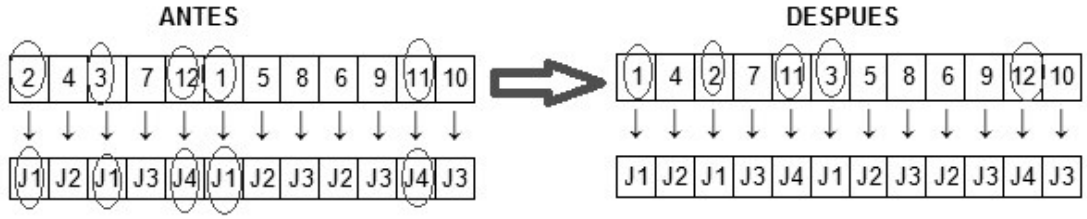


Figura 36. **Corrección de Secuencias en el Cromosoma**. *Fuente*. Elaboración propia

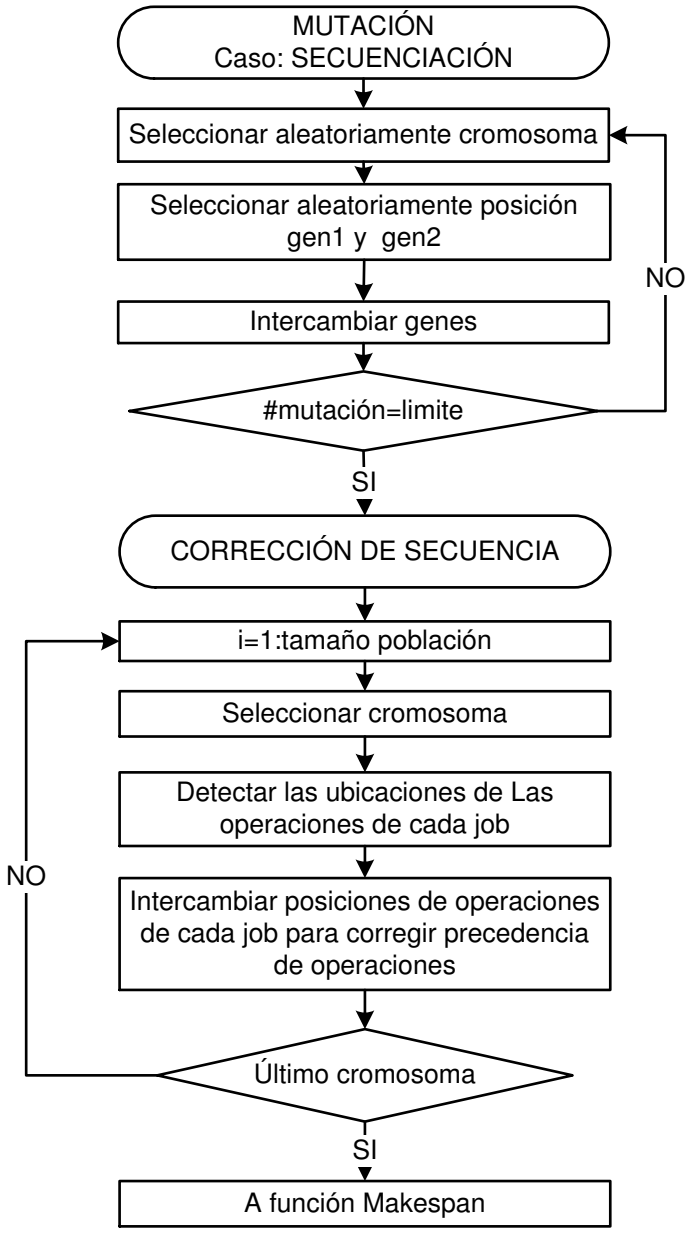


Figura 37. **Mutación y Corrección de Secuencias**. *Fuente*. Elaboración propia.

3.9.4 Evaluación en el Algoritmo Genético de Secuencias

Se ha diseñado la Función *Makespan* que calcula o evalúa del Cromosoma de Secuencias el tiempo en que finaliza el procesamiento de todos los *Jobs* en el sistema de fabricación. La información con los tiempos de procesamiento se van registrando en un matriz denominada TMK. Previamente se construyen tres matrices MAQ, JOB y TI, ellos registran la máquina, el *Job* y el tiempo de proceso que se corresponde con cada operación del Cromosoma de Secuencias, de acuerdo a lo que fue explicado en el ítem 3.9.1 con la Figura 34.

Para lograr lo anterior, se utiliza al Cromosoma de Secuencias y a los vectores: “máquinas” (contiene las máquinas asignadas a las operaciones del 1 al 12, en ese orden), “tiempos” (contiene los tiempos en relación al vector máquinas) y “listaJob*s*xopera” (contiene el número de *Job* que se corresponde con cada operación). Así por ejemplo, si el primer elemento del cromosoma es 4, entonces, indica que estamos en la operación 4, por lo que el cuarto elemento del vector “maquinas” lo copiamos al primer elemento del arreglo MAQ, el cuarto elemento del vector *Jobs*xoperacion lo copiamos al primer elemento de TR y el cuarto elemento del vector “tiempos” lo copiamos al primer elemento de TI, de manera, similar se van encontrando los demás elementos de MAQ, JOB y TI.

Para construir el vector TMK, se utilizan al Cromosoma de Secuencias y los arreglos MAQ, JOB y TI. Los cálculos se realizan examinando cada operación de izquierda a derecha del cromosoma. El proceso brevemente se muestra en la Figura 38, así para la primera operación del cromosoma que es la operación 4, en MQ tenemos la máquina 1, la cual si se revisan las celdas de más a la izquierda no se encuentra la misma máquina (evidentemente porque es la primera operación) y en JOB el *Job* 2 tampoco está antes (por la misma razón). Entonces, el tiempo de proceso, tiene un tiempo inicial nulo más el tiempo de proceso de la operación (TMK=0+TI=2).

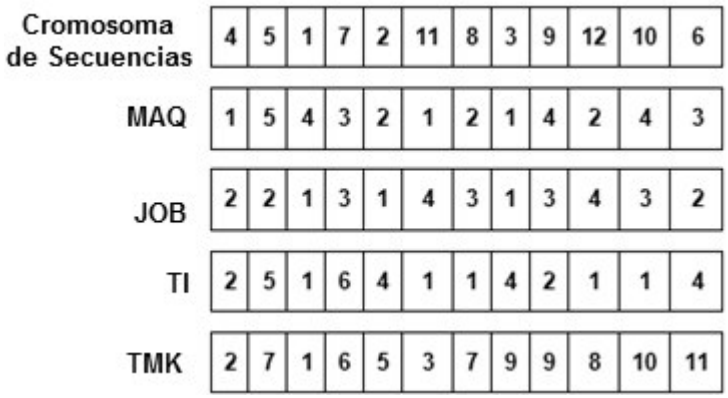


Figura 38. **Procedimiento para Calcular TMK y el *Makespan***. *Fuente.* Elaboración propia

En cambio, por ejemplo, cuando se llega a la operación 12, se observa en MAQ que la máquina 2 ya fue utilizada antes y en JOB se observa que la operación pertenece al *Job* 4 que también ocurrió antes. Entonces el tiempo transcurrido, tiene como punto de partida el mayor tiempo entre lo contabilizado para la máquina 2 (TMK=7) y lo contabilizado para *Job* 4 (TMK=3) más el tiempo de proceso de la operación (mayor (3,7) + TI) dando como resultado 8. El proceso continua, hasta haberse calculado TMK para la operación 6, que es la última.

Por tanto, el *Makespan* (C_M) del Cromosoma de Secuencias es el mayor elemento del vector TMK, el cual en este ejemplo es de 11.

Basado en el procedimiento descrito, la Figura 39, muestra el Diagrama de Flujo de la Función *Makespan*.

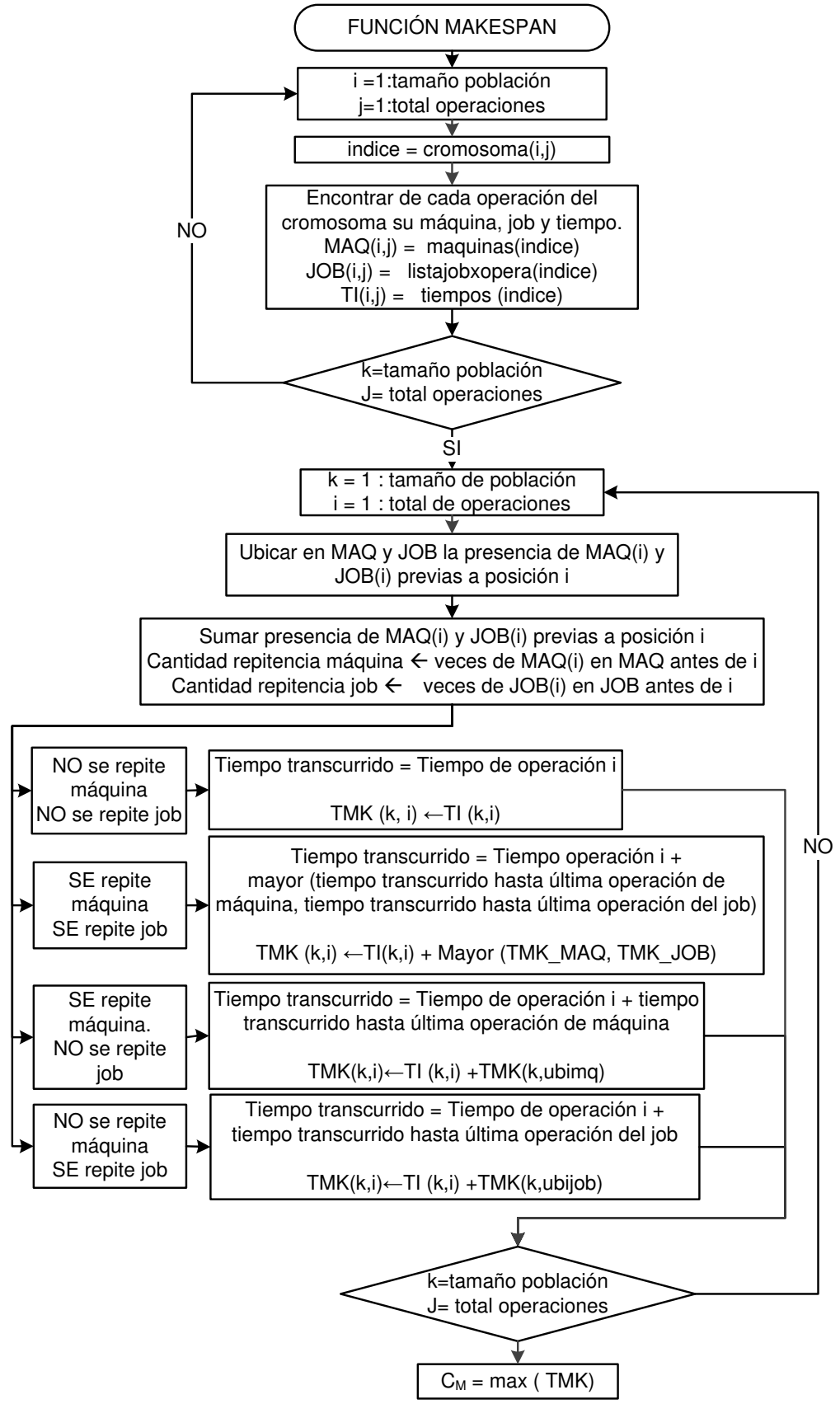


Figura 39. **Función *Makespan***. *Fuente.* Elaboración propia

3.10 Diseño del Algoritmo de Presentación de Resultados en Diagramas de Gantt

Luego de solucionar el Subproplema de Enrutamiento y el Subproblema de Secuenciación el resultado es una secuencia ordenada de operaciones, en el cual se ha minimizado los criterios de optimización de W_T, W_M y C_M. Para facilitar visualmente la verificación de los resultados, el programa construye un Diagrama de Gantt, el cual presenta en el eje de las ordenadas a las máquinas y en el eje de las abscisas el tiempo transcurrido. También en el Diagrama se presentan los resultados numéricos de las variables.

Antes de generar el Diagrama de Gantt se necesita preparar los datos, para ello teniendo como dato el mejor Cromosoma de Secuencias se obtienen los vectores MAQ, JOB, TI, TMK y OPE. Los vectores MAQ, JOB, TI se obtienen de acuerdo al procedimiento que se explicó antes. Para obtener TMK se invoca a la Función *Makespan* quien como respuesta devuelve a ese vector así como a C_M. El vector OPE se construye en el mismo acto que se construyen MAQ, JOB y TI y contiene el número de la operación de cada *Job*. Supongamos que los resultados del Subproblema de Enrutamiento son:

Operaciones	1	2	3	4	5	6	7	8	9	10	11	12
Máquinas	4	2	1	1	5	3	3	2	4	4	1	2
Tiempos	1	4	4	2	5	4	6	1	2	1	1	1

El resultado del Subproblema de Secuenciación es:

Secuencia operaciones	4	5	1	7	2	11	8	3	9	12	10	6

Entonces los vectores que se generan son los mostrados en la Figura 40.

Figura 40. **Vectores Generados para Diagrama de Gantt**. *Fuente*. Elaboración propia

La Figura 41, muestra el algoritmo para generar el Diagrama de Gantt. El bloque principal del algoritmo es el correspondiente al ploteo de los segmentos de recta que representan las operaciones de los *Jobs*. La acción de ploteo de este bloque se repite a continuación:

plot([(TMK(i)-TI(i));TMK(i)],[MAQ(i);MAQ(i)])

Donde:
I: toma el valor de 1 hasta el número de operaciones totales del problema.
(TMK (i)-TI (i)): es el punto de inicio del segmento de recta, medido en el eje de tiempos.
TMK (i): es el punto final del segmento de recta, medido en el eje de tiempos. [MAQ (i); MAQ (i)]: es respectivamente el punto inicial y final del segmento de recta sobre el eje de ordenadas (es un punto MAQ (i)).

Así por ejemplo para i= 5
TMK (5)-TI (5) = 5-4=1
TMK (5) = 5
MAQ (5) = 2

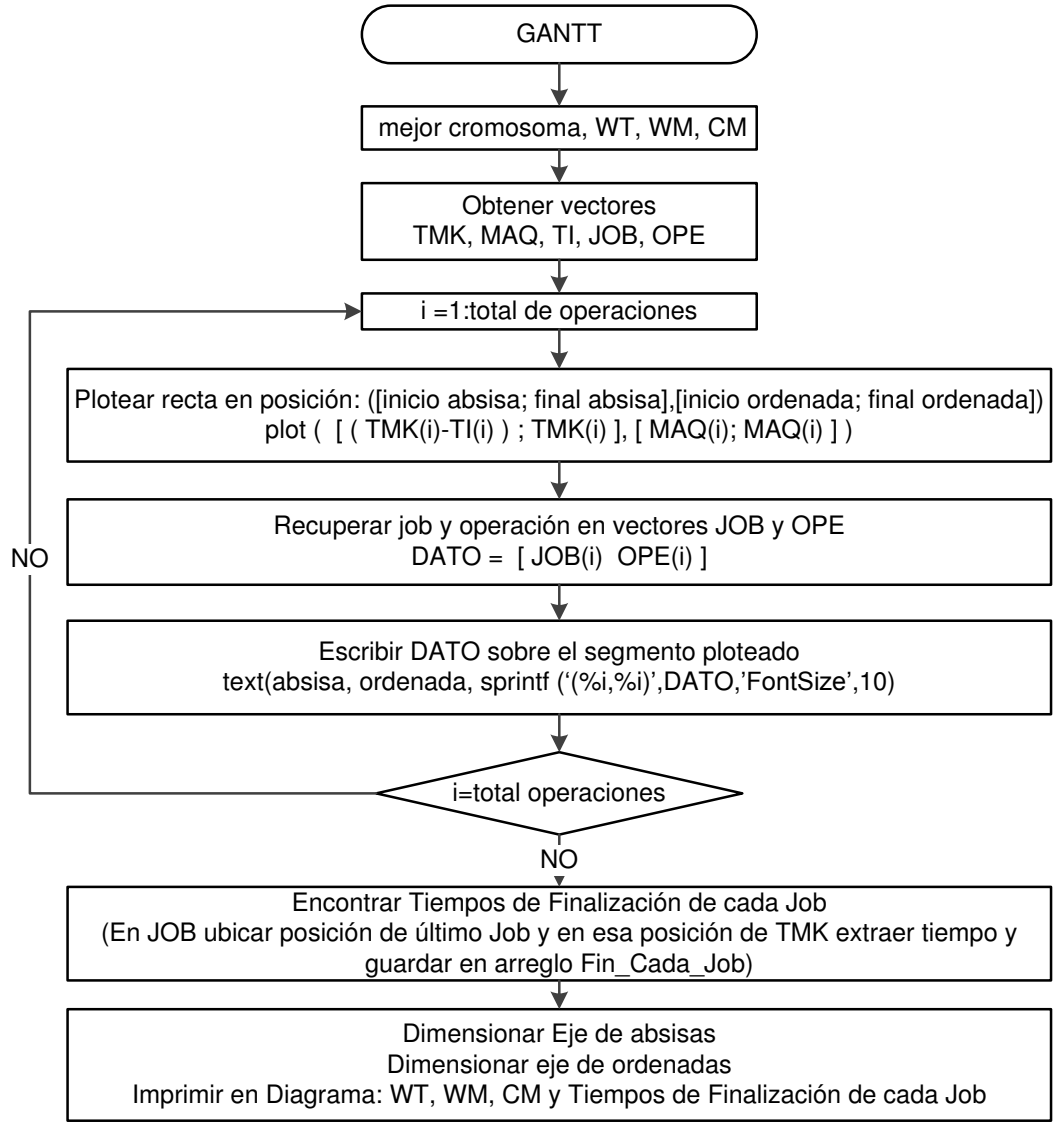


Figura 41. **Algoritmo Gantt**. *Fuente*. Elaboración propia

También para i=5, se puede encontrar en el vector OPE la operación 2 y en el vector JOB el *Job* 1. Por tanto, el ploteo del segmento de recta para i = 5 tiene entonces la forma aproximada de la Figura 42.

Figura 42. **Ejemplo de Ploteo para i=5**. *Fuente*: Elaboración propia

De esta manera se grafican sobre el mismo Diagrama los otros segmentos de rectas que representan a las demás operaciones.

Otro cálculo que merece destacarse, para su presentación en el Diagrama de Gantt, es el relativo a los tiempos en que finaliza cada *Job*, para ello se utilizan los vectores JOB y TMK. Se ubica en JOB cada *Job* uno por uno. Para el *Job* que está siendo analizado, se encuentra la posición de su última aparición en el vector. Luego en esa misma posición se busca en TMK el tiempo utilizado por ese Job y se guarda en el arreglo Fin_cada_Job. Esta información que se presenta en el Diagrama de Gantt puede ser de utilidad para establecer fechas de compromiso de finalización de los Jobs.

Las demás bloques de la Figura 41, se utilizan para la ubicación de los títulos y variables sobre el Diagrama, se ha tenido en cuenta especificar las coordenadas de los títulos dependientes con la escala máxima de la abscisa, esto con la intención de obtener Diagramas con una presentación adecuada de los títulos independiente del tamaño del problema.

3.11 Metodología para Probar el Funcionamiento, la Eficacia y la Eficiencia de los Algoritmos

Como se mencionó anteriormente, los algoritmos propuestos demostraran su eficacia y eficiencia solucionando casos de sistemas de fabricación tipo *FJSS* utilizados también por otros investigadores. Estos casos han sido planteados por Kacem et al. (2002) y Kacem et al. (2002a) y solucionados por diversos investigadores para probar sus algoritmos. Es así, que los resultados de *Maximum Workload* (W_M), *Total Workload* (W_T), *Makespan* (C_M), así como el tiempo de ejecución de los algoritmos, han sido comparados (Capítulo 4). La Eficacia será evaluada al comparar los resultados (W_M, W_T y C_M) y la Eficiencia por la velocidad con que se obtiene los resultados en cada caso solucionado.

Cada caso de *FJSSP* se presenta en tablas que tienen las características que fueron descritas en el ítem 3.7 (con la Figura 23). A continuación son presentados los casos de *FJSSP* planteados por Kacem et al. (2002) y Kacem et al. (2002a).

En la Tabla 9, se muestra los datos para el caso *FJSSP* correspondiente a un sistema de fabricación de cuatro *Jobs* (J1, J_2, J_3, J_4). En donde, J_1 está constituido de tres operaciones ($O_{1,1}$, $O_{1,2}$, $O_{1,3}$), J_2 de tres operaciones ($O_{2,1}$, $O_{2,,2}$, $O_{2,3}$), J_3 de cuatro operaciones ($O_{2,1}$, $O_{2,,2}$, $O_{2,3}$, $O_{3,4}$) y finalmente J_4 de dos operaciones ($O_{4,1}$, $O_{4,2}$), totalizando doce operaciones. Cada operación puede procesarse en una de las cinco máquinas disponibles (M_1, M_2, M_3, M_4, M_5), los números en cada celda corresponde a los tiempos de procesamiento de la operación en la máquina indicada**.** Abreviadamente, este caso, corresponde a uno de 4 *Jobs* x 5 máquinas de 12 operaciones y Flexibilidad Total (todas las operaciones tienen la opción de ser destinadas a procesarse en cualquiera de las cinco máquinas).

Tabla 9.
Caso FJSSP 4x5 con 12 Operaciones (Flexibilidad Total)

JOB	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5
J_1	$O_{1,1}$	2	5	4	1	2
	$O_{1,2}$	5	4	5	7	5
	$O_{1,3}$	4	5	5	4	5
J_2	$O_{2,1}$	2	5	4	7	8
	$O_{2,2}$	5	6	9	8	5
	$O_{2,3}$	4	5	4	54	5
J_3	$O_{3,1}$	9	8	6	7	9
	$O_{3,2}$	6	1	2	5	4
	$O_{3,3}$	2	5	4	2	4
	$O_{3,4}$	4	5	2	1	5
J_4	$O_{4,1}$	1	5	2	4	12
	$O_{4,2}$	5	1	2	1	2

Fuente. Kacen et al. (2002)

En la Tabla 10, se muestra los datos para el caso de un *FJSSP* de 8 *Jobs* x 8 máquinas, pero es de Flexibilidad Parcial, porque, no todas las operaciones pueden ser procesadas en cualquiera de las ocho máquinas. La imposibilidad se indica con "- ". Para el manejo computacional, a esas ubicaciones, se les asigna números muy grandes, de tal modo que si el Algoritmo Genético de Rutas está buscando la máquina más rápida, ésta es descartada. En otro caso, en que el Algoritmo Genético de Rutas, por selección arbitraria, eligiera esa máquina, entonces la evaluación del Cromosoma de Máquinas resultará en un número tan grande que igualmente se descartará en el proceso. Las Tablas 11, 12 y 13 son los otros casos, van avanzando en complejidad, hasta el último en donde se tiene un *FJSSP* de quince *Jobs* (15), diez (10) máquinas de Flexibilidad Total de cincuenta y seis operaciones (56).

También para mayor ilustración, sin la intención de comparar resultados, en el Capítulo 5 se han solucionado casos del mundo real encontrados en la literatura.

Tabla 10.
Caso FJSSP 8x8 con 27 Operaciones (Flexibilidad Parcial)

JOB	O_{ij}	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
J_1	$O_{1,1}$	5	3	5	3	3	-	10	9
	$O_{1,2}$	10	-	5	8	3	9	9	6
	$O_{1,3}$	-	10	-	5	6	2	4	5
J_2	$O_{2,1}$	5	7	3	9	8	-	9	-
	$O_{2,2}$	-	8	5	2	6	7	10	9
	$O_{2,3}$	-	10	-	5	6	4	1	7
	$O_{2,4}$	10	8	9	6	4	7	-	-
J_3	$O_{3,1}$	10	-	-	7	6	5	2	4
	$O_{3,2}$	-	10	6	4	8	9	10	-
	$O_{3,3}$	1	4	5	6	-	10	-	7
J_4	$O_{4,1}$	3	1	6	5	9	7	8	4
	$O_{4,2}$	12	11	7	8	10	5	6	9
	$O_{4,3}$	4	6	2	10	3	9	5	7
J_5	$O_{5,1}$	3	6	7	8	9	-	10	-
	$O_{5,2}$	10	-	7	4	9	8	6	-
	$O_{5,3}$	-	9	8	7	4	2	7	-
	$O_{5,4}$	11	9	-	6	7	5	3	6
J_6	$O_{6,1}$	6	7	1	4	6	9	-	10
	$O_{6,2}$	11	-	9	9	9	7	6	4
	$O_{6,3}$	10	5	9	10	11	-	10	-
J_7	$O_{7,1}$	5	4	2	6	7	-	10	-
	$O_{7,2}$	-	9	-	9	11	9	10	5
	$O_{7,3}$	-	8	9	3	8	6	-	10
J_8	$O_{8,1}$	2	8	5	9	-	4	-	10
	$O_{8,2}$	7	4	7	8	9	-	10	-
	$O_{8,3}$	9	9	-	8	5	6	7	1
	$O_{8,4}$	9	-	3	7	1	5	8	-

Fuente. Kacen et al. (2002)

Tabla 11.
Caso FJSSP 10x7 con 29 Operaciones (Flexibilidad Total)

JOB	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5	M_6	M_7
J_1	$O_{1,1}$	1	4	6	9	3	5	2
	$O_{1,2}$	8	9	5	4	1	1	3
	$O_{1,3}$	4	8	10	4	11	4	3
J_2	$O_{2,1}$	6	9	8	6	5	10	3
	$O_{2,2}$	2	10	4	5	9	8	4
J_3	$O_{3,1}$	15	4	8	4	8	7	1
	$O_{3,2}$	9	6	1	10	7	1	6
	$O_{3,3}$	11	2	7	5	2	3	14
J_4	$O_{4,1}$	2	8	5	8	9	4	3
	$O_{4,2}$	5	3	8	1	9	3	6
	$O_{4,3}$	1	2	6	4	1	7	2
J_5	$O_{5,1}$	7	1	8	5	4	3	9
	$O_{5,2}$	2	4	5	10	6	4	9
	$O_{5,3}$	5	1	7	1	6	6	2
J_6	$O_{6,1}$	8	7	4	56	9	8	4
	$O_{6,2}$	5	14	1	9	6	5	8
	$O_{6,3}$	3	5	2	5	4	5	7
J_7	$O_{7,1}$	5	6	3	6	5	15	2
	$O_{7,2}$	6	5	4	9	5	4	3
	$O_{7,3}$	9	8	2	8	6	1	7
J_8	$O_{8,1}$	6	1	4	1	10	4	3
	$O_{8,2}$	11	13	9	8	9	10	8
	$O_{8,3}$	4	2	7	8	3	10	7
J_9	$O_{9,1}$	12	5	4	5	4	5	5
	$O_{9,2}$	4	2	15	99	4	7	3
	$O_{9,3}$	9		11	2		4	2
J_{10}	$O_{10,1}$	9	4	13	10	7	6	8
	$O_{10,2}$	4	3	25	3	8	1	2
	$O_{10,3}$	1	2	6	11	13	3	5

Fuente. Kacen et al. (2002)

Tabla 12
Caso FJSSP 10x10 con 30 Operaciones (Flexibilidad Total)

JOB	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
	$O_{1,1}$	1	4	6	9	3	5	2	8	9	5
J_1	$O_{1,2}$	4	1	1	3	4	8	10	4	11	4
	$O_{1,3}$	3	2	5	1	5	6	9	5	10	3
	$O_{2,1}$	2	10	4	5	9	8	4	15	8	4
J_2	$O_{2,2}$	4	8	7	1	9	6	1	10	7	1
	$O_{2,3}$	6	11	2	7	5	3	5	14	9	2
	$O_{3,1}$	8	5	8	9	4	3	5	3	8	1
J_3	$O_{3,2}$	9	3	6	1	2	6	4	1	7	2
	$O_{3,3}$	7	1	8	5	4	9	1	2	3	4
	$O_{4,1}$	5	10	6	4	9	5	1	7	1	6
J_4	$O_{4,2}$	4	2	3	8	7	4	6	9	8	4
	$O_{4,3}$	7	3	12	1	6	5	8	3	5	2
	$O_{5,1}$	7	10	4	5	6	3	5	15	2	6
J_5	$O_{5,2}$	5	6	3	9	8	2	8	6	1	7
	$O_{5,3}$	6	1	4	1	10	4	3	11	13	9
	$O_{6,1}$	8	9	10	8	4	2	7	8	3	10
J_6	$O_{6,2}$	7	3	12	5	4	3	6	9	2	15
	$O_{6,3}$	4	7	3	6	3	4	1	5	1	11
	$O_{7,1}$	1	7	8	3	4	9	4	13	10	7
J_7	$O_{7,2}$	3	8	1	2	3	6	11	2	13	3
	$O_{7,3}$	5	4	2	1	2	1	8	14	5	7
	$O_{8,1}$	5	7	11	3	2	9	8	5	12	8
J_8	$O_{8,2}$	8	3	10	7	5	3	4	6	8	4
	$O_{8,3}$	6	2	13	5	4	3	5	7	9	5
	$O_{9,1}$	3	9	1	3	8	1	6	7	5	4
J_9	$O_{9,2}$	4	6	2	5	7	3	1	9	6	7
	$O_{9,3}$	8	5	4	8	6	1	2	3	10	12
	$O_{10,1}$	4	3	1	6	7	1	2	6	20	6
J_{10}	$O_{10,2}$	3	1	8	1	9	4	1	4	17	15
	$O_{10,3}$	9	2	4	2	3	5	2	4	10	23

Fuente. Kacen et al. (2002)

Tabla 13
Caso FJSSP 15x10 con 56 Operaciones (Flexibilidad Total)

JOB	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
J_1	$O_{1,1}$	1	4	6	9	3	5	2	8	9	4
	$O_{1,2}$	1	1	3	4	8	10	4	11	4	3
	$O_{1,3}$	2	5	1	5	6	9	5	10	3	2
	$O_{1,4}$	10	4	5	9	8	4	15	8	4	4
J_2	$O_{2,1}$	4	8	7	1	9	6	1	10	7	1
	$O_{2,2}$	6	11	2	7	5	3	5	14	9	2
	$O_{2,3}$	8	5	8	9	4	3	5	3	8	1
	$O_{2,4}$	9	3	6	1	2	6	4	1	7	2
J_3	$O_{3,1}$	7	1	8	5	4	9	1	2	3	4
	$O_{3,2}$	5	10	6	4	9	5	1	7	1	6
	$O_{3,3}$	4	2	3	8	7	4	6	9	8	4
	$O_{3,4}$	7	3	12	1	6	5	8	3	5	2
J_4	$O_{4,1}$	6	2	5	4	1	2	3	6	5	4
	$O_{4,2}$	8	5	7	4	1	2	36	5	8	5
	$O_{4,3}$	9	6	2	4	5	1	3	6	5	2
	$O_{4,4}$	11	4	5	6	2	7	5	4	2	1
J_5	$O_{5,1}$	6	9	2	3	5	8	7	4	1	2
	$O_{5,2}$	5	4	6	3	5	2	28	7	4	5
	$O_{5,3}$	6	2	4	3	6	5	2	4	7	9
	$O_{5,4}$	6	5	4	2	3	2	5	4	7	5
J_6	$O_{6,1}$	4	1	3	2	6	9	8	5	4	2
	$O_{6,2}$	1	3	6	5	4	7	5	4	6	5
J_7	$O_{7,1}$	1	4	2	5	3	6	9	8	5	4
	$O_{7,2}$	2	1	4	5	2	3	5	4	2	5
J_8	$O_{8,1}$	2	3	6	2	5	4	1	5	8	7
	$O_{8,2}$	4	5	6	2	3	5	4	1	2	5
	$O_{8,3}$	3	5	4	2	5	49	8	5	4	5
	$O_{8,4}$	1	2	36	5	2	3	6	4	11	2
J_9	$O_{9,1}$	6	3	2	22	44	11	10	23	5	1
	$O_{9,2}$	2	3	2	12	15	10	12	14	18	16
	$O_{9,3}$	20	17	12	5	9	6	4	7	5	6
	$O_{9,4}$	9	8	7	4	5	8	7	4	56	2
J_{10}	$O_{10,1}$	5	8	7	4	56	3	2	5	4	1
	$O_{10,2}$	2	5	6	9	8	5	4	2	5	4
	$O_{10,3}$	6	3	2	5	4	7	4	5	2	1
	$O_{10,4}$	3	2	5	6	5	8	7	4	5	2
J_{11}	$O_{11,1}$	1	2	3	6	5	2	1	4	2	1
	$O_{11,2}$	2	3	6	3	2	1	4	10	12	1
	$O_{11,3}$	3	6	2	5	8	4	6	3	2	5
	$O_{11,4}$	4	1	45	6	2	4	1	25	2	4
J_{12}	$O_{12,1}$	9	8	5	6	3	6	5	2	4	2
	$O_{12,2}$	5	8	9	5	4	75	63	6	5	21
	$O_{12,3}$	12	5	4	6	3	2	5	4	2	5
	$O_{12,4}$	8	7	9	5	6	3	2	5	8	4
J_{13}	$O_{13,1}$	4	2	5	6	8	5	6	4	6	2
	$O_{13,2}$	3	5	4	7	5	8	6	6	3	2
	$O_{13,3}$	5	4	5	8	5	4	6	5	4	2
	$O_{13,4}$	3	2	5	6	5	4	8	5	6	4
J_{14}	$O_{14,1}$	2	3	5	4	6	5	4	85	4	5
	$O_{14,2}$	6	2	4	5	8	6	5	4	2	6
	$O_{14,3}$	3	25	4	8	5	6	3	2	5	4
	$O_{14,4}$	8	5	6	4	2	3	6	8	5	4
J_{15}	$O_{15,1}$	2	5	6	8	5	6	3	2	5	4
	$O_{15,2}$	5	6	2	5	4	2	5	3	2	5
	$O_{15,3}$	4	5	2	3	5	2	8	4	7	5
	$O_{15,4}$	6	2	11	14	2	3	6	5	4	8

Fuente. Kacen et al. (2002)

CAPÍTULO 4: RESULTADOS Y DISCUSIÓN

El Algoritmo Genético de Rutas, el Algoritmo Genético de Secuencias y Algoritmo de Presentación de Resultados, tal como se indicó anteriormente, han sido ensamblados en un solo programa. En este sentido, en el ítem 4.1, se describen las características generales del Programa. Para, luego, al inicio del ítem 4.2, se realiza un resumen de los resultados numéricos, que por extenso se presentan en el Anexo B, cuando se ejecutó el programa para solucionar los casos de *FJSSP* planteados por Kacem et al. (2002) y Kacem et al. (2202a). Inmediatamente después, en el ítem 4.21, se realiza un análisis detallado del funcionamiento interno del programa, o mejor dicho, del Algoritmo Genético de Rutas y del Algoritmo Genético de Secuencias con el objeto de verificar el cumplimiento de las dos primeras hipótesis de la tesis. Finalmente, en el ítem 4.3 se describen los resultados que prueban la tercera hipótesis de la tesis.

4.1 Características Generales del Programa

El programa que soluciona los casos *FJSSP*, cuyo Diagrama de Flujo se explicó con la Figura 22, incluyen al Algoritmo Genético de Rutas, el cual soluciona el problema de Enrutamiento, al Algoritmo Genético de Secuencias, el cual soluciona el Subproblema de Secuenciación, y al Algoritmo de Presentación de Resultados, el cual presenta los datos en Diagrama de Gantt. También se incluyen, líneas de código para exportar los datos numéricos a un archivo de Excel y líneas de código para pedir los datos al usuario. El Programa ha sido denominado *Multiobjective Flexible Job Shop Scheduling Problem*, su codificación se ha realizado en el

Lenguaje matemático de Matlab y los principales detalles del código se presentan en el Anexo A.

El programa puede correr más de una vez para entregar varias soluciones óptimas de un mismo problema, es decir, todos los bloques de la Figura 22 se pueden reiterar más de una vez para tener varias respuestas del mismo caso *FJSSP*.

Al ejecutarse el programa, se solicita del usuario los datos relativos del *FJSSP* y los datos relativos del Algoritmo Genético de Rutas y del Algoritmo Genético de Secuencias. En la Figura 43 y la Figura 44, se pueden ver todos los requerimientos que el programa solicita del usuario para dos casos diferentes.

La Figura 43, muestra un ejemplo de datos que el usuario ingresa cuando se ejecuta el programa con "OPCIÓN_NÚMERO? = 1". Es decir, la opción que corresponde al de "MINIMIZAR LIBREMENTE A MAXIMUM WORKLOAD Y TOTAL WORKLOAD". Por consiguiente, no existirán valores umbrales de parada para *Maximum Workload* y *Total Workload*.

El "TAMAÑO_POBLACIÓN_ALGORITMO_GENÉTICO_RUTAS = 200" y el "TAMAÑO_POBLACIÓN_ALGORITMO_GENÉTICO_SECUENCIAS =200" indican el número de individuos o cromosomas para las poblaciones en cada Algoritmo Genético.

"MÁXIMA_GENERACIONES_ALGORITMO_GENÉTICO_RUTAS = 50" y "MÁXIMA_GENERACIONES_ALGORITMO_GENÉTICO_SECUENCIAS = 50" significan que en ambos Algoritmos Genéticos sus poblaciones evolucionarán pasando por las etapas de: Selección, Cruce, Mutación y Evaluación, durante cincuenta veces.

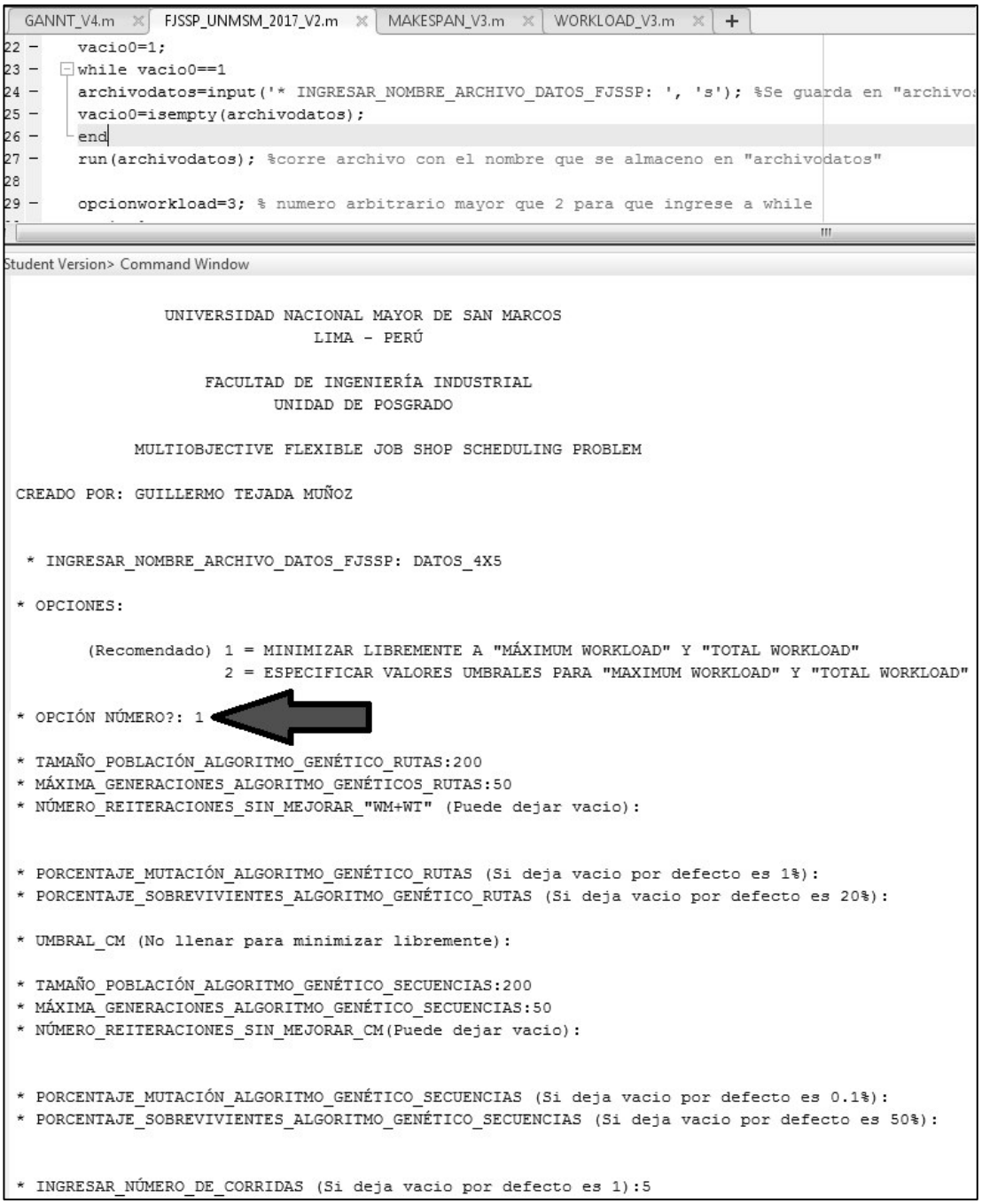


Figura 43. **Datos con la Opción 1**. *Fuente*: Elaboración propia

En las líneas “PORCENTAJE_MUTACIÓN_ALGORITMO_...” y en “PORCENTAJE_SOBREVIVIENTES_ALGORITMO…”, si el usuario no ingresa valores, entonces por defecto el programa designa los porcentajes que figuran entre paréntesis. Los valores por defecto han sido seleccionados dando buenos resultados para muchos casos.

En “NÚMERO_REITERACIONES_SIN_MEJORAR_WM+WT” es un criterio adicional de parada del Algoritmo Genético de Rutas. Un número escrito en esta línea indica el número de veces que el valor de la función objetivo del Algoritmo Genético de Rutas, (W_M+W_T), se reitera sin mejorar en generaciones consecutivas. Similarmente, sucede lo mismo para caso del Algoritmo Genético de Secuencias en la línea: “NÚMERO_REITERACIONES_SIN_MEJORAR_CM”. En ambos casos, si se dejan vacías las líneas, internamente se les designa valores iguales al número máximo de generaciones, quedando de esta manera inhabilitadas.

La Figura 44, muestra las alternativas cuando se ejecuta el programa con la opción 2. Es decir, con la opción de: “ESPECIFICAR VALORES UMBRALES PARA MAXIMUM WORKLOAD Y TOTAL WORKLOAD”.

Para la opción 2, el programa requiere del usuario tres nuevos datos, que en la Figura 44, se muestran como ejemplos:

“UMBRAL_MAXIMUM_WORKLOAD (U_WM) = 10”.
“UMBRAL_TOTAL_WORKLOAD (U_WT) = 32”.
“MÁXIMO_NÚMERO_REINICIOS_ALGORITMO_GENÉTICO_RUTAS…= 100”.

Con los dos primeros datos (U_WM y U_WT) el Algoritmo Genético de Rutas se detiene, si se cumple las condiciones: (W_M+W_T) ≤. (U_WM + U_WT) con W_M ≤ U_WM y W_T ≤ U_WT. Pero si no se cumplen, entonces, se contínua. Así, por ejemplo, para los datos umbrales de la Figura 44, la búsqueda puede detenerse cuando el mejor Cromosoma de Rutas alcanza un

(W_M+W_T) ≤. 42 y si además W_M ≤ 10 y W_T ≤ 32; en caso contrario el Algoritmo Genético de Rutas sigue con su proceso de búsqueda.

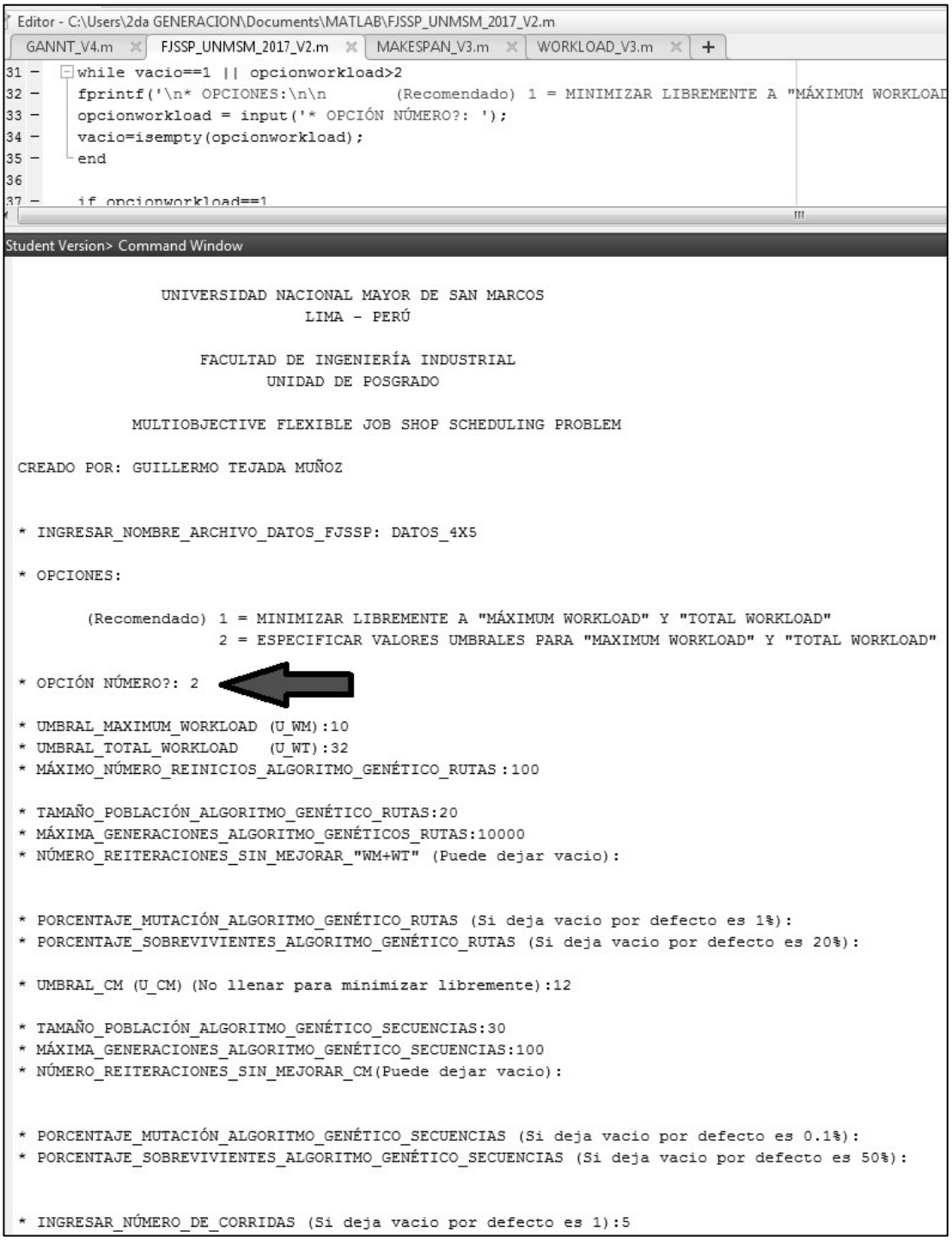


Figura 44. **Datos con la Opción 2**. *Fuente*: Elaboración propia

El tercer dato requerido “MÁXIMO_NÚMERO_REINICIOS…= 100”, significa el máximo número de reinicios o intentos, generándose en cada intento una nueva población de individuos, para que el Algoritmo Genético de Rutas pueda cumplir con los umbrales requeridos. En el ejemplo, el número de reinicios (100) es arbitrario, conviene colocar un número alto para dar libertad al programa.

Se ha considerado para la población del Algoritmo Genético de Rutas un tamaño de decenas de unidades (en este caso 20) y generaciones del orden de los miles (en este caso 10000), ello ha proveído de mucha rapidez a la búsqueda. Además, un número alto de generaciones provee libertad al Algoritmo Genético de Rutas para ejecutarse disminuyendo la probabilidad de reinicios.

Para el caso del Algoritmo Genético de Secuencias, según el ejemplo de los datos de la Figura 44, el dato “UMBRAL_CM (U_CM) = 12” y el dato “MÁXIMA_GENERACIONES_ALGORITMO_GENÉTICO_SECUENCIAS = 100” significan respectivamente que la búsqueda se detiene cuando el mejor Cromosoma de Secuencias tiene un $C_M \leq$ U_CM (Makespan menor o igual a 12) o cuando el número de generaciones alcanza el valor de 100. Es muy importante, especificar también el número de corridas si se desean tener varias soluciones para el mismo problema, en este ejemplo el número de corridas es 5.

La Figura 45, muestra una vista del ambiente de Matlab, cuando el programa está ejecutándose, los datos ingresados son los correspondientes al de la Figura 44, se observa que automáticamente se van generando figuras, una por cada corrida, éstas contienen los Diagramas de Gantt de las secuencias obtenidas. También los resultados son exportados a una hoja Excel.

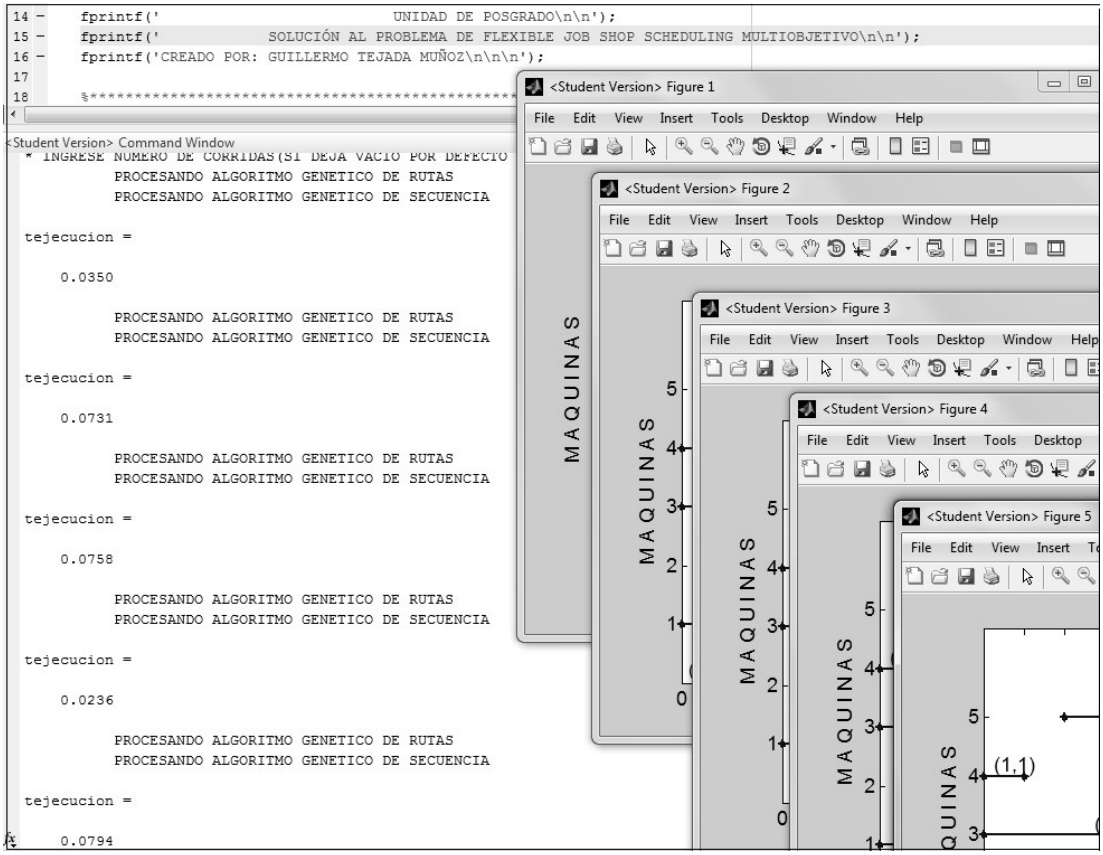


Figura 45. **Vista de Ejecución del Programa con la Generación de Diagramas de Gantt**.
Fuente: Elaboración propia.

La Figura 46, es uno de los Diagramas de Gantt generado automáticamente por el programa, se muestra el primero de los cinco diagramas generados. Los valores de W_M, W_T, C_M y precedencia de operaciones pueden ser validados objetivamente del análisis del Diagrama.

La Figura 47, muestra los resultados exportados por el programa a un archivo de Excel. Las cinco soluciones, según se muestra, son W_M=10, W_T = 32 y $C_M \leq 12$, cumpliéndose con los umbrales de búsqueda especificados por el usuario.

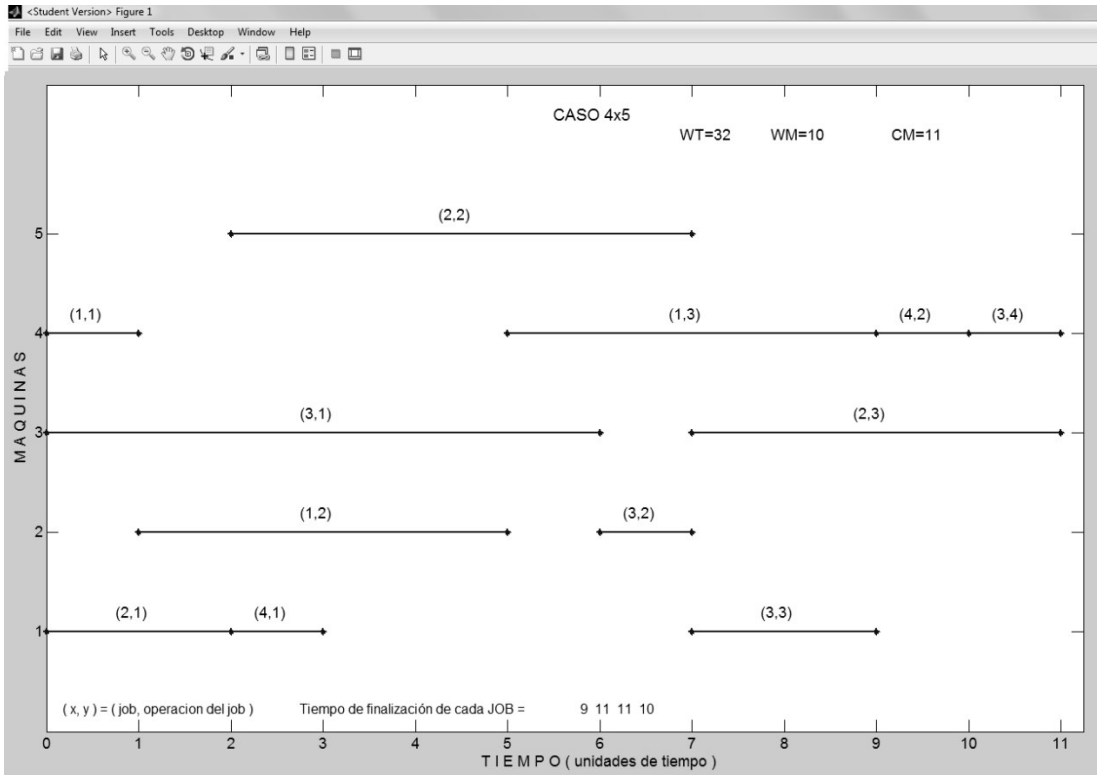


Figura 46. **Resultado en Diagramas de Gantt**. *Fuente*. Elaboración propia

Pegar | Copiar formato | Portapapeles | Fuente | Alineación | Combinar y centrar

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
3	WM	WT	CM	t_ejecución														
4	10	32	11	0.0350		MÁQUINAS	4	2	4	1	5	3	3	2	1	4	1	4
5						TIEMPOS	1	4	4	2	5	4	6	1	2	1	1	1
6						SECUENCIA ÓPTIMA	1	7	2	4	5	3	8	11	12	9	10	6
7																		
8	10	32	12	0.0731		MÁQUINAS	4	2	4	1	1	3	3	2	4	4	1	4
9						TIEMPOS	1	4	4	2	5	4	6	1	2	1	1	1
10						SECUENCIA ÓPTIMA	1	11	12	4	2	3	7	8	9	10	5	6
11																		
12	10	32	12	0.0758		MÁQUINAS	4	2	4	1	1	3	3	2	1	4	1	4
13						TIEMPOS	1	4	4	2	5	4	6	1	2	1	1	1
14						SECUENCIA ÓPTIMA	4	7	1	11	2	5	8	3	9	12	6	10
15																		
16	10	32	11	0.0236		MÁQUINAS	4	2	1	1	5	3	3	2	4	4	1	4
17						TIEMPOS	1	4	4	2	5	4	6	1	2	1	1	1
18						SECUENCIA ÓPTIMA	4	1	7	11	2	12	8	5	6	9	3	10
19																		
20	10	32	11	0.0794		MÁQUINAS	4	2	1	1	5	3	3	2	4	4	1	2
21						TIEMPOS	1	4	4	2	5	4	6	1	2	1	1	1
22						SECUENCIA ÓPTIMA	4	1	2	7	11	5	8	3	9	10	12	6
23																		
24	T_Promedio			0.0574														

Figura 47. **Resultados Exportados a Excel**. *Fuente*. Elaboración propia

4.2 Solución de casos *FJSSP* y Monitoreo del Funcionamiento del Algoritmo Genético de Rutas y del Algoritmo Genético de Secuencias

Los cinco casos de *FJSSP* planteados por Kacem et al. (2002) y Kacem et al. (2002a) que han sido solucionados para probar el funcionamiento de los algoritmos son:

1) *FJSSP* de 4 *Jobs*, 5 máquinas, 12 operaciones de Flexibilidad Total (Tabla 9).
2) *FJSSP* de 8 *Jobs*, 8 máquinas, 27 operaciones de Flexibilidad Parcial (Tabla 10)
3) *FJSSP* de 10 *Jobs*, 7 máquinas, 29 operaciones de Flexibilidad Total (Tabla 11)
4) *FJSSP* de 10 *Jobs*, 10 máquinas, 30 operaciones de Flexibilidad Total (Tabla 12)
5) *FJSSP* de 15 *Jobs*, 10 máquinas, 56 operaciones de Flexibilidad Total (Tabla 13)

Para cada caso, los algoritmos han corrido veinte veces. Es decir, se han obtenido veinte soluciones para el caso *FJSSP* 4x5, veinte para el caso *FJSSP* 8x8, veinte para el caso *FJSSP* 10x7, veinte para el caso *FJSSP* 10x10 y veinte para el caso *FJSSP* 15x10. El programa ha exportado automáticamente los resultados numéricos a un archivo de Excel (estos resultados aparecen en el Anexo B). También automáticamente el programa ha generado Diagramas de Gannt que son utilizados para validar visualmente los resultados numéricos.

En las tablas del **Anexo B**, en las filas de **Máquinas** aparecen las máquinas que el Algoritmo Genético de Rutas designó para cada una de las

operaciones, en las filas de **Tiempos**, están los tiempos que cada máquina utiliza para procesar cada operación, en las filas de **Secuencia** aparece la secuencia de operaciones que el Algoritmo Genético de Secuencias entrega como resultado. Aparecen también, las columnas $\boldsymbol{W_M}$, $\boldsymbol{W_T}$, $\boldsymbol{C_M}$ y **t** que corresponden respectivamente al *Maximum* Workload ($\boldsymbol{W_M}$), *Total Workload* ($\boldsymbol{W_T}$), *Makespan* ($\boldsymbol{C_M}$) y Tiempo de Ejecución de los Algoritmos (**t**), este ultimo valor se refiere al tiempo consumido por el Algoritmo Genético de Rutas y el Algoritmo Genético de Secuencias.

La Tabla 14, muestra el resumen de las soluciones numéricas presentadas en el Anexo B.

Tabla 14.
Resultados de W_T, W_M, C_M y Tiempos de Ejecución del Programa

Caso *FJSSP*	Respuestas	Total Workload (W_T)	*Maximum* Workload (W_M)	*Makespan* (C_M)	Frecuencia (Sobre 20)	Tiempo Promedio (20 corridas)
4x5	R1	32	10	11	65 %	0.09 seg.
	R2	32	10	12	35 %	
	R3	32	9	13	5%	
8x8	R1	75	12	15	40 %	0.80 seg.
	R2	76	12	15	60 %	
10x7	R1	61	11	11	70 %	1.14 seg.
	R2	61	11	12	30 %	
10x10	R1	42	6	7	15 %	1.62 seg.
	R2	42	6	8	70 %	
	R3	42	6	9	15 %	
15x10	R1	91	11	12	70 %	11.53 seg.
	R2	91	11	13	30 %	
R1 = Respuesta 1, R2 = Respuesta 2, R3 = Respuesta 3						

Fuente. Elaboración propia

De las veinte veces que corrió el programa, para un mismo caso, se han obtenido diferentes respuestas o soluciones, indicadas en la columna

Respuestas de la Tabla 14 y el número de veces que se repite la respuesta, sobre un total de 20, se indica en la columna **Frecuencia** en términos de porcentaje. Esto último, quiere decir, por ejemplo, en el caso de *FJSSP* 4x5, ver Tabla 14, se ha encontrado una tercera respuesta R3 con valores de: W_T = 32, W_M = 9, C_M = 13, con una frecuencia de 5%, es decir, existe solo una (**1**) sola respuesta con estos valores, si consideramos el total de veinte (20). Por otro lado, la columna de **Tiempo Promedio**, registra el tiempo de ejecución promedio utilizado por el Algoritmo Genético de Rutas conjuntamente con el Algoritmo Genético de Secuencias para encontrar la respuesta. Las condiciones en que se ejecutó el programa, para cada caso, se resumen en la Tabla 15.

Tabla 15
Condiciones del Programa

	FJSSP Jobs x Máquinas				
CONDICIONES	4X5	8X8	10X7	10X10	15X10
Umbral_Maximum_Workload (U_WM)	10	12	11	6	11
Umbral_Total_Workload (U_WT)	32	76	61	42	91
Máximo_Número_Reinicios_Algoritmo_Genético_Rutas	100	100	100	100	100
Tamaño_Población_Algoritmo_Genético_Rutas	20	20	20	20	20
Máxima_Generaciones_Algoritmo_Genético_Rutas	10000	10000	5000	5000	5000
Número_Reiteraciones_Sin_Mejorar_WM+WT	-	-	-	-	-
Porcentaje_Mutación_Algoritmo_Genético_Rutas	1	1	1	1	1
Porcentaje_Sobrevivientes_Algoritmo_Genético_Rutas	20	20	20	20	20
Umbral_Cm (U_CM)	12	15	11	7	12
Tamaño_Población_Algoritmo_Genético_Secuencias	50	200	200	300	400
Máxima_Generaciones_Algoritmo_Genético_Secuencias	10	30	10	10	40
Número_Reiteraciones_Sin_Mejorar_CM	-	-	-		
Porcentaje_Mutación_Algoritmo_Genético_Secuencias	0.1	0.1	0.1	0.1	0.1
Porcentaje_Sobrevivientes_Algoritmo_Genético_Secuencias	50	50	50	50	50

Fuente. Elaboración propia

4.2.1 Resultado al Detalle para el Caso FJSSP 4x5

A continuación se muestra los resultados al detalle cuando se corrió el programa para resolver el caso *FJSSP* de 4 *Jobs* y 5 máquinas (primer caso de la Tabla 14). Los valores numéricos que fueron exportados directamente por el programa a una hoja de Excel se muestran en la Tabla 16, que puede ser presentada como dos tablas, tal como se muestra en la Tabla 17 y la Tabla 18.

Tabla 16
Resultado General del Caso FJSSP 4x5

													W_M	W_T	C_M	t (seg)
Operaciones	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	10	32	11	0.10
Máquinas	4	2	4	1	5	3	3	2	1	4	1	2				
Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
Secuencia	*4*	*7*	*1*	*2*	*8*	*5*	*11*	*9*	*12*	*6*	*3*	*10*				

Fuente. Elaboración propia

En la Tabla 17, aparecen las columnas: Job, (Operación general) $O_{Job,\ operación\ del\ Job}$, Máquina y Tiempo. La presentación de la Tabla 17 perrmite observar con mayor claridad cuáles de las máquinas han sido designadas a cada operación por el Algoritmo Genético de Rutas. Se indica además el W_M y el W_T reportado por el Algoritmo Genético de Rutas.

En la Tabla 18, se presenta a la Secuencia, pero acompañadas de las máquinas en donde se ejecutan y el tiempo que demora en procesarse, así se tiene: (**Operación general)** $O_{Job,operación\_del\_Job}$ Maquina (Tiempo). Se indicá además el valor de C_M, que le corresponde a la secuencia.

Los resultados de la Tabla 17 han sido logrados con la ejecución del Algoritmo Genético de Rutas. Por consiguiente, el Algoritmo Genético

de Rutas ha sido capaz de designar máquinas a cada operación minimizando W_M y W_T.

Tabla 17
Resultado del Algoritmo Genético de Rutas del Caso FJSSP 4x5

JOB	(Operación general) $O_{Job,\ operación\_\ del\_Job}$	Máquina	Tiempo
J_1	*1* ($O_{1,1}$)	4	1
	2 ($O_{1,2}$)	2	4
	3 ($O_{1,3}$)	4	4
J_2	4 ($O_{2,1}$)	1	2
	5 ($O_{2,2}$)	5	5
	6 ($O_{2,3}$)	3	4
J_3	*7* ($O_{3,1}$)	3	6
	8 ($O_{3,2}$)	2	1
	9 ($O_{3,3}$)	1	2
	10 ($O_{3,4}$)	4	1
J_4	*11* ($O_{4,1}$)	1	1
	1*2* ($O_{4,2}$)	2	1
		W_M=10 W_T= 32	

Fuente. Elaboración propia

Tabla 18
Resultado Secuencia con Operación, Máquina y Tiempo del Caso FJSSP 4x5

Orden en que se deben ejecutar las Operaciones (C_M=11)								
4 ($O_{2,1}$) M_1 (2)	→	*7* ($O_{3,1}$) M_3 (6)	→	*1* ($O_{1,1}$) M_4 (1)	→	*2* ($O_{1,2}$) M_2 (4)	→
8 ($O_{3,2}$) M_2 (1)	→	*5* ($O_{2,2}$) M_5 (5)	→	*11* ($O_{4,1}$) M_1 (1)	→	*9* ($O_{3,3}$) M_1 (2)	→
12 ($O_{4,2}$) M_2 (1)	→	*6* ($O_{2,3}$) M_3 (4)	→	*3* ($O_{1,3}$) M_4 (4)	→	*10* ($O_{3,4}$) M_4 (1)	

Fuente. Elaboración propia

Mientras los resultados de la Tabla 18 han sido logrados con la ejecución del Algoritmo Genético de Secuencias. Por consiguiente, el Algoritmo Genético de Secuencias ha sido capaz de secuenciar u

ordenar el procesamento de las operaciones que aparecen en la Tabla 17, de forma de minimizar el C_M. Operaciones, que en el paso anterior, el Algoritmo Genético de Rutas les designó máquina para su procesamiento.

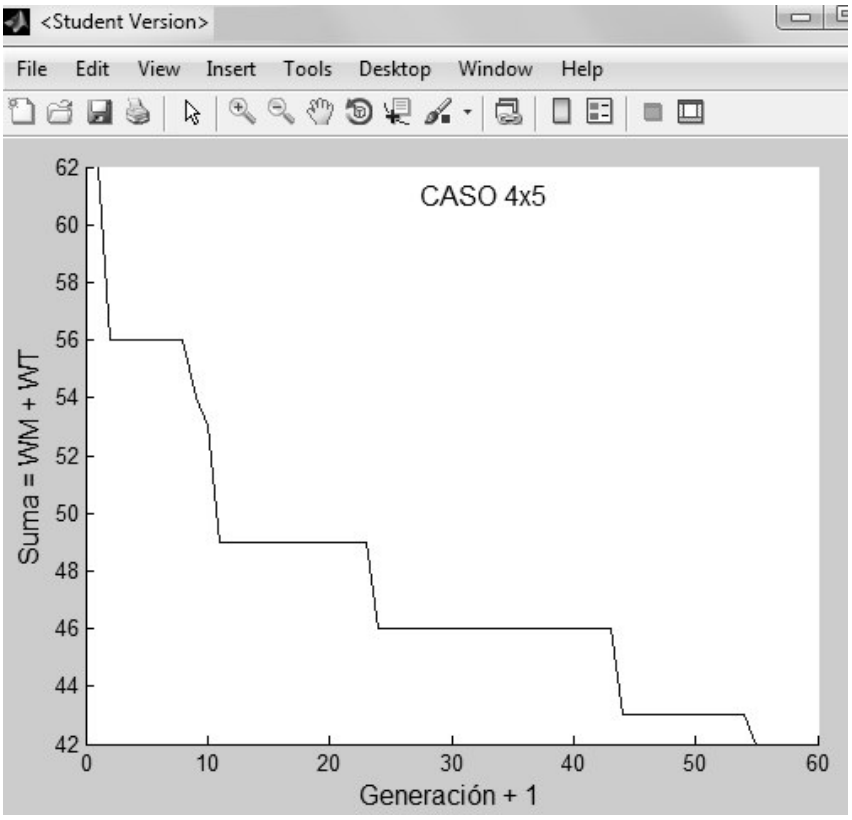
Se analizará, los resultados de la Tabla 17, dejando momentáneamente de lado los resultados de la Tabla 18. Es oportuno indicar que la primera hipótesis de la investigación, la cual se transcribe literalmente a continuación:

H1: Se solucionará el Subproblema de Enrutamiento óptimo, en un sistema de fabricación tipo Flexible Job Shop (FJS), consistente en asignar a cada operación de los Jobs, una entre muchas máquinas candidatas para su procesamiento, minimizando los criterios de: Total Workload (W_T, suma de carga de trabajo de todas las máquinas) y Maximum Workload (W_M, máxima carga de trabajo entre todas las máquinas) mediante la utilización de Algoritmos Genéticos.

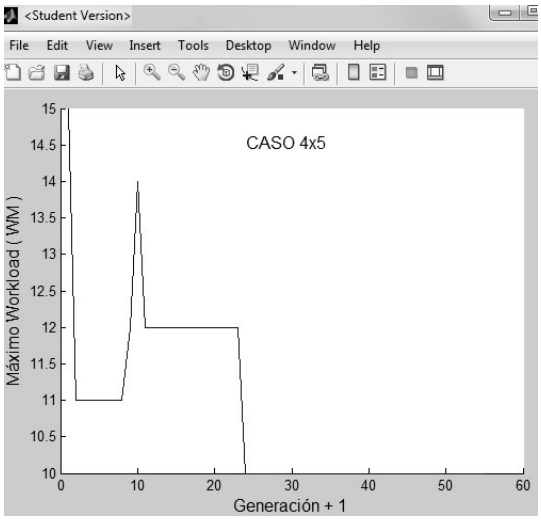
Ha sido probada según los resultados descritos con la Tabla 17 y como se sigue confirmando de acuerdo al monitoreo del funcionamiento del Algoritmo Genético de Rutas, que se describe en los parráfos siguientes.

Se ha monitoreado el funcionamiento del programa, registrando en un vector los resultados parciales logrados en cada generación, para su ploteo. Es así, que la Figura 48a, muestra el funcionamiento del Algoritmo Genético de Rutas, se verifica que en cada generación va logrando minimizar la suma de ($W_M + W_T$), se detiene, luego de aproximadamente sesenta generaciones, cuando alcanza una suma mínima de 42 que es el valor umbral de parada. En este punto, el Algoritmo Genético de Rutas verifica que cada sumando, W_M y W_T cumplan con sus umbrales. En este caso, como se observa en (b) y (c) de la Figura 48, los sumandos alcanzan los valores de $W_M = 10$ y $W_T = 32$, cumpliendo con los umbrales especificados. Si no fuera así, el algoritmo se reinicia, tal como ha sido explicado anteriormente. En el caso particular

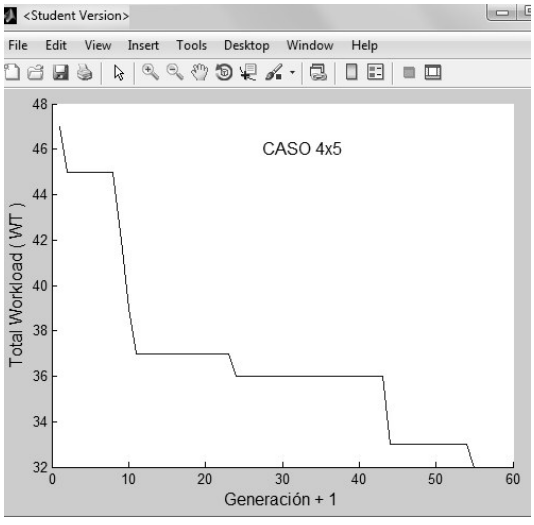
de la Figura 48, el proceso de minimización, según el monitoreo realizado, ha correspondido durante su tercer reinicio.



(a)



(b)



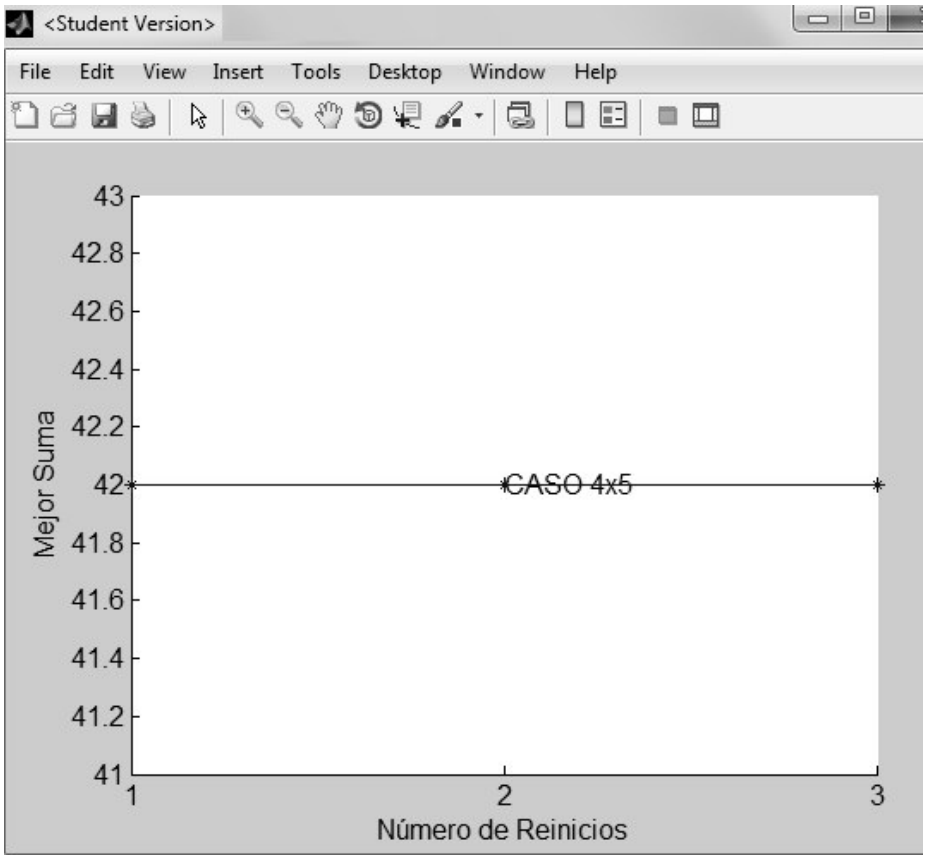
(c)

Figura 48. **Progreso de Búsqueda del Algoritmo de Genético de Rutas del Caso *FJSSP* 4x5: (a) suma mínima (W_T+W_M), (b) comportamiento de W_M, (c) comportamiento de W_T.** *Fuente*. Elaboración propia

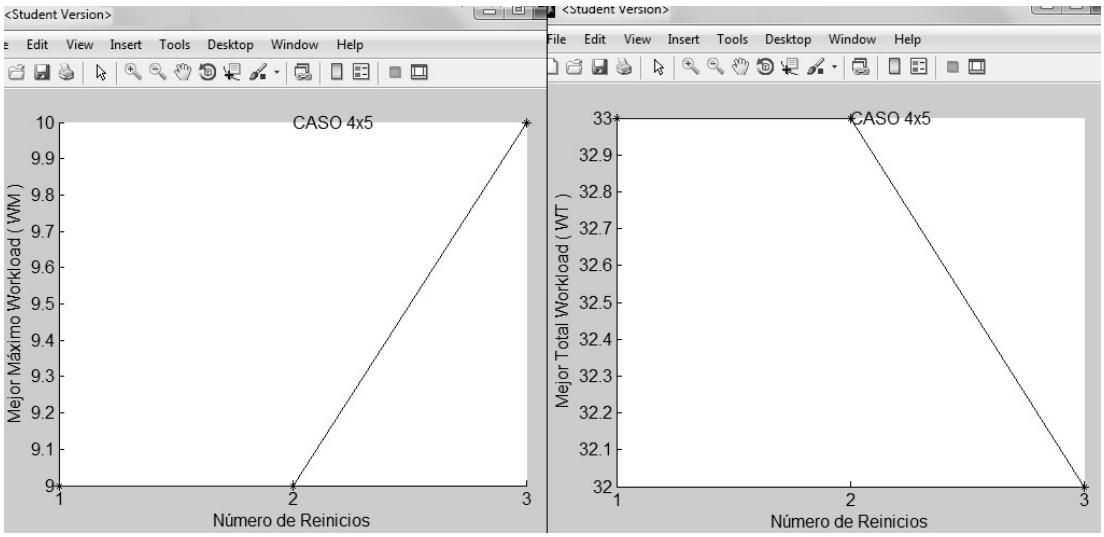
Cuando ocurre un reinicio, se producen saltos a otro sector del Epacio de Soluciones, facilitando, desde ese sector, encontrar con mayor rapidez un objetivo de mínima suma y sumandos que igualen o sean menores que sus umbrales. También, como se indicó en la Métologia, en lugar de optar por los reinicios, se pudó incrementar las generaciones y principalmente mediante los procesos de cruzamiento y mutación encontrar esa diversidad dentro del Espacio de Soluciones, pero ese proceso resultó ser muy lento.

Cada reinicio, es el comienzo de un nuevo proceso de minimización del Algoritmo Genético de Rutas, por esta razón, después de cada reinicio se tiene respuestas de la mejor suma de $W_M + W_T$ alcanzada y respuestas individuales de los valores de cada sumado W_M y W_T. Para demostrar la efectividad de los reinicios, se ha monitoreado estas respuestas después de cada reinicio para el *caso FJSSP* 4x5. La Figura 49a muestra la mejor suma obtenida en cada reinicio, mientras que la Figura 49b muestra el valor obtenido por W_T en cada reinicio y la Figura 49c el valor obtenido por W_M en cada reinicio. Comparando las tres Figuras, en cada reinicio, se observa que en la primera minimización el Algoritmo Genético de Rutas alcanzó una suma de 42 pero con sumando de 33 (W_T) y 9 (W_M), reiniciándose el Algoritmo Genético de Rutas, para luego alcanzar nuevamente una suma minima de 42 con sumandos son 33 y 9, hasta que finalmente en el tercer reinicio se obtiene una suma de 42 con sumando de 32 y 10 respectivamente que, se correspnden con los umbrales que fueron especificados ($W_M + W_T \leq 42$, con $W_M \leq 32$ y $W_T \leq 10$).

Una vez que el Algoritmo Genético de Rutas entrega los vectores Máquinas y Tiempos con mínimos valores de W_T y W_M. Entonces, el Algoritmo Genético de Secuencias comienza la búsqueda de la mejor secuencia de operaciones minimizando el C_M. También, se ha monitoreado dentro del Algoritmo Genético de Secuencias este proceso.



(a)



(b) (c)

Figura 49. **Valores Alcanzados en cada Reinicio: (a) Mejor suma, (b) mejor W_M y (c) mejor W_T.** *Fuente*. Elaboración propia

Es pertinente indicar que la segunda hipótesis de la investigación, la cual se transcribe literalmente a continuación:

H2: Se solucionará el Subproblema de Secuenciación óptima, en un sistema de fabricación tipo Flexible Job Shop (FJS), consistente en secuenciar el orden en que se deben procesar las operaciones de los Jobs en las máquinas seleccionadas previamente, minimizando el objetivo del Makespan (C_M, tiempo de finalización de todos los Jobs) mediante la utilización de Algoritmo Genéticos.

Ha sido probada de acuerdo a los resultados de la Tabla 18, que indica la secuencia óptima generada por el Algoritmo Genético de Secuencias con un mínimo C_M. Además, el cumplimiento de la hipótesis ha sido confirmada según la información obtenida del monitoreo del funcionamiento del Algortimo Genético de Secuencias, según se decribe en los siguientes párrafos.

En la Figura 50, se observa el monitoreo realizado al Algoritmo Genético de Secuencias cuando en cada generación va encontrando Cromosomas de Secuencias con menores valores de *Makespan*, se observa como en la tercera generación del Algoritmo Genético de Secuencias logra minimizar el *Makespan* inicial de 12 a 11, el cual fue el umbral de parada.

Es pertinente mencionar, que otra de la propuesta de la tesis ha consistido en evaluar y clasificar a la población genética recien creada, antes que haya pasado por su primer cruce y su primera mutación, la idea es detectar si el mejor individuo de la población pueda ya estar cumpliendo con el umbral de parada, redundando en un ahorro significativo de tiempo búsqueda. Pues bien en muchos casos pequeños como el que se está analizando ha ocurrido lo descrito. La Figura 51, es uno de los casos detectados, se ha podido verificar que el *Makespan* del mejor individuo de la población recien creada es de 11, no necesitando, por tanto, ingresar a las etapas de cruzamiento y mutación, como si ocurrió para el caso de la Figura 50. De esta forma se ha ganado mucha velocidad en el proceso de búsqueda.

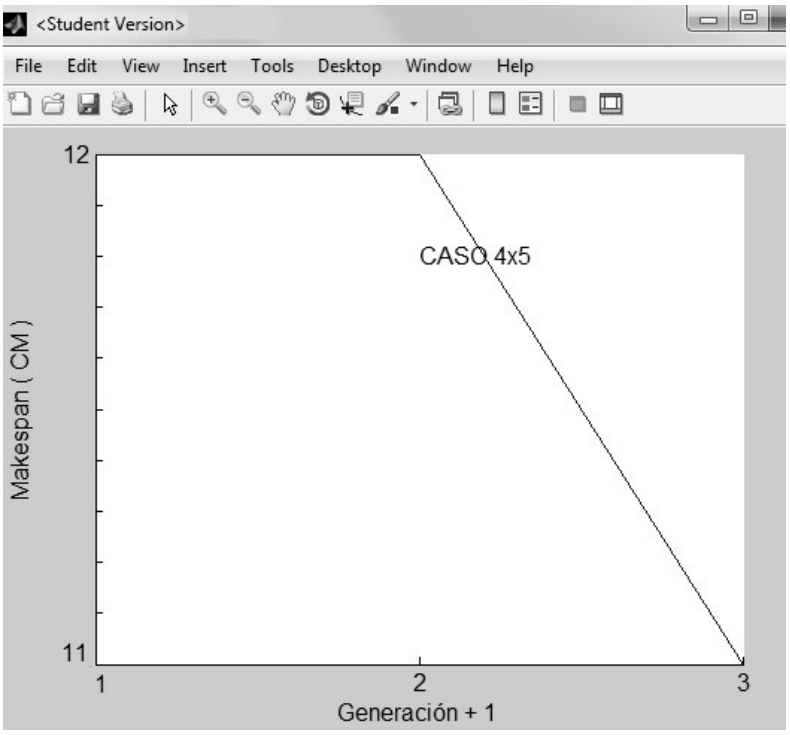


Figura 50. **Mejor *Makespan* en cada Generación del Caso *FJSSP* 4x5**. *Fuente*. Elaboración propia

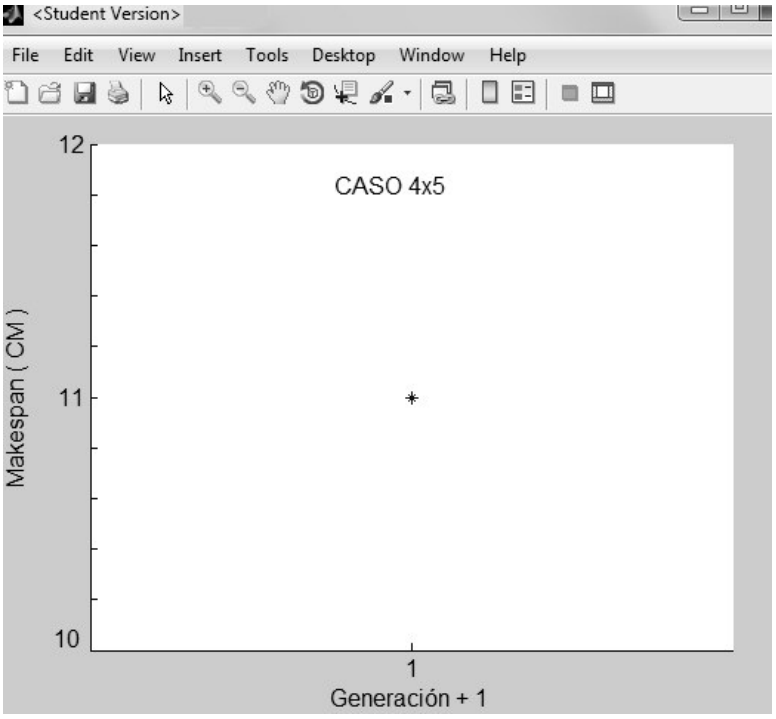


Figura 51. ***Makespan* Inicial Cumple con Umbral de Parada**. *Fuente*. Elaboración propia

En las figuras presentadas las generaciones aparecen aumentadas en una unidad porque el primer número es el valor detectado de la función objetivo antes de ingresar a los procesos de cruzamiento y mutación.

Con los resultados de Secuencia, Máquinas y Tiempos como se muestra en la Tabla 18, el Algoritmo de Presentación de Resultados construye los Diagramas de Gantt. La secuencia se transcribe a continuación:

4 ($O_{2,1}$) M_1 (2) → **7** ($O_{3,1}$) M_3 (6) → **1** ($O_{1,1}$) M_4 (1) → **2** ($O_{1,2}$) M_2 (4) →
8 ($O_{3,2}$) M_2 (1) → **5** ($O_{2,2}$) M_5 (5) → 11 ($O_{4,1}$) M_1 (1) → **9** ($O_{3,3}$) M_1 (2) →
12 ($O_{4,2}$) M_2 (1) → **6** ($O_{2,3}$) M_3 (4) → **3** ($O_{1,3}$) M_4 (4) → **10** ($O_{3,4}$) M_4 (1)

Si se desea realizar el Diagrama de Gantt manualmente, no hay más que seguir el orden de la secuencia. Asi, la operación $O_{2,1}$ se debe dibujar en la máquina 1 como un segmento de recta ocupando 2 unidades de tiempo, luego la operación $O_{3,1}$ en la máquina 3 como un segmento de recta de 6 unidades de tiempo, y así sucesivamente. Hay que tener cuidado de respetar la precedencia de operaciones del mismo *Job,* como por ejemplo, la operación **2** ($O_{1,2}$) M_2 (4) a pesar que es la primera en la máquina 2, debe ser colocada en el eje de tiempos del Diagrama después de la operación **1** ($O_{1,1}$) M_4 (1) ya que ambas son del mismo *Job* (restricción de precedencia). El Diagrama de Gannt generado automáticamente por el programa se muestra en la Figura 52.

Realizando, un análisis del Diagrama de Gantt de la Figura 52, se puede verificar el cumplimiento de los resultados numéricos de W_M; W_T y C_M generados por los algoritmos. Así por ejemplo, si se observa las operaciones distribuidas en cada máquina (M1, M2, M3, M4, M5), se puede verificar la secuencia, es decir, el orden de la ejecución de las operaciones y también calcular directamente las cargas de trabajo en cada máquina, como sigue:

M5: $O_{2,2}$ = 5 unidades de tiempo (u.t.)
M4: $O_{1,1} + O_{1,3} + O_{3,4}$ = 1+ 4 + 1= 6 u.t.
M3: $O_{3,1} + O_{2,3}$ = 6 + 4 = 10 u.t.
M2: $O_{1,2} + O_{3,2} + O_{4,2}$ = 4 + 1 + 1 = 6 u.t.
M1: $O_{2,1} + O_{4,1} + O_{3,3}$ =2 + 1 + 2 = 5 u.t.

Por lo que, la máxima carga de trabajo o *Maximun Workload* (W_M) la tiene la máquina M3 con 10 unidades de tiempo. Totalizando, todas las máquinas, una carga de trabajo o *Total Workload* (W_T) de 32 unidades de tiempo.

Por otro lado, también se puede verificar los tiempos que le toma a cada *Job* completar la ejecución de todas sus operaciones en las distintas máquinas que le fueron asignadas, así tenemos:

J1 = $O_{1,1}$ (M4) → $O_{1,2}$ (M2) → $O_{1,3}$ (M4) = 9 unidades de tiempo (u.t.)
J2 = $O_{2,1}$ (M1) → $O_{2,2}$ (M5) → $O_{2,3}$ (M3) = 11 u.t.
J3 = $O_{3,1}$ (M3) → $O_{3,2}$ (M2) → $O_{3,3}$ (M1) → $O_{3,4}$ (M4) = 10 u.t.
J4 = $O_{4,1}$ (M1) → $O_{4,2}$ (M2) = 8 u.t.

Es decir, cada *Job* (J1, J2, J3, J4) concluye respectivamente sus operaciones en: 9, 11, 10, 8 unidades de tiempo. Por lo que el tiempo que se utiliza para completar todas las operaciones de los *Jobs* o *Makespan* (C_M) es de 11 unidades de tiempo (Recordemos que C_M = Max [11,9,10,8]). Estos valores son directamente verificables de la escala de tiempos del Diagrama de Gantt.

También, del Diagrama de Gannt se puede verificar que la solución está cumpliendo con las restricciones tecnológicas u orden de precedencia de las operaciones de cada *Job*.

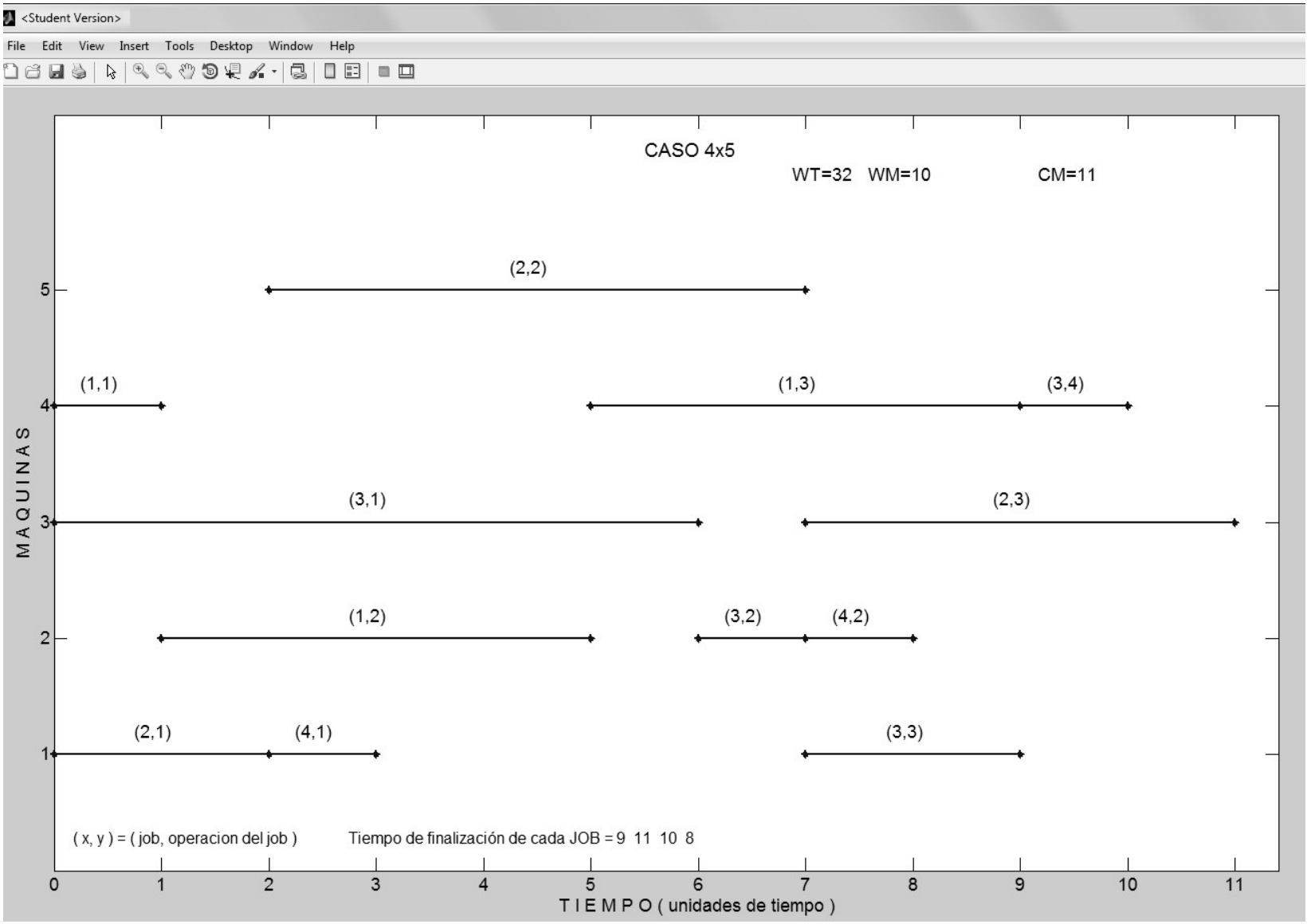


Figura 52. **Diagrama de Gantt: *FJSSP* 4 *Jobs*, 5 máquinas, 12 operaciones**. *Fuente*. Elaboración propia.

Se verifica entonces que secuencia de operaciones con las máquinas que fueron seleccionadas, tal como se muestra, en la Tabla 18 genera los valores de W_T = 32, W_M = 10 y C_M = 11. Observese que en la Figura 52 se entrega impreso los resultados de W_M, W_T, C_M, así como también los correspondietes a los tiempos de finalización de cada *Job* (ver en la parte inferior del Diagrama) lo cual es importante para establecer fechas de compromiso de terminación para cada *Job*.

4.2.2 Resultados al Detalle para los Casos: FJSSP 8x8, FJSSP 10x7 y FJSSP 10x15

Al igual que en caso anterior, a continuación se muestran los resultados al detalle cuando se corrió el programa para resolver los casos de *FJSSP* 8x8, *FJSSP* 10x7, *FJSSP* 10x10 y *FJSSP* 15x10, los resultados numéricos que fueron directamente exportados por el programa a un archivo de Excel se muestran respectivamente en: Tabla 19, Tabla 20, Tabla 21 y Tabla 22.

El Progreso de búsqueda del Algoritmo Genético de Rutas en cada generación, la Búsqueda del mejor *Makespan* del Algoritmo Genético de Secuencias en cada generación, los Diagramas de Gantt para cada caso se muestran en las Figuras numeradas del 53 al 64.

Los resultados mostrados desde la Tabla 19 a la Tabla 22 y desde la Figura 53 a la Figura 64, donde se muestran el proceso de minimización de las variables W_M, W_T y C_M, no hace más que confirmar, como en el caso anteriormente analizado del *FJSSP* 4x5, el cumplimiento de la hipótesis H1 e hipótesis H2. Resultados que, además, pueden ser validados objetivamente mediante los Diagramas de Gantt mostrados desde la Figura 61 a la Figura 64.

Tabla 19

Resultado General del Caso FJSSP 8x8

Operaciones	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*
Máquinas	5	5	6	3	4	7	5	7	4	1	2	6	3	1
Tiempos	3	3	2	3	2	1	4	2	4	1	1	5	2	3
Secuencia	*11*	*8*	*14*	*9*	*18*	*1*	*2*	*24*	*25*	*10*	*21*	*4*	*12*	*13*

														W_M	***W_T***	***C_M***	***t (seg.)***
Operaciones	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	12	76	15	0.51
Máquinas	7	6	7	3	8	2	3	8	4	1	2	8	5				
Tiempos	6	2	3	1	4	5	2	5	3	2	4	1	1				
Secuencia	*5*	*15*	*6*	*19*	*20*	*22*	*16*	*7*	*26*	*23*	*3*	*27*	*17*				

Fuente. Elaboración propia

Tabla 20

Resultado General del Caso FJSSP 10x7

Operaciones	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
Máquinas	1	6	6	7	1	7	6	5	1	4	1	2	1	2	3
Tiempos	1	1	4	3	2	1	1	2	2	1	1	1	2	1	4
Secuencia	*6*	*21*	*27*	*9*	*1*	*15*	*2*	*18*	*28*	*24*	*29*	*4*	*7*	*8*	*12*

														W_M	***W_T***	***C_M***	**t (seg.)**
Operaciones	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	11	61	11	0.20
Máquinas	3	3	7	7	6	4	4	2	5	2	7	2	6	1				
Tiempos	1	2	2	3	1	1	8	2	4	2	2	4	1	1				
Secuencia	*16*	*13*	*19*	*25*	*22*	*10*	*5*	*17*	*14*	*11*	*23*	*26*	*3*	*20*				

Fuente. Elaboración propia

Tabla 21

Resultado General del Caso FJSSP 10x10

Operaciones	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
Máquinas	1	3	4	1	10	10	10	8	7	7	3	4	9	9	4
Tiempos	1	1	1	2	1	2	1	1	1	1	3	1	2	1	1
Secuencia	*25*	*28*	*7*	*10*	*1*	*22*	*8*	*13*	*2*	*19*	*20*	*16*	*21*	*14*	*29*

															W_M	***W_T***	***C_M***	***t (seg.)***
Operaciones	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	6	42	7	0.25
Máquinas	6	9	9	1	3	6	5	2	2	6	7	6	6	4	7				
Tiempos	2	2	1	1	1	1	2	3	2	1	1	1	1	1	2				
Secuencia	*30*	*26*	*17*	*23*	*9*	*15*	*24*	*4*	*5*	*18*	*3*	*11*	*27*	*6*	*12*				

Fuente. Elaboración propia

Tabla 22

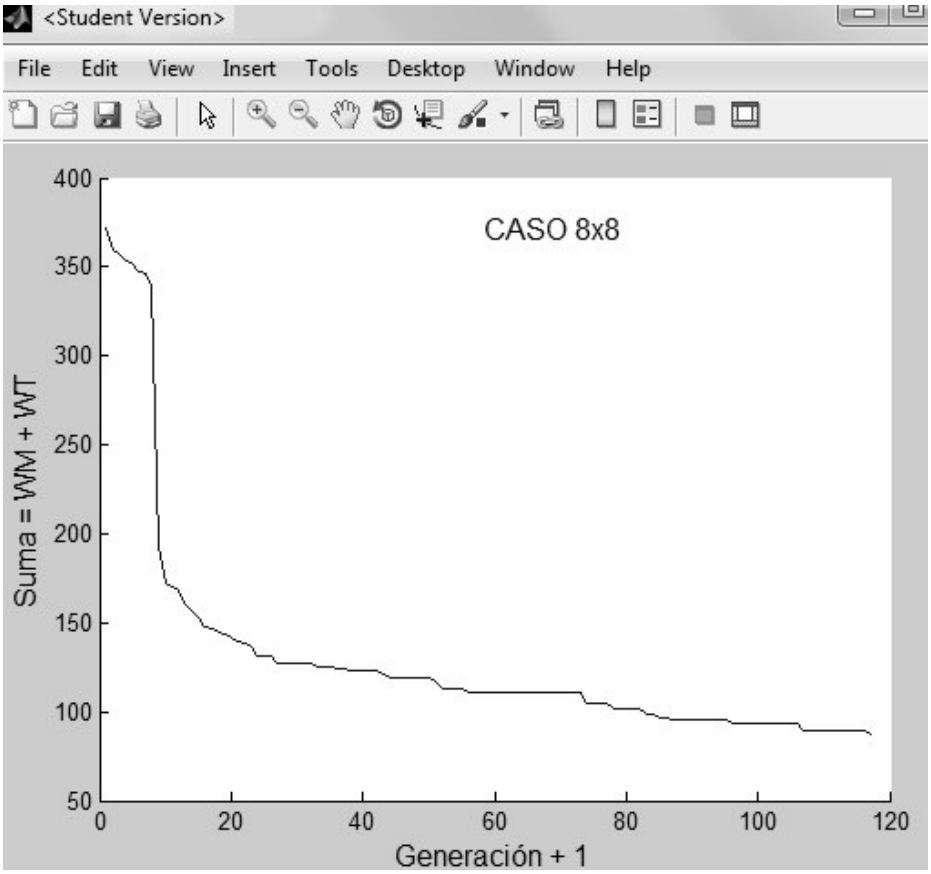
Resultado General del Caso FJSSP 15x10

Operaciones	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*
Máquinas	1	1	3	9	4	3	10	4	7	7	2	4	5	5	6	10	9	6	7
Tiempos	1	1	1	4	1	2	1	1	1	1	2	1	1	1	1	1	1	2	2
Secuencia	*13*	*17*	*45*	*41*	*49*	*23*	*29*	*5*	*18*	*50*	*30*	*25*	*33*	*37*	*14*	*9*	*21*	*10*	*42*

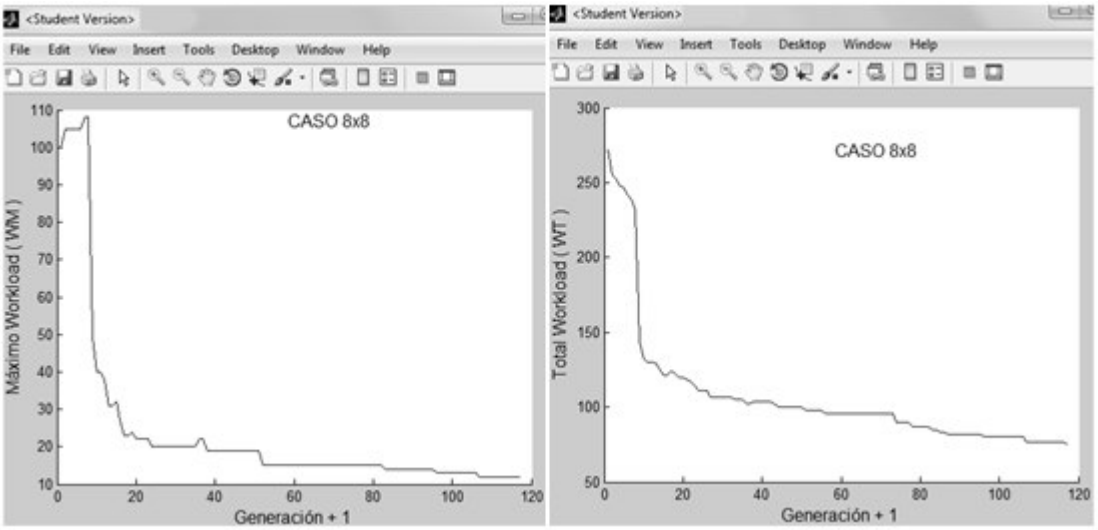
Operaciones	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*
Máquinas	6	2	1	1	2	7	8	4	1	10	3	7	10	10	1	10	2	1
Tiempos	2	1	1	1	1	1	1	2	1	1	2	4	2	1	2	1	2	1
Secuencia	*46*	*53*	*54*	*1*	*2*	*6*	*19*	*7*	*38*	*11*	*34*	*47*	*20*	*24*	*39*	*40*	*31*	*35*

																			W_M	***W_T***	***C_M***	***t (seg)***
Operaciones	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*	*47*	*48*	*49*	*50*	*51*	*52*	*53*	*54*	*55*	*56*	11	91	12	3.68
Máquinas	6	3	2	8	5	9	7	2	10	10	2	1	9	8	5	8	9	6	5				
Tiempos	1	2	1	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
Secuencia	*15*	*55*	*48*	*51*	*22*	*3*	*43*	*16*	*52*	*26*	*12*	*32*	*27*	*8*	*56*	*36*	*28*	*44*	*4*				

Fuente. Elaboración propia

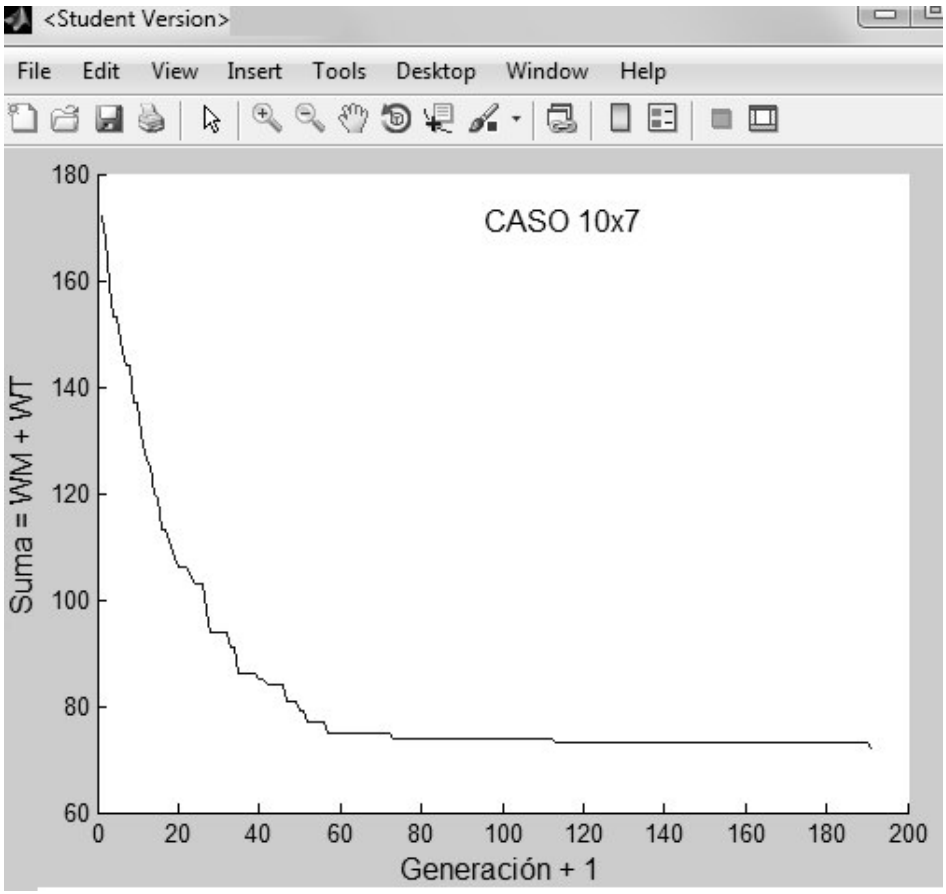


(a)

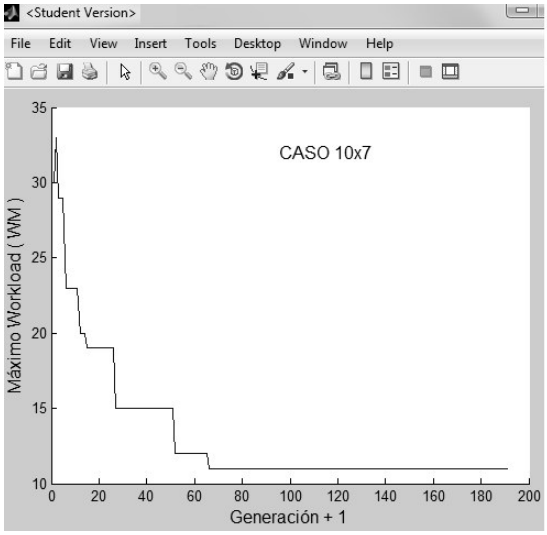


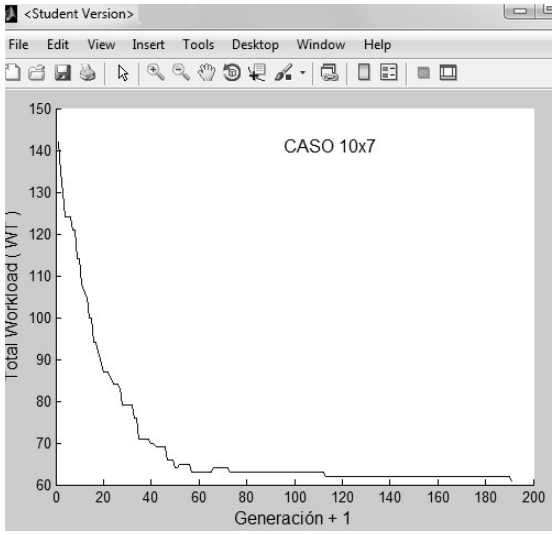
(b) (c)

Figura 53. **Progreso de Búsqueda del Algoritmo Genético de Rutas del Caso *FJSSP* 8x8: (a) suma mínima (W_T+W_M), (b) comportamiento de W_M, (c) comportamiento de W_T.** *Fuente*. Elaboración propia



(a)





(b) (c)

Figura 54. **Progreso de Búsqueda del Algoritmo Genético de Rutas del Caso *FJSSP* 10x7: (a) suma mínima (W_T+W_M), (b) comportamiento de W_M, (c) comportamiento de W_T.** *Fuente*. Elaboración propia

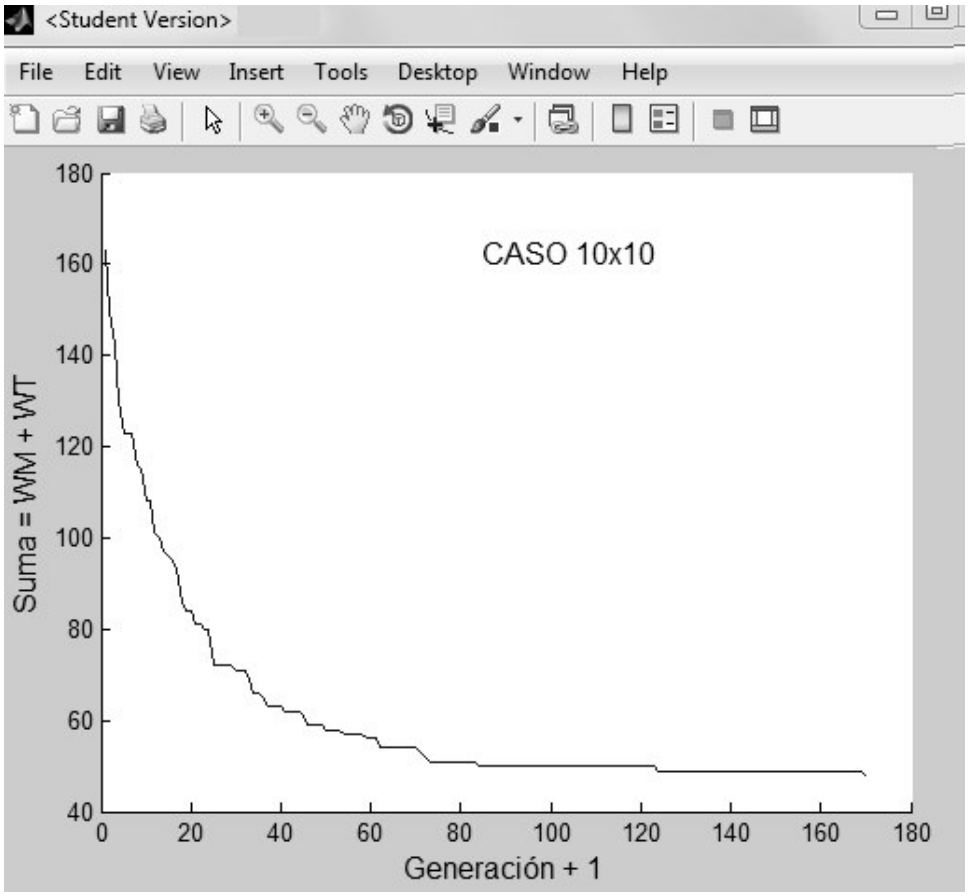
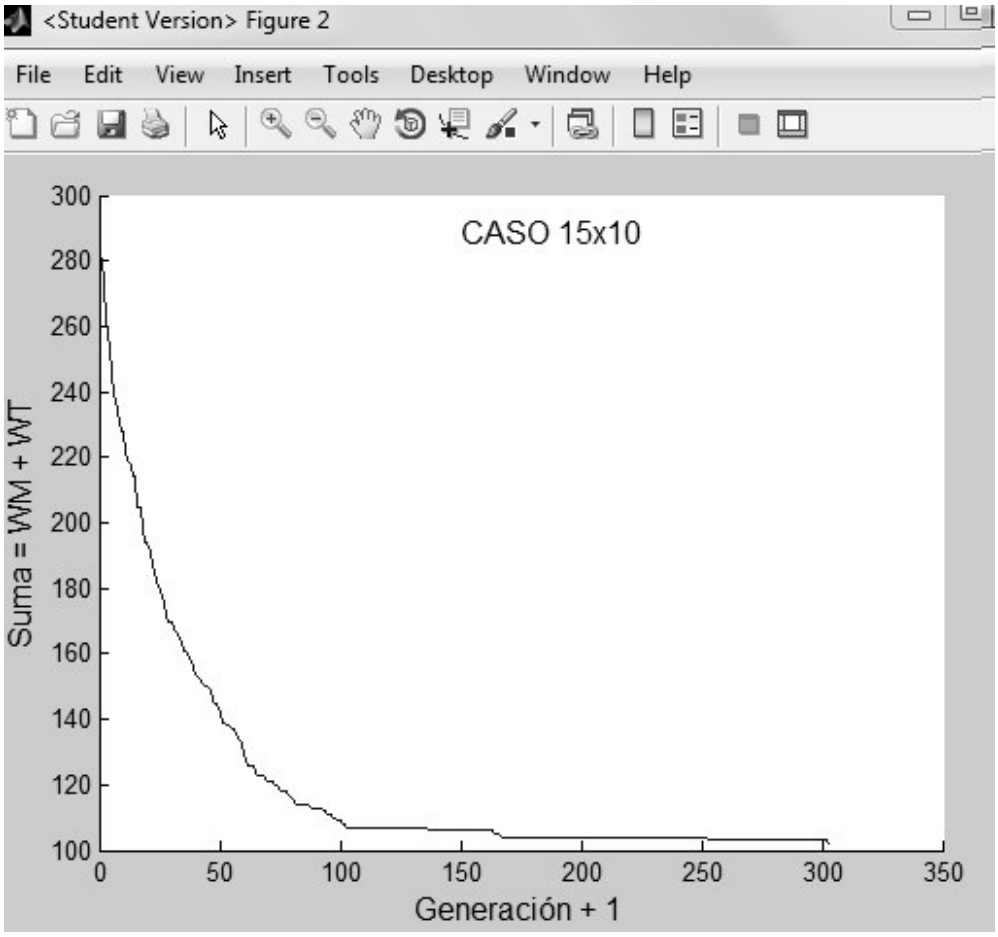
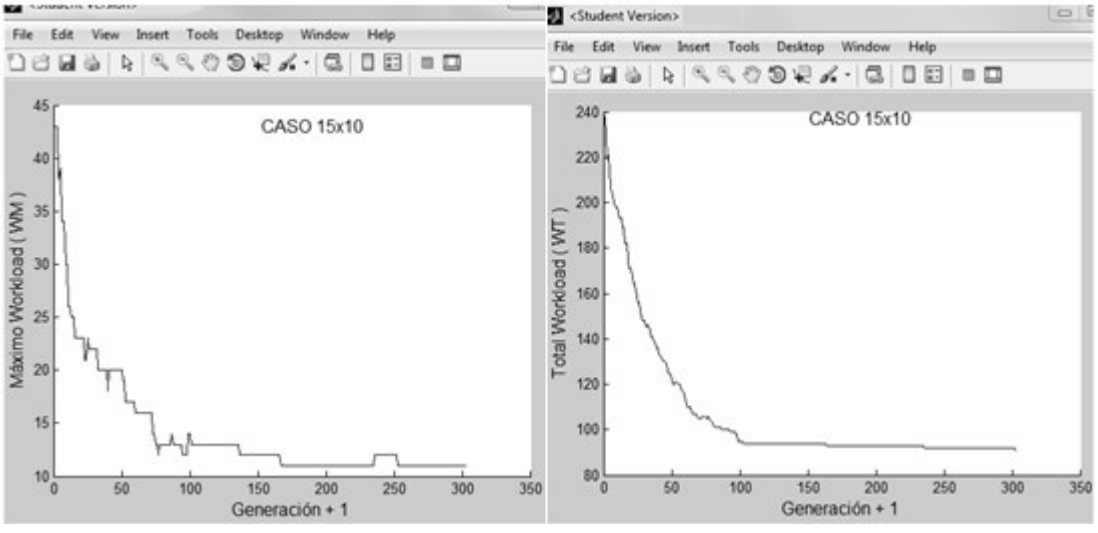


Figura 55. **Progreso de Búsqueda del Algoritmo de Genético de Rutas del Caso *FJSSP* 10x10: (a) suma mínima (W_T+W_M), (b) comportamiento de W_M, (c) comportamiento de W_M.** *Fuente*. Elaboración propia



(a)



(b) (c)

Figura 56 **Progreso de Búsqueda del Algoritmo Genético de Rutas del Caso *FJSSP* 15x10: (a) suma mínima (W_T+W_M), (b) comportamiento de W_M, (c) comportamiento de W_T.** *Fuente*. Elaboración propia

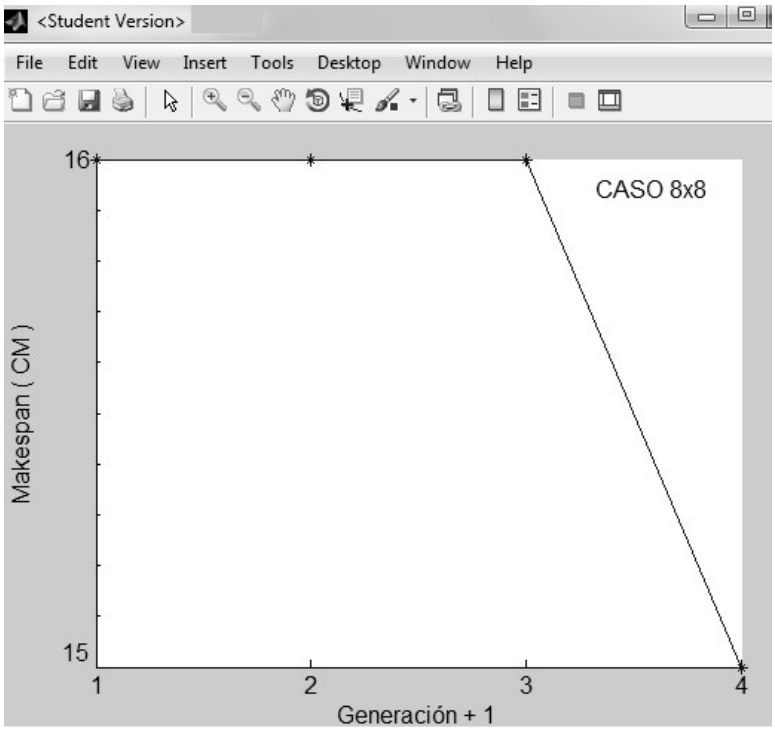


Figura 57. **Mejor *Makespan* en cada Generación del Caso *FJSSP* 8x8**. *Fuente*. Elaboración propia

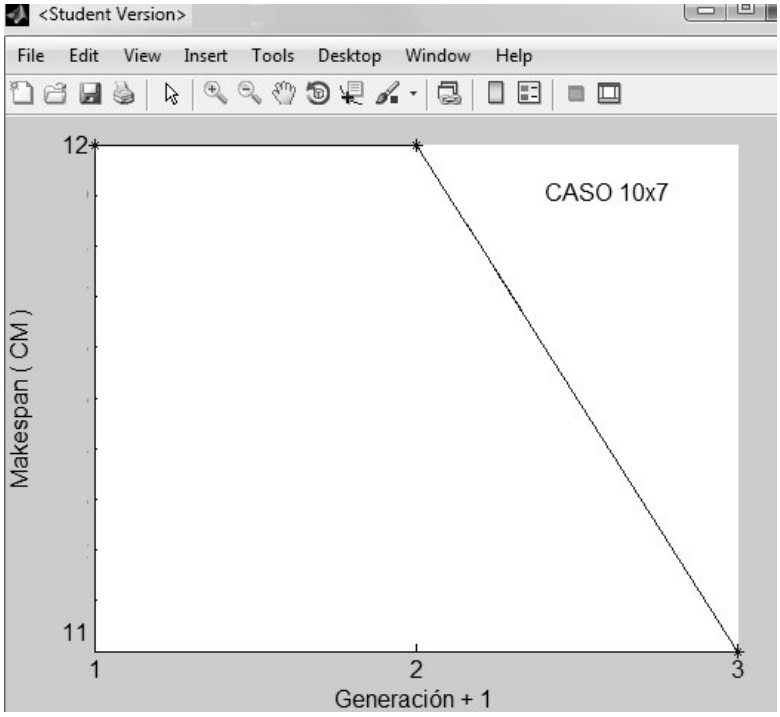


Figura 58. **Mejor *Makespan* en cada Generación del Caso *FJSSP* 10x7**. *Fuente*. Elaboración propia.

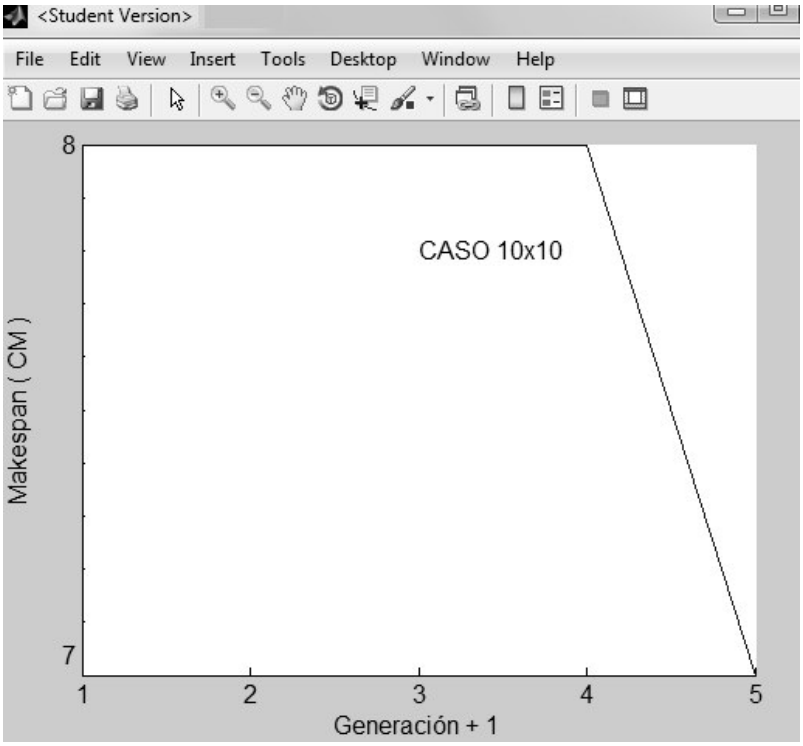


Figura 59. **Mejor *Makespan* en cada Generación del Caso *FJSSP* 10x10**. *Fuente*. Elaboración propia

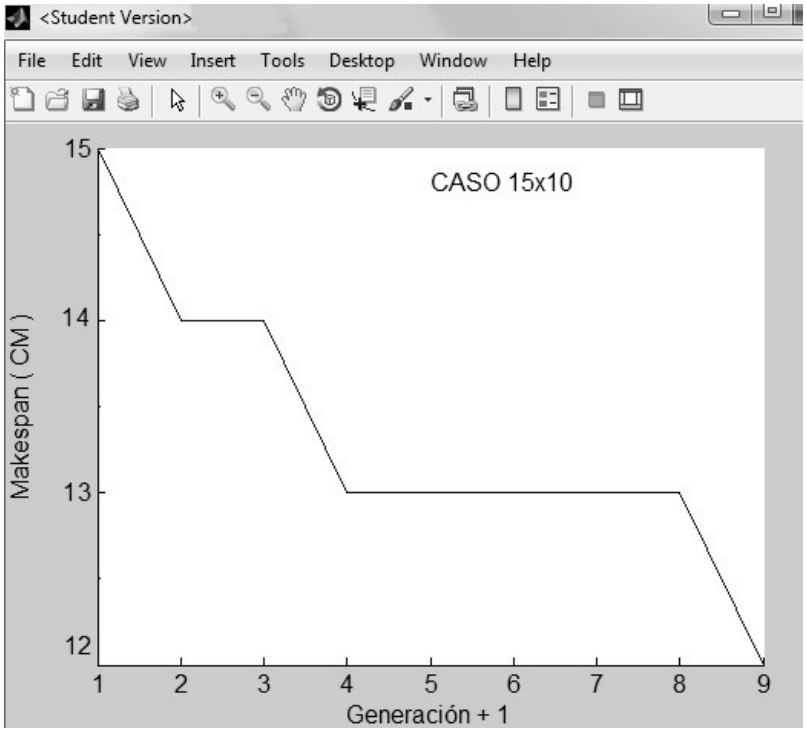


Figura 60. **Mejor *Makespan* en cada Generación del Caso *FJSSP* 15x10.** *Fuente*. Elaboración propia

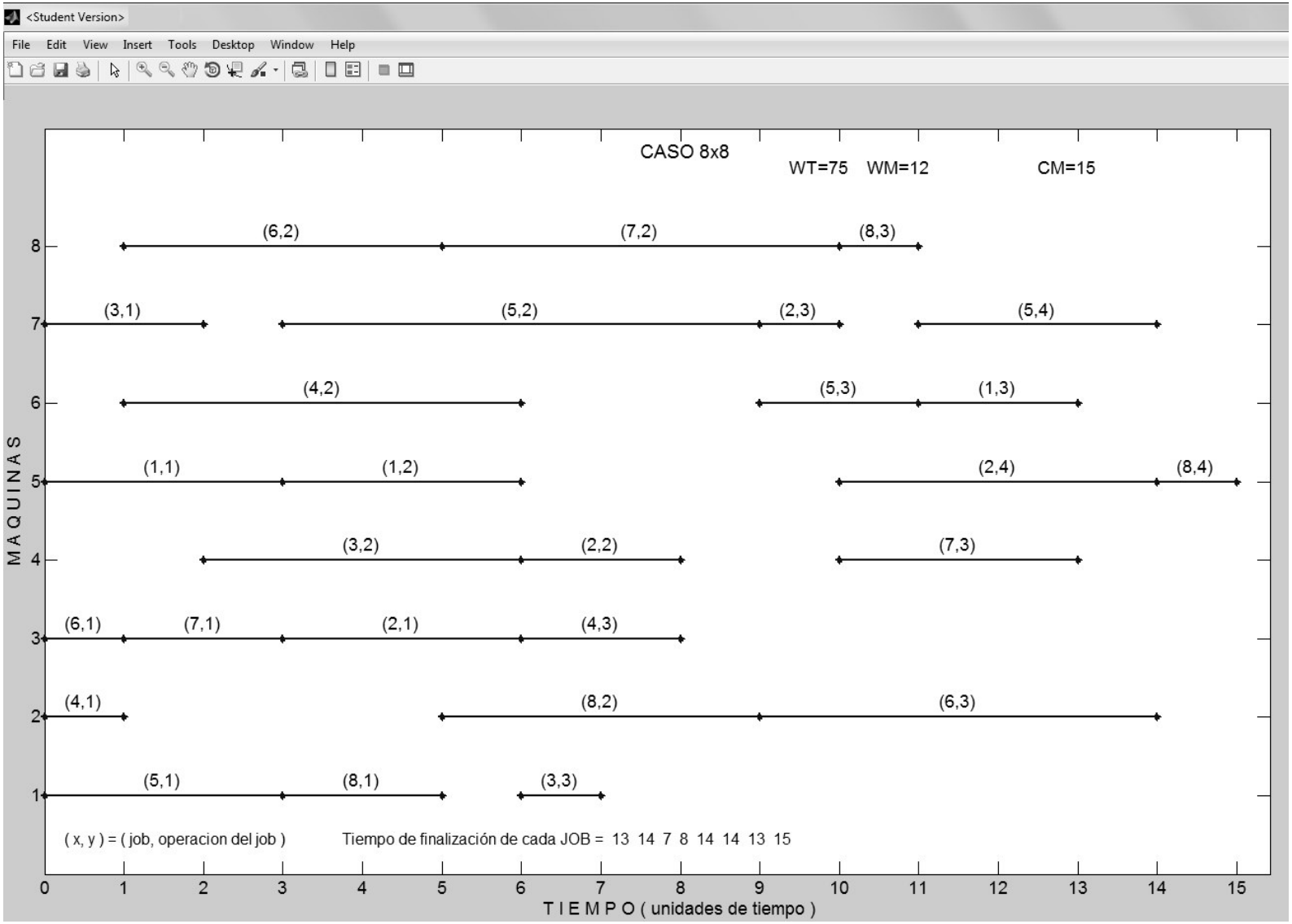


Figura 61. **Diagrama de Gantt: *FJSSP* 8 *Jobs*, 8 máquinas, 27 operaciones**. *Fuente*. Elaboración propia

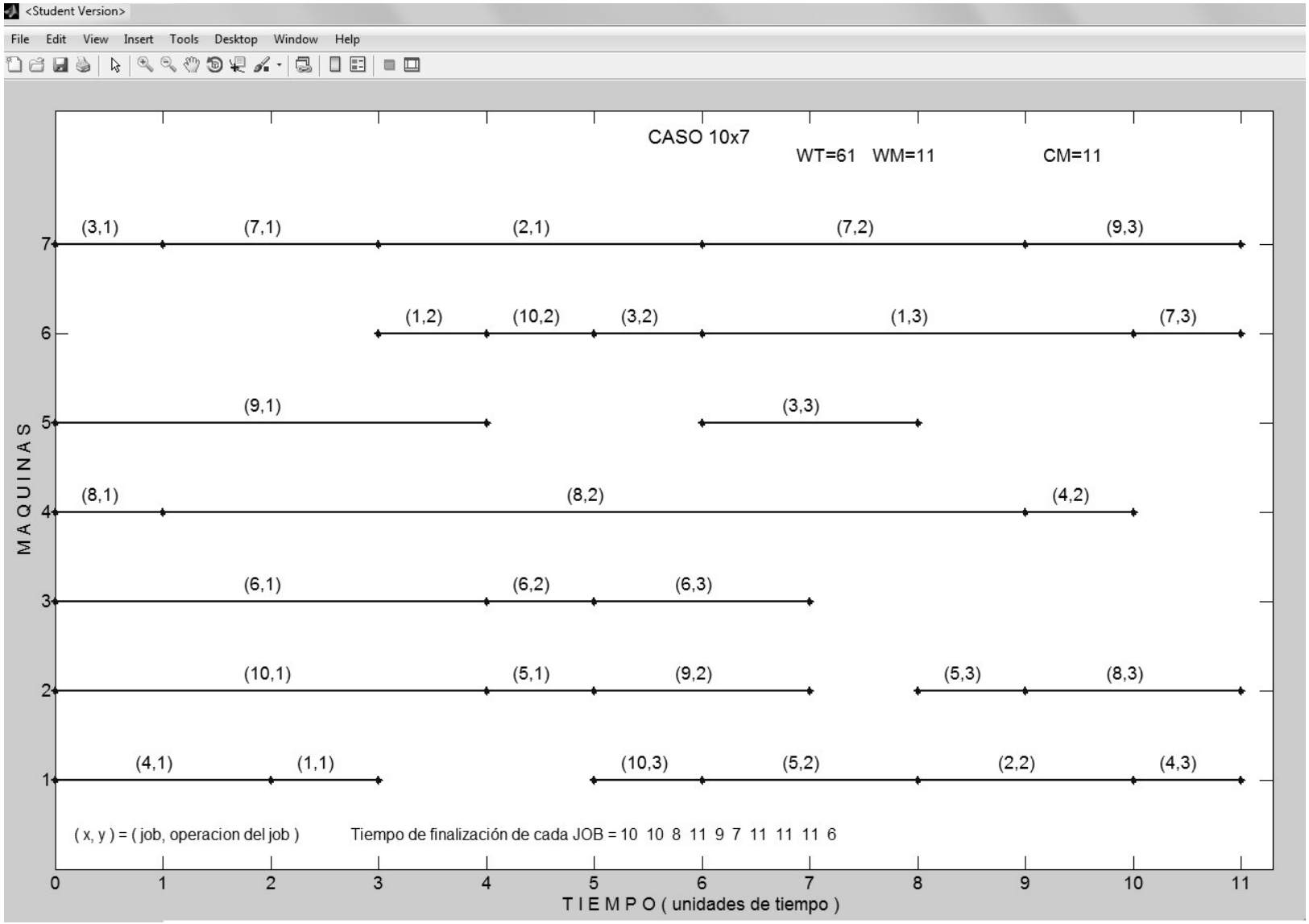


Figura 62. **Diagrama de Gantt: *FJSSP* 10 *Jobs*, 7 máquinas, 29 operaciones**. *Fuente*. Elaboración propia

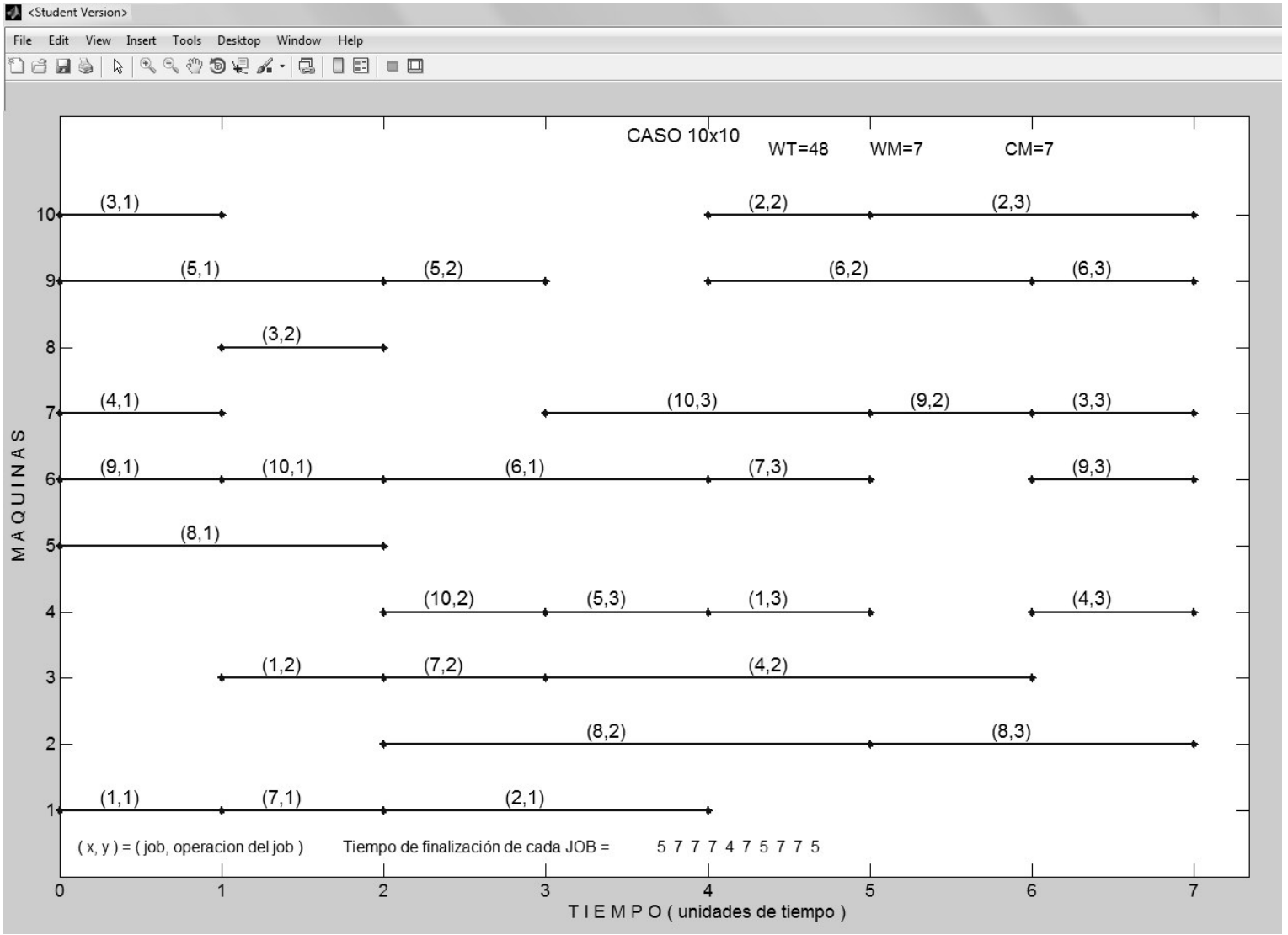


Figura 63. **Diagrama de Gantt: *FJSSP* 10 *Jobs*, 10 máquinas, 30 operaciones**. *Fuente*. Elaboración propia.

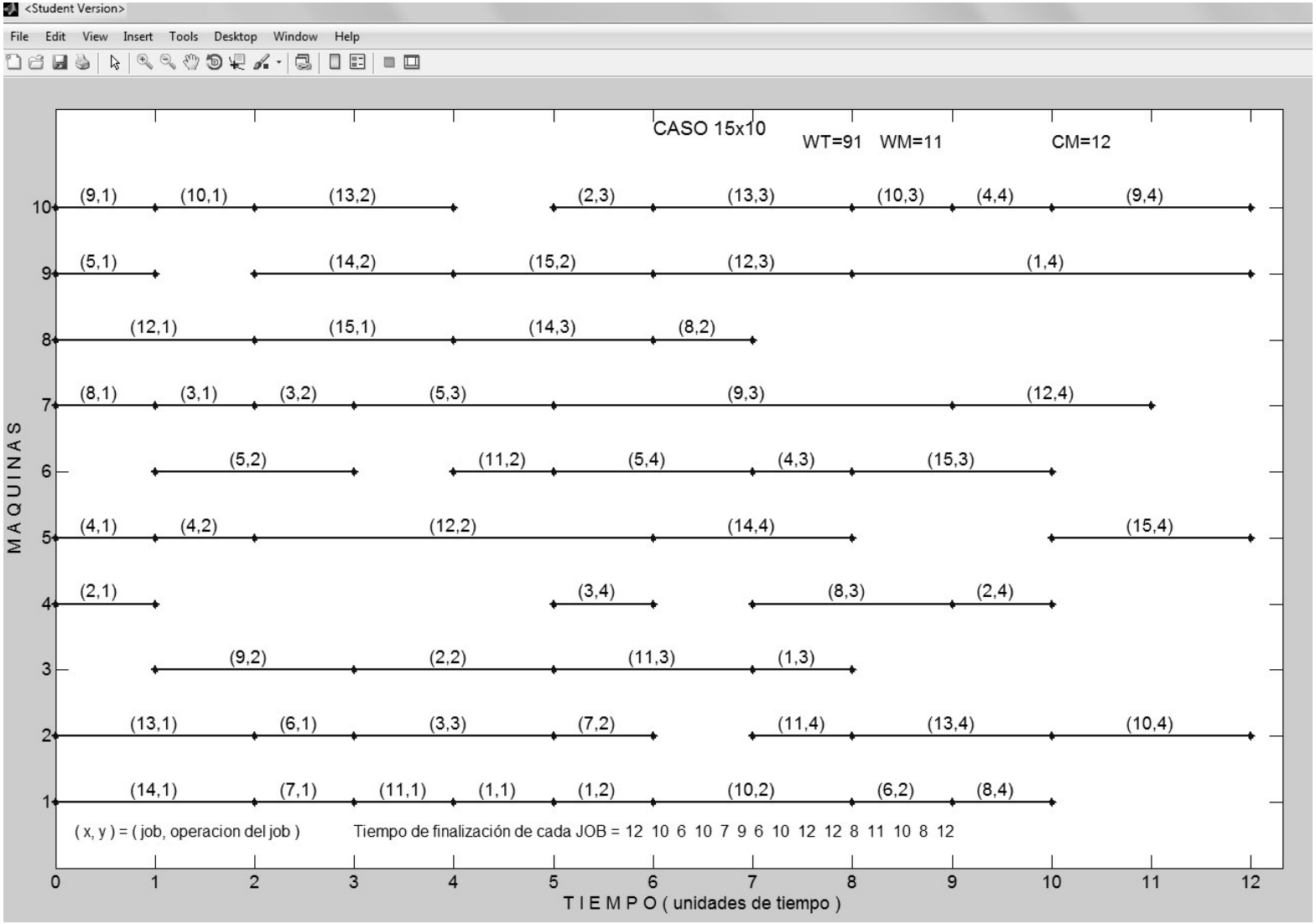


Figura 64. **Diagrama de Gantt: *FJSSP* 15 *Jobs*, 10 máquinas, 56 operaciones**. *Fuente*. Elaboración propia.

4..3 Eficacia y Eficiencia de los Algoritmos

Se comparan los resultados de la propuesta de la tesis con la de otros investigadores con la finalidad de probar la tercera hipótesis, la cual se transcribe a continuación:

H3: Se probará la eficacia y eficiencia de los algoritmos propuestos considerando la existencia de resultados obtenidos por otros investigadores de la literatura que han probado sus algoritmos con los casos de sistemas de fabricación tipo FJS planteados por Kacem.

Los cinco casos de *FJSSP* especificados por Kacem et al. (2002) y Kacem et al. (2002a) han sido utilizados para probar el funcionamiento y desempeño del Algoritmo Genético de Rutas y del Algoritmo Genético de Secuencias, tal como lo han realizado otros investigadores de la literatura con la finalidad de probar sus técnicas de solución, por esta razón los resultados han podido ser comparados.

Para cada caso, el Algoritmo Genético de Rutas y el Algoritmo Genético de Secuencias, han corrido veinte veces. Es decir, se han obtenido veinte soluciones para el caso *FJSSP* 4x5, veinte para el caso *FJSSP* 8x8, veinte para el caso *FJSSP* 10x7, veinte para el caso *FJSSP* 10x7 y veinte para el caso *FJSSP* 15x10. En la Tabla 23, se muestra la comparación de los resultados de los Algoritmos propuestos en este trabajo de tesis con los reportados por otros investigadores, quienes han utilizado diferentes técnicas para solucionar el *FJSSP* (vease ítem 2.2). También, los investigadores consutados han reportado más de una respuesta para cada caso. El mejor criterio para evaluar las diferentes respuestas es el *Makespan* (C_M), ya que éste criterio determina el tiempo necesario con el cual se completan el procesamiento de todos los *Jobs*.

Tabla 23
Comparación de Resultados de la Propuesta de la Tesis con Otras Propuestas

		CASO *FJSSP* (JOBS X MÁQUINAS)														
		4x5			8x8			10x7			10x10			15x10		
INVESTIGADOR		W_T	W_M	C_M	W_T	W_M	C_M	W_T	W_M	C_M	W_T	W_M	C_M	W_T	W_M	C_M
Kacem et al. (2002a)	R1	32	7	16				60	9	15	41	5	7	91	10	23
	R2	34	10	16				64	10	17	45	5	7	95	11	23
	R3	33	7	18				63	10	18	41	7	8	91	11	24
Kacem et al. (2002)	R1	-	-	-	79	13	15	-	-	-	45	5	7	-	-	-
	R2	-	-	-	75	13	16	-	-	-	-	-	-	-	-	-
Xia & Wu (2005)	R1	-	-	-	75	12	15	-	-	-	44	6	7	91	11	12
	R2	-	-	-	73	13	16	-	-	-	-	-	-	-	-	-
Zhang, et al. (2009)	R1	32	10	11	77	12	14	-	-	-	43	6	7	93	11	11
	R2	-	-	-	75	12	15	-	-	-	-	-	-	-	-	-
Xing et al. (2009)	R1	32	8	12	77	12	14	61	11	11	42	6	7	91	11	11
	R2	-	-	-	76	12	15	62	10	11	42	5	8	93	10	11
Li et al. (2010)	R1	32	10	11	77	12	14	61	11	11	43	5	7	91	11	11
	R2	32	8	12	75	12	15	62	10	11	42	6	7	93	10	11
	R3	-	-	-	-	-	-	-	-	-	42	5	8	-	-	-
PROPUESTA DE TESIS (2016)	R1	32	10	11	75	12	15	61	11	11	42	6	7	91	11	12
	R2	32	10	12	76	12	15	61	11	12	42	6	8	91	11	13
	R3	32	9	13	-	-	-	-	-	-	42	6	9	-	-	-

R1= Respuesta 1, R2= Respuesta 2, R3= Respuesta 3

Fuente. Elaboración propia

En la última fila de la Tabla 23, están los resultados de la propuesta de la tesis para todos los casos de *FJSSP*. El resultado obtenido, C_M=11, para el caso de *FJSSP* 4x5 supera a Kacem e iguala a la de los otros investigadores. En el caso de *FJSSP* 8x8 el resultado, C_M=15, no supera por una unidad de tiempo a los resultados de tres investigadores, en los otros casos iguala a las soluciones. En

el caso de *FJSSP* 10x10 el resultado, C_M=11, iguala a las demás y finalmente en el caso de *FJSSP* 15x10 el resultado, C_M=12, supera a Kacem, iguala con Xia&Wu (2005), pero no supera en una unidad de tiempo a los otros investigadores. **Por consiguiente, al conseguirse resultados similares con la mayoría de los investigadores, se comprueba la Eficacia del Algoritmo Genético de Rutas y del Algoritmo Genético de Secuencias.**

En la última fila de la Tabla 24, se muestra los tiempos promedio de ejecución del Algoritmo Genético de Rutas y el Algoritmo Genético de Secuencias propuestos en la tesis para todos los casos de *FJSSP*, los resultados son comparados con los obtenidos por investigadores que han reportado estos datos. Se puede verificar que en todos los casos la propueta de este trabajo supera a las demás. **Por consiguiente, se comprueba la Eficiencia del Algoritmo Genético de Rutas y del Algoritmo Genético de Secuencias.**

Por lo tanto, la hipótesis H3 ha ha quedado comprobada.

Tabla 24.
Comparación de Tiempos de Ejecución de los Algoritmos Propuestos con otras Propuestas

	CASO *FJSSP* (JOBS X MÁQUINAS)				
INVESTIGADOR	4x5	8x8	10x7	10x10	15x10
Xing et al. (2009) (promedio 10 corridas)	2.58 s.	39.37 s.	109.99 s.	39.74 s.	865.18 s.
Li et al.(2010) (promedio 20 corridas)	0.15 s.	3.08 s.	2.58 s.	3.12 s.	25.13 s.
PROPUESTA DE TESIS (promedio 20 corridas)	0.09 s.	0.80 s.	1.14 s.	1.62 s.	11.53 s.

Fuente. Elaboración propia

Finalmente, como se ha mencionado anteriormente en el Anexo B, se puede encontrar los resultados de la propuesta de la tesis para veinte corridas.

CAPÍTULO 5: EJEMPLOS REALES DE JSSP, FJSSP Y FUNCIONES OBJETIVO DERIVADAS DEL *MAKESPAN*

A manera de ejemplos ilustrativos se presentan casos de la industria que pueden ser modelados como *Job Shop* y *Flexible Job Shop* y solucionados con los algoritmos propuestos en la tesis. También, en el ítem 5.3, se sugiere la manera como realizar una adaptación a los algoritmos propuestos con la finalidad de optimizar otros criterios de búsqueda derivadas del *Makespan* (como por ejemplo Latencias y Tardanzas). Aunque, esta adaptación pueda ser innecesaria toda vez que el algoritmo propuesto también entrega información del tiempo en que finaliza cada *Job* (ver Diagramas de Gantt del capítulo IV), dato que puede ser utilizado para establecer fechas de compromisos de entrega.

5.1 Ejemplo 1: *Job Shop* en la Industria de Cosméticos

A continuación se presenta un caso real de *Job Shop* encontrado en Vitorino & Tadeu (2015), quienes solucionan un problema real de *Job Shop* con el software Lingo. Primero se describirá el caso y los cálculos que realizan. Luego, con esos mismos datos se encontrará la solución con el algoritmo propuesto en esta tesis.

Vitorino & Tadeu (2015), analizan el caso de la industria de los cosméticos localizada en la Región Metropolitana de Curitiba (Brasil), concretamente, en la sección de Acabado de esa industria. En esta sección, cada producto (tarea) puede seguir por diferentes rutas, los cuales dependen de las características

finales del embalaje. Por lo tanto, la secuencia de la producción no es simple. Los autores modelan matemáticamente el problema para optimizar el *Makespan* con el uso del software Lingo. Posteriormente, a partir de sus resultados con Excel elaboran Diagramas de Gantt.

A continuación, se describe el análisis de Vitorino & Tadeu (2015). El sector productivo de la planta se compone de las secciones de: Manipulación, Control de Calidad, Etiquetado, Llenado, Acabado y Envío (Figura 65).

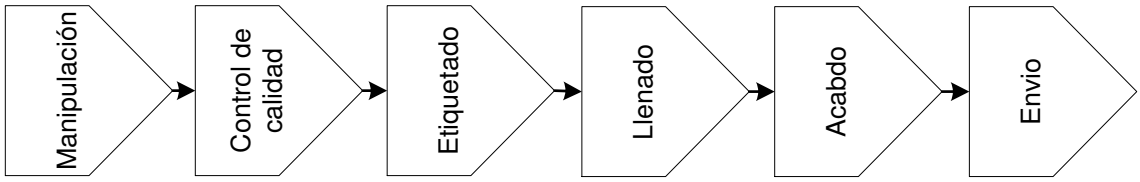


Figura 65. **Flujograma Simplificado del Proceso Productivo de la Industria de Cosméticos**. *Fuente*: Vitorino & Tadeu (2015).

El trabajo se focaliza en la sección de Acabado, que aparece con mayores detalles en la Figura 66. En esta sección hay cuatro posibles rutas de producción y cada producto dependiendo de su especificación o del cliente, puede seguir un camino diferente.

El problema es modelado como *Job Shop*, cada actividad dentro del área de Acabado se define como una máquina y las tareas son los productos que deben pasar por ese proceso de acuerdo con una ruta fija y con tiempos de procesamiento predeterminados. En la Tabla 25, se muestra la operación realizada por cada máquina y su tiempo de procesamiento promedio, obtenido por observación práctica (según reportan los autores).

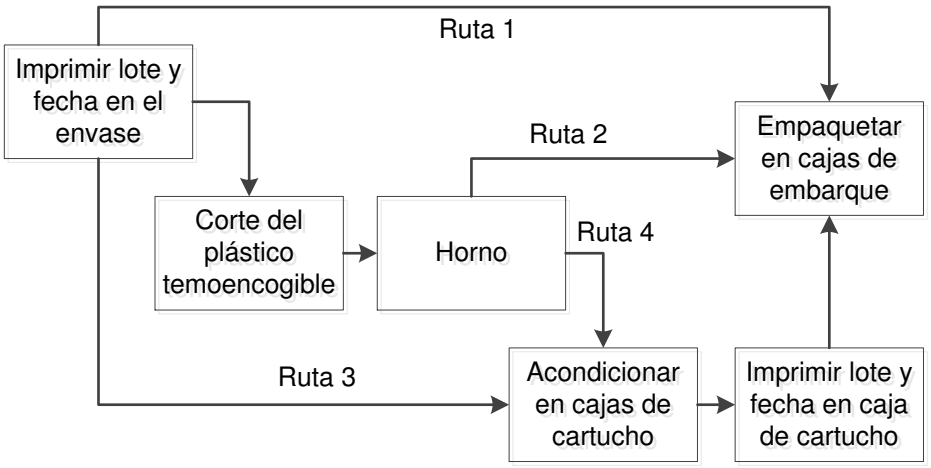


Figura 66. **Rutas Posibles de Producción.** *Fuente*: Vitorino & Tadeu (2015).

Tabla 25
Funciones de las Máquina y Tiempos de Procesamiento

Máquina	Descripción	Tiempo medio de procesamiento (min)
M1	Imprimir lote y fecha en el envase	0.08
M2	Empaquetar en cajas de embarque	0.20
M3	Corte de plástico termoencogible	0.12
M4	Horno	0.17
M5	Acondicionar en cajas de cartucho	0.08
M6	Imprimir lote y fecha en cajas de cartucho	0.08

Fuente: Vitorino & Tadeu (2015).

En la Tabla 26, se muestran las tareas que se programan, indicando la ruta y demanda, este último dato, los autores lo han estimado a partir de información proporcionada por la empresa. La Tabla 27 corresponde a la matriz de operaciones, que indica el orden de las máquinas que cada tarea recorre dentro de la sección de acabado. La Tabla 28, define el tiempo que cada tarea permanece en cada máquina. Cada tarea, en este caso, se determina como el conjunto de productos que deben ser procesados, el cual corresponde a la demanda (ver Tabla 26). Por lo que, el tiempo de procesamiento de cada tarea

es el tiempo de procesamiento de un ítem multiplicado por su demanda. Los autores de la investigación, a partir del modelo matemático del problema, con el software Lingo, obtienen los resultados de la Tabla 29.

Tabla 26
Tareas con su Respectiva Ruta y Demanda

Tarea	Descripción	Ruta	Demanda
J1	Crema Hidratante	4	1300
J2	Colonia	3	4000
J3	Jabón líquido	1	2800
J4	Protector solar	2	1100
J5	Gel crema nutritiva	4	1500
J6	Desodorante	3	4500
J7	Shampoo	1	3100
J8	Aceite	2	2800

Fuente: Vitorino & Tadeu (2015).

Tabla 27
Matriz de Operaciones

Tarea	Orden de las máquinas					
J1	M1	M3	M4	M5	M6	M2
J2	M1	M5	M6	M2	-	-
J3	M1	M2	-	-	-	-
J4	M1	M3	M4	M2	-	-
J5	M1	M3	M4	M5	M6	M2
J6	M1	M5	M6	M2	-	-
J7	M1	M2	-	-	-	-
J8	M1	M3	M4	M2	-	-

Fuente. Vitorino & Tadeu (2015).

Tabla 28
Matriz de Tiempos de Procesamiento

Tarea	Tiempos de procesamiento (min)					
	Máquinas					
	M1	M2	M3	M4	M5	M6
J1	108.33	260.00	151.67	216.67	108.33	104.00
J2	333.33	800.00	-	-	333.33	320.00
J3	233.33	560.00	-	-	-	-
J4	91.67	220.00	128.33	183.33	-	-
J5	125.00	300.00	175.00	250.00	125.00	120.00
J6	375.00	900.00	-	-	375.00	360.00
J7	258.33	620.00	-	-	-	-
J8	233.33	560.00	326.67	466.67	-	-

Fuente. Vitorino & Tadeu (2015).

Tabla 29
Resultados Obtenidos con el Software Lingo

Tarea	Tiempos de procesamiento (min)					
	Máquinas					
	M1	M2	M3	M4	M5	M6
J1	453.33	1325.67	605.00	853.33	961.67	1065.67
J2	1228.33	3628.33	-	-	156.67	1881.67
J3	233.33	793.33	-	-	-	-
J4	325.00	1013.33	453.33	636.67	-	-
J5	621.67	1648.33	853.33	1103.33	1228.33	1348.33
J6	1950.00	4528.33	-	-	3268,33	3628.33
J7	2208.33	2828.33	-	-	-	-
J8	855.00	2208.33	1181.67	1648.33	-	-

Fuente. Vitorino & Tadeu (2015).

La última operación en realizarse es la que corresponde a la tarea 6 en la máquina 2 con un tiempo de 4,548.33 min, equivalente a aproximadamente 75 horas. Por lo tanto, para la realización de todas las tareas son necesarios cinco días de trabajo, teniendo en cuenta dos turnos de 8 horas.

5.1.1. Cálculos con Algoritmos Propuestos en la Tesis

Primeramente, los datos de la Tabla 27 y Tabla 28 del trabajo de Vitorino & Tadeu (2015) se han transferido al formato de tabla utilizado en la tesis, tal como se muestra en la Tabla 30.

En la Tabla 30, la primera columna corresponde a las tareas (*Jobs*), las cuales son ocho (J1…J8), la segunda corresponde a las operaciones, en donde los números entre paréntesis ordenan sin distinción a todas las operaciones del problema, mientras que en la notación "$O_{i,\ j}$" el subíndice **i** hace referencia al número de *Job* y el subíndice **j** a la operación del *Job*. Así, por ejemplo, el *Job* 1 (J1) correspondiente a la Crema Hidratante, procesada por la ruta 4, tiene seis operaciones en total, que también se indican como: $O_{1,1}$, $O_{1,2}$, $O_{1,3}$, $O_{1,4}$, $O_{1,5}$ y $O_{1,6}$. Cada una explicada como sigue:

- $O_{1,1}$: Imprimir lote y fecha en el envase (Realizada por M1 en 108.33 min.).
- $O_{1,2}$: Corte de plástico termoencogible (Realizada por M3 en 151.67 min).
- $O_{1,3}$: Horno (realizada por M4 en 216,67).
- $O_{1,4}$: Acondicionar en cajas de cartucho (realizada por M5 en 108.33 min.).
- $O_{1,5}$: Imprimir lote y fecha en cajas de cartucho (realizada por M6 en 104 min).
- $O_{1,6}$: Empaquetar en cajas de embarque (realizada por M2 en 260 min).

Tabla 30
Datos del JSSP 8x6

Tarea (*Job*)	Operación	M_1	M_2	M_3	M_4	M_5	M_6
J1 (Crema Hidratante)	(1) $O_{1,1}$	108.33	-	-	-	-	-
	(2) $O_{1,2}$	-	-	151.67	-	-	-
	(3) $O_{1,3}$	-	-	-	216.67	-	-
	(4) $O_{1,4}$	-	-	-	-	108.33	-
	(5) $O_{1,5}$	-	-	-	-	-	104.00
	(6) $O_{1,6}$	-	260.00	-	-	-	-
J2 (Colonia)	(7) $O_{2,1}$	333.33	-	-	-	-	-
	(8) $O_{2,2}$	-	-	-	-	333.33	-
	(9) $O_{2,3}$	-	-	-	-	-	320.00
	(10) $O_{2,4}$	-	800.00	-	-	-	-
J3 (Jabón líquido)	(11) $O_{,3,1}$	233.33	-	-	-	-	-
	(12) $O_{3,2}$	-	560.00	-	-	-	-
J4 (Protector Solar)	(13) $O_{4,1}$	91.67	-	-	-	-	-
	(14) $O_{4,2}$	-	-	128.33	-	-	-
	(15) $O_{4,3}$	-	-	-	183.33	-	-
	(16) $O_{4,4}$	-	220.00	-	-	-	-
J5 (Gel crema nutritiva)	(17) $O_{5,1}$	125.00	-	-	-	-	-
	(18) $O_{5,2}$	-	-	175.00	-	-	-
	(19) $O_{5,3}$	-	-	-	250.00	-	-
	(20) $O_{5,4}$	-	-	-	-	125.00	-
	(21) $O_{5,5}$	-	-	-	-	-	120.00
	(22) $O_{5,6}$	-	300.00	-	-	-	-
J6 (Desodorante)	(23) $O_{6,1}$	375.00	-	-	-	-	-
	(24) $O_{6,2}$	-	-	-	-	375.00	-
	(25) $O_{6,3}$	-	-	-	-	-	360.00
	(26) $O_{6,4}$	-	900.00	-	-	-	-
J7 (Shampoo)	(27) $O_{7,1}$	258.33	-	-	-	-	-
	(28) $O_{7,2}$	-	620.00	-	-	-	-
J8 (Aceite)	(29) $O_{8,1}$	233.33	-	-	-	-	-
	(30) $O_{8,2}$	-	-	326.67	-	-	-
	(31) $O_{8,3}$	-	-	-	466.67	-	-
	(32) $O_{8,4}$	-	560.00	-	-	-	-
	Suma	1758.32	4220	781.67	1116.67	941.66	904

Fuente: Elaboración propia, datos: Vitorino & Tadeu (2015).

En la Tabla 30 se puede distinguir que se trata de un problema de *Job Shop* (existe una máquina disponible para cada operación). Sin embargo, el algoritmo no lo sabe y trata el problema como si fuera uno de *Flexible Job Shop*. Así, el

Algoritmo Genético de Rutas, progresivamente en cada generación de la población de cromosomas debe encontrar las máquinas indicadas. Esto se vuelve posible, porque en los espacios en donde no hay máquinas se colocan números muy grandes, de tal manera que durante el proceso se irán descartando. En la última fila de la Tabla 30 se ha calculado las cargas por máquina, de donde $W_M = 4220$ y al sumar la fila $W_T = 9722.32$, estos valores, al ser conocidos, son los umbrales de parada para el Algoritmo Genético de Rutas. Para evitar errores por decimales, vamos a considerar umbrales de $W_M = 4220$ y $W_T = 9723$ y para el *Makespan* tomaremos un criterio de parada hasta que encuentre un umbral menor que obtenido en el problema anterior, por ejemplo: 4500.

Se realizaron 20 corridas, en todas ellas se lograron obtener las máquinas esperadas y el *Makespan* en todos los casos fue menor a 4500, siendo el mejor de 4453.33. El tiempo medio de ejecución de cada corrida fue de 5.46 seg.

La secuencia obtenida para la mejor respuesta se muestra en la Tabla 31. En esta tabla, los números entre paréntesis son los mismos que figuran en la columna "Operaciones" de la Tabla 30. Así por ejemplo la secuencia de operaciones: (11) → (1) → (27) es equivalente a: ($O_{3,1}$) → ($O_{1,1}$) → ($O_{7,1}$).

Tabla 31
Respuesta de Secuencia Óptima del JSSP 8x6

Orden en que se deben ejecutar las Operaciones (C_M =4453.33)
(11)→(1)→(27)→(2)→(12)→(7)→(3)→(13)→(17)→(14)→(18)→(28)→(15)→(4)→(8)→(16)→(19)→(23)→(9)→(20)→(24)→(10)→(29)→(5)→(30)→(31)→(25)→(21)→(6)→ (26)→(22)→(32)

Fuente: Elaboración propia

El Diagrama de Gantt, de la secuencia de la Tabla 31, generada automáticamente, se muestra en la Figura 67. La última operación en ejecutarse es la $O_{8,4}$ que corresponde a la última operación del *Job* "Aceite"

ejecutada en la máquina 2. En la Figura 67 puede verificarse objetivamente el resultado del *Makespan* obtenido, es de 4,453.33 minutos, el cual es mejor que el obtenido con el software Lingo descrito en el ítem anterior (4,548.33 min). También en el Diagrama de Gantt se muestra impreso los tiempos de conclusión de cada *Job*. Las condiciones en la que corrió el programa fueron:

- Número de reinicio de poblaciones de rutas = 1 (Sin reinicio)
- MAXIMOWORKLOAD (umbral) = 4220
- TOTALWORKLOAD (umbral) = 9723
- *MAKESPAN* (umbral) = 4500
- Tamaño de población de rutas = 20
- Generaciones en población de rutas = 20000
- Porcentaje mutación en población rutas = 1
- Porcentaje sobrevivientes en población rutas= 20
- Tamaño de población de secuencias = 600
- Porcentaje mutación en población secuencia = 1.000000e-01
- Porcentaje sobrevivientes en población secuencia = 50
- Generaciones en población de secuencias = 200
- Número de corridas = 20

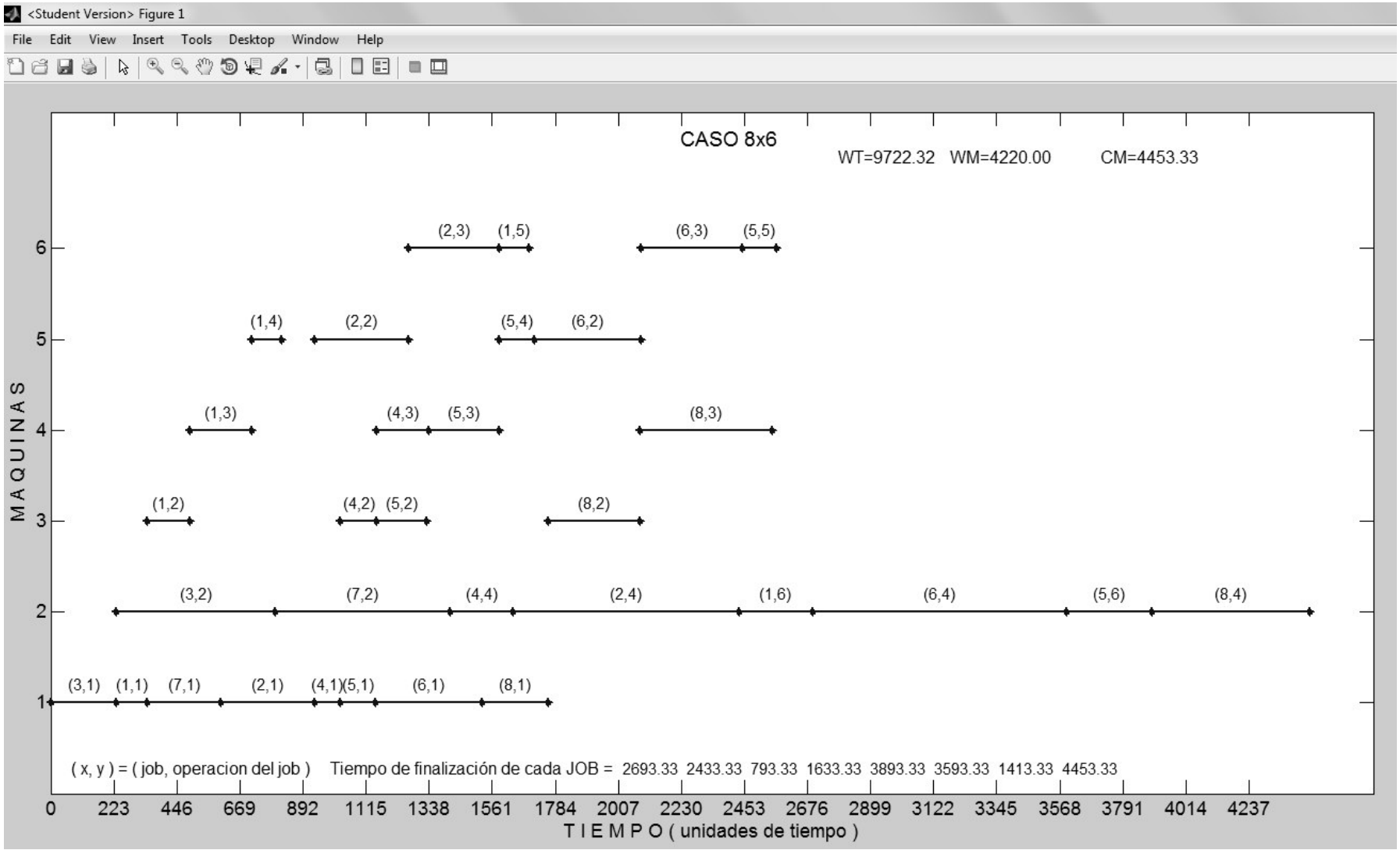


Figura 67. **Diagrama de Gantt *JSSP* 8x6 - Obtenido con Algoritmos Propuestos.** *Fuente*: Elaboración propia

5.2 Ejemplo 2: Moldes Utilizados en la Industria Automovilística

A continuación se presenta un caso real encontrado en Roshanaei (2012), se trata de una empresa tipo *FJSS* que fabrica moldes para la industria automovilística de América del Norte. Roshanaei en su trabajo desarrolla la solución al problema de *FJSS* minimizando solo el objetivo del *Makespan,* ha utilizado un algoritmo hibrido Recocido Simulado Inmune y Artificial (*Artificial Immune and Simulated Annealing* - AISA).

Los tiempos de procesamiento de todas las máquinas disponibles para cada operación no se informan en el trabajo de Roshanaei (2012), solo aparecen las que figuran en su Diagrama de Gantt. Es por eso, que con algunos de esos datos y la información de la Tabla 33 se han generado datos para solucionar el problema. Por esta razón, no se tiene la intención de comparar resultados con los de Roshanaei (2012). Si no más bien, se pretende mostrar un caso complejo de *FJSS* del mundo real, el cual luego de analizar su funcionamiento puede ser sintetizado a un formato de tabla utilizado en esta tesis para viabilizar su solución mediante los algoritmos propuestos.

En Roshanaei (2012), se describen las actividades detalladas de un taller de trabajo de moldeo de una empresa que suministra a otras de la industria automovilística de América del Norte. El estudio se limita a la programación de operaciones de mecanizado en el taller, mientras que las operaciones de montaje del molde después de completar las operaciones de mecanizado no se consideran.

La compañía fabrica moldes y troqueles (matrices) para crear productos tales como lentes posteriores o reflectores de lámpara delantera para los fabricantes de automóviles. Los datos utilizados en el estudio incluyen:

- Tipo de grupos de máquinas, nombres y número de máquinas en cada grupo.
- Capacidades para cada máquina en un grupo que incluye uso típico/principal, alternado, etc.
- Turnos de trabajo: número normal de turnos / por día, número de horas por turno. Esto podría ser cambiado debido al uso de horas extraordinarias.
- Jerarquía del molde (componentes del molde y sus instancias y códigos para el molde compuesto).
- Tiempos de procesamiento normales para cada operación.
- El estado actual de la máquina con carga (cuál máquina está trabajando en cuál parte/operación).
- Aproximado tiempo de configuración y desmantelamiento: Constante o fracción de cada tiempo de operación, las cuales están incluidas en los tiempos de procesamiento.

Un molde típico consiste de las siguientes partes:

a) Cavidad
b) Núcleo
c) Corredoras
d) Retractor
e) Placas de sujeción.

Se utilizan catorce (14) máquinas de control numérico capaces de realizar las siguientes operaciones:

1) Desbaste
2) Alivio del estrés
3) Semi-acabados
4) Acabado
5) Mandrinar
6) Perforar a pistola
7) Carbono

8) Maquinado por descarga eléctrica (EDM)

La Cavidad y Núcleo requieren de las ocho operaciones: Desbaste, Alivio del estrés, Semi-acabados, Acabado, Mandrinar, Perforar a pistola, Carbono. Las Corredoras necesitan sólo cuatro operaciones: Desbaste, Acabado, carbono y EDM. Los Retractores requieren dos operaciones: Desbaste y Acabado. Las Placas de Sujeción sólo necesitan la operación: Mandrinar.

La compañía utiliza las máquinas CNC para el maquinado de los moldes como se muestra en la Tabla 32, en donde se ilustra los requisitos operacionales para la fabricación de cuatro (4) moldes, seccionados en 20 Partes. La Parte 1 tiene como primera operación al Desbaste y se denota como O_{11} que significa primera operación de la Parte 1. La Parte 4 tiene como segunda operación al Acabado y se denota por O_{42} que representa la segunda operación de la Parte 4. Por tanto, el primer índice j de O_{jl} indica el número de parte y el segundo índice i, denota la operación requerida.

La Tabla 33, muestra los nombres y funcionalidades de cada máquina, revela que las máquinas son flexibles pero no son de flexibilidad idénticas. Para cada operación hay un número reducido de máquinas, no las catorce, que están disponibles de procesar cada operación. Por lo tanto, el sistema de fabricación es de Flexibilidad Parcial (*FP-JSSP*) ya que cada máquina realiza ciertas operaciones y por lo tanto diferentes partes no tienen total libertad de enrutamiento.

La compañía para asegurar una carga de trabajo equitativa entre las máquinas ha priorizado o ponderado a cada una de ellas, ver Tabla 34. Los coeficientes se han multiplicado por los tiempos de procesamiento de las operaciones en cada máquina. Cuanto mayor sea el coeficiente asignado a una máquina, es el menos deseable para el procesamiento de ciertas operaciones. La imagen típica del molde se muestra en la Figura 68.

Tabla 32
Código Operacional y Requisitos de las Partes

No	No	Nombre de Parte	Operaciones requeridas
	Parte 1	Cavidad 1	O_{11}, O_{12}, O_{13}, O_{14}, O_{15}, O_{16}, O_{17}, O_{18}
	Parte 2	Núcleo 1	O_{21}, O_{22}, O_{23}, O_{24}, O_{25}, O_{26}, O_{27}, O_{28}
Molde #1	Parte 3	Corredoras 1	O_{31}, O_{32}, O_{33}, O_{34}
	Parte 4	Retractor 1	O_{41}, O_{42}
	Parte 5	Placas de sujeción 1	O_{51}
	Parte 6	Cavidad 2	O_{61}, O_{62}, O_{63}, O_{64}, O_{65}, O_{66}, O_{67}, O_{68}
	Parte 7	Núcleo 2	O_{71} O_{72}, O_{73}, O_{74}, O_{75}, O_{76}, O_{77}, O_{78}
Molde #2	Parte 8	Corredoras 2	O_{81}, O_{82}, O_{83}, O_{84}
	Parte 9	Retractor 2	O_{91}, O_{92}
	Parte 10	Placas de sujeción 2	O_{101}
	Parte 11	Cavidad 3	O_{111}, O_{112}, O_{113}, O_{114}, O_{115}, O_{116}, O_{117}, O_{118}
	Parte 12	Núcleo 3	O_{121} O_{122}, O_{123}, O_{124}, O_{125}, O_{126}, O_{127}, O_{128}
Molde #3	Parte 13	Corredoras 3	O_{131}, O_{132}, O_{133}, O_{134}
	Parte 14	Retractor 3	O_{141}, O_{142}
	Parte 15	Placas de sujeción 3	O_{151}
	Parte 16	Cavidad 4	O_{161}, O_{162}, O_{163}, O_{164}, O_{165}, O_{166}, O_{167}, O_{168}
	Parte 17	Núcleo 4	O_{171} O_{172}, O_{173}, O_{174}, O_{175}, O_{176}, O_{177}, O_{178}
Molde #4	Parte 18	Corredoras 4	O_{181}, O_{182}, O_{183}, O_{184}
	Parte 19	Retractor 4	O_{191}, O_{192}
	Parte 20	Placas de sujeción 4	O_{201}

Fuente. Roshanaei, 2012.

Tabla 33
Funciones de las Máquinas

Operaciones	Nombre de máquinas elegibles
Desbaste	M_1 (AWEA) - M_2 (Johnford) - M_3 (Dynamic) - M_4 (Eumach)
Alivio del estrés	M_5 (Outsourced)
Semi-acabados	M_4 (Eumach) - M_3 (Dynamic)
Acabado	M_6 (Exceeder 1) - M_7 (Exceeder 2)
Mandrinar	M_8 (Kuraki) - M_9 (Parpas) - M_{10} (Takumi)
Perforar a pistola	M_{11} (Outsourced)
Carbono	M_3 (Dynamic) - M_{12} (Datic)
Maquinado por descarga eléctrica (EDM)	M_{13} (Techno) - M_{14} (NX8)

Fuente. Roshanaei, 2012

Tabla 34
Coeficientes de Prioridad Asignadas en la Planta a las Máquinas

Operaciones	Nombre de máquinas elegibles
Desbaste	(1) M_1 - (1.1) M_2- (1.2) M_3 - (1.3) M_4
Alivio del estrés	(1) M_5
Semi-acabados	(1) M_4 - (1.1) M_3
Acabado	(1) M_6 - (1)M_7
Mandrinar	(1) M_8 - (1.1) M_9 - (1.2) M_{10}
Perforar a pistola	(1) M_{11}
Carbono	(1) M_3 - (1.1) M_{12}
Maquinado por descarga eléctrica (EDM)	(1) M_{13} -(1.1) M_{14}

Fuente: Roshanaei, 2012

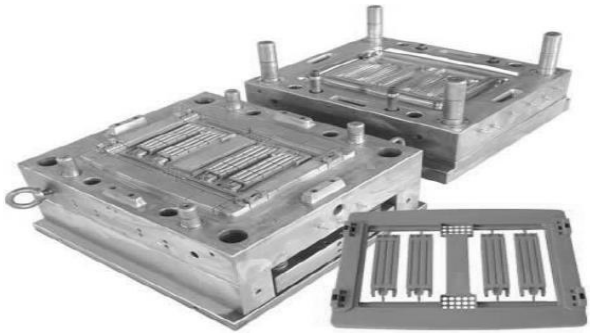


Figura 68. **Molde (Superior) - Producto Moldeado (Inferior Derecha)**. *Fuente*: Roshanaei, 2012

La Figura 69, presenta la solución en un diagrama de Gantt. Se cuenta con 20 *Jobs* (j = 20) y 14 máquinas CNC (m = 14). Esto se considera un ejemplo de gran tamaño de *FJSSP*. En cada celda, el nombre de cada operación se ha incluido. Debajo de cada célula, existen dos números. El primer número es el tiempo de procesamiento de la operación y el segundo número es el tiempo de terminación de esa operación. Por ejemplo, O_{13} en M_4 tiene tiempo de procesamiento de 24 y el tiempo total es 169, que es la suma de los tiempos de procesamiento de las operaciones precedentes de O_{13} (O_{11} = 60 en M_1, O_{12} = 85 en M_5 y O_{13} = 24 en M_4). Cada parte se muestra con un color único para que la ruta de procesamiento de cada parte en diferentes máquinas pueda ser rastreada.

El Algoritmo Recocido Simulado Inmune y Artificial (AISA) es aplicado para resolver el caso real de *PF-JSSP* con 20 partes (92 operaciones) en 14 centros de mecanizado flexibles. El AISA utiliza un tiempo de ejecución de 56 segundos, obteniendo un *Makespan* de 881 horas. Este tiempo es equivalente a 110 turnos de trabajo. La empresa utilizó para la fabricación de estas 92 operaciones en sus recursos actuales alrededor de 960 horas, que era equivalente a 120 turnos de trabajo (dos turnos por día, incluyendo días laborables y fines de semana).

La diferencia entre el tiempo utilizado por la compañía y el tiempo encontrado por la aplicación de la AISA es de 79 horas, lo que equivale a casi diez turnos de trabajo de ahorro. Teniendo en cuenta este ahorro de tiempo de producción, si la empresa decide aplicar el AISA, pueden reducir el tiempo de fabricación de moldes en un 8,3%. Teniendo en cuenta los salarios de los trabajadores, el coste de inventario, gastos generales, etc. Esta reducción del tiempo de mecanizado puede potencialmente resultar en una diferencia significativa en el tiempo, el costo y la eficiencia operativa.

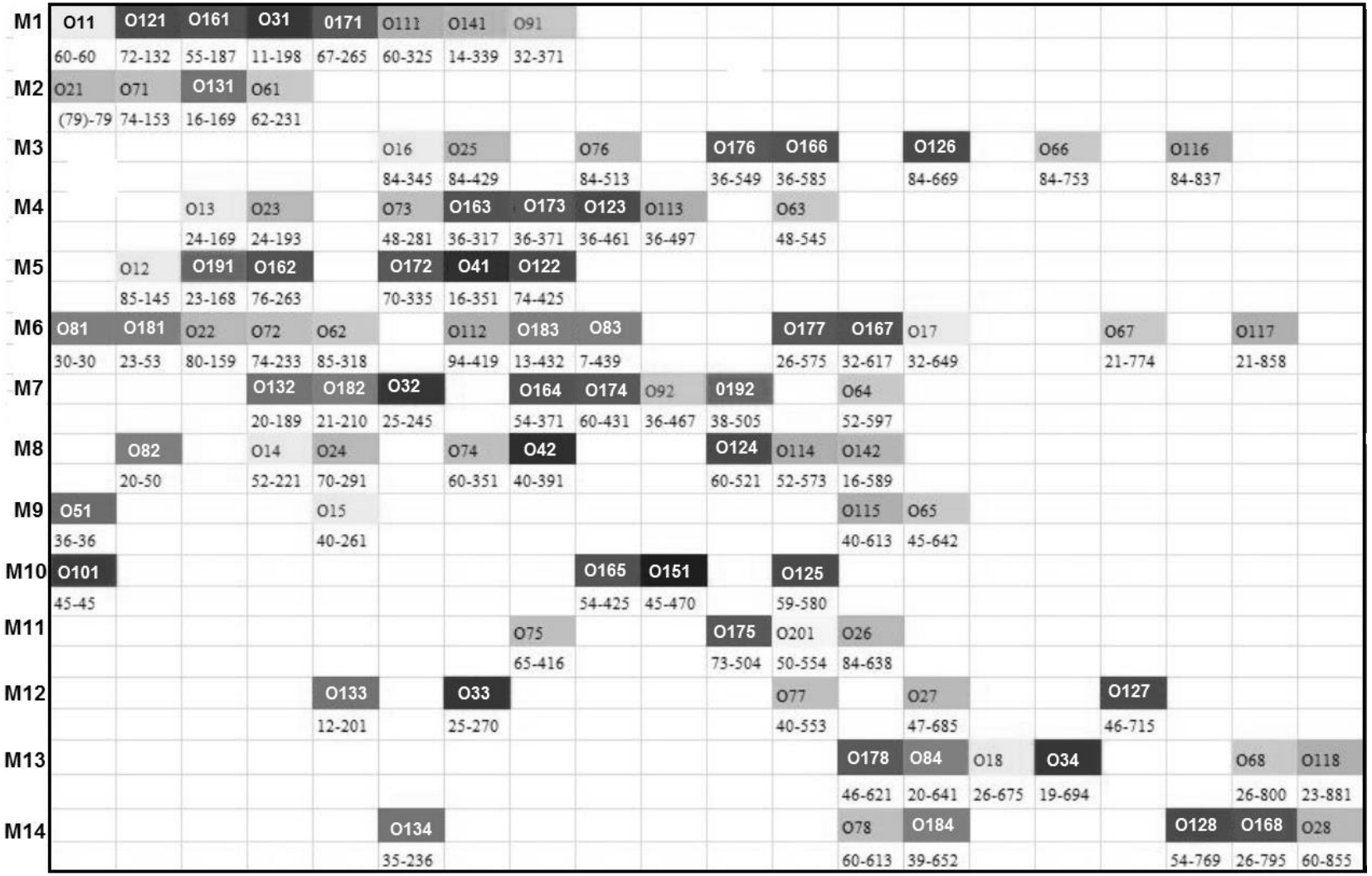


Figura 69. **Solución en Diagrama de Gantt**. *Fuente*: Roshanaei (2012)

5.2.1 Cálculos con Algoritmos Propuestos en la Tesis

En este ítem aplicaremos los Algoritmos Genéticos propuestos en la tesis, al caso de la compañía de moldes y troqueles que ha servido de ilustración para Roshanaei (2012). Al existir 92 operaciones, deberían existir datos de tiempo de ejecución de por lo menos dos máquinas para cada operación. Pero, los datos, es decir, los tiempos de ejecución del conjunto de máquinas disponibles para cada una de las 92 operaciones, utilizados por Roshanai (2012), no se pueden deducir de sus tablas. Sin embargo, del Diagrama de Gantt que presenta el autor (Figura 69) se han extraído algunos datos, los datos faltantes han sido completados al azar pero teniendo en cuenta la flexibilidad de cada operación por procesarse en una de las máquinas según se indica en la Tabla 33.

De acuerdo a lo anterior, en la Tabla 35 se muestran los datos, aquí las operaciones son indicadas como números entre paréntesis que identifican a las 92 operaciones en total, o mediante la notación Oij que identifican a la operación de un *Job* (i) y el número de operación relativo a ese *Job* (j).

La Tabla 35, corresponde a una de *Flexible Job Shop* de Flexibilidad Parcial, ya que las catorce máquinas no están disponibles de ser designadas para cada operación, sino solamente una de un subconjunto más reducido. Así por ejemplo, la operación ochenta seis (86), es decir, la primera operación del *Job* 18 ($O_{18,1}$) que corresponde a la operación de “Desbaste” tiene la opción de ser procesada en una de las cuatro máquinas disponibles: M1, M2, M3, M4.

Tabla 35
Datos Estimados para el FJSSP 20x14

Job	Operación	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}
J1	(1) $O_{1,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(2) $O_{1,2}$	-	-	-	-	70	-	-	-	-	-	-	-	-	-
	(3) $O_{1,3}$	-	-	26	24	-	-	-	-	-	-	-	-	-	-
	(4) $O_{1,4}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
	(5) $O_{1,5}$	-	-	-	-	-	-	-	36	36	36		-	-	-
	(6) $O_{1,6}$	-	-	-	-	-	-	-	-	-	-	36	-	-	-
	(7) $O_{1,7}$	-	-	26	-	-	-	-	-	-	-	-	26	-	-
	(8) $O_{1,8}$	-	-	-	-	-	-	-	-	-	-	-	-	23	23
J2	(9) $O_{2,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(10) $O_{2,2}$	-	-	-	-	70	-	-	-	-	-	-	-	-	-
	(11) $O_{2,3}$	-	-	26	24	-	-	-	-	-	-	-	-	-	-
	(12) $O_{2,4}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
	(13 $O_{2,5}$	-	-	-	-	-	-	-	36	36	36	-	-	-	-
	(14) $O_{2,6}$	-	-	-	-	-	-	-	-	-	-	36	-	-	-
	(15) $O_{2,7}$	-	-	26	-	-	-	-	-	-	-	-	26	-	-
	(16) $O_{2,8}$	-	-	-	-	-	-	-	-	-	-	-	-	23	23
J3	(17) $O_{3,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(18) $O_{3,2}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
	(19) $O_{3,3}$	-	-	26	-	-	-	-	-	-	-	-	26	-	-
	(20) $O_{3,4}$	-	-	-	-	-	-	-	-	-	-	-	-	23	23
J4	(21) $O_{4,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(22) $O_{4,2}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
J5	(23) $O_{5,1}$	-	-	-	-	-	-	-	36	36	36	-	-	-	-
J6	(24) $O_{6,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(25) $O_{6,2}$	-	-	-	-	70	-	-	-	-	-	-	-	-	-
	(26) $O_{6,3}$	-	-	26	24	-	-	-	-	-	-	-	-	-	-
	(27) $O_{6,4}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
	(28) $O_{6,5}$	-	-	-	-	-	-	-	36	36	36		-	-	-
	(29) $O_{6,6}$	-	-	-	-	-	-	-	-	-	-	36	-	-	-
	(30) $O_{6,7}$	-	-	26	-	-	-	-	-	-	-	-	26	-	-
	(31) $O_{6,8}$	-	-	-	-	-	-	-	-	-	-	-	-	23	23
J7	(32) $O_{7,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(33) $O_{7,2}$	-	-	-	-	70	-	-	-	-	-	-	-	-	-
	(34) $O_{7,3}$	-	-	26	24	-	-	-	-	-	-	-	-	-	-
	(35) $O_{7,4}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
	(36) $O_{7,5}$	-	-	-	-	-	-	-	36	36	36	-	-	-	-
	(37) $O_{7,6}$	-	-	-	-	-	-	-	-	-	-	36	-	-	-
	(38) $O_{7,7}$	-	-	26	-	-	-	-	-	-	-	-	26	-	-
	(39) $O_{7,8}$	-	-	-	-	-	-	-	-	-	-	-	-	23	23
J8	(40) $O_{8,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(41) $O_{8,2}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
	(42) $O_{8,3}$	-	-	26	-	-	-	-	-	-	-	-	26	-	-
	(43) $O_{8,4}$	-	-	-	-	-	-	-	-	-	-	-	-	23	23
J9	(44) $O_{9,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(45) $O_{9,2}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
J10	(46) $O_{10,1}$	-	-	-	-	-	-	-	36	36	36	-	-	-	-

Fuente: Elaboración propia, con algunos datos de Roshanaei, 2012

Tabla 35 (continuación…)
Datos Estimados para el FJSSP 20x14

Job	Operación	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}
J11	(47) $O_{11,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(48) $O_{11,2}$	-	-	-	-	70	-	-	-	-	-	-	-	-	-
	(49) $O_{11,3}$	-	-	26	24	-	-	-	-	-	-	-	-	-	-
	(50) $O_{11,4}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
	(51) $O_{11,5}$	-	-	-	-	-	-	-	36	36	36		-	-	-
	(52) $O_{11,6}$	-	-	-	-	-	-	-	-	-	-	36	-	-	-
	(53) $O_{11,7}$	-	-	26	-	-	-	-	-	-	-	-	26	-	-
	(54) $O_{11,8}$	-	-	-	-	-	-	-	-	-	-	-	-	23	23
J12	(55) $O_{12,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(56) $O_{12,2}$	-	-	-	-	70	-	-	-	-	-	-	-	-	-
	(57) $O_{12,3}$	-	-	26	24	-	-	-	-	-	-	-	-	-	-
	(58) $O_{12,4}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
	(59) $O_{12,5}$	-	-	-	-	-	-	-	36	36	36	-	-	-	-
	(60) $O_{12,6}$	-	-	-	-	-	-	-	-	-	-	36	-	-	-
	(61) $O_{12,7}$	-	-	26	-	-	-	-	-	-	-	-	26	-	-
	(62) $O_{12,8}$	-	-	-	-	-	-	-	-	-	-	-	-	23	23
J13	(63) $O_{13,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(64) $O_{13,2}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
	(65) $O_{13,3}$	-	-	26	-	-	-	-	-	-	-	-	26	-	-
	(66) $O_{13,4}$	-	-	-	-	-	-	-	-	-	-	-	-	23	23
J14	(67) $O_{14,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(68) $O_{14,2}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
J15	(69) $O_{15,1}$	-	-	-	-	-	-	-	36	36	36	-	-	-	-
J16	(70) $O_{16,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(71) $O_{16,2}$	-	-	-	-	70	-	-	-	-	-	-	-	-	-
	(72) $O_{16,3}$	-	-	26	24	-	-	-	-	-	-	-	-	-	-
	(73) $O_{16,4}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
	(74) $O_{16,5}$	-	-	-	-	-	-	-	36	36	36		-	-	-
	(75) $O_{16,6}$	-	-	-	-	-	-	-	-	-	-	36	-	-	-
	(76) $O_{16,7}$	-	-	26	-	-	-	-	-	-	-	-	26	-	-
	(77) $O_{16,8}$	-	-	-	-	-	-	-	-	-	-	-	-	23	23
J17	(78) $O_{17,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(79) $O_{17,2}$	-	-	-	-	70	-	-	-	-	-	-	-	-	-
	(80) $O_{17,3}$	-	-	26	24	-	-	-	-	-	-	-	-	-	-
	(81) $O_{17,4}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
	(82) $O_{17,5}$	-	-	-	-	-	-	-	36	36	36	-	-	-	-
	(83) $O_{17,6}$	-	-	-	-	-	-	-	-	-	-	36	-	-	-
	(84) $O_{17,7}$	-	-	26	-	-	-	-	-	-	-	-	26	-	-
	(85) $O_{17,8}$	-	-	-	-	-	-	-	-	-	-	-	-	23	23
J18	(86) $O_{18,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(87) $O_{18,2}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
	(88) $O_{18,3}$	-	-	26	-	-	-	-	-	-	-	-	26	-	-
	(89) $O_{18,4}$	-	-	-	-	-	-	-	-	-	-	-	-	23	23
J19	(90) $O_{19,1}$	60	66	72	78	-	-	-	-	-	-	-	-	-	-
	(91) $O_{19,2}$	-	-	-	-	-	52	52	-	-	-	-	-	-	-
J20	(92) $O_{20,1}$	-	-	-	-	-	-	-	36	36	36	-	-	-	-

Fuente: Elaboración propia, con algunos datos de Roshanaei, 2012

Luego de introducir los datos de la Tabla 35 al programa, y realizar 20 corridas, se han encontrado secuencias de operaciones con un *Makespan* de 817, *Maximum Workload* de 560 y *Total Workload* de 3894.

El programa corrió libremente, solo limitada por el máximo número de generaciones, tanto para el Algoritmo Genético de Rutas como el Algoritmo Genético de Secuencias. Las condiciones ingresadas al programa fueron:

- Tamaño de población de rutas = 20
- Generaciones en población de rutas = 500
- Tamaño de población de secuencias = 400
- Generaciones en población de secuencias = 150
- Porcentaje mutación en población rutas = 1
- Porcentaje sobrevivientes en población rutas = 20
- Porcentaje mutación en población secuencia = 0.1
- Porcentaje sobrevivientes en población secuencia = 50
- Número de corridas = 20

La Tabla 36, muestra la designación de máquinas a cada una de las 92 operaciones (solución del Algoritmo Genético de Rutas).

La Taba 5.13, muestra la secuencia u orden de procesamiento de las operaciones (solución del Algoritmo Genético de Secuencias). Las operaciones se muestran en números entre paréntesis y son los mismos números que figuran en la columna “operación” de la Tabla 36.

Para explicar brevemente la secuencia, se toma por ejemplo los cuatro primeros números, que son: (9)→(10)→(55)→(24), los cuales equivalen a: $O_{2,1}$→$O_{2,2}$→$O_{12,1}$ →$O_{6,1.}$ Con la ayuda de la Tabla 36, se deduce que la operación $O_{2,1}$ es procesada en la máquina 1 en 60 segundos, $O_{2,2}$ en la máquina 5 en 70 segundos, $O_{12,1}$ en la máquina 2 en 66 segundos y $O_{6,1}$ en la máquina 1 en 60 segundos. Es decir, en la máquina 1, primero se procesa $O_{2,1}$ y luego $O_{6,1}$. Este tipo de información es útil para construir un Diagrama de Gantt, el cual es mostrado en la Figura 70.

Tabla 36

Respuesta de Rutas para FJSSP 20x14 (W_M = 560, W_T = 3894)

Job	Operación	Máquina	Tiempo
J1	(1) $O_{1,1}$	1	60
	(2) $O_{1,2}$	5	70
	(3) $O_{1,3}$	4	24
	(4) $O_{1,4}$	7	52
	(5) $O_{1,5}$	10	36
	(6) $O_{1,6}$	11	36
	(7) $O_{1,7}$	12	26
	(8) $O_{1,8}$	13	23
J2	(9) $O_{2,1}$	1	60
	(10) $O_{2,2}$	5	70
	(11) $O_{2,3}$	4	24
	(12) $O_{2,4}$	6	52
	(13 $O_{2,5}$	10	36
	(14) $O_{2,6}$	11	36
	(15) $O_{2,7}$	12	26
	(16) $O_{2,8}$	14	23
J3	(17) $O_{3,1}$	2	66
	(18) $O_{3,2}$	7	52
	(19) $O_{3,3}$	12	26
	(20) $O_{3,4}$	14	23
J4	(21) $O_{4,1}$	2	66
	(22) $O_{4,2}$	6	52
J5	(23) $O_{5,1}$	8	36
J6	(24) $O_{6,1}$	1	60
	(25) $O_{6,2}$	5	70
	(26) $O_{6,3}$	4	24
	(27) $O_{6,4}$	6	52
	(28) $O_{6,5}$	9	36
	(29) $O_{6,6}$	11	36
	(30) $O_{6,7}$	12	26
	(31) $O_{6,8}$	13	23
J7	(32) $O_{7,1}$	2	66
	(33) $O_{7,2}$	5	70
	(34) $O_{7,3}$	4	24
	(35) $O_{7,4}$	7	52
	(36) $O_{7,5}$	10	36
	(37) $O_{7,6}$	11	36
	(38) $O_{7,7}$	12	26
	(39) $O_{7,8}$	14	23
J8	(40) $O_{8,1}$	2	66
	(41) $O_{8,2}$	6	52
	(42) $O_{8,3}$	3	26
	(43) $O_{8,4}$	14	23
J9	(44) $O_{9,1}$	1	60
	(45) $O_{9,2}$	6	52
J10	(46) $O_{10,1}$	9	36
J11	(47) $O_{11,1}$	1	60
	(48) $O_{11,2}$	5	70
	(49) $O_{11,3}$	4	24
	(50) $O_{11,4}$	7	52
	(51) $O_{11,5}$	8	36
	(52) $O_{11,6}$	11	36
	(53) $O_{11,7}$	12	26
	(54) $O_{11,8}$	14	23
J12	(55) $O_{12,1}$	2	66
	(56) $O_{12,2}$	5	70
	(57) $O_{12,3}$	4	24
	(58) $O_{12,4}$	7	52
	(59) $O_{12,5}$	10	36
	(60) $O_{12,6}$	11	36
	(61) $O_{12,7}$	12	26
	(62) $O_{12,8}$	13	23
J13	(63) $O_{13,1}$	1	60
	(64) $O_{13,2}$	6	52
	(65) $O_{13,3}$	12	26
	(66) $O_{13,4}$	13	23
J14	(67) $O_{14,1}$	2	66
	(68) $O_{14,2}$	7	52
J15	(69) $O_{15,1}$	9	36
J16	(70) $O_{16,1}$	2	66
	(71) $O_{16,2}$	5	70
	(72) $O_{16,3}$	4	24
	(73) $O_{16,4}$	7	52
	(74) $O_{16,5}$	9	36
	(75) $O_{16,6}$	11	36
	(76) $O_{16,7}$	3	26
	(77) $O_{16,8}$	14	23
J17	(78) $O_{17,1}$	1	60
	(79) $O_{17,2}$	5	70
	(80) $O_{17,3}$	4	24
	(81) $O_{17,4}$	6	52
	(82) $O_{17,5}$	9	36
	(83) $O_{17,6}$	11	36
	(84) $O_{17,7}$	12	26
	(85) $O_{17,8}$	13	23
J18	(86) $O_{18,1}$	1	60
	(87) $O_{18,2}$	6	52
	(88) $O_{18,3}$	12	26
	(89) $O_{18,4}$	13	23
J19	(90) $O_{19,1}$	1	60
	(91) $O_{19,2}$	6	52
J20	(92) $O_{20,1}$	9	36

Fuente. Elaboración propia, con datos generados por el algoritmo

Tabla 37
Respuesta de Secuencias para FJSSP 20x14

Orden en que se deben ejecutar las Operaciones (C_M=817)
(9)→ (10)→ (55)→ (24)→ (17)→ (25)→ (26)→ (11)→ (56)→ (78)→ (18)→ (32)→ (27)→ (86)→ (63)→ (40)→ (12)→ (87)→ (57)→ (33)→ (79)→ (34)→ (80)→ (64)→ (28)→ (1)→ (70)→ (13)→ (23)→ (21)→ (2)→ (88)→ (90)→ (41)→ (35)→ (71)→ (58)→ (47)→ (3)→ (89)→ (4)→ (19)→ (81)→ (72)→ (29)→ (36)→ (37)→ (48)→ (59)→ (46)→ (14)→ (82)→ (83)→ (67)→ (44)→ (65)→ (73)→ (38)→ (49)→ (60)→ (5)→ (92)→ (50)→ (51)→ (42)→ (15)→ (68)→ (43)→ (74)→ (6)→ (75)→ (20)→ (69)→ (84)→ (30)→ (39)→ (7)→ (66)→ (31)→ (16)→ (8)→ (52)→ (76)→ (61)→ (45)→ (53)→ (62)→ (77)→ (54)→ (91)→ (85)→ (22)

Fuente. Elaboración propia

El *Makespan* de 817 horas es mejor que el obtenido por Roshanaei (2012) de 881 horas. Sin embargo, por las razones que se explicaron no se tiene el propósito de comparar ambos resultados. Es de resaltar que el algoritmo propuesto ha minimizado también las cargas de trabajo de las máquinas (*Maximum Workload* y *Total Workload*).

Con este ejemplo se demuestra que los algoritmos Genéticos de Rutas y de Secuencias han sido capaces de solucionar un problema de gran tamaño de *FJSS* (20 *Jobs* x 14 máquinas), aún más importante, el ejemplo ha servido para ilustrar el procedimiento por el cual un caso de gran complejidad de *FJSS* de la vida real, se puede sintetizar a un formato de tabla (Tabla 35) para viabilizar su solución mediante los algoritmos propuestos.

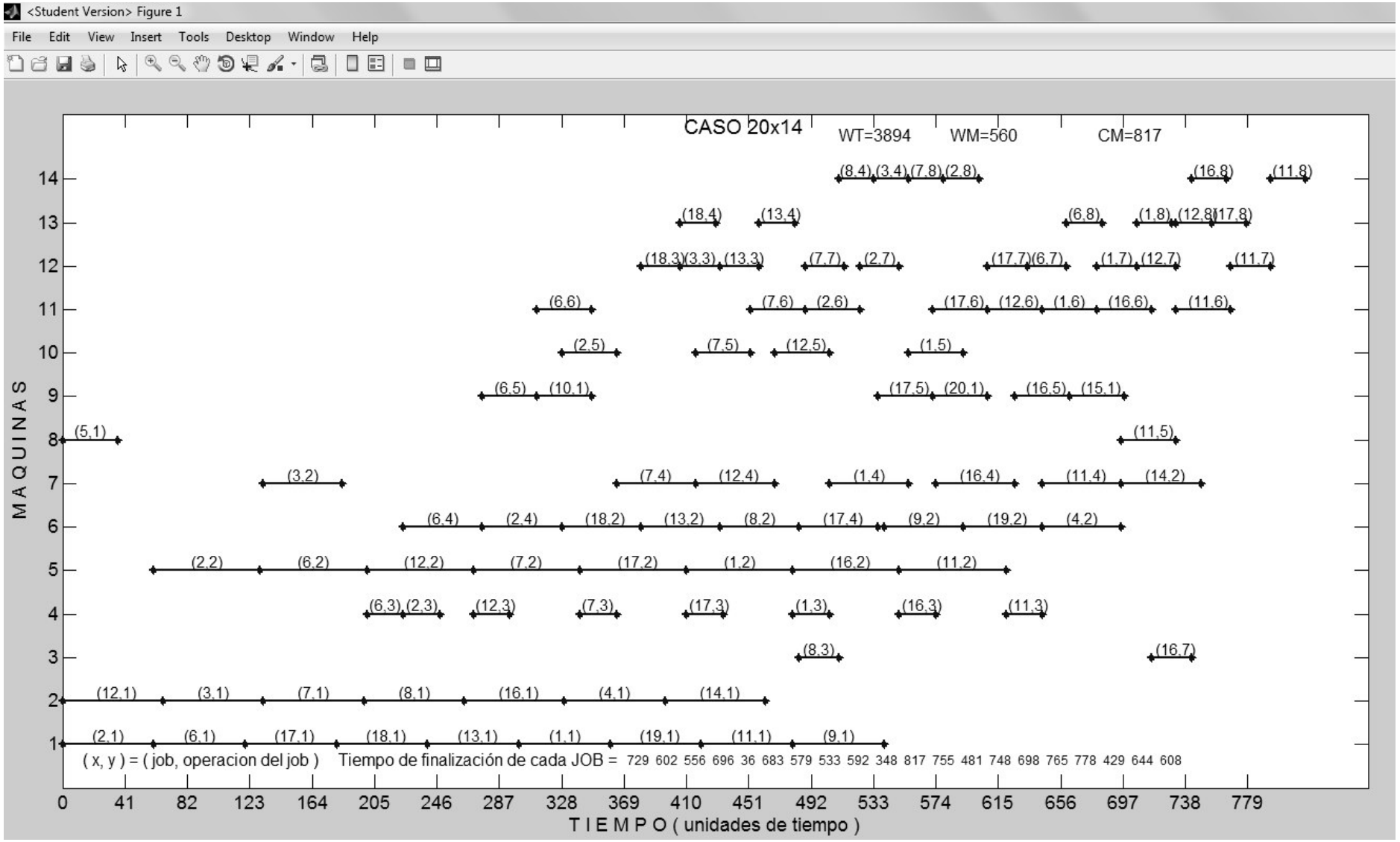


Figura 70. **Diagrama de Gantt *FJSSP* 20x14 - Obtenido con Algoritmos Propuestos.** *Fuente*: Elaboración propia.

5.3 Adaptación del Programa a Funciones Objetivos Derivadas del *Makespan* (Latencias y Tardanzas)

Con el título, se desea resaltar la existencia de otras funciones objetivos, que se pueden obtener fácilmente a partir del desarrollo de la presente tesis para calcular el *Makespan*.

Se recuerda que siendo C_i el instante de término de procesamiento del *Job* i, para $i=1,\ldots, n$, entonces, el *Makespan* es:

$$C_M = \max \{C_i : i = 1, \quad ,n\} \qquad (5.1)$$

La formulación de minimizar el *Makespan* se presenta como:

$$\text{Min } (C_M) \qquad (5.2)$$

De acuerdo a la literatura, existen otras funciones objetivos, que se utilizan con cierta frecuencia, consisten en minimizar la Latencia (*Lateness*), Tardanza (*Tardiness*) y variantes de éstas. Para estos casos, a cada *Job* le corresponde una fecha o plazo de entrega (*Due-Date*). Pero para calcular los criterios de latencia, tardanza y sus variantes, es necesario conocer los tiempos de finalización (C_i) de cada *Job*, los cuales pueden ser obtenidos con el procedimiento de cálculo del *Makespan* del algoritmo que ha sido propuesto en esta tesis.

A continuación de Gomes (2016), Laviós (2013) y Pinedo (2016), se extraen las siguientes definiciones, las cuales también se explicaron en el ítem 2.3.1.2 del presente volumen de tesis.

- Minimización de la latencia máxima. La latencia del *Job* i es definida como:

$$L_i = C_i - d_i \quad i = 1,\ldots, n \qquad (5.3)$$

Donde:

Li= Latencia del *Job* i, Ci=Tiempo de finalización del *Job* i, di= fecha de entrega del *Job* i

Li, es positiva cuando el *Job* i es finalizada tarde, nula cuando finaliza a tiempo y negativa cuando es finalizada temprano o con anticipación. En la minimización de la latencia máxima: $L_{max} = \max L_i$, la función objetivo consiste en:

$$\min L_{max} \qquad L_{max} \geq L_i \qquad i = 1,\ldots, n \tag{5.4}$$

- Minimización de la latencia total. En este caso se tiene:

$$\min \sum_{i=1}^{n} Li \qquad i=1,..,n \tag{5.5}$$

- Minimización de la tardanza máxima. La tardanza máxima, designada por Tmax, corresponde al *Job* con mayor diferencia Ti entre o instante de término Ci y la fecha o plazo de entrega di, siempre y cuando el *Job* es entregado después de la fecha de entrega. En este contexto:

$$Tmax \geq Ti \qquad i = 1,\ldots, \tag{5.6}$$

$$Ti \geq Ci - di \qquad i = 1, \ldots, n \tag{5.7}$$

$$Ti \geq 0 \tag{5.8}$$

Es decir,

$$Ti = \max \{0; Ci - d_i\} \tag{5.9}$$

En que, Ti *es* igual al mayor valor positivo, no pudiendo ser menor que cero, pues eso significaría adelanto en lugar de tardanza. La función objetivo a minimizar es:

$$\min Tmax \tag{5.10}$$

- Minimización de la suma de tardanzas o tardanza total. En este caso, la función objetivo consiste en determinar la menor suma de todos las tardanzas Ti, o sea:

$$\min \sum_{i=1}^{n} Ti \qquad i=1,..,n \tag{5.11}$$

- Minimización del número de *Jobs* con tardanza o número de tardanzas. En la minimización del número de *Jobs* con tardanza, la función objetivo consiste en minimizar el número de *Jobs* con tardanzas, puede ser formulado de la siguiente forma:

$$\min \sum_{i=1}^{n} yi \qquad i=1, ..,n \tag{5.12}$$

Donde,

$$yi = \begin{cases} 1, & \text{tarea con tardanza} \\ 0, & \text{caso contrario} \end{cases} \tag{5.13}$$

Se observa que si Ti> 0, implica que $yi = 1$.

5.3.1 Modificaciones al Diagrama de Flujo del Makespan

Si se desea minimizar las funciones objetivos anteriormente descritas, entonces en el Algoritmo del *Makespan* de la Figura 39 se tendrá que realizar ligeros agregados. El procedimiento es sencillo, el primer paso consiste en calcular los tiempos de terminación de cada *Job,* FIN_CADA_JOB= [C_1 C_2 C_3…C_n], de la misma manera como lo realiza el algoritmo Gantt de la Figura 41, y como a continuación se explica mediante la Figura 71.

En la Figura 71, se representan al CROMOSOMA DE SECUENCIAS, a los arreglos JOB y TMK. En el vector JOB se ubica la última posición que ocupa cada *Job*, por ejemplo, la última posición del *Job* 1, en el vector JOB, es la 8, entonces el contenido de la posición 8 de TMK (vector generado por la

Función Makespan) se guarda en la primera posición del vector FIN_CADA_JOB. Así sucesivamente se va encontrando los contenidos del vector FIN_CADA_JOB.

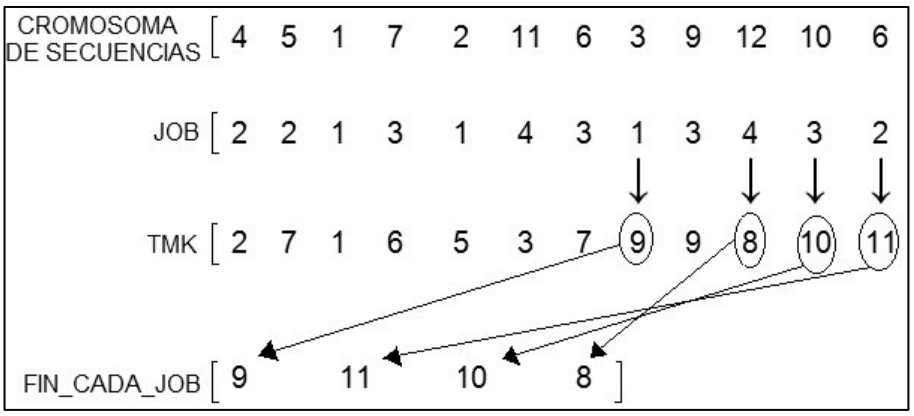


Figura 71. Cálculo de Fin de cada Job**.** *Fuente*: Elaboración propia

El segundo paso, consiste en restar al vector FIN_CADA_JOB el vector DUE_DATE, que es dato del usuario y contiene los plazos de entrega de cada *Job*. Así por ejemplo si DUE_DATE es [7 12 9 5], entonces, al restar ambos arreglos se obtiene el vector LATENCIA = [2 -1 1 3]. De este resultado, están plenamente identificadas las tardanzas, que son todos los miembros mayores de cero (2 1 3). Las ubicaciones de las tardanzas son identificables como el vector UBITARDANZAS = [1 0 1 1]. Finalmente, con esta información se puede calcular cualquier objetivo, como por ejemplo: La suma de tardanzas (6), máximo de tardanzas (3), la suma de latencias (5), el número de tardanzas (3). El procedimiento completo y los códigos en lenguaje **m** de Matlab que hay que incluir al algoritmo del *Makespan* de la Figura 39 se muestra en la Figura 72. Luego, de evaluar cada cromosoma, el programa principal se encargará de clasificarlos de menor a mayor para los propósitos de minimización.

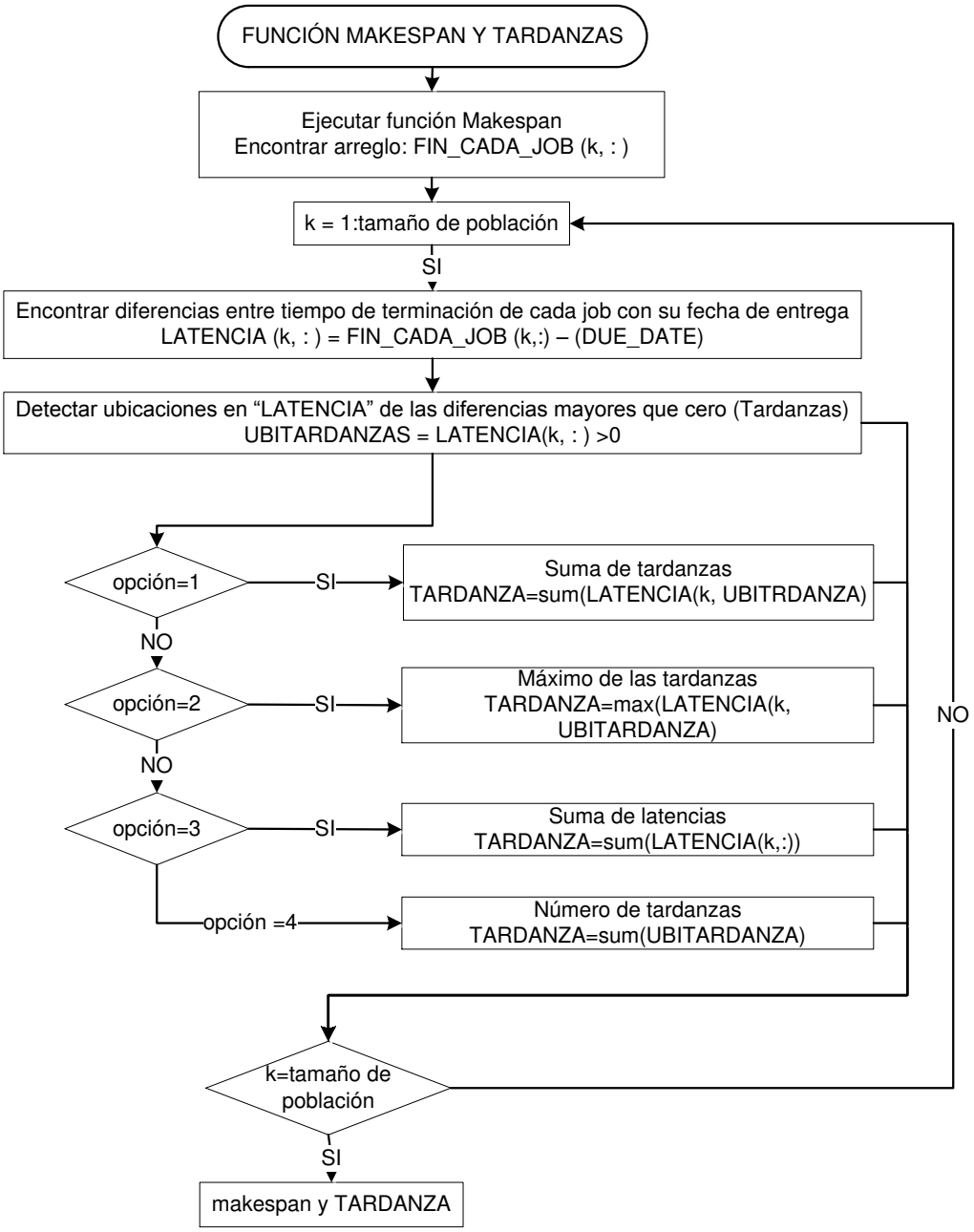


*Figura 72***. Función *Makespan* y Tardanzas**. *Fuente*: Elaboración propia

5.3.2. Ejemplo Aplicativo

En Vidal et al. (2011), página 1605, se encuentra un problema de *Flexible Job Shop*, en donde se tienen cuatro *Job*s (J1, J2, J3 y J4) con tres operaciones cada una y cuatro máquinas posibles para la ejecución de las operaciones. Este problema, servirá como ejemplo para solucionarlo con las

adaptaciones a la propuesta indicadas en la Figura 5.15. En la Tabla 38, se muestran los datos.

Tabla 38
FJSSP 4x4

JOB	di	Orden	$O_{i,j}$	M_1	M_2	M_3	M_4
J_1	9	1, 3, 2	$O_{1,1}$	1	4	6	9
			$O_{1,2}$	3	2	5	1
			$O_{1,3}$	4	2	1	3
J_2	7	2,1,3	$O_{2,1}$	2	8	7	1
			$O_{2,2}$	2	2	4	5
			$O_{2,3}$	6	11	2	7
J_3	8	1,2,3	$O_{3,1}$	8	5	4	9
			$O_{3,2}$	3	3	6	1
			$O_{3,3}$	7	1	8	1
J_4	11	1,2,3	$O_{4,1}$	5	10	6	4
			$O_{4,2}$	4	2	3	8
			$O_{4,3}$	7	3	4	1

Fuente. Vidal et al. (2011)

La columna “di” es la fecha o plazo de entrega (en unidades de tiempo) de cada *Job*, es decir, por ejemplo, el *Job* J1 tiene un plazo de entrega de 9 (días, horas, etc.). La columna “Orden” de la tabla, es el orden deseado de ejecución de las operaciones de cada *Job*. Así, por ejemplo las operaciones de J_1 que son: $O_{1,1}$, $O_{1,2}$, $O_{1,3}$; se deben ejecutar en el orden 1, 3, 2, es decir, en el orden: $O_{1,1}$, $O_{1,3}$ y $O_{1,2}$. Los tiempos en cada celda representan la suma de los tiempos de ejecución de cada operación más el tiempo de configuración (*setup*) de cada máquina.

Teniendo en cuenta el orden descrito en la columna “Orden” y volviendo a enumerar los subíndices de cada operación, para los dos primeros *Job*s (J_1 y J_2), entonces la Tabla 38, es presentada en la Tabla 39, esto se realiza para adaptarnos al formato que tratamos en la tesis. También en la tabla, se ha

agregado números entre paréntesis que ordena consecutivamente todas las operaciones del problema.

Tabla 39

FJSSP 4x4 Adaptada
(Subíndices de las operaciones de los *Job*s J_1 y J_2 modificados)

JOB	di	$O_{i,j}$	M_1	M_2	M_3	M_4
J_1	9	(1) $O_{1,1}$	1	4	6	9
		(2) $O_{1,2}$	4	2	1	3
		(3) $O_{1,3}$	3	2	5	1
J_2	7	(4) $O_{2,1}$	2	2	4	5
		(5) $O_{2,2}$	2	8	7	1
		(6) $O_{2,3}$	6	11	2	7
J_3	8	(7) $O_{3,1}$	8	5	4	9
		(8) $O_{3,2}$	3	3	6	1
		(9) $O_{3,3}$	7	1	8	1
J_4	11	(10) $O_{4,1}$	5	10	6	4
		(11) $O_{4,2}$	4	2	3	8
		(12) $O_{4,3}$	7	3	4	1

Fuente. Elaboración propia, datos de Tabla 38

Con los datos de la Tabla 39, el programa se ejecutó con la opción de minimizar el número de tardanzas. Las máquinas seleccionadas para cada una de las operaciones se muestran en la Tabla 40, el *Maximum Workload* y *Total Workload* encontrados se muestra en la tabla inferior de la tabla.

La secuencia óptima de operaciones encontrada se muestra en la Tabla 41, los números entre paréntesis son los mismos que corresponden a la Tabla 40. El Diagrama de Gantt de la secuencia se muestra en la Figura 73 (generado automáticamente por el programa). En la parte superior del Diagrama de Gantt se tienen los siguientes resultados:

- *Total Workload* = WT = 21
- *Maximum Workload* = WM = 8
- *Makespan* = CM = 8
- Tardanza= 0

En la parte inferior de la Figura 73 se tiene:

- Tiempo de finalización de cada JOB= [7 7 7 8]
- Due_Date = [9 7 8 11]

Es decir, el programa minimizó las tardanzas a cero, ya que al restar los tiempos de finalización de cada *Job* con los plazos de entrega para cada *Job* (Due_Date) obtenemos cifras negativas o nulas. Objetivamente, del Diagrama podemos comprobar directamente estos resultados.

De esta forma, se pone de relieve el hecho que el algoritmo propuesto en la presente tesis es fácilmente adaptable a otras funciones objetivos con ligeras modificaciones.

Tabla 40
Respuesta de Rutas para FJSSP 4x4

JOB	Operación	Máquina	Tiempo
J_1	(1) $O_{1,1}$	1	1
	(2) $O_{1,2}$	3	1
	(3) $O_{1,3}$	4	1
J_2	(4) $O_{2,1}$	1	2
	(5) $O_{2,2}$	4	1
	(6) $O_{2,3}$	3	2
J_3	(7) $O_{3,1}$	3	4
	(8) $O_{3,2}$	4	1
	(9) $O_{3,3}$	2	1
J_4	(10) $O_{4,1}$	4	4
	(11) $O_{4,2}$	2	2
	(12) $O_{4,3}$	4	1
			$W_T = 21$, $W_M = 8$

Fuente. Elaboración propia

Tabla 41
Respuesta de Secuencia para FJSSP 4x4

Orden en que se deben ejecutar las Operaciones (C_M=12)
(10) → (1)→ (7)→ (4) →(5) →(11)→ (2) →(8)→ (3)→ (9)→ (6) → (12)

Fuente. Elaboración propia

Las condiciones sobre el programa modificado son:

- Tamaño de población de rutas = 100
- Generaciones en población de rutas = 100
- Tamaño de población de secuencias = 500
- Generaciones en población de secuencias = 150
- Porcentaje mutación en población rutas = 1
- Porcentaje sobrevivientes en población rutas = 20
- Porcentaje mutación en población secuencia = 0.1
- Porcentaje sobrevivientes en población secuencia = 50

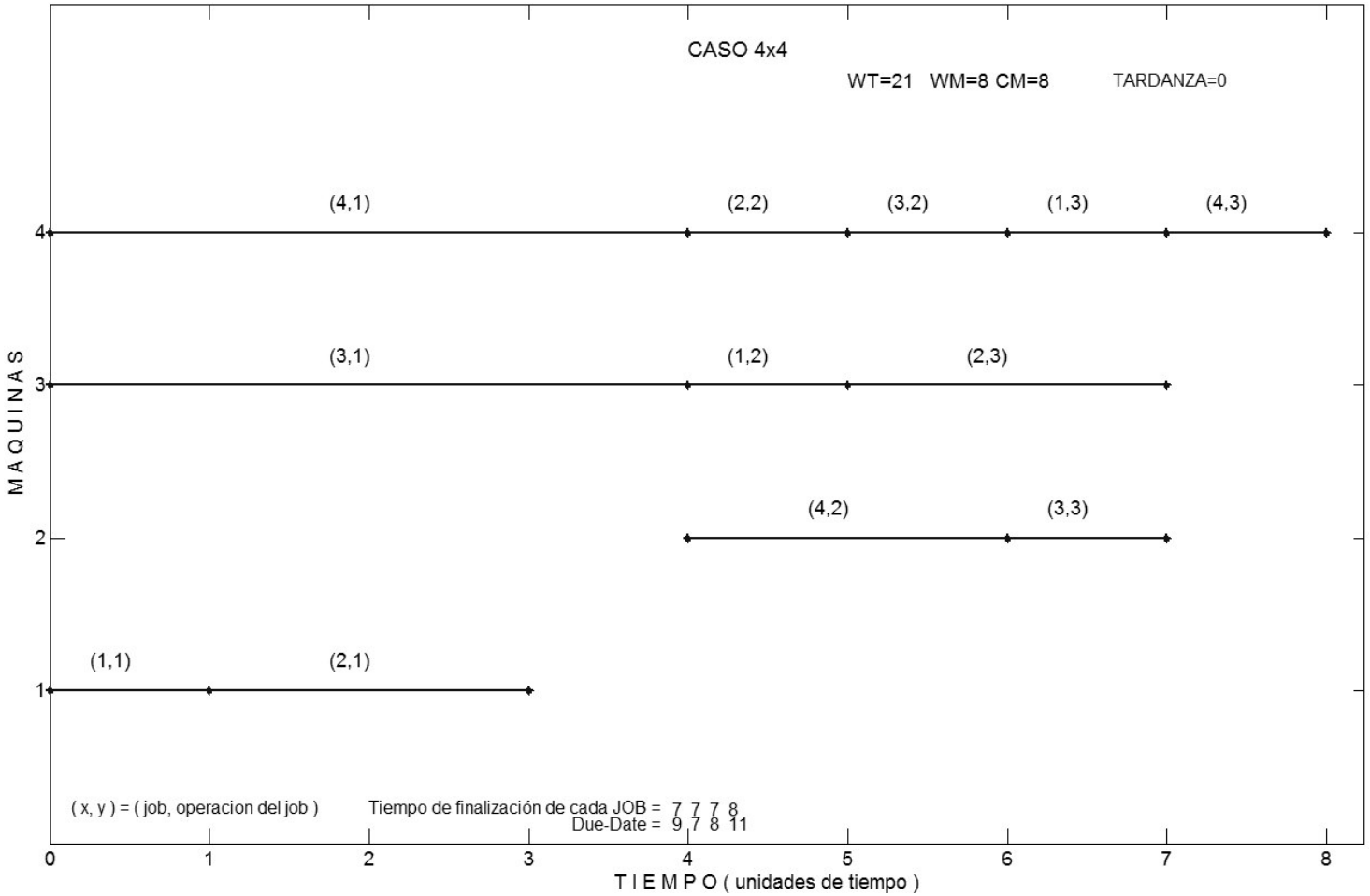


Figura 73. **Resultado de minimización de tardanzas**. *Fuente*: Elaboración propia

CONCLUSIONES

Se solucionó el problema del *Flexible Job Shop Scheduling* (*FJSS*) desde un enfoque jerárquico que dividió el problema en dos subproblemas: El Subproblema de Enrutamiento y el Subproblema de Secuenciación.

Se solucionó el Subproblema de Enrutamiento mediante el denominado Algoritmo de Genético de Rutas, quien designó a cada una de las operaciones de los *Jobs* una de las máquinas entre un conjunto de máquinas disponibles, minimizando dos criterios: La carga de trabajo de la máquina más cargada (*Maximum Workload* - $W_{M)}$ y la carga de trabajo total de las máquinas (*Total Workload* - W_T).

Se solucionó el Subproblema de Secuenciación, mediante el denominado Algoritmo Genético de Secuencias, quien utilizó los resultados del Algoritmo Genético de Rutas para encontrar la secuencia o el orden óptimo en el cual se deben ejecutar las operaciones distribuidas en cada máquina, minimizando el criterio del tiempo de terminación de todos los *Job*s (*Makespan* -C_M).

Se ha probado la Eficacia y la Eficiencia del Algoritmo Genético de Rutas y del Algoritmo Genético de Secuencias propuestos, mediante la solución de casos complejos de *FJSSP* planteados por Kacem et al. (2002) y Kacem et al. (2002 a), los resultados comparados con las de otros investigadores es superado solamente en el caso de 15x10, en donde algunos investigadores han alcanzado una solución de C_M=11 frente a la propuesta de la tesis en donde se alcanza el valor de C_M=12. Sin embargo, los tiempos de ejecución del algoritmo propuesto superan a las demás propuestas.

De acuerdo a la literatura consultada, no se ha podido identificar trabajos de enfoque jerárquico, que hayan solucionado el *FJSSP,* aplicando simúltaneamente, en ambos subproblemas, Algoritmos Genéticos. En cambio, en la propuesta de la tesis para la solución de ambos subproblemas se han utilizado simultáneamente Algoritmos Genéticos, en el primer subproblema se ha diseñado un Algoritmo Genético de pocos miembros de la población con ciertas flexibilidades, mientras que en el segundo un Algoritmo Genético más convencional, dando en ambos casos buenos resultados.

En el Algoritmo Genético de Rutas ha dado buenos resultados la propuesta de la tesis con respecto al cruce de los cromosomas, en donde los *Jobs,* que son seleccionados aleatoriamente, intercambian las máquinas de sus operaciones. Así mismo, ha sido también de mucha utilidad los tres tipos de mutación propuestos en la tesis.

En el Cromosoma de Secuencias, a diferencia de otras propuestas, se han representado numéricamente a todas las operaciones del *FJSSP,* por este motivo, se ha tenido que crear un método, con muy buenos resultados, que ha corregido las posiciones relativas de las operaciones de cada *Job,* ésto con la finalidad de mantener la precedencia de las operaciones en cada *Job.*

Ha diferencia de otras propuestas, en donde una vez creada la población de individuos, se procede a las etapas de: selección, cruce, mutación, evaluación, clasificación, etc. En la propuesta de la tesis, en cambio, a las poblaciones iniciales se las evalúa y clasifica, sin pasar por la selección, cruce y mutación. El objeto, es detectar, desde el inicio si el mejor cromosoma de la población pueda estar cumpliendo con el criterio de parada. La propuesta de la tesis ha dado mucha rapidez tanto al Algoritmo Genético de Rutas como al Algoritmo Genético de Secuencias, sobre todo para los tamaños pequeños y medianos de *FJSSP*.

En el Algoritmo Genético de Rutas, cuando se ejecuta con umbrales de parada para W_M y W_T, el problema pasa de ser de bi objetivo a mono

objetivo, mediante la suma de los objetivos W_T y W_M. Sin embargo, a diferencia de otros trabajos, en la tesis se propone no ponderar a cada sumando y posteriormente realizar una verificación si cada sumando cumple con sus umbrales, si no cumplen, se cambia a la población, la propuesta ha tenido muy buenos resultados.

El algoritmo diseñado que genera los Diagramas de Gantt al final del programa, ha sido de mucha utilidad, debido a que ha permitido validar objetivamente todos los resultados. Se destaca también en los Diagramas la presentación impresa de los tiempos de finalización de cada *Job* (verificable también objetivamente), lo que permitiría a un usuario no comprometerse en finalizar con un *Job* o con todos ellos en fechas inoportunas.

Sugerencias para Futuros Trabajos

Diseñar y adicionar otros operadores de mutación para el Algoritmo Genético de Secuencias, éstos conjuntamente con el existente pueden aportar con aumentar la diversidad en el espacio de búsqueda, mejorando el tiempo para alcanzar su objetivo, del mismo modo como se ha hecho para el caso del Algoritmo Genético de Rutas en donde se ha propuesto hasta tres tipos de mutaciones los cuales se ejecutan probabilísticamente.

Optimizar el algoritmo *Makespan* con la intención de reducir el tiempo de ejecución, de este modo en un enfoque integrado puede calcularse simultáneamente en cada cromosoma los criterios de C_M, W_M y W_T en tiempos no prohibitivos.

Incluir las Redes de Petri como una opción para calcular el *Makespan*, podría ser una alternativa para reducir el tiempo de cálculo toda vez que esta herramienta permite modelar matemáticamente mediante matrices las máquinas, los tiempos de procesamiento y los estados del sistema.

REFERENCIAS

Aguiar V. (2014). *Resolución de problemas de optimización combinatoria utilizando técnicas de computación evolutiva*. Una aplicación a la biomedicina. Tesis Doctoral. Tecnologías de la Información y las Comunicaciones, Universidade da Coruña, España.

Azardoost, E. B., Manipour, N. I. (2011). A Hybrid Algorithm for Multi Objective *Flexible Job Shop Scheduling* Problem. *Proceeding of the 2011 International Conference on Industrial Engineering and Operations Management*, 795-801.

Beasley, D., Bull, D., Martin, R. R. (1993). An Overview of Genetic Algorithms: Part I, Fundamentals. *University Computing*, 15(2), 58-69. Recuperado de: http://mat.uab.cat/~alseda/MasterOpt/Beasley93GA1.pdf.

Beck, F. L. (2000). *Escalonaiviento de Tarefas Job-Shop Realistas utilizando Algoritmos a Genéticos em Matlab*. Dissertação Mestrado, Universidade Federal de Santa Catarina, Brasil.

Behnke, D. and Geiger M.J. (2012). *Test Instances for the Flexible Job Shop Scheduling Problem with Work Centers*. Research Report, H -Universität Der Bundeswehr Hamburg.

Blum, C. and Roli, A. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, 35(3), 268–308.

Brandimarte, P. (1993). Routing and *Scheduling* in a *Flexible Job Shop* by Tabu Search. *Annals of Operations Research*, 41, 157–183.

Chankong, V., Haimes, Y. (1983). Multiobjective Decision Making Theory and Methodology, New York: North-Holland.

Chiang, T. Ch. and Lin, H. J. (2012). *Flexible Job Shop Scheduling* using a Multiobjective Memetic Algorithm. *Advanced Intelligent Computing Theories and Applications with Aspects of Artificial Intelligence. Lecture Notes in Computer Science*, 6839, 49-56.

Chiang, T., Lin, H. (2013). A Simple and Effective Evolutionary Algorithm for Multiobjective *Flexible Job Shop Scheduling*. *International Journal of Production Economics*, 141, 87-98.

Claudio, J.E. (2002). *Heurísticas e Metaheurísticas para Otimização Combinatória Multiobjetivo*. Tese Doutorado. Faculdade de Engenharia Elétrica e de Computação, Departamento de Engenharia de Sistemas, Universidade Estadual De Campinas, Brasil.

Cook, S. (1971). The Complexity of Theorem Proving Procedures. *Proceedings Third Annual ACM Symposium on Theory of Computing*, 151-158.

Cuatrecasas L (2009). *Diseño avanzado de procesos y plantas de produccion flexible*. Barcelona, España: Profit Editorial.

De Melo, E.L. (2014). *Meta-heurísticas Iterated Local Search, GRASP e Artificial Bee Colony aplicadas ao Job Shop Flexível para minimização do atraso total.* Tese Doutor em Ciências, Escola Politécnica (Engenharia de Produção) da Universidade de São Paulo, Brasil.

Deb, K. (2005). *Multi-Objective Optimization Using Evolutionary Algorithms.* UK: John Wiley & Sons.

Dehuri, S., Ghosh, A., Cho, S. (2008) Particle swarm optimized polynomial neural network for classification: a multi-objective view. *International Journal of Intelligent of Defence Support Systems*, 1(3), 225–253.

Demir, Y., Isleyen, K. (2013). Evaluation of mathematical models for flexible *Job-Shop Scheduling* problems. *Applied Mathematical modeling*, 37(2013), 977-988.

Dorigo, M., Maniezzo, V., Colorini, A. (1996). Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics – Part B*, 26(1), 29-41.

Fonseca, C.M., P.J., Fleming, P. J. (1995). *Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms*. Research Report 564, Departament of Automatic Control and Systems Engineering, The University of Sheffield, UK.

Freitas K. (2007). *Escalonamento Genético FJSP com tempo de configuração dependente de seqüencia*. Dissertação de mestrado, Universidad Federal de Uberlândia, Brasil.

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flow *Shop* and *Job Shop Scheduling*. *Mathematics of Operations Research*, 1(2), 117–129.

Gen M., Lin, L. (2012*).* Multiobjective Genetic Algorithm for *Scheduling* Problems in Manufacturing Systems*. Industrial Engineering & Management Systems*, 11(4), 310-330. doi : 10.7232/iems.2012.11.4.310.

Genova K., Kirilov L., Guliashki V. (2015). A Survey of Solving Approaches for Multiple Objective *Flexible Job Shop Scheduling* Problems. *Cybernetics And Information Technologies,* 15(2), 3-22.

Gestal, M., Rivero, D., Rabuñal, J. R., Dorado, J., Pazos, A. (2010). Introducción a los Algoritmos Genéticos y la Programación Genética. *Universidade da Coruña. Servizo de Publicacións.*

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 5, 533-549.

Goldberg,D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, MA, USA: Addison-Wesley Publishing Company.

Gomes, H. (2015). *Estudo e Análise do Sequenciamento de Tarefas de Produção: Job Shop Scheduling.* Dissertação de Mestrado em Engenharia e

Gestão Industrial. Escola Superior de Estudos Industriais e de Gestão, Instituto Politécnico do Porto, Portugal.

Haupt, R.L., Haupt, S.E. (2004). *Practical genetic algorithms*. New Jersey U.S.:Published by John Wiley & Sons, Inc.

Hayes RH, Wheelwright SC (1979). Link manufacturing process and product life cycles. *Harvard Business Review,*135–142.

Heidegger, M. (1963). *El final de la filosofía y la tarea del pensar*. Sartre.

Holland, J. H. (1975). Adaption in natural and artificial systems. USA: The University of Michigan Press.

Hsu T., Dupas R., Jolly D., Goncalves G. (2002). Evaluation of mutation heuristics for the solving of multiobjective flexible *Job-Shop* by an evolutionary algorithm. *IEEE International Conference on Systems, Man and Cybernetics,* 5, 6-9.

Jacobson, Sheldom H. (2004). Analizing the Performance of Generalized Hill Climbing Algorithms. *Journal of Heuristics,* 10, 387-405.

Jiang, J., Wen, M., Maa, K., Long, X., Li, J. (2011). Hybrid Genetic Algorithm for Flexible *Job-Shop Scheduling* with Multi-Objective. *Journal of Information & Computational Science*, 8(11), 2197-2205.

Kacem I., Hammadi S., Borne, P. (2002). Approach by localization and multiobjective evolutionary optimization for flexible *Job-Shop Scheduling* problems. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 32, 1–13.

Kacem, I., Hammadi, S., Borne, P. (2002a). Pareto-Optimality Approach for *Flexible Job Shop Scheduling* Problems: Hybridization of Evolutionary Algorithms and Fuzzy Logic. *Mathematics and Computers in Simulation*, 60, 245–276.

Kirkpatrick S., Gelatt C.D., Vecchi M.P. (1983). Optimization by Simulated Annealing. *Science*, 220, 45-54.

Kuhn, Th. (1971). *Estructura de las revoluciones científicas*. México: Fondo de Cultura Económica.

Lakatos, I. (1978). *The Methodology of Scientific Research Programmes Philosophical Papers*. USA: John Worrall And Gregory Currie.

Laviós, J. J. (2013*). Análisis de la Relajación Lagrangiana como Método de Programación de Talleres Flexibles en un entorno Multiagente*. Tesis

Doctoral. Universidad de Burgos, Departamento de Ingeniería Civil, España.

Li, J.Q., Pan, Q.K., Liang ,YC. (2010). An Effective Hybrid Tabu Search Algorithm for Multi-Objective Flexible *Job-Shop Scheduling* Problems. *Computers & Industrial Engineering*, 59, 647-662.

López, A.J. (2005). *Diseño de un Algoritmo Evolutivo Multiobjetivo Paralelo*. Tesis de Maestría en Ciencias. Centro de Investigación y de Estudios Avanzados del IPN. Departamento de Ingeniería Eléctrica. Sección de Computación, Mexico.

Lozano, J.A., Larrañaga, P. (2001). Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publisher.

Maldonado, C.E. (2005). Heurística y producción de conocimiento nuevo en la perspectiva CTS. *Estética, Ciencia y Tecnología. Creaciones electrónicas y numéricas (I.Hernández, compiladora)*. Colombia: Editorial Pontificia Universidad Javeriana.

Malhotra R., Singh N., Singh Y. (2011). Genetic Algorithms: Concepts, Design for Optimization of Process Controllers. *Computer and Information Science,* 4(2), 39-54.

Márquez, J.E., Ávila, R.L, Gómez, M.A., González, E., Herrera, C.R. (2012). Algoritmo genético aplicado a la programación en talleres de maquinado. *Ingeniería Mecánica,* 15(3), 201-212.

Martí, R. (2003). Multistart Methods en Fred Glover y Gary A. Kochenberger (eds), Handbook of Metaheuristics (pp. 355-368).

Maturana, H., Varela, F. (1990). *El árbol del conocimiento. Las raíces biológicas del conocimiento humano*. Madrid: Debate.

Mekni, S., Chaâr, B. (2015). Multiobjective *Flexible Job Shop Scheduling* Using A Modified Invasive Weed Optimization. *International Journal on Soft Computing (IJSC)*, 6(1). doi: 10.5121/ijsc.2015.6103 25.

Melián B., Moreno J, Moreno J. (2003). Metaheuristics: A global view. Inteligencia Artificial. *Revista Iberoamericana de Inteligencia Artificial*, 19, 7-28.

Menna S. (2014). Heurísticas y Metodología de la Ciencia. *Mundo Siglo XXI*, *revista del CIECAS-IPN*, IX (32), 67-77

Minsky, Marvin, L. (1967). *Computation: Finite and Infinite Machines*. New Jersey U.S: Prentice-Hall.

Mitchell, M. (1999). *An Introduction to Genetic Algorithms*. Massachusetts USA: A Bradford Book The MIT Press.

Moreno, P., Huecas, G., Sánchez, García, A. (2007). Metaheurísticas de optimización combinatoria: Uso se Simulated Annealing para un problema de calendarización. *Revista de Ciencia, Tecnología y Medio Ambiente*, 5.

Morillo, D., Moreno, L., Díaz, J. (2014). Metodologías Analíticas y Heurísticas para la Solución del Problema de Programación de Tareas con Recursos Restringidos (RCPSP): una revisión. Parte 1. *Ingeniería y Ciencia*. 10(19), 247–271

Morin, E. (1994). El método III: El conocimiento del conocimiento. Madrid: Cátefra.

Moscato, P., Cotta-Porras, C. (2003). Una introducción a los algoritmos meméticos. Inteligencia Artificial. *Revista Iberoamericana de Inteligencia Artificial*. 19.

Moslehi, G., Mahnam, M. (2011). A Pareto Approach to Multi-Objective *Flexible Job Shop Scheduling* Problem Using Particle Swarm Optimization and Local Search. *International Journal Production Economics*, 129, 14-22.

Motaghedi-Iarijani, A. K., Sabri-Iaghaie, M., Heydari. (2010). Solving *Flexible Job Shop Scheduling* with Multi Objective Approach. *International Journal of Industrial Engineering and Production Research*, 21(4), 197-209.

Pezzella, F., Morganti, G., Ciaschetti, G. (2008). A genetic algorithm for the Flexible *Job Shop Scheduling* Problem. *Computers & Operations Research*, 35, 3202-3212.

Pinedo. (2016). *Scheduling Theory, Algorithms and Systems*. New York, USA: Springer, Fifth Edition

Ranjini, A., Zoraida, B.S.E. (2013). Analysis of selection schemes for solving *Job Shop Scheduling* problem using genetic algorithm. *IJRET: International Journal of Research in Engineering and Technology,* 2(11), 775-779.

Roshanaei, V. (2012). *Mathematical Modelling and Optimization of Flexible Job Shops Scheduling Problem*. Master Thesis, University of Windsor, Ontario, Canada.

Sadaghiani, J., Boroujerdi, S., Mirhabibi, M., Sadaghiani, P. (2014). A Pareto Archive Floating Search Procedure for Solving Multi-Objective *Flexible Job Shop Scheduling* Problem*. Decision Science Letters*, 3(2), 157-168.

Sastry, K., Goldberg, D, Kendall, G. (2005). *Genetic Algorithms en Search Methodologies – Introductory Tutorials in optimization and Decision Support Techniques.* US: Springer, 97-125. doi: 10.1007/0-387-28356-0-4.

Schaffer, J. D., Eshelman, L. J. (1991). On Crossover as an Evolutionarily Viable Strategy. *Internation Conference on Genetic Algotithms* 4, 61-68.

Shahsavari-Pour, N., Ghasemishabankareh, B. (2013). A novel hybrid meta-heuristic algorithm for solving multi objective flexible *JobShop Scheduling*. *Journal of Manufacturing Systems* 32, 771– 780.

Shao, X., Liu, W., Liu, Q., Zhang, C. (2013). Hybrid Discrete Particle Swarm Optimization for Multi-Objective Flexible *Job-Shop Scheduling* Problem. *International Journal of Advanced Manufacturing Technology*, 67(9-12), 2885-2901. doi: 10.1007/s00170-012-4701-3.

Srinivas, M., Patnaik, L. M. (1994). Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. *IEEE Transactions on System, Man and Cybernetics*, 24(4), 656-666.

Sun W., Pan Y., Lu, X., Ma, Q. (2010). Research on flexible *Job-Shop Scheduling* problem based on a modified genetic algorithm. *Journal of Mechanical Science and Technology*, 24(10), 2119-2115. doi: 10.1007/s12206-010-0526-x.

Vidal, J., Mucientes, M., Bugarín, A., Lama, M. (2011). Machine *Scheduling* in custom furniture industry through neuro-evolutionary hybridization. *Applied Soft Computing.* 11(2), 1600–1613.

Vitorino, S., Tadeu, C. (2015). Otimização da programação de produção em ambiente *Job Shop* através da minimização do *Makespan*: um estudo de caso em uma indústria de cosméticos. Monografia apresentada ao Curso

de Especialização em Engenharia de Produção, Universidade Federal do Parana, Brasil.

Vollmann T, Berry W, Whybark C (1997). *Manufacturing planning and control systems*. USA: McGraw-Hill.

Voß S. (2001). *Meta-heuristics: The State of the Art. Local Search for Planning and Scheduling*. ECAI 2000 Work*Shop*, 1-23.

Wojakowski, P., Warzolek D. (2013). Research Study of State-of-the-Art Algorithms for Flexible *Job-Shop Scheduling* Problem. *Technical Transactions, Mechanics,* 1, 381-388.

Womack, JP, Jones DT, Roos D (2007). *The machine that changed the world.* Free Press.

Xia, W., Wu, Z. (2005). An Effective Hybrid Optimization Approach for Multi-Objective *Flexible Job Shop Scheduling* Problems. *Computers & Industrial Engineering*, 48(2), 409-425.

Xing, L.N., Chen, Y.W., Yang K.W. (2009a). Multi-objective flexible *Job Shop* schedule: Design and evaluation by simulation modeling. *Applied Soft Computing*, 9, 362–376.

Xing, L.N., Chen, Y.W., Yang, K.W. (2009b). An Efficient Search Method for Multi Objective *Flexible Job Shop Scheduling* Problems. *Journal of Intelligent manufacturing*, 20, 283-293.

Xiong, J., Tan, X., Yang, K.W., Xing, L.N, Chen, Y.W. (2012). A Hybrid Multiobjective Evolutionary Approach for Flexible *Job-Shop Scheduling* Problems. *Mathematical Problems in Engineering*, 1(27). doi: 10.1155/2012/478981.

Yu, X., Gen, M. (2010). *Introduction to Evolutionary Algorithms*. London UK: Springer-Verlag London Limited.

Zhang, G., Shao, X., Li, P., Gao, L. (2009). An Effective Hybrid Particle Swarm Optimization Algorithm for Multi-Objective Flexible *Job-Shop Scheduling* Problem. *Computers & Industrial Engineering*, 56(4), 1309-1318.

Ziaee, M. (2014). An Efficient Heuristic Algorithm for *Flexible Job Shop Scheduling* with Maintenance Constraints. *Applied Mathematics and Sciences: An International Journal (MathSJ)*, 1(1), 19-31.

ANEXO A: PRINCIPALES DETALLES DEL CÓDIGO FUENTE

```
% UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
% UNIDAD DE POSGRADO DE INGENIERÍA INDUSTRIAL
% Autor: GUILLERMO TEJADA MUÑOZ (gtejadam@unmsm.edu.pe)
% LIMA –PERÚ

clear;
clc;
in=0;
fprintf('\n              UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS\n');
fprintf('                        LIMA - PERÚ\n\n')
fprintf('                FACULTAD DE INGENIERÍA INDUSTRIAL\n');
fprintf('                     UNIDAD DE POSGRADO\n\n');
fprintf('          MULTIOBJECTIVE FLEXIBLE JOB SHOP SCHEDULING PROBLEM\n\n');
fprintf('CREADO POR: GUILLERMO TEJADA MUÑOZ\n\n\n');


%************************************************************
%****************INGRESO DE DATOS DE USUARIO*****************
%************************************************************


vacio0=1;
while vacio0==1
%Se guarda en "archivosdatos" el nombre del archivo con los datos
archivodatos=input('* INGRESAR_NOMBRE_ARCHIVO_DATOS_FJSSP: ', 's');
vacio0=isempty(archivodatos);
end

%corre archivo con el nombre que se almaceno en "archivodatos"
run(archivodatos);

% numero mayor que 2 para “opcionworkload” para que se ingrese al siguiente while
opcionworkload=3;
vacio=1;
while vacio==1 || opcionworkload>2
fprintf('\n* OPCIONES:\n\n       (Recomendado) 1 = MINIMIZAR LIBREMENTE A
"MÁXIMUM WORKLOAD" Y "TOTAL WORKLOAD"\n                  2 = ESPECIFICAR
VALORES UMBRALES PARA "MAXIMUM WORKLOAD" Y "TOTAL WORKLOAD"\n\n');
opcionworkload = input('* OPCIÓN NÚMERO?: ');
vacio=isempty(opcionworkload);
end

if opcionworkload==1
maxwork=0;
totalwork=0;
NpoblaNuevas=1;
else
maxwork=input('\n* UMBRAL_MAXIMUM_WORKLOAD (U_WM):');
totalwork=input('* UMBRAL_TOTAL_WORKLOAD   (U_WT):');
NpoblaNuevas=input('*
MÁXIMO_NÚMERO_REINICIOS_ALGORITMO_GENÉTICO_RUTAS: ');
if isempty(NpoblaNuevas)
  NpoblaNuevas=1; % valor por defecto
```

```
end
end

vacio1=1;
while vacio1==1 %Sale de while solo cuando hay dato
tamapoblamq=input('\n* TAMAÑO_POBLACIÓN_ALGORITMO_GENÉTICO_RUTAS:');
vacio1=isempty(tamapoblamq);
end

vacio2=1;
while vacio2==1 %Sale de while solo cuando hay dato
maxgenmaq=input('* MÁXIMA_GENERACIONES_ALGORITMO_GENÉTICOS_RUTAS:');
vacio2=isempty(maxgenmaq);
end

reiteracionmq=input('* NÚMERO_REITERACIONES_SIN_MEJORAR_"WM+WT" (Puede
dejar vacio):');
  if isempty(reiteracionmq)
  reiteracionmq=maxgenmaq; % Si no se indica reiteraciones
  end

mutmaq=input('\n\n* PORCENTAJE_MUTACIÓN_ALGORITMO_GENÉTICO_RUTAS (Si
deja vacio por defecto es 1%):');
mutmaq=mutmaq/100;
if isempty(mutmaq)
mutmaq=0.01; % valor por defecto
end

sobremaq=input('* PORCENTAJE_SOBREVIVIENTES_ALGORITMO_GENÉTICO_RUTAS
(Si deja vacio por defecto es 20%): ');
sobremaq=sobremaq/100;
if isempty(sobremaq) % Si no selecciona
sobremaq=0.20; % valor por defecto
end

mkdeseado=input('\n* UMBRAL_CM (U_CM) (No llenar para minimizar libremente):');
  if isempty(mkdeseado)
   mkdeseado=0; % valor por defecto
  end

vacio3=1;
while vacio3==1 %Sale de while solo cuando hay dato
tampoblasecuen=input('\n*
TAMAÑO_POBLACIÓN_ALGORITMO_GENÉTICO_SECUENCIAS:');
vacio3=isempty(tampoblasecuen);
end

vacio4=1;
while vacio4==1 %Sale de while solo cuando hay dato
maxgensec=input('*
MÁXIMA_GENERACIONES_ALGORITMO_GENÉTICO_SECUENCIAS:');
vacio4=isempty(maxgensec);
end

reiteracion=input('* NÚMERO_REITERACIONES_SIN_MEJORAR_CM(Puede dejar
vacio):');
  if isempty(reiteracion)
  reiteracion=maxgensec; % Si no se indica reiteraciones
  end
```

```
mutsec=input('\n\n* PORCENTAJE_MUTACIÓN_ALGORITMO_GENÉTICO_SECUENCIAS (Si deja vacio por defecto es 0.1%): ');
mutsec=mutsec/100;
if isempty(mutsec)
mutsec=0.001; % valor por defecto
end

sobresec=input('*PORCENTAJE_SOBREVIVIENTES_ALGORITMO_GENÉTICO_SECUENCIAS (Si deja vacio por defecto es 50%): ');
sobresec=sobresec/100;
if isempty(sobresec) % Si no selecciona
sobresec=0.5; % valor por defecto
end

corridas=input('\n\n* INGRESAR_NÚMERO_DE_CORRIDAS (Si deja vacio por defecto es 1):');
if isempty(corridas) % Si no selecciona
corridas=1; % valor por defecto
end


%***** PUNTO DE INICIO DE UNA CORRIDA DEL PROGRAMA  ***********

for numerocorridas=1:corridas


tic; % Se inicia el tiempo que mide la ejecución del programa


%       ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
%       ¬¬¬¬¬¬¬¬ SUBPROBLEMA DE ENRUTAMIENTO ¬¬¬¬¬¬¬¬
%       ¬¬¬¬¬¬¬¬ ALGORITMO GENÉTICO DE RUTAS ¬¬¬¬¬¬¬¬
%       ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


display('        PROCESANDO ALGORITMO GENÉTICO DE RUTAS');

%                              DESCRITO EN LA FIGURA 25


%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
%  PREPARAR DATOS (VECTORES, ESCALARES) CON INFORMACIÓN DE USUARIO
%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


Datos=size(T); % T Datos=[(jobs) (operaciones maxima por job) (máquinas)]
jobs=Datos(1);
nmaq=Datos(3);

cantidadoperxjob=[]; % Vector para almacenar la cantidad de operaciones de cada job.
listaoperxjob=[];   % Vector que almacena números de operaciones de cada job.

for i=1:jobs
[f,c]=size(find(T(i,:)));
C=c/nmaq; % "C" es número de operaciones de cada job
cantidadoperxjob(i)=C;  % Ejemplo: cantidadoperxjob=[3 3 4 2]
if i==1
listaoperxjob(1,1:C)=[1:C];% Introduce la primera secuencia numerada del primer Job
else
long=length(listaoperxjob);
```

```
listaoperxjob(1,(long+1):(long+C))=[1:C]; % Ejemplo: listaoperxjob=[1 2 3  1 2 3  1 2 3 4  1 2]
end
end

totalopera=sum(cantidadoperxjob); % Ejemplo: totalopera=12
nq=0;
n=0;
ma=0;
listajobxopera=[]; % Indica el job de cada operación consecutiva

for i=1:jobs
for iter=1:cantidadoperxjob(i)
ma=ma+1;
listajobxopera(ma)=i;  % Ejemplo: listajobxopera=[1 1 1  2 2 2  3 3 3 3  4 4]
end
end

puntinijob=[]; % Vector que contiene localización de donde se inicia cada job.
puntosinicio=1;

for i=2:jobs
puntosinicio=puntosinicio+cantidadoperxjob(i-1);
puntinijob(i)=puntosinicio; %Ejemplo: puntinijob=[1 4 7 11]
end

puntinijob(1)=1; %El primer job siempre se inicia en 1.Ejemplo: puntinijob=[1 4 7 11]
sobrevivenmq=floor(sobremaq*tamapoblamq); % Número de población que sobreviven.
casamientosmq=ceil((tamapoblamq-sobrevivenmq)/2); % Cada casamiento es un cruce
mutacionesmq=ceil((tamapoblamq)*totalopera*mutmaq); % Número de alteración de genes
cuponesxpadremq=1;

for i=2:sobrevivenmq
cuponesxpadremq=[cuponesxpadremq i*ones(1,i)];
 % Ejemplo: para sobrevivenmq=4, cuponesxpadremq= (1 2 2 3 3 3 4 4 4 4)
end

cuponesxpadremq=sobrevivenmq+1-cuponesxpadremq;
%cuponesxpadremq=[4 3 3 2 2 2 1 1 1 1]
totalcuponesmq=length(cuponesxpadremq);
% Ejemplo para sobreviven=4, totalcuponesmq =10


% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬ CALCULAR SUMA UMBRAL DEL USUARIO ¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


sumaesperada=maxwork+totalwork; % Suma umbral puede ser 0 en caso de no especificar


%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
%  PUNTO DE REINICIO (Solo se reinicia para buscar workload y totalworkoad umbrales )
%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


for yy=1:NpoblaNuevas % Solo si usuario lo seleccionó. Normalmente NpoblaNuevas= 1.
```

```
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% ¬¬¬¬¬¬¬ CREAR POBLACIÓN ¬¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


for i=1:tamapoblamq
for j=1:totalopera
x=ceil(nmaq*rand);
poblamq(i,j)=x;
end
end


% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
%¬¬¬¬¬¬ CALCULAR WM Y WT DE MIEMBROS DE POBLACIÓN ¬¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬

% SE EJECUTA FUNCIÓN WORKLOAD - DESCRITO EN LA FIGURA 32


[totalworkload,maxworkload]=WORKLOAD(T,poblamq,tamapoblamq,jobs,cantidadoperxjob,
nmaq,totalopera);


%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% ¬¬¬¬¬¬¬ SUMAR WM Y WT DE CADA MIEMBRO DE LA POBLACIÓN ¬¬¬¬¬
%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬

costosuma=totalworkload+maxworkload;

% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% ¬¬¬¬¬¬ ORDENAR MIEMBROS DE POBLACIÓN DE MENOR A MAYOR SUMA ¬¬¬
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


[costosuma,posminmax]=sort(costosuma);
poblamq=poblamq(posminmax,:);
totalworkload=totalworkload(posminmax);
maxworkload=maxworkload(posminmax);


% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% ¬¬¬¬ REGISTRAR LA MEJOR SUMA (MENOR SUMA DE LA POBLACIÓN) ¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


mejorsuma=costosuma(1); % Para comparar

generamq=0;     % Incializar contador


% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬ EVALUAR MEJOR SUMA¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬

% Es posible que no ingrese a while si la mejor suma evaluada anteriormente cumple
% con el umbral

while mejorsuma>sumaesperada
```

```
%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% INCREMENTAR CONTADOR DE GENERACIONES Y ABORTAR EN CASO DE LIMITE
%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


generamq=generamq+1;

if generamq==maxgenmaq
   break
end


% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% CONTAR REITERACIONES DE "Mejor Suma" Y ABORTAR EN CASO DE LIMITE
%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


 x1(generamq)=mejorsuma;
 if generamq>2
  if x1(generamq)==x1(generamq-1)
     nq=nq+1;
       if nq== reiteracionmq
       break;
       end
  else
  nq=0;
  end
 end


% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬ SELECCIONAR LISTA DE PADRES PARA CRUZAMIENTO ¬¬¬¬
%                              DESCRITO EN LA  FIGURA 28
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


sorteo1mq=ceil(totalcuponesmq*rand(1,casamientosmq)); %sorteo1mq= (7 4 8 6 5 7 8 4)
sorteo2mq=ceil(totalcuponesmq*rand(1,casamientosmq)); % sorteo2mq= (6 10 4 1 9 3 2 6)
listapadres1mq=cuponesxpadremq(sorteo1mq); % listapadres1mq=(4 3 4 3 3 4 4 3)
listapadres2mq=cuponesxpadremq(sorteo2mq); % listapadres2mq=(3 4 3 1 4 2 2 3)
% Se van aparear (4,3), (3,4). . .(4,2), (3,3). Cada apareamiento genera dos nuevos hijos.
%Solo en (3,3) %es inútil, esto en más probable que aparezca en poblaciones pequeñas y
%no es necesario corregir.


% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬ CRUCE ¬¬¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


for i=1:casamientosmq % Desde 1 hasta número de apareos(casamientos)
padre1mq=poblamq(listapadres1mq(i),:); %Ejemplo: padre1mq= [1 4 3 3 4 1 1 5 3 2 1 2]
padre2mq=poblamq(listapadres2mq(i),:); %Ejemplo: padre2mq= [3 4 3 2 5 5 1 4 4 3 3 3]
ubicamq=2*(i-1)+1; % Indicador de posición para guardar los hijos ejemplo: i=1,3,5
hijo1mq=padre1mq;
hijo2mq=padre2mq;
j=ceil(jobs*rand);% En Job arbitrario sus máquinas se intercambien. Ejemplo j=job=2
hijo1mq(puntinijob(j):(puntinijob(j)+cantidadoperxjob(j)-
1))=padre2mq(puntinijob(j):(puntinijob(j)+cantidadoperxjob(j)-1));
```

```
% hijo1mq=[1 4 3 2 5 5 1 5 3 2 1 2]
hijo2mq(puntinijob(j):(puntinijob(j)+cantidadoperxjob(j)-
1))=padre1mq(puntinijob(j):(puntinijob(j)+cantidadoperxjob(j)-1));
% hijo2mq=[3 4 3 3 4 1 1 4 4 3 3 3]
poblamq(sobrevivenmq+ubicamq,:)=hijo1mq;
% pobla(5) se remplaza con hijo1mq, primer cruce
poblamq(sobrevivenmq+ubicamq+1,:)=hijo2mq;
% pobla(6) se remplaza con hijo2mq
end


% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬ MUTACIÓN ¬¬¬¬¬¬¬¬¬¬
%    DESCRITO EN LA  FIGURA 30
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


probabimq=rand; % Selección aleatoria


% ¬¬ PRIMERA MUTACION: una de las máquinas más rápidas de operación ¬¬¬¬¬¬¬¬


if probabimq>0 && probabimq<=0.9

 for i = 1:mutacionesmq
 cromosomq=ceil(rand*(tamapoblamq)); %Selección de cromosoma

 gen1mq=ceil(rand*totalopera);
 jobmq=listajobxopera(gen1mq); %seleccion de un job
 opermq=listaoperxjob(gen1mq);%seleccion de operación del job
 [ti, maqui]=sort(T(jobmq,opermq,:)); %se ordena de menor a mayor tiempo maquina
 indicador=ceil((nmaq/2)*rand); % Selecciona una de la mitad de las maquinas más veloces
 poblamq(cromosomq,gen1mq)=maqui(indicador); %reemplazo
 end

else


% ¬¬¬¬¬¬¬¬ SEGUNDA MUTACIÓN: Selecciona aleatoriamente una máquina¬¬¬¬¬¬¬¬


if probabimq>0.9 && probabimq<=0.95

for i = 1:mutacionesmq
cromosomq=ceil(rand*(tamapoblamq)); %Seleccion de cromosoma
gen1mq=ceil(rand*totalopera);
poblamq(cromosomq,gen1mq)=ceil(nmaq*rand);
end

else


% ¬¬¬¬¬¬¬¬ TERCERA MUTACION: No selecciona máquina más frecuente ¬¬¬¬¬¬¬¬


 for i = 1:mutacionesmq
 cromosomq=ceil(rand*(tamapoblamq)); %Seleccion de cromosoma
 A=poblamq(cromosomq,:); % A contiene el Cromosoma seleccionado
 gen1mq=ceil(rand*totalopera); %Ubicación del gen que cambiara
```

```
 maqmasfrecuente=mode(A); % mode identifica la máquina que más se repite
 maqui=ceil(nmaq*rand); %Selecciono una maquina al azar

while maqui==maqmasfrecuente
  maqui=ceil(nmaq*rand);
  end
 poblamq(cromosomq,gen1mq)= maqui;
 end

end
end


% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% ¬¬¬¬¬¬¬ ¬¬¬ CALCULAR WM Y WT DE MIEMBROS DE POBLACIÓN ¬¬¬¬¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬

%        SE EJECUTA FUNCIÓN WORKLOAD - DESCRITO EN LA  FIGURA 32


[totalworkload,maxworkload]=WORKLOAD(T,poblamq,tamapoblamq,jobs,cantidadoperxjob,
nmaq,totalopera);


%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
%¬¬¬ CALCULAR SUMA DE WM Y WT DE CADA MIEMBRO DE LA POBLACIÓN¬¬¬¬¬¬
%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


costosuma=totalworkload+maxworkload;


% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% ¬¬¬¬¬ ORDENAR MIEMBROS DE POBLACIÓN DE MENOR A MAYOR SUMA ¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


[costosuma,posminmax]=sort(costosuma);
poblamq=poblamq(posminmax,:);
totalworkload=totalworkload(posminmax);
maxworkload=maxworkload(posminmax);


% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% ¬¬¬ REGISTRAR LA MEJOR SUMA (MENOR SUMA DE LA POBLACIÓN) ¬¬¬¬¬
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


mejorsuma=costosuma(1);% Comparar


 end % FIN DE WHILE - Retorna a while mejorsuma>sumaesperada
```

```
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬¬ WM Y WT SON LOS ESPERADOS POR USUARIO ¬¬¬¬¬¬¬
% ¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


% Se ejecuta cuando se aborta de while

if maxworkload(1)<=maxwork && totalworkload(1)<=totalwork
   break  % Si se cumple condición se aborta de FOR
end
 end % Fin de For (for yy=1:NpoblaNuevas)



%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬
%                    RESULTADO MEJOR MIEMBRO DE LA POBLACIÓN
%                        (MÁQUINAS CON ÓPTIMO WM Y WT)
%¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬¬


maquinas=poblamq(1,:); %  Mejor miembro de la población

tiempos=[];
tiem=0;
for i=1:jobs
for j=1:cantidadoperxjob(i)
tiem=tiem+1;
tiempos(tiem)=T(i,j,maquinas(tiem)); % Se registra tiempo de cada máquina
end
end


% SE CONTINUA CON LA EJECUCIÓN DEL ALGORITMO GENÉTICO DE SECUENCIAS
```

```
%                ##############################################
%                ####### SUBPROBLEMA DE SECUENCIAMIENTO   ####
%                ####### ALGORITMO GENÉTICO DE SECUENCIAS ####
%                ##############################################
display('        PROCESANDO ALGORITMO GENETICO DE SECUENCIA');

%                              DESCRITO EN LA  FIGURA 33


%##############################
%####### PREPARAR DATOS #####
%##############################


sobreviven=floor(sobresec*tampoblasecuen);
casamientos=floor((tampoblasecuen-sobreviven)/2);
cuponesxpadre=1;
for i=2:sobreviven
cuponesxpadre=[cuponesxpadre i*ones(1,i)];
end
 cuponesxpadre=sobreviven+1-cuponesxpadre;
 totalcupones=length(cuponesxpadre);
mutaciones=ceil((tampoblasecuen-1)*totalopera*mutsec);


% ######################################################
% ####### CREAR POBLACIÓN DE SECUENCIAS VÁLIDAS ####
% ######################################################


poblasecuen=[];% Arreglo que contiene población
individuo=[];
secuenciacorrecta=[];

for i=1:tampoblasecuen
poblasecuen(i,1:totalopera)=randperm(totalopera);
end

for i=1:tampoblasecuen
posicion1=[];
posicion2=[];
operaciones=zeros;
individuo=poblasecuen(i,:); %Cada cromosoma pasa al vector individuo

% Se ordena con sort al individuo para ubicar posición original de cada operación en
% vector individuo

[B, posicion1]=sort(individuo);

for j=1:jobs
operaciones=cantidadoperxjob(j);  % Se toma el número de operaciones de cada job
if j==1
posicion2(1:operaciones)=sort(posicion1(1:operaciones));
else
limitinf=length(posicion2)+1;
limitsup=limitinf+(operaciones-1);
posicion2(limitinf:limitsup)= sort(posicion1(limitinf:limitsup));
end
```

```
end
secuenciacorrecta(posicion2)=B;
poblasecuen(i,:)=secuenciacorrecta; %Población cumple criterio de precedencia
end


% ############################################################
% ####### CALCULAR MAKESPAN (CM) DE MIEMBROS POBLACIÓN ###
% ############################################################


% SE EJECUTA FUNCIÓN MAKESPAN – DESCRITO EN LA FIGURA 39

costo=MAKESPAN(maquinas,listajobxopera,tiempos,poblasecuen);


% ############################################################
% ####### ORDENAR POBLACIÓN DE MENOR A MAYOR MAKESPAN #
% ############################################################


[costo,posminmax1]=sort(costo); % Ordena costo de menor a mayor
poblasecuen=poblasecuen(posminmax1,:); % Ordena a los miembros de la población

generacion=0;     % Inicializar contador
x1=[];
n=0;


% ##################################################
% ######## SI MAKESPAN ES EL DESEADO ABORTAR ####
% ##################################################


% Si costo(1) del paso previo cumple con umbral ya no se ingresa a while.
while costo(1)>mkdeseado


%##################################################
%####### SI GENERACIONES EN MÁXIMA ABORTAR  ####
%##################################################


generacion=generacion+1; % Incrementar contador

if generacion>maxgensec
break;
end
```

```
%#############################################################
%####### SI REITERACIÓN DE MAKESPAN ES MÁXIMA ABORTAR  #####
%#############################################################


x1(generacion)=costo(1);
if generacion>2
 if x1(generacion)==x1(generacion-1)
    n=n+1;
     if n== reiteracion
        break;
     end
 else
 n=0;
 end
end


% ############################################################
% ####### SELECCIONAR LISTA DE PADRES PARA CRUZAMIENTO ###
% ############################################################


sorteo1=ceil(totalcupones*rand(1,casamientos));
% Ejemplo sorteo1=(30 30 77 117 29 64 84)

sorteo2=ceil(totalcupones*rand(1,casamientos));
% Ejemplo sorteo2=(111 44 14 16 56 45 29)

listapadres1=cuponesxpadre(sorteo1); % lista de padres1=(8 8 12 15 8 11 13)
listapadres2=cuponesxpadre(sorteo2); % lista de padres2=(15 9 5 6 11 9 8)

% Ejemplo: se van acruzar: (8,15),(8,9),(12,5),(15,6),(8,11),(11,9),(13,8)


%######################
%####### CRUCE #######
%######################


for i=1:casamientos % Desde 1 hasta número de cruces (casamientos)
padre1=poblasecuen(listapadres1(i),:); %Ejemplo: padre1= [7  1 4 11 2 12  5 3 8 9 6 10]
padre2=poblasecuen(listapadres2(i),:); %Ejemplo: padre2= [7 11 4  1 2  3 12 5 6 8 9 10]
ubica=2*(i-1)+1;

gen=ceil(rand*totalopera);   % Ejemplo gen=9
origenpadre1=padre1;    % origenpadre1 = [7 1 4 11 2 12 5 3 8 9 6 10]
inicialgen=gen;     % inicialgen=9
padre1(gen)=padre2(gen);
% Ejemplo: padre1= [7  1 4 11 2 12  5 3 6 (se intercambio con 8) 9  6 (conflicto con
% intercambiado) 10]
padre2(gen)=origenpadre1(gen);
 % Ejemplo: padre2= [7 11 4  1 2  3 12 5 8(se intercambio con 6) 8  9 10]
genrepetido=find(origenpadre1 == padre1(gen));
% Encuentra posicion de bit repetido ejemplo: genrepetido=11 (en la posicion 11 de padre1
% está el bit repetido (6)
gen=genrepetido;  % Guarda posición del bit repetido gen=11
while gen~=inicialgen
% Si gen==inicialgen termina proceso. Además si padre1 = padre2 o genes por intercambiar
% son iguales,en ese caso no hay cruzameinto
```

```
padre1(gen)=padre2(gen);        % Final, padre1=[7 1 4 11 2 12 5 3 6 8 9 10]
padre2(gen)=origenpadre1(gen);   % Final, padre2=[7 11 4 1 2 3 12 5 8 9 6 10]
genrepetido=find(origenpadre1 == padre1(gen));
gen=genrepetido;
end
poblasecuen(sobreviven+ubica,:)=padre1;
poblasecuen(sobreviven+ubica+1,:)=padre2;
end


%########################
%####### MUTACIÓN #######
%########################


for i = 1:mutaciones
cromosoma=ceil(rand*(tampoblasecuen-1))+1;

gen1=ceil(rand*totalopera);
gen2=ceil(rand*totalopera);

temp=poblasecuen(cromosoma,gen1);
poblasecuen(cromosoma,gen1)=poblasecuen(cromosoma,gen2);
poblasecuen(cromosoma,gen2)=temp;

end


% ######################################################################
% ## DETECTAR EN CADA CROMOSOMA LAS OPERACIONES DE CADA JOB #####
% #######        Y  CORREGIR  PRECEDENCIA                    #######
% ######################################################################


for i=2:tampoblasecuen
posicion1=zeros;
posicion2=zeros;
operaciones=zeros;
individuo=poblasecuen(i,:);

[B, posicion1]=sort(individuo);
for j=1:jobs
operaciones=cantidadoperxjob(j);
if j==1
posicion2(1:operaciones)=sort(posicion1(1:operaciones));
else
limitinf=length(posicion2)+1;
limitsup=limitinf+(operaciones-1);
posicion2(limitinf:limitsup)= sort(posicion1(limitinf:limitsup));
end
end
secuenciacorrecta(posicion2)=B;
poblasecuen(i,:)=secuenciacorrecta;
end
```

```
% ############################################################
% ####### CALCULAR MAKESPAN (CM) DE MIEMBROS POBLACIÓN ####
% ############################################################

% Se ejecuta Función MAKESPAN – Descrito en la Figura 39


costo=MAKESPAN(maquinas,listajobxopera,tiempos,poblasecuen);


% ############################################################
% ### ORDENAR POBLACIÓN DE MENOR A MAYOR MAKESPAN #######
% ############################################################


[costo,posminmax]=sort(costo);
poblasecuen=poblasecuen(posminmax,:);


end %generacion


tejecucion=toc  % Termina medida del tiempo de ejecución del algoritmo


% ################################################
% ####### EXPORTAR RESULTADOS HOJAS DE EXCEL ###
% ################################################


in=in+4;
vari1=['H', num2str(in)];   % Concatena Variable literal con numero ejemplo C4= ['C',4]
vari2=['B', num2str(in)];
resultado=[maquinas;tiempos;poblasecuen(1,:)];
resultado2=[maxworkload(1),totalworkload(1),costo(1),tejecucion];
xlswrite('RESULTADO_FJSS.xlsx', resultado,'Hoja1',vari1);
xlswrite('RESULTADO_FJSS.xlsx', resultado2,'Hoja1',vari2);



% ########################################
% ####### GENERAR DIAGRAMA DE GANNT ####
% ########################################

% A CONTINUACIÓN SE EJECUTA ARCHIVO “GANTT” QUE CONTIENE CÓDIGO DEL
% ALGORITMO DESCRITO EN LA FIGURA 41

GANTT


end %PUNTO FINAL DE UNA CORRIDA
```

ANEXO B: SOLUCIONES CASOS DE *FJSSP*

A continuación se presentan todos los resultados de las veinte (20) corridas por cada uno de los casos de Kacem que fueron especificados en el ítem 4.1. Los resultados fueron colectados en una hoja Excel exportados directamente por el programa desarrollado. Las soluciones obtenidas son el fundamento del desarrollo del ítem 4.2 y 4.3.

Cada una de las veinte corridas se muestran en las siguientes tablas de este anexo, las cuales son:

- Tabla B1: Solución 4 *Jobs*, 5 máquinas, 12 operaciones de flexibilidad total (20 corridas).
- Tabla B2: Solución 8 *Jobs*, 8 máquinas, 27 operaciones de flexibilidad parcial (20 corridas).
- Tabla B3: Solución 10 *Jobs*, 7 máquinas, 29 operaciones de flexibilidad total (20 corridas).
- Tabla B4: Solución 10 *Jobs*, 10 máquinas, 30 operaciones de flexibilidad total (20 corridas).
- Tabla B5: Solución 15 *Jobs*, 10 máquinas, 56 operaciones de flexibilidad total (20 corridas).

Cada una de las veinte (20) soluciones describe, la relación de “Máquinas” que han sido designadas a cada operación, los “Tiempos” de ejecución de cada máquina y la “Secuencia” (orden) en que se deben ejecutar las operaciones. A cada solución por tanto le corresponde un resultado de *Maximum Workload* (W_M), *Total Workload* (W_T), *Makespan* (C_M) y el tiempo que le toma al algoritmo solucionar el problema. También se presenta el tiempo promedio de la ejecución del algoritmo para las veinte soluciones.

Tabla B1

Respuestas FJSSP 4 Jobs, 5 máquinas, 12 operaciones (20 corridas)

													W_M	W_T	C_M	t (seg)
1	Máquinas	4	2	4	1	5	3	3	2	1	4	1	2	10	32	11	0.06
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	4	7	1	2	8	5	11	9	12	6	3	10				
2	Máquinas	4	2	4	1	5	3	3	2	1	4	1	2	10	32	11	0.04
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	7	4	1	5	2	3	6	11	8	9	12	10				
3	Máquinas	4	2	4	1	5	3	3	2	4	4	1	4	10	32	12	0.11
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	1	11	12	7	2	3	8	4	9	5	6	10				
4	Máquinas	4	2	4	1	1	3	3	2	4	4	1	2	10	32	12	0.06
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	7	1	2	4	3	5	11	8	9	10	6	12				
5	Máquinas	4	2	1	1	5	3	3	2	4	4	1	2	10	32	11	0.08
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	4	1	11	2	7	12	8	9	3	5	10	6				
6	Máquinas	4	2	1	1	5	3	3	2	4	4	1	4	10	32	11	0.09
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	7	1	4	2	5	8	11	6	9	3	12	10				
7	Máquinas	4	2	4	1	1	3	3	2	4	4	1	2	10	32	12	0.03
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	1	2	11	7	8	4	5	3	9	6	10	12				
8	Máquinas	4	2	4	1	5	1	3	2	4	4	1	4	9	32	12	0.03
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	7	1	2	4	8	11	5	6	12	3	9	10				
9	Máquinas	4	2	4	1	5	1	3	2	1	4	1	4	9	32	13	0.57
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	1	2	7	8	11	4	12	9	5	6	3	10				
10	Máquinas	4	2	4	1	5	3	3	2	1	4	1	4	10	32	11	0.04
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	4	1	5	2	7	3	11	8	9	12	10	6				
11	Máquinas	4	2	4	1	5	3	3	2	4	4	1	4	10	32	12	0.22
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	7	1	2	4	11	12	5	6	3	8	9	10				
12	Máquinas	4	2	4	1	1	3	3	2	1	4	1	2	10	32	11	0.10
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	4	1	7	5	11	2	8	12	3	6	9	10				

Fuente: Elaboración propia

Tabla B1 (continuación)
Respuestas FJSSP 4 Jobs, 5 máquinas, 12 operaciones (20 corridas)

13	Máquinas	4	2	4	1	1	3	3	2	1	4	1	4	10	32	11	0.07
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	1	4	7	5	2	3	8	11	12	9	10	6				
14	Máquinas	4	2	4	1	1	3	3	2	1	4	1	2	10	32	11	0.03
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	1	4	5	2	7	8	9	3	10	11	6	12				
15	Máquinas	4	2	4	1	1	3	3	2	4	4	1	4	10	32	12	0.03
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	1	4	7	11	12	5	2	3	8	6	9	10				
16	Máquinas	4	2	1	1	5	3	3	2	4	4	1	4	10	32	11	0.05
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	4	5	7	1	2	8	11	6	12	9	3	10				
17	Máquinas	4	2	4	1	5	3	3	2	1	4	1	2	10	32	11	0.03
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	1	7	2	8	4	5	11	6	9	12	3	10				
18	Máquinas	4	2	4	1	5	3	3	2	4	4	1	4	10	32	12	0.04
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	4	11	1	2	12	7	8	3	5	9	10	6				
19	Máquinas	4	2	4	1	5	3	3	2	4	4	1	2	10	32	12	0.09
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	1	7	4	11	2	3	8	5	9	6	10	12				
20	Máquinas	4	2	4	1	5	3	3	2	1	4	1	4	10	32	11	0.05
	Tiempos	1	4	4	2	5	4	6	1	2	1	1	1				
	Secuencia	1	7	2	4	11	5	3	6	8	12	9	10				
														Tiempo de ejecución promedio (seg)			0.09

Fuente: Elaboración propia, datos del programa

Tabla B2

Respuestas FJSSP 8 Jobs, 8 máquinas, 27 operaciones (20 corridas)

																											W_M	W_T	C_M	t (Seg.)
	Máquinas	5	5	6	3	4	7	5	7	4	1	2	6	3	1	7	6	7	3	8	2	3	8	4	1	2	8	5	12	76	15	0.51
1	Tiempos	3	3	2	3	2	1	4	2	4	1	1	5	2	3	6	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	11	8	14	9	18	1	2	24	25	10	21	4	12	13	5	15	6	19	20	22	16	7	26	23	3	27	17				
	Máquinas	4	5	6	3	4	7	5	7	4	1	2	6	5	1	7	6	7	3	8	2	3	8	4	1	2	8	5	12	76	15	0.53
2	Tiempos	3	3	2	3	2	1	4	2	4	1	1	5	3	3	6	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	14	11	1	8	12	21	18	24	2	19	13	9	15	16	4	5	25	22	20	6	26	17	10	7	23	3	27				
	Máquinas	5	5	6	3	4	7	5	7	4	1	2	6	3	1	7	6	7	3	8	2	3	8	4	1	2	8	5	12	75	15	0.30
3	Tiempos	3	3	2	3	2	1	4	2	4	1	1	5	2	3	6	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	8	14	18	4	11	5	19	21	15	24	12	13	6	9	10	25	1	20	16	2	3	22	26	23	7	27	17				
	Máquinas	4	5	6	3	4	7	5	7	3	1	2	6	5	1	4	6	7	3	8	2	3	8	4	1	2	8	5	12	76	15	0.19
4	Tiempos	3	3	2	3	2	1	4	2	6	1	1	5	3	3	4	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	8	18	1	4	21	11	14	5	6	19	24	25	2	12	13	22	15	16	9	26	7	20	23	27	3	17	10				
	Máquinas	4	5	6	3	4	7	5	7	3	1	2	6	5	1	4	6	7	3	8	2	3	8	4	1	2	8	5	12	76	15	0.86
5	Tiempos	3	3	2	3	2	1	4	2	6	1	1	5	3	3	4	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	18	14	19	24	1	11	4	21	12	2	8	15	5	6	22	25	16	23	20	13	26	9	10	3	7	27	17				
	Máquinas	4	5	6	3	4	7	5	7	3	1	2	6	5	1	4	6	7	3	8	2	3	8	4	1	2	8	5	12	76	15	1.19
6	Tiempos	3	3	2	3	2	1	4	2	6	1	1	5	3	3	4	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	18	1	4	11	2	8	24	19	14	5	15	25	6	7	12	16	21	22	17	9	23	20	10	13	3	26	27				
	Máquinas	4	5	6	3	4	7	5	7	4	1	2	6	3	1	7	6	7	3	8	2	3	8	4	1	2	8	5	12	75	15	0.61
7	Tiempos	3	3	2	3	2	1	4	2	4	1	1	5	2	3	6	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	8	14	18	15	11	21	24	12	19	1	4	25	5	16	6	2	3	17	22	13	26	9	7	20	23	10	27				

Fuente: Elaboración propia

Tabla B2 (Continuación)
Respuestas FJSSP 8 Jobs, 8 máquinas, 27 operaciones (20 corridas)

																											W_M	W_T	C_M	t (Seg.)
	Máquinas	4	5	6	3	4	7	5	7	4	1	2	6	3	1	7	6	7	3	8	2	3	8	4	1	2	8	5	12	75	15	0.67
8	Tiempos	3	3	2	3	2	1	4	2	4	1	1	5	2	3	6	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	8	18	4	11	1	9	2	14	15	24	21	19	12	22	25	3	5	20	16	10	6	13	7	23	26	27	17				
	Máquinas	4	5	6	3	4	7	5	7	3	1	2	6	5	1	4	6	7	3	8	2	3	8	4	1	2	8	5	12	76	15	0.34
9	Tiempos	3	3	2	3	2	1	4	2	6	1	1	5	3	3	4	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	18	4	8	21	11	12	1	19	2	14	5	6	24	13	15	9	10	3	22	25	26	20	23	7	16	27	17				
	Máquinas	5	5	6	3	4	7	5	7	4	1	2	6	3	1	7	6	7	3	8	2	3	8	4	1	2	8	5	12	75	15	0.46
10	Tiempos	3	3	2	3	2	1	4	2	4	1	1	5	2	3	6	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	14	18	24	1	11	4	8	19	21	25	5	12	9	15	2	16	20	10	22	26	6	3	7	23	27	17	13				
	Máquinas	4	5	6	3	4	7	5	7	3	1	2	6	5	1	4	6	7	3	8	2	3	8	4	1	2	8	5	12	76	15	2.78
11	Tiempos	3	3	2	3	2	1	4	2	6	1	1	5	3	3	4	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	1	11	24	18	4	12	19	5	14	21	25	15	2	8	26	16	6	22	20	3	7	13	23	9	10	17	27				
	Máquinas	4	5	6	3	4	7	5	7	3	1	2	6	5	1	4	6	7	3	8	2	3	8	4	1	2	8	5	12	76	15	0.46
12	Tiempos	3	3	2	3	2	1	4	2	6	1	1	5	3	3	4	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	24	1	18	11	8	4	2	12	5	14	19	15	6	3	21	25	22	16	23	13	26	9	17	10	7	27	20				
	Máquinas	4	5	6	3	4	7	5	7	3	1	2	6	5	1	4	6	7	3	8	2	3	8	4	1	2	8	5	12	76	15	1.19
13	Tiempos	3	3	2	3	2	1	4	2	6	1	1	5	3	3	4	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	18	4	19	1	24	11	12	5	21	14	2	8	25	15	6	26	22	27	20	7	9	16	23	3	13	17	10				
	Máquinas	4	5	6	3	4	7	5	7	3	1	2	6	5	1	4	6	7	3	8	2	3	8	4	1	2	8	5	12	76	15	0.28
14	Tiempos	3	3	2	3	2	1	4	2	6	1	1	5	3	3	4	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	18	1	4	24	21	5	14	11	19	25	8	9	26	2	12	27	22	10	15	13	20	3	6	7	23	16	17				

Fuente: Elaboración propia

Tabla B2 (Continuación)

Respuestas FJSSP 8 Jobs, 8 máquinas, 27 operaciones (20 corridas)

																											W_M	W_T	C_M	t (Seg.)
	Máquinas	4	5	6	3	4	7	5	7	4	1	2	6	3	1	7	6	7	3	8	2	3	8	4	1	2	8	5	12	75	15	0.70
15	Tiempos	3	3	2	3	2	1	4	2	4	1	1	5	2	3	6	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	21	11	1	18	4	19	5	12	8	14	15	22	24	2	3	13	16	25	6	26	9	7	20	27	10	17	23				
	Máquinas	4	5	6	3	4	7	5	7	4	1	2	6	5	1	7	6	7	3	8	2	3	8	4	1	2	8	5	12	76	15	1.12
16	Tiempos	3	3	2	3	2	1	4	2	4	1	1	5	3	3	6	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	18	14	11	24	4	25	1	8	5	9	15	2	12	19	21	6	13	16	22	17	3	23	10	26	20	7	27				
	Máquinas	4	5	6	3	4	7	5	7	3	1	2	6	5	1	4	6	7	3	8	2	3	8	4	1	2	8	5	12	76	15	0.77
17	Tiempos	3	3	2	3	2	1	4	2	6	1	1	5	3	3	4	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	18	1	19	2	4	21	8	11	14	12	24	5	25	15	22	9	20	10	13	16	6	3	26	7	27	17	23				
	Máquinas	4	5	6	3	4	7	5	7	4	1	2	6	3	1	7	6	7	3	8	2	3	8	4	1	2	8	5	12	75	15	0.67
18	Tiempos	3	3	2	3	2	1	4	2	4	1	1	5	2	3	6	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	18	4	1	8	14	5	21	9	11	24	2	12	25	15	19	22	6	7	16	20	23	17	10	13	26	3	27				
	Máquinas	4	5	6	3	4	7	5	7	4	1	2	6	3	1	7	6	7	3	8	2	3	8	4	1	2	8	5	12	75	15	1.33
19	Tiempos	3	3	2	3	2	1	4	2	4	1	1	5	2	3	6	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	14	11	24	8	18	21	19	22	12	1	4	15	5	2	25	6	16	17	20	9	10	26	23	7	27	3	13				
	Máquinas	4	5	6	3	4	7	5	7	4	1	2	6	3	1	7	6	7	3	8	2	3	8	4	1	2	8	5	12	75	15	1.04
20	Tiempos	3	3	2	3	2	1	4	2	4	1	1	5	2	3	6	2	3	1	4	5	2	5	3	2	4	1	1				
	Secuencia	11	18	14	19	24	8	12	21	15	1	4	9	22	13	5	6	25	2	26	23	20	16	10	3	17	7	27				
																													Tiempo de ejecución promedio (seg.)			0.80

Fuente: Elaboración propia

Tabla B3

Respuestas FJSSP 10 Jobs, 7 máquinas, 29 operaciones (20 corridas)

																												W_M	W_T	C_M	t (Seg.)
	Máquinas	1	6	6	7	1	7	6	5	1	4	1	2	1	2	3	3	3	7	7	6	4	4	2	5	2	7	2	6	1	11	61	11	0.20
1	Tiempos	1	1	4	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	3	1	1	8	2	4	2	2	4	1	1				
	Secuencia	6	21	27	9	1	15	2	18	28	24	29	4	7	8	12	16	13	19	25	22	10	5	17	14	11	23	26	3	20				
	Máquinas	1	5	7	7	1	7	3	5	1	4	5	2	1	2	3	3	3	7	6	6	2	4	2	5	2	4	2	6	1	11	61	12	2.27
2	Tiempos	1	1	3	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	4	1	1	8	2	4	2	2	4	1	1				
	Secuencia	21	4	15	12	18	13	22	9	1	27	19	24	25	10	28	2	6	7	16	23	5	17	26	8	14	3	29	20	11				
	Máquinas	1	5	6	7	1	7	6	2	1	4	1	2	1	4	3	3	3	7	7	6	4	4	2	5	2	7	2	6	1	11	61	11	0.51
3	Tiempos	1	1	4	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	3	1	1	8	2	4	2	2	4	1	1				
	Secuencia	6	7	27	15	4	28	9	24	1	29	21	18	22	12	13	25	19	16	2	26	8	10	23	17	5	11	3	14	20				
	Máquinas	1	6	7	7	1	7	6	5	1	4	5	2	1	2	3	3	3	7	6	6	4	4	2	5	2	7	2	6	1	11	61	11	0.53
4	Tiempos	1	1	3	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	4	1	1	8	2	4	2	2	4	1	1				
	Secuencia	21	4	15	27	12	22	24	1	9	16	2	28	6	10	7	18	8	5	13	25	11	3	19	26	20	14	17	29	23				
	Máquinas	1	6	7	7	1	7	6	5	1	4	1	2	1	4	3	3	3	7	3	6	2	4	2	5	2	7	2	6	1	11	61	11	0.35
5	Tiempos	1	1	3	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	4	1	1	8	2	4	2	2	4	1	1				
	Secuencia	9	21	12	24	27	18	15	4	1	2	6	22	28	7	16	5	10	19	13	25	29	3	17	23	11	26	8	20	14				
	Máquinas	1	6	7	7	1	7	6	2	1	4	1	2	1	4	3	3	3	7	6	6	4	4	2	5	2	7	2	6	1	11	61	11	0.31
6	Tiempos	1	1	3	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	4	1	1	8	2	4	2	2	4	1	1				
	Secuencia	12	15	6	1	13	18	24	2	7	27	4	21	25	22	19	9	3	20	5	10	28	29	16	26	8	11	23	14	17				
	Máquinas	1	6	6	7	1	7	3	5	1	4	5	2	1	2	3	3	3	7	7	6	2	4	2	5	2	7	2	6	1	11	61	11	0.35
7	Tiempos	1	1	4	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	3	1	1	8	2	4	2	2	4	1	1				
	Secuencia	24	1	18	21	4	12	9	15	13	6	7	19	27	2	16	22	25	10	3	28	8	29	26	14	5	20	23	17	11				

Fuente: Elaboración propia

Tabla B3 (Continuación)
Respuestas FJSSP 10 Jobs, 7 máquinas, 29 operaciones (20 corridas)

																													W_M	W_T	C_M	t (Seg.)
	Máquinas	1	6	6	7	1	7	6	5	1	4	5	2	1	2	3	3	3	7	7	6	2	4	2	3	2	7	2	6	1	11	61	11	0.61
8	Tiempos	1	1	4	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	3	1	1	8	2	4	2	2	4	1	1				
	Secuencia	24	1	15	2	21	12	9	13	27	16	3	22	18	4	28	6	19	5	25	10	26	7	29	20	8	17	14	23	11				
	Máquinas	1	5	6	7	1	7	3	2	1	4	1	2	1	4	3	3	3	7	7	6	4	4	2	5	2	7	2	6	1	11	61	11	0.40
9	Tiempos	1	1	4	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	3	1	1	8	2	4	2	2	4	1	1				
	Secuencia	21	1	4	9	22	18	15	16	6	12	27	24	28	2	5	19	13	29	7	3	25	20	17	26	10	11	8	14	23				
	Máquinas	1	6	6	7	1	7	3	5	1	4	1	2	1	2	3	3	3	7	7	6	2	4	2	5	2	4	2	6	1	11	61	12	0.91
10	Tiempos	1	1	4	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	3	1	1	8	2	4	2	2	4	1	1				
	Secuencia	15	24	1	6	21	4	27	9	7	28	16	22	29	12	2	25	18	10	3	13	23	8	19	26	17	5	11	14	20				
	Máquinas	1	6	7	7	1	7	3	2	1	4	1	2	1	4	3	3	3	7	6	6	4	4	2	5	2	7	2	6	1	11	61	11	1.07
11	Tiempos	1	1	3	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	4	1	1	8	2	4	2	2	4	1	1				
	Secuencia	27	24	4	15	9	1	28	12	29	2	21	22	13	6	7	18	10	5	25	16	14	17	3	11	8	19	23	26	20				
	Máquinas	1	6	7	7	1	7	6	5	1	4	5	2	1	2	3	3	3	7	3	6	2	4	2	5	2	4	2	6	1	11	61	12	1.06
12	Tiempos	1	1	3	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	4	1	1	8	2	4	2	2	4	1	1				
	Secuencia	15	24	16	1	2	9	21	27	22	28	18	25	4	12	19	17	23	6	7	10	29	13	20	8	3	14	11	26	5				
	Máquinas	1	6	6	7	1	7	6	5	1	4	1	2	1	2	3	3	3	7	7	6	4	4	2	3	2	7	2	6	1	11	61	11	0.37
13	Tiempos	1	1	4	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	3	1	1	8	2	4	2	2	4	1	1				
	Secuencia	1	4	9	21	22	18	12	27	13	24	15	5	25	2	16	3	6	19	28	17	7	29	14	23	20	10	11	26	8				
	Máquinas	1	6	7	7	1	7	6	5	1	4	5	2	1	2	3	3	3	7	6	6	2	4	2	5	2	4	2	6	1	11	61	12	3.77
14	Tiempos	1	1	3	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	4	1	1	8	2	4	2	2	4	1	1				
	Secuencia	21	24	6	7	18	9	27	25	22	10	19	20	4	1	15	5	12	23	8	13	11	14	2	28	16	3	26	29	17				

Fuente: Elaboración propia

Tabla B3 (Continuación)

Respuestas FJSSP 10 Jobs, 7 máquinas, 29 operaciones (20 corridas)

																													W_M	W_T	C_M	t (Seg.)
	Máquinas	1	6	6	7	1	7	3	5	1	4	5	2	1	4	3	3	3	7	7	6	2	4	2	5	2	7	2	6	1	11	61	11	0.73
15	Tiempos	1	1	4	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	3	1	1	8	2	4	2	2	4	1	1				
	Secuencia	24	15	21	1	27	2	12	9	18	4	22	13	5	28	3	6	19	25	16	10	7	23	14	29	26	8	17	20	11				
	Máquinas	1	5	6	7	1	7	6	5	1	4	5	2	1	2	3	3	3	7	7	6	2	4	2	5	2	4	2	6	1	11	61	12	1.67
16	Tiempos	1	1	4	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	3	1	1	8	2	4	2	2	4	1	1				
	Secuencia	15	1	6	24	21	22	27	25	9	16	28	18	10	29	12	13	7	26	2	19	4	23	8	17	3	11	20	5	14				
	Máquinas	1	6	7	7	1	7	6	5	1	4	5	2	1	2	3	3	3	7	3	6	4	4	2	5	2	7	2	6	1	11	61	11	5.77
17	Tiempos	1	1	3	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	4	1	1	8	2	4	2	2	4	1	1				
	Secuencia	27	12	6	18	1	2	9	24	21	4	13	25	22	5	15	16	28	14	23	19	7	8	17	29	10	3	26	20	11				
	Máquinas	1	6	7	7	1	7	6	5	1	4	1	2	1	2	3	3	3	7	6	6	4	4	2	5	2	7	2	6	1	11	61	11	0.47
18	Tiempos	1	1	3	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	4	1	1	8	2	4	2	2	4	1	1				
	Secuencia	24	12	1	21	9	6	27	4	15	18	16	13	25	22	2	7	3	10	28	26	29	5	14	8	19	17	23	20	11				
	Máquinas	1	5	7	7	1	7	3	2	1	4	5	2	1	4	3	3	3	7	6	6	4	4	2	5	2	7	2	6	1	11	61	11	0.39
19	Tiempos	1	1	3	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	4	1	1	8	2	4	2	2	4	1	1				
	Secuencia	21	12	6	4	18	27	1	22	5	24	9	25	15	13	2	7	10	8	16	3	17	28	14	26	19	23	20	29	11				
	Máquinas	1	6	7	7	1	7	6	5	1	4	1	2	1	2	3	3	3	7	6	6	2	4	2	5	2	4	2	6	1	11	61	12	1.06
20	Tiempos	1	1	3	3	2	1	1	2	2	1	1	1	2	1	4	1	2	2	4	1	1	8	2	4	2	2	4	1	1				
	Secuencia	4	21	6	9	24	7	18	1	27	12	8	25	22	15	16	13	2	10	14	3	23	5	11	17	19	28	20	29	26				
																															Tiempo de ejecución promedio (seg)			1.14

Fuente: Elaboración propia

Tabla B4

Respuestas FJSSP 10 Jobs, 10 máquinas, 30 operaciones (20 corridas)

																														W_M	W_T	C_M	t (Seg.)
	Máquinas	1	3	4	1	10	10	10	8	7	7	3	4	9	9	4	6	9	9	1	3	6	5	2	2	6	7	6	6	4	7	6	42	7	0.25
1	Tiempos	1	1	1	2	1	2	1	1	1	1	3	1	2	1	1	2	2	1	1	1	1	2	3	2	1	1	1	1	1	2				
	Secuencia	25	28	7	10	1	22	8	13	2	19	20	16	21	14	29	30	26	17	23	9	15	24	4	5	18	3	11	27	6	12				
	Máquinas	1	3	4	1	10	10	10	8	2	9	2	4	9	9	4	6	9	7	1	3	6	5	7	2	6	7	6	6	2	4	6	42	8	2.40
2	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	4	2	1	1	1	1	1	2				
	Secuencia	19	10	7	8	22	28	29	13	14	4	30	1	20	9	23	15	16	11	25	26	5	17	21	12	2	3	18	6	24	27				
	Máquinas	1	3	4	1	10	3	10	8	7	9	2	4	9	9	2	6	9	7	1	3	4	5	10	2	6	7	6	6	4	4	6	42	9	1.49
3	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	4	2	1	1	1	1	1	2				
	Secuencia	16	7	13	1	2	19	25	10	22	20	17	26	23	11	3	8	4	14	15	18	28	21	9	24	5	29	12	27	30	6				
	Máquinas	1	3	4	1	4	3	10	8	2	7	2	4	9	9	4	6	9	9	1	3	6	5	7	2	3	7	6	6	2	4	6	42	8	1.81
4	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	4	2	1	1	1	1	1	2				
	Secuencia	10	4	13	25	7	1	5	11	26	12	28	8	19	2	16	6	29	14	20	17	27	22	3	9	15	18	21	23	24	30				
	Máquinas	1	3	4	1	7	10	10	8	2	9	3	4	9	9	4	6	9	7	1	3	6	5	2	2	6	7	6	3	4	4	6	42	8	1.62
5	Tiempos	1	1	1	2	1	2	1	1	1	1	3	1	2	1	1	2	2	1	1	1	1	2	3	2	1	1	1	1	1	2				
	Secuencia	10	28	7	25	8	9	16	19	22	20	13	11	4	29	23	26	30	21	14	17	5	6	15	12	1	2	24	27	18	3				
	Máquinas	1	3	4	1	7	10	10	4	7	9	2	4	9	9	4	6	9	7	1	3	4	5	2	6	3	7	6	3	2	7	6	42	8	1.92
6	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	3	3	1	1	1	1	1	2				
	Secuencia	22	13	10	14	7	19	16	28	8	1	29	25	2	26	30	23	27	17	4	5	6	24	3	20	15	9	18	21	11	12				
	Máquinas	1	2	4	1	10	3	10	4	7	7	2	4	9	9	2	6	9	7	1	3	4	5	10	2	3	7	6	3	4	7	6	42	9	1.86
7	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	4	2	1	1	1	1	1	2				
	Secuencia	22	10	28	7	25	4	23	13	8	5	19	26	11	16	1	17	27	12	18	29	2	20	30	14	3	6	15	24	9	21				

Fuente: Elaboración propia

Tabla B4 (Continuación)
Respuestas FJSSP 10 Jobs, 10 máquinas, 30 operaciones (20 corridas)

																														W_M	W_T	C_M	t (Seg.)
	Máquinas	1	3	4	1	10	10	10	8	2	9	2	4	9	9	2	6	9	7	1	3	4	5	7	2	3	7	6	3	4	4	6	42	8	2.79
8	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	4	2	1	1	1	1	1	2				
	Secuencia	22	28	10	13	1	7	14	19	11	4	2	16	29	25	5	17	8	9	20	23	30	12	3	26	21	6	15	18	27	24				
	Máquinas	1	3	4	1	10	3	10	8	7	9	2	4	9	9	2	6	9	7	1	3	4	5	2	6	3	7	6	3	4	7	6	42	8	1.80
9	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	3	3	1	1	1	1	1	2				
	Secuencia	22	10	7	25	13	28	26	23	11	16	8	1	19	20	2	29	4	27	9	17	21	5	3	18	12	14	30	24	15	6				
	Máquinas	1	2	4	1	4	3	10	8	2	7	2	4	9	9	4	6	9	9	1	3	4	5	10	2	3	7	6	3	4	7	6	42	8	1.53
10	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	4	2	1	1	1	1	1	2				
	Secuencia	22	1	25	16	19	10	28	13	17	29	14	18	4	26	7	27	11	2	20	30	21	3	5	12	23	8	9	24	15	6				
	Máquinas	1	3	4	1	7	3	10	4	2	9	2	4	9	9	2	6	9	7	1	3	6	5	10	2	6	7	6	3	4	7	6	42	8	1.60
11	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	4	2	1	1	1	1	1	2				
	Secuencia	7	10	25	16	11	28	13	26	14	4	29	15	19	27	12	30	22	5	20	17	1	21	2	8	6	23	3	9	24	18				
	Máquinas	1	3	4	1	7	3	10	4	2	7	2	4	9	9	4	6	9	9	1	3	6	5	10	2	6	7	6	6	4	7	6	42	8	1.43
12	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	4	2	1	1	1	1	1	2				
	Secuencia	7	25	13	14	10	26	4	16	17	19	8	18	11	15	1	28	2	29	5	3	12	9	22	20	30	23	6	24	27	21				
	Máquinas	1	2	4	1	7	10	10	8	7	7	3	4	9	9	4	6	9	9	1	3	6	5	2	2	3	7	6	6	4	4	6	42	7	0.23
13	Tiempos	1	1	1	2	1	2	1	1	1	1	3	1	2	1	1	2	2	1	1	1	1	2	3	2	1	1	1	1	1	2				
	Secuencia	22	10	25	13	14	28	29	1	11	4	5	26	7	16	6	17	19	18	2	27	8	30	23	24	20	15	21	12	9	3				
	Máquinas	1	3	4	1	4	3	10	4	2	7	2	4	9	9	4	6	9	9	1	3	4	5	10	2	3	7	6	3	7	7	6	42	8	1.45
14	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	4	2	1	1	1	1	1	2				
	Secuencia	22	13	16	7	8	10	11	4	25	28	19	9	29	12	20	23	21	30	26	1	17	5	2	18	27	14	6	3	15	24				

Fuente: Elaboración propia

Tabla B4 (Continuación)
Respuestas FJSSP 10 Jobs, 10 máquinas, 30 operaciones (20 corridas)

																														W_M	W_T	C_M	t (Seg.)
	Máquinas	1	3	4	1	7	3	10	4	7	7	2	4	9	9	4	6	9	7	1	3	4	5	10	2	3	7	6	3	7	2	6	42	8	2.51
15	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	4	2	1	1	1	1	1	2				
	Secuencia	19	7	22	13	1	28	8	20	25	10	11	29	21	30	2	4	16	17	23	5	6	14	3	26	15	27	9	24	18	12				
	Máquinas	1	2	4	1	4	10	10	4	7	9	3	4	9	9	4	6	9	7	1	3	6	5	2	2	3	7	6	6	4	7	6	42	7	1.17
16	Tiempos	1	1	1	2	1	2	1	1	1	1	3	1	2	1	1	2	2	1	1	1	1	2	3	2	1	1	1	1	1	2				
	Secuencia	7	25	1	10	19	22	2	4	8	28	13	29	3	16	5	23	20	26	14	11	15	24	9	6	12	27	17	30	18	21				
	Máquinas	1	3	4	1	10	3	10	4	7	9	2	4	9	9	2	6	9	7	1	3	6	5	10	2	3	7	6	6	7	7	6	42	9	1.47
17	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	4	2	1	1	1	1	1	2				
	Secuencia	10	4	25	7	8	19	22	20	13	28	16	1	11	9	26	14	27	23	24	21	12	29	5	15	17	18	2	6	30	3				
	Máquinas	1	3	4	1	10	10	10	4	2	7	2	4	9	9	4	6	9	9	1	3	6	5	7	2	6	7	6	3	2	4	6	42	8	2.20
18	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	4	2	1	1	1	1	1	2				
	Secuencia	28	13	22	7	8	29	10	30	11	1	25	19	4	26	12	5	14	27	23	2	20	15	16	17	18	9	6	24	3	21				
	Máquinas	1	3	4	1	10	10	10	8	2	9	2	4	9	9	4	6	9	7	1	3	6	5	7	2	6	7	6	3	2	4	6	42	8	1.45
19	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	4	2	1	1	1	1	1	2				
	Secuencia	10	16	22	4	19	28	13	7	1	29	2	8	23	25	9	11	17	20	21	24	26	30	27	18	3	5	14	6	12	15				
	Máquinas	1	2	4	1	4	3	10	8	7	9	2	4	9	9	4	6	9	7	1	3	6	5	10	2	3	7	6	3	2	7	6	42	8	1.47
20	Tiempos	1	1	1	2	1	2	1	1	1	1	2	1	2	1	1	2	2	1	1	1	1	2	4	2	1	1	1	1	1	2				
	Secuencia	7	19	10	28	13	22	4	8	11	12	29	9	1	20	30	25	16	17	26	14	21	5	23	2	3	18	6	27	15	24				
																																Tiempo de ejecución promedio (seg)			1.62

Fuente: Elaboración propia

Tabla B5

Respuestas FJSSP 15 Jobs, 10 máquinas, 56 operaciones (20 corridas)

																																																										W_M	W_T	C_M	t (seg)
01	Máquinas	1	1	3	9	4	3	10	8	7	9	2	4	5	5	6	10	9	6	7	4	2	1	1	2	7	8	4	1	10	3	7	10	10	1	10	2	1	6	9	2	8	5	6	7	2	10	10	2	1	9	8	5	8	6	3	5	11	91	12	4.97
	Tiempos	1	1	1	4	1	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	2	1	1	2	4	2	1	2	1	2	1	1	2	1	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
	Secuencia	13	25	37	21	5	17	45	41	9	29	53	49	33	46	18	23	1	38	10	6	30	26	7	2	47	54	3	50	34	14	42	11	43	19	12	27	35	51	39	24	52	31	55	15	8	22	28	16	20	56	44	32	40	4	48	36				
02	Máquinas	1	1	3	9	4	3	10	4	2	9	2	4	5	5	6	10	9	6	7	4	2	1	1	2	7	8	4	1	10	1	7	10	10	8	10	2	7	6	3	7	8	5	9	7	2	10	10	2	1	9	8	5	8	6	3	5	11	91	12	12.68
	Tiempos	1	1	1	4	1	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	2	1	1	2	4	2	1	2	1	2	1	1	2	1	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
	Secuencia	13	25	17	45	18	1	9	37	29	5	33	41	10	53	6	49	38	21	46	14	42	39	30	7	54	2	34	3	15	50	19	51	11	55	47	35	31	23	24	22	20	8	48	43	52	26	4	12	27	16	28	56	40	44	36	32				
03	Máquinas	1	1	3	6	4	3	10	8	7	9	2	4	5	5	6	10	9	6	7	4	2	1	1	2	7	8	4	1	10	3	7	10	10	8	10	2	7	6	3	2	8	5	9	7	2	10	10	2	1	9	8	5	1	9	6	5	11	91	12	7.75
	Tiempos	1	1	1	4	1	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	2	1	1	2	4	2	1	2	1	2	1	1	2	1	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
	Secuencia	37	13	1	17	29	9	14	53	41	5	45	38	18	33	25	46	49	26	21	30	23	34	2	10	42	39	11	54	47	50	22	19	31	40	43	27	35	6	3	51	15	48	20	36	28	7	16	32	44	12	55	56	4	8	52	24				
04	Máquinas	1	1	3	9	4	3	10	8	2	7	2	4	5	5	6	10	9	6	7	6	2	1	1	2	7	8	4	1	10	3	7	10	10	8	10	2	1	6	9	7	8	5	9	7	2	10	10	2	1	9	8	5	8	6	6	5	11	91	12	9.16
	Tiempos	1	1	1	4	1	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	2	1	1	2	4	2	1	2	1	2	1	1	2	1	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
	Secuencia	37	41	49	9	13	23	29	45	30	25	42	17	1	33	38	5	21	6	39	50	10	53	14	26	18	2	31	54	34	11	51	27	22	19	3	46	47	55	24	15	43	20	35	7	44	48	32	28	36	12	56	16	52	4	8	40				
05	Máquinas	1	1	3	9	7	3	10	4	7	9	2	4	5	5	6	10	9	6	7	4	2	1	1	2	7	8	4	1	10	3	7	10	10	8	10	2	1	6	9	2	8	5	6	7	2	10	10	2	1	9	8	5	8	3	3	5	11	91	12	6.99
	Tiempos	1	1	1	4	1	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	2	1	1	2	4	2	1	2	1	2	1	1	2	1	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
	Secuencia	5	17	45	9	10	29	25	49	41	18	30	33	37	13	50	46	14	53	38	34	23	51	26	47	39	1	6	42	11	19	7	24	2	21	48	54	3	31	15	22	35	20	43	40	44	32	8	16	27	55	4	52	36	56	12	28				
06	Máquinas	1	2	3	6	4	3	10	4	7	9	2	4	5	5	6	10	9	6	7	4	2	1	1	2	7	8	4	1	10	3	7	10	10	1	10	2	1	6	3	7	8	5	9	7	2	10	10	2	1	9	8	5	1	9	6	5	11	91	12	5.38
	Tiempos	1	1	1	4	1	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	2	1	1	2	4	2	1	2	1	2	1	1	2	1	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
	Secuencia	45	17	53	29	13	25	26	54	46	5	18	30	41	9	33	21	31	27	10	1	37	14	6	42	47	7	2	49	15	55	50	51	3	34	11	35	56	32	43	38	19	22	12	23	16	28	48	39	36	24	20	4	52	40	8	44				
07	Máquinas	1	1	3	9	4	3	10	8	7	9	2	4	5	5	6	10	9	6	7	4	2	1	1	2	7	8	4	1	10	3	7	10	10	8	10	2	1	6	9	7	8	5	6	7	2	10	10	2	1	9	8	5	1	3	6	5	11	91	12	8.52
	Tiempos	1	1	1	4	1	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	2	1	1	2	4	2	1	2	1	2	1	1	2	1	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
	Secuencia	53	49	25	29	5	21	17	13	37	33	18	45	41	1	9	10	11	12	34	30	31	6	14	15	46	54	2	50	26	23	27	19	35	42	47	3	20	51	52	38	7	43	28	36	55	39	16	56	40	8	22	4	24	32	44	48				

Fuente: Elaboración propia

Tabla B5 (Continuación)

Respuestas FJSSP 15 Jobs, 10 máquinas, 56 operaciones (20 corridas)

			W_M	W_T	C_M	t (seg)
08	Máquinas	1 1 3 6 7 3 10 4 7 9 2 4 5 5 6 10 9 6 7 4 2 1 1 2 7 8 4 1 10 1 7 10 10 8 10 2 1 6 9 2 8 5 9 7 2 10 10 2 1 9 8 5 8 9 6 5	11	91	12	12.28
	Tiempos	1 1 1 4 1 2 1 1 1 1 2 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 2 1 1 2 4 2 1 2 1 2 1 1 2 1 2 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2				
	Secuencia	53 37 17 9 38 33 25 5 45 1 10 13 11 41 29 49 6 18 26 34 54 19 50 30 2 14 46 42 55 27 3 12 39 47 43 21 23 51 31 15 56 7 48 35 4 24 40 22 28 36 20 16 52 8 32 44				
09	Máquinas	1 1 3 6 4 3 10 8 2 9 2 4 5 5 6 10 9 6 7 6 2 1 1 2 7 8 4 1 10 1 7 10 10 8 10 2 7 6 3 7 8 5 9 7 2 10 10 2 1 9 8 5 1 9 3 5	11	91	13	18.08
	Tiempos	1 1 1 4 1 2 1 1 1 1 2 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 2 1 1 2 4 2 1 2 1 2 1 1 2 1 2 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2				
	Secuencia	33 13 29 45 37 53 5 17 6 18 41 38 21 54 14 34 19 9 25 30 42 46 39 26 47 49 10 27 11 50 35 15 31 16 48 55 1 2 56 12 3 7 51 20 52 36 43 40 28 23 24 22 32 4 8 44				
10	Máquinas	1 1 3 6 7 3 10 8 7 9 2 4 5 5 6 10 9 6 7 6 2 1 1 2 7 8 4 1 10 3 7 10 10 8 10 2 1 6 3 2 8 5 9 7 2 10 10 2 1 9 8 5 1 3 3 5	11	91	13	12.69
	Tiempos	1 1 1 4 1 2 1 1 1 1 2 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 2 1 1 2 4 2 1 2 1 2 1 1 2 1 2 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2				
	Secuencia	13 53 17 29 9 41 45 30 37 33 54 25 5 46 49 1 18 14 34 35 47 6 2 10 23 50 38 3 42 19 21 55 11 51 15 39 26 48 27 36 7 22 31 52 43 20 56 44 4 32 12 16 24 40 8 28				
11	Máquinas	1 1 3 6 7 3 10 4 2 9 2 4 5 5 6 10 9 6 7 4 2 1 1 2 7 8 4 1 10 3 7 10 10 8 10 2 1 6 9 7 8 5 9 7 2 10 10 2 1 9 8 5 8 9 3 5	11	91	13	16.53
	Tiempos	1 1 1 4 1 2 1 1 1 1 2 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 2 1 1 2 4 2 1 2 1 2 1 1 2 1 2 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2				
	Secuencia	53 17 9 13 37 45 29 25 33 23 38 54 10 49 30 41 14 1 50 34 46 24 5 26 21 47 35 11 18 42 55 48 2 6 27 3 31 19 7 12 20 8 39 40 15 22 51 36 28 52 43 44 16 4 56 32				
12	Máquinas	1 1 3 6 4 3 10 4 7 9 2 4 5 5 6 10 9 6 7 4 2 1 1 2 7 8 4 1 10 3 7 10 10 1 10 2 1 6 9 2 8 5 9 7 2 10 10 2 1 9 8 5 8 3 3 5	11	91	12	4.45
	Tiempos	1 1 1 4 1 2 1 1 1 1 2 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 2 1 1 2 4 2 1 2 1 2 1 1 2 1 2 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2				
	Secuencia	41 29 33 9 1 49 53 45 17 30 5 6 13 14 46 37 21 42 7 2 25 23 18 10 11 47 26 27 38 50 51 15 34 22 52 39 16 3 24 35 54 55 12 19 31 40 4 48 20 28 43 8 32 44 56 36				
13	Máquinas	1 1 3 6 4 3 10 8 7 9 2 4 5 5 6 10 9 6 7 4 2 1 1 2 7 8 4 1 10 3 7 10 10 8 10 2 1 6 9 7 8 5 9 7 2 10 10 2 1 9 8 5 1 9 6 5	11	91	12	5.27
	Tiempos	1 1 1 4 1 2 1 1 1 1 2 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 2 1 1 2 4 2 1 2 1 2 1 1 2 1 2 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2				
	Secuencia	17 53 41 33 45 9 37 18 5 49 10 29 21 1 25 54 2 46 13 6 42 11 50 19 23 47 43 12 14 38 55 34 56 48 7 20 26 35 27 30 31 51 44 15 3 22 4 52 24 39 32 16 28 40 36 8				
14	Máquinas	1 1 3 6 7 3 10 8 7 9 2 4 5 5 6 10 9 6 7 4 2 1 1 2 7 8 4 1 10 3 7 10 10 8 10 2 1 6 3 2 8 5 6 7 2 10 10 2 1 9 8 5 8 9 3 5	11	91	12	20.40
	Tiempos	1 1 1 4 1 2 1 1 1 1 2 1 1 1 1 1 1 2 2 2 1 1 1 1 1 1 2 1 1 2 4 2 1 2 1 2 1 1 2 1 2 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2				
	Secuencia	45 29 37 17 33 41 53 13 14 5 9 25 1 10 2 21 30 46 15 11 26 23 49 6 18 27 34 42 31 54 38 19 28 50 51 43 47 24 7 16 3 55 8 39 56 35 52 48 12 44 4 32 36 20 40 22				

Fuente: Elaboración propia

Tabla B5 (Continuación)

Respuestas FJSSP 15 Jobs, 10 máquinas, 56 operaciones (20 corridas)

																																																										W_M	W_T	C_M	t (seg)
15	Máquinas	1	1	3	6	4	3	10	4	7	9	2	4	5	5	6	10	9	6	7	4	2	1	1	2	7	8	4	1	10	3	7	10	10	8	10	2	7	6	9	2	8	5	9	7	2	10	10	2	1	9	8	5	1	3	6	5	11	91	12	6.26
	Tiempos	1	1	1	4	1	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	2	1	1	2	4	2	1	2	1	2	1	1	2	1	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
	Secuencia	41	23	29	30	53	5	37	17	45	49	9	21	13	33	46	54	38	34	42	1	47	18	25	35	2	14	10	31	39	11	55	56	6	50	40	36	12	26	24	43	7	19	3	32	15	22	27	51	16	44	52	4	8	20	28	48				
16	Máquinas	1	1	3	6	4	3	10	8	2	9	2	4	5	5	6	10	9	6	7	4	2	1	1	2	7	8	4	1	10	3	7	10	10	8	10	2	1	6	9	7	8	5	6	7	2	10	10	2	1	9	8	5	8	3	3	5	11	91	13	26.49
	Tiempos	1	1	1	4	1	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	2	1	1	2	4	2	1	2	1	2	1	1	2	1	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
	Secuencia	5	49	17	45	13	1	9	41	50	14	53	21	2	33	37	18	29	34	42	10	46	23	51	30	25	15	19	35	38	26	6	7	11	39	31	47	36	48	43	54	40	20	3	24	52	55	22	16	56	27	28	4	12	32	44	8				
17	Máquinas	1	1	3	9	4	3	10	8	7	7	2	4	5	5	6	10	9	6	7	6	2	1	1	2	7	8	4	1	10	3	7	10	10	8	10	2	1	6	3	2	8	5	6	7	2	10	10	2	1	9	8	5	8	9	3	5	11	91	12	4.72
	Tiempos	1	1	1	4	1	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	2	1	1	2	4	2	1	2	1	2	1	1	2	1	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
	Secuencia	17	53	41	9	13	1	37	33	49	29	18	45	5	10	25	26	14	54	46	6	30	38	11	21	15	47	55	50	22	34	48	19	51	31	2	42	23	56	27	35	3	43	28	24	7	12	4	52	39	32	20	8	36	40	44	16				
18	Máquinas	1	1	3	6	7	3	10	8	7	9	2	4	5	5	6	10	9	6	7	4	2	1	1	2	7	8	4	1	10	3	7	10	10	8	10	2	1	6	3	2	8	5	6	7	2	10	10	2	1	9	8	5	8	9	3	5	11	91	12	5.61
	Tiempos	1	1	1	4	1	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	2	1	1	2	4	2	1	2	1	2	1	1	2	1	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
	Secuencia	5	9	21	41	29	17	10	53	25	30	18	13	26	45	33	23	49	46	14	6	15	37	38	19	42	1	11	50	51	2	31	47	24	54	22	39	27	40	34	43	44	3	7	20	28	16	35	12	48	4	55	52	56	36	32	8				
19	Máquinas	1	2	3	6	4	3	10	4	7	9	2	4	5	5	6	10	9	6	7	4	2	1	1	2	7	8	4	1	10	3	7	10	10	1	10	2	1	6	9	7	8	5	9	7	2	10	10	2	1	9	8	5	8	9	6	5	11	91	12	14.71
	Tiempos	1	1	1	4	1	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	2	1	1	2	4	2	1	2	1	2	1	1	2	1	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
	Secuencia	5	1	17	53	29	45	13	21	9	25	30	37	49	18	33	10	6	46	14	38	41	34	2	15	54	42	22	39	19	50	26	55	43	11	47	20	27	35	48	12	3	23	31	28	7	4	24	32	8	40	56	36	51	52	44	16				
20	Máquinas	1	1	3	9	7	3	10	4	2	9	2	4	5	5	6	10	9	6	7	6	2	1	1	2	7	8	4	1	10	1	7	10	10	8	10	2	1	6	3	7	8	5	6	7	2	10	10	2	1	9	8	5	8	9	3	5	11	91	13	27.72
	Tiempos	1	1	1	4	1	2	1	1	1	1	2	1	1	1	1	1	1	2	2	2	1	1	1	1	1	1	2	1	1	2	4	2	1	2	1	2	1	1	2	1	2	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2				
	Secuencia	53	21	25	37	5	38	9	33	17	49	54	50	29	10	18	13	41	34	45	46	23	6	30	1	14	7	2	55	8	39	24	42	3	11	43	26	4	35	51	15	19	56	20	22	31	36	44	40	52	47	48	27	32	12	16	28				

Tiempo promedio ejecución (Seg.) 11.53

Fuente: Elaboración propia