UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA
MAESTRÍA Y DOCTORADO EN CIENCIAS E INGENIERÍA

"ANÁLISIS DE MÉTODOS DE LA PROGRAMACIÓN DE LOS TRABAJOS EN TALLERES DE FLUJO HÍBRIDO CON TIEMPOS DE CAMBIO DE PARTIDA"

T E S I S

Que para obtener el grado de
MAESTRO EN INGENIERÍA
Presenta

Elia Leyva Sánchez

DIRECTOR
Dra. Larisa Burtseva

Mexicali, B. C. **Marzo, 2009**

RESUMEN de la Tesis de **Elia Leyva Sánchez,** presentada como requisito parcial para la obtención del grado de MAESTRO EN INGENIERÍA en CIENCIAS DE LA COMPUTACIÓN. Mexicali, Baja California, México. Marzo de 2009.

ANÁLISIS DE MÉTODOS DE LA PROGRAMACIÓN DE LOS TRABAJOS EN TALLERES DE FLUJO HÍBRIDO CON TIEMPOS DE CAMBIO DE PARTIDA

Resumen aprobado por:

Dra. Larisa Burtseva
Director de Tesis

Los procesos industriales discretos tales como manufactura de tableros de circuitos impresos, cerámica, etc., presentan frecuentemente problemas en la programación de la producción. Entre todos los modelos, el Taller de Flujo Híbrido con tiempos de cambio de partida dependientes de la secuencia de los trabajos, pertenece a los más difíciles de optimizar. En consecuencia de la elevada complejidad no son aplicables los procedimientos analíticos. Los métodos evolutivos tales como algoritmos genéticos e inmunes son apropiados para su resolución, lo que se confirma mediante el análisis bibliográfico.

Las partes inmanentes de un algoritmo evolutivo son los métodos de selección, cruzamiento y mutación, mediante los cuales las permutaciones de trabajos se modifican. En Talleres de Flujo Híbrido con tiempos de cambio de partida dependientes de la secuencia, los operadores de cruzamiento apropiados tienen mayor efecto sobre la calidad de la solución, ya que toman en consideración tanto los tiempos de procesamiento de trabajos y de cambio de partida, como la disposición de los tiempos libres en las máquinas para realizar ajustes. En esta investigación se analizan siete operadores de cruzamiento apropiados para este tipo de modelos buscando el tiempo mínimo de ejecución de todos los trabajos: OBX, PPX, OSX, 2PX, KPX, PMX y OX, con el objetivo de contrastar y especificar cuales son los que contribuyen a generar un mejor resultado. El experimento computacional realizado para Talleres de Flujo Híbrido con dos y tres etapas muestra que el mejor resultado se obtiene con el operador OBX. Para el análisis de los datos del experimento fue aplicada la metodología que utiliza las variancias medias, lo que permitió comparar los resultados de algoritmos para diferentes talleres. Los algoritmos de cruzamiento fueron implementados en el sistema computacional PLARETF.

Palabras Clave: *Taller de Flujo Híbrido, Tiempo de Cambio de Partida Dependiente de la Secuencia, Algoritmo Evolutivo, Operador de Cruzamiento.*

ABSTRACT of the thesis, presented by Elia Leyva Sánchez, in order to obtain the MASTER of ENGINEERING DEGREE. Mexicali, Baja California, México. March, 2009.

ANALYSIS OF METHODS OF JOBS SCHEDULING IN HYBRID FLOWSHOPS WITH SETUP TIMES

Approved by:

Dr. Larisa Burtseva Thesis Advisor

Discrete industrial processes such as manufacture of printing circuit board, ceramic, etc., present often problems in the production scheduling. Among all models, the Hybrid Flowshop with a sequence dependent setup times, belongs to the most diffícult for optimizing. As a consequent of its high complexity the analytic procedures are not applicable. The evolutionary methods, such as genetic and immune algorithms, are appropriated for its resolution, as confírmed by the bibliographical analysis.

The immanent parts of an evolutionary algorithm are the selection methods, crossing and mutation, that allow the modification of the job permutations. In an Hybrid Flowshop with a sequence dependent setup times, the appropriate operators of crossover have a greater effect on the solution quality, since they consider the times of Jobs processing and setup, also the disposition of machine free time to carry out adjustments. In mis investigation, seven operators of crossoever for this type of models with the criterion of minimum time of execution of all Jobs are analyzed; applaying OBX, PPX, OSX, 2PX, KPX, PMX and OX. The objective is to contrast and specify the contributions to genérate better results. The computational experiment carried out for Hybrid Flowshop with two and three stages indicates that the best result is obtained with the OBX operator. In this experiment a modification of the methodology of the statistic analysis was utilized, applying the variance average. It permits to compare the algorithm results for different shops. The algorithms of crossover were implemented in the computational system PLARETF.

Keywords: *Hybrid Flowshop, Sequence Dependent Setup Times, Evolutionary Algorithm, Crossover Operator*.

AGRADECIMIENTOS

Por permitirme realizar este proyecto a CONACYT brindando su valioso apoyo económico; a la Universidad Autónoma de Baja California y su Facultad de Ingeniería unidad Mexicali, por permitirme ingresar a su programa de investigación y culminar este proyecto de tesis; a la Dra. Larisa Burtseva, por su asesoría y dirección, a los Maestros y Doctores, por compartir en las clases y en la revisión de este proyecto sus conocimientos y experiencia no solo en lo académico; a Rainier Romero Parra por su asesoría y a Rodolfo Ruiz Nangusé por su camaradería. Finalmente a mi pareja Dr. Octavio Lázaro Mancilla, a mis hijos David Leyva Sánchez y Ievelia Lázaro Leyva por su amor, apoyo, paciencia y comprensión.

DEDICATORIA

A la SOA

Al SMA Serge Ranauld de la Ferriere,

al SHM José Manuel Estrada Vázquez y

al VSA José Marcelli Noli

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

CAPÍTULO 1

INTRODUCCIÓN

1.1 Definición del problema

La programación de trabajos en la manufactura incluye la asignación de órdenes de pedidos a los recursos disponibles en la producción, con el propósito de cumplir con criterios de optimización previamente establecidos, tales como la minimización de la fecha máxima de finalización de todos los trabajos procesados, minimización de la tardanza de los pedidos con respecto a las fechas límites de cumplimiento, etc. Por lo regular, estas decisiones se toman en el nivel operativo afectando directamente a la eficiencia y rentabilidad de la empresa. Por tal motivo se busca constantemente mejorar las técnicas y metodologías que permitan dicha toma de decisiones tomando en cuenta las particularidades en los procesos productivos.

Es conocido que muchos problemas en la programación de la producción son difíciles de resolver por lo complicado de la naturaleza de los mismos (Jungwattanakit et al, 2007). El primer problema se planteó y resolvió de manera abstracta en los años 50´s a través de un algoritmo para un taller de dos máquinas sucesivas (Johnson, 1954). A partir de esa fecha se desarrolló una nueva línea de investigación con el fin de crear nuevos algoritmos capaces de resolver los problemas en la programación de trabajos, avanzando con la sofisticación de los talleres reales.

Con el tiempo los algoritmos han aumentado su complejidad y han desarrollado su terminología. Así por ejemplo un Taller de Flujo (TF), es físicamente una línea de máquinas que realizan operaciones a un objeto determinado formando un flujo de trabajos por las máquinas.

Los talleres más complejos tienen conjuntos de máquinas dispuestos en paralelo, a los cuales se les conoce como Talleres de Flujo Flexible (TFF) si las máquinas en cada conjunto son iguales e Híbrido si alguna de las máquinas posee diferentes características a las restantes del conjunto al que pertenece.

El propósito de esta tesis es la investigación de un problema frecuente en la industria electrónica, así como los métodos que se aplican para resolverlo. Consiste en la minimización de tiempo de ejecución de los trabajos en plantas de producción del tipo Taller de Flujo Híbrido (TFH) cuando el cambio de partida implica el ajuste de la máquina y los tiempos del ajuste dependen directamente del trabajo anteriormente procesado por la máquina.

Para aplicar los métodos que resuelven a TFH, se utiliza el sistema computacional PLARETF (Programación de Recursos en Talleres Flujo) (Romero et al, 2007), desarrollado para la resolución e investigación de diversos problemas en la programación de la producción, el cual se ha complementado agregando operadores frecuentes en el desarrollo de algoritmos tanto heurísticos como de la inteligencia artificial, permite brindar una mejor resolución a talleres de producción cada vez más sofisticados, como los flexibles e híbridos.

1.2 Antecedentes

La primera investigación del TFH se debe a Arthanary y Ramaswany [Arthanary et al, 1971]. Posteriormente los trabajos Gupta (1971), Graham (1979), Lawler (1993), Gupta (1998), Allahverdi (1999) se enfocan también al TFH. Estas investigaciones dieron lugar al concepto de Tiempos de Cambio de Partida (TCP, *setup time*). Una unidad de los TCP, es el tiempo para cambiar de un trabajo a otro en la misma máquina. Los resultados obtenidos mencionan que el TCP se incluye en el tiempo de proceso, o se descarta por ser insignificante.

Los casos más estudiados en los últimos años, sobre los TF son aquellos en que los TCP son separados de los tiempos de proceso. Estos se definen como anticipatorios, entendiendo que el tiempo de partida del siguiente trabajo inicia tan pronto como una máquina se encuentre libre para procesarlo (Cheng, 2003).

En muchas situaciones reales los tiempos de cambio de partida no son anticipatorios y una máquina inicia solo cuando llega el trabajo a la misma, y esta se encuentra lista para procesarlo, surgiendo el concepto de Tiempos de Cambio de Partida dependientes de la secuencia (TCP) de los trabajos. Basados en varias situaciones prácticas se ha clasificado el TF con TCP en varias categorías, las cuales se describen a detalle posteriormente.

La presente investigación se enfoca al problema de TFH TCP, el cual ha tenido modificaciones a través de los años y se ha clasificado según su complejidad como *NP*-duro, cuando los tiempos de partida se presentan en todas las máquinas del taller. Por ejemplo, Gupta y Darrow muestran que la programación de calendarios permutacionales (cuando la secuencia de todos los trabajos en todas las etapas es idéntica) no siempre alcanza al óptimo en caso de tener dos máquinas (Gupta et al, 1986). Sin embargo, los estudios solo se enfocan a las metodologías generales para hallar las permutaciones, no así a sus componentes que afectan directamente a la calidad de la solución.

En el desarrollo de las técnicas para resolver problemas tales como TF TCP se han generado restricciones en la programación de las permutaciones. Por ejemplo Bellman extiende la teoría del análisis para TF de dos a tres etapas donde una sola máquina posee TCP, mostrando que en este caso el conjunto de permutaciones planificadas contiene un calendario $C_{\max}$ mínimo, el cual se resuelve mediante programación dinámica (Bellman et al, 1982). Mas tarde Gupta y Darrow mencionan que la planeación de permutaciones en un problema semejante (mientras la secuencia de los trabajos es idéntica) no siempre se obtiene la permutación óptima cuando se tiene el caso de dos máquinas (Gupta et al, 1986). Posteriormente Szware y Gupta extienden el problema de Taller de Flujo de dos etapas añadiendo tiempos de cambio de partida a problemas de Taller de Flujo con múltiples etapas y desarrollan un análisis heurístico para resolverlo (Szware et al, 1987). Diez años más tarde Parthasarathy y Rajendra añadieron al problema de Taller de Flujo SDJST el peso de la tardanza (Parthasarathy et al, 1997). Cheng, mencionan que si el problema Taller de Flujo SDJST se trata con TSP usando una máquina es *NP-duro* (Cheng et al, 2000).

Existen muchas referencias desarrolladas sobre este problema, en la Tabla 1.1 se enlistan algunos de los problemas tratados por diferentes autores con sus respectivas restricciones. La mayoría de las investigaciones se han dirigido a tratar de resolver el criterio de minimizar el tiempo de finalización de los trabajos en un taller, denotado por $C_{\max}$. También se incluyen algunas restricciones sobre las características de los trabajos, tales como trabajo sin espera (*nwt, no wait*), asignación de fecha de entrada a la primera operación (*release dates*), retraso de un trabajo (*tardines*), peso o prioridad de un trabajo (*weight*).

Tabla 1.1. Investigaciones sobre Talleres de Flujo con *tiempos de cambio de partida*, Cheng (2000).

No. de etapas	Referencia	Año	Técnica de solución	**Criterios y/o Condiciones**
2	Corwin y Esogbue	1974	Programación dinámica	**Maximizar el *makespan*** $C_{\max}$ **(Tiempo de partida en una sola máquina, óptimo)**
	Bellman, Esogbue y Nabeshima	1982	Programación dinámica	**Maximizar el *makespan*** $C_{\max}$ **(Tiempo de partida en una sola máquina, óptimo)**
	Uskup y Smith	1975	Brach & Bound	**TSC (Con due dates, óptimo)**
	Szware y Gupta	1987	Tipo-Múltiple	$C_{\max}$ **(tiempos de partida adicionados, óptimo)**
	Gupta y Darrow	1985 1986	Constructiva	$C_{\max}$ **(aproximado)**
	Voß	1993	Heurística	$C_{\max}$ **(máquinas múltiples en dos etapas, aproximado)**
3	Bellman, Esogbue y Nabeshima	1982	Programación dinámica	$C_{\max}$ **(tiempo de partida en una sola máquina, óptimo)**
m	Gupta	1969 1975	Lexi Search	$C_{\max}$ **TC (óptimo)**
	Srikar y Ghosh ()	1986	MILP	$C_{\max}, \sum C_i$, **(óptimo)**
	Stafford y Tseng ()	1990	MILP	$C_{\max}, \sum C_i$, **(*nwt*, optimo)**
	Gupta	1986	TSP	$C_{\max}, \sum C_i$, **(nwt, Buffer Limitado, óptimo o aproximado)**
	Gupta	1988	Multisort	$C_{\max}$ **(aproximado)**
	Gupta,Das y Ghosh	1995	Brach & Bound	$C_{\max}$ **(óptimo)**
	Rios-Mercado y Bard	1996	MILP	$C_{\max}$ **(óptimo)**
		1998b	Brach & Cut	$C_{\max}$ **(óptimo)**
		1998b	Brach & Bound	$C_{\max}$ **(óptimo)**
	Simons	1992	TPS constructivo	$C_{\max}$ **(aproximado)**
	Das, Gupta y Khumawala	1995	Savings Index	$C_{\max}$ **(aproximado)**
	Rios-Mercado y Bard	1998a 1999a	TSP,GRASP	$C_{\max}$ **(aproximado)**
	Szware y Gupta	1987	Multi-sort	$C_{\max}$ **(tiempos de partida adicionados, aproximado)**
	Norman	1999	Constructive local Search	$C_{\max}$ **(Buffer Finito, aproximado)**
	Bianco, Dell'Olmo y Giordani	1999	Heurística Brach & Bound	$C_{\max}$ **(nwt, release date, optimo y aproximado)**
	Parthasarathy y Rajendra	1997a	Simulated Annealing	$\sum w_i T_i$ **(aproximado)**
		1997b	Simulated Annealing	$\max w_i T_i$ **(aproximado)**

La presente investigación se enfocara sobre el problema específico de TFH TCP dependientes de la secuencia de los trabajos, de dos y tres etapas, aplicando un algoritmo genético para resolverlo.

Es necesario realizar un esfuerzo para desarrollar investigaciones acerca de este problema, ya que la generalización del mismo conlleva a un campo muy fértil de

investigación de los métodos para hallar el calendario que cumpla con criterios establecidos, así como de las técnicas con las cuales se analizan los datos de resultantes de los métodos. Hallar un calendario óptimo es de gran utilidad en la industria, porque en muchos casos una producción óptima depende de una eficaz programación de sus talleres de trabajo, por ejemplo en industrias como las textiles, electrónica, cerámica, etc.

1.3 Objetivos

- Analizar los métodos para la resolución de problemas de programación de trabajos para distintos modelos de THF TCP
- Desarrollar recomendaciones para la resolución de problemas de programación de trabajos en THF TCP

1.4 Metodología

La metodología que se sigue en esta investigación considera:

- El aprendizaje del modelo bajo estudio y su representación práctica.
- El análisis bibliográfico para aprender el problema profundamente y hacer las conclusiones acerca de los algoritmos disponibles para su resolución.
- El análisis de los talleres donde se presentan TCP y su clasificación.
- El estudio de los algoritmos y métodos aplicables para resolución del problema de THF TCP.
- El diseño del experimento
- La implementación del algoritmo a aplicar
    - La generación de datos de entrada
    - La aplicación del algoritmo
    - La configuración de los datos de salida
- El análisis de datos de salida
- El desarrollo de recomendaciones para la resolución de problemas de programación de trabajos con distintos modelos de TFH TCP, así como las conclusiones de este trabajo

1.5 Esquema general de la tesis

El contenido de esta investigación inicia con el Capítulo 2 donde se describe el escenario general de los TF, así como el taller objeto de investigación de esta tesis que es el TFH TCP, se describen formalmente los conceptos necesarios que definen al THF, las características mediante las cuales se asignan las tareas a procesar en los recursos, se relatan sus antecedentes. Se clasifica el problema en cuanto a su complejidad computacional. Se describen los diferentes tiempos de cambio de partida y su relación con el TFH. Posteriormente se describen los diferentes métodos y los algoritmos, que se han aplicado al problema de TFH TCP dependientes de la secuencia. En el Capítulo 3 se describe el funcionamiento del algoritmo que se aplicara en la experimentación, así como sus componentes, considerando como objetivo analizar los operadores de cruzamiento que son componentes fundamentales para el buen funcionamiento de ese tipo de algoritmos. En el Capítulo 4 se describe la implementación del algoritmo y sus componentes, se presenta el diseño del experimento y el análisis de resultados mediante métodos estadísticos. En el Capítulo 5 se presenta las conclusiones generales sobre el trabajo de investigación.

CAPÍTULO 2

PROBLEMA DE TALLER DE FLUJO HÍBRIDO CON TIEMPOS DE CAMBIO DE PARTIDA DEPENDIENTES DE LA SECUENCIA

Muchos ambientes en la producción suponen varias etapas, u operaciones, en la elaboración de los trabajos. Cada operación se ejecuta por una máquina especial, o una de varias máquinas. Las máquinas capaces de ejecutar la misma operación forman un grupo. Estas máquinas son completamente idénticas o se distinguen entre sí por la velocidad, marca, o algunas características tecnológicas.

Las actividades se definen como trabajos, y los recursos de cualquier tipo como las máquinas. Se supone que el número de trabajos y máquinas es conocido de antemano.

Los problemas de programación de los trabajos que se llevan a cabo en un ámbito industrial se formulan a través de una notación abstracta llamada “taller”. Se entiende por taller a la disposición física de las máquinas en una planta productiva. El procesamiento de los trabajos forma un flujo al pasar por las máquinas. En este capítulo se describirán los diferentes patrones de flujo que se presentan en los ambientes de producción.

2.1 Taller de Flujo Híbrido con Tiempos de Cambio de Partida

Enseguida se describe cada patrón de flujo mediante la disposición física de los recursos dentro del taller, culminando con el Taller de Flujo Híbrido, objeto de investigación de esta tesis. Además, se presenta la clasificación de los Tiempos de Cambio de Partida relacionados a con el flujo en la asignación de los trabajos a los Talleres.

2.1.1 Patrón de Flujo.

Taller de Flujo

En un Taller de Flujo (TF) las máquinas son empleadas en serie para procesar los trabajos, siguiendo el mismo orden, es decir, una línea tecnológica para la producción Figura 2.1.

El primer problema de TF fue estudiado por Johnson quien planteó el problema de encontrar una solución óptima para fecha máxima de finalización de todos los trabajos en dos máquinas sucesivas (Johnson, 1954). A partir de este estudio en los últimos 50 años el TF ha evolucionado en su complejidad, hasta considerar múltiples máquinas. En la siguiente figura se presenta un esquema abstracto de este taller.

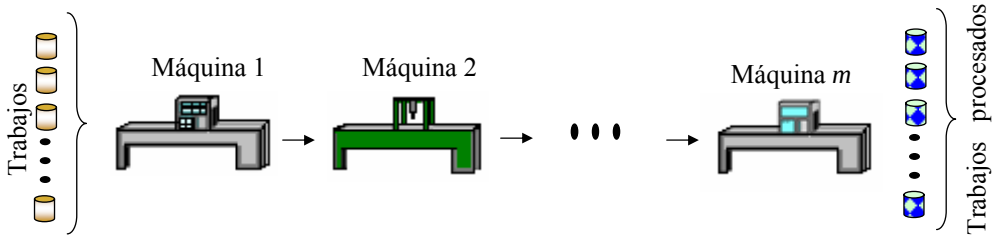


Figura 2.1 Diagrama que representa el modelo de TF con tres máquinas en serie.

Taller de Flujo Flexible

En un Taller de Flujo Flexible (TFF) los trabajos se procesan en m etapas distintas, en una sola máquina por cada etapa. Las máquinas de cada etapa son idénticas Figura 2.2.

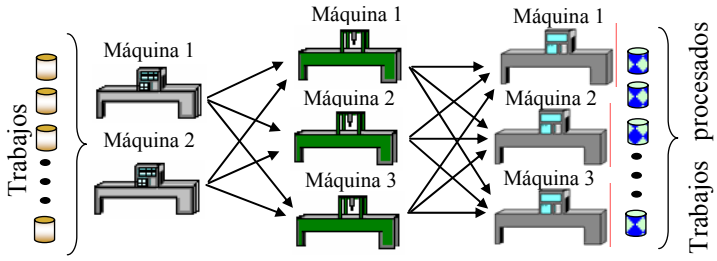


Figura 2.2 Modelo abstracto de TFF con tres etapas, con 2 máquinas en la primera etapa, 3 en la segunda y 3 en la tercera, todas ellas llevan a cabo el mismo tipo de operación en cada etapa.

Taller de Flujo Hibrido.

Un Taller de Flujo Híbrido (TFH) esta compuesto por m etapas, cada trabajo se procesa en una sola máquina en cada etapa. Al menos una de las etapas contiene máquinas de

distintas velocidades u otras características, su forma gráfica es similar a la de TFF como se observa en la Figura 2.3.

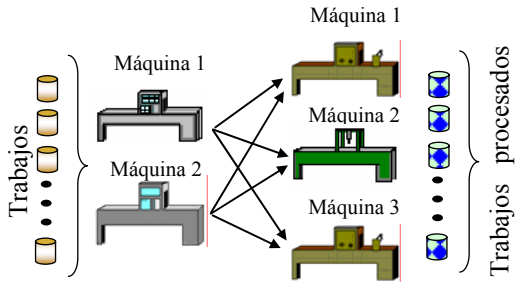


Figura 2.3 Modelo abstracto de un TFF con tres etapas, donde cada $m_i, i = 1..3$ es un conjunto de máquinas que llevan a cabo el mismo tipo de operación.

Se sigue en la presente investigación la terminología de acuerdo a la Teoría de Programación (*Scheduling Theory*) proporcionada por Pinedo (2002), así como su notación formal.

El *THF* está definido por un conjunto de m etapas de proceso, $M = \{1,\ldots,m\}$, y cada etapa contiene un conjunto $M_i = \{1,\ldots,m_i\}$ con m_i máquinas paralelas no relacionadas que procesan los trabajos, $|M_i| \geq 1$. Se tiene un conjunto de *n* trabajos $J = \{1,2,\ldots,n\}$ que deben ser procesados en un TFH. Cada trabajo j se considera como una secuencia de m operaciones a realizar $j = \{O_{1j},\ldots O_{mj}\}$. La operación O_{ij} en la secuencia es procesada en una sola máquina de cada etapa con $p_{il,j}$ unidades de tiempo, que indica el tiempo de procesamiento del trabajo j, $j \in J$, en la máquina l, $l \in M_i$ de la etapa $i, i = \overline{1,m}$.

Sea una permutación π de n trabajos. Si el trabajo se encuentra en la $j-esima$ posición de la permutación, se denota como $\pi_{(j)}, j \in J$. Como todo trabajo es procesado en cada etapa, así se tendrán m operaciones por trabajo. $C_{i,\pi_{(j)}}$ es el tiempo

de finalización de cada trabajo $\pi_{(j)}$ en la etapa i, donde $i \in M$. L_{i_l} es el trabajo que ya paso por la máquina l en la etapa $i, l \in M_i$.

El término $S_{i_l,j,k}$ representa el tiempo de cambio de partida que depende de la secuencia de los trabajos, es decir, el tiempo necesario de preparación de la máquina l en la etapa i para procesar el trabajo k una vez que ya fue procesado el trabajo j.

Se aplican los criterios de la optimización, como la minimización de tiempo total de producción (*makespan*), fecha límite de finalización (*production due dates*), mínimo costo de almacenamiento, etc.

Asignación a las máquinas:

Un problema de programación se describe mediante la tripleta $\alpha|\beta|\gamma$ (Graham, 1979), donde:

α especifica el ambiente de recursos (o máquinas);
β describe las propiedades de las trabajos y tareas;
γ es el criterio de optimización.

Las notaciones que toma el parámetro α son:

- *Fm (Flowshop)* denota al taller de flujo con *m* etapas.
- *FF(Flexible Flowshop)* es el taller de flujo flexible con *m* etapas.
- *HF(Hybrid Flowshop)* es el taller de flujo híbrido con *m* etapas.

Las opciones del parámetro β son:

- ϕ sin descripción específicas sobre los trabajos.
- r_j fechas de entrada.
- s_{ijk} y s_{ij} tiempos de cambio de partida independientes y dependientes de la secuencia respectivamente.
- *prmp* o *pmtn,* permite la interrupción durante las ejecuciones de las operaciones.
- *Prec,(precedence)* existen relaciones de precedencia entre los trabajo, es decir no se podrá realizar otro trabajo hasta que el trabajo predecesor este terminado.
- *Brkdwn* (*breakdowns*), las máquinas están sujetas a averías o a períodos donde no se procesan trabajos.

- M_j *(machine eligibility constraints),* restricción de uso de máquinas.
- *Prmu,* (*permutation*)la solución se busca entre las permutaciones.
- *Block,* (*blocking*) es la ejecución sin parar de trabajos en una máquina.
- *nwt, s*in espera (*no wait*) un trabajo no espera entre dos máquinas.
- *Recrc,* recirculación (*recirculation*) un trabajo es procesado más de una vez por una máquina.

Las opciones para γ son:

- C_j, tiempo o fecha en que se finaliza el proceso del trabajo j en el taller.
- d_j (*due days*), los tiempos de inicio de los primeros trabajos.
- L_j, holgura del trabajo frente a su fecha de finalización.
- T_j, retraso "*tardiness*" de un trabajo *j.*
- E_j, adelanto "earliness" de un trabajo j.
- U_j, tiempo máximo de finalización del proceso del trabajo.

2.1.2 Clasificación de los Tiempos de Cambio de Partida

Por tiempo de preparación de una máquina, se entiende a la cantidad de tiempo para su preparación antes de iniciar la ejecución de un trabajo después de uno anterior. En este periodo la máquina queda inactiva. Se suele expresar al inicio de la operación de cambio de partida como tiempo de ajuste, o al final como tiempo de desmontaje, o se asumen ambos. Entonces, se entiende por Tiempo de Cambio de Partida como el tiempo requerido para que una máquina dada cambie de un trabajo a otro nuevo. Este incluye el tiempo de ajuste y calibración de la máquina, así como la preparación de componentes para el trabajo que sigue. Pinedo indica que el Tiempo de Cambio de Partida (TCP) es un factor significativo para la programación de la producción en todos los modelos de flujo, ya que fácilmente se consume más de 20% de la capacidad disponible de las máquinas si no son bien manejadas (Pinedo, 1995).

En general los Tiempos de Cambio de Partida en problemas de programación de trabajos en una máquina se clasifican de la siguiente manera (Cheng, 2000):

1. Tiempos de Cambio independiente del trabajo procesado. Incluye tiempos para las actividades tales como los saltos requeridos para la instalación fija de cada trabajo en la máquina.
2. Tiempos de Cambio dependiente del trabajo procesado. Incluye tiempos para las actividades de instalación de un trabajo y el tiempo de ajuste de la máquina.
3. Tiempo de Remoción independiente del trabajo procesado. Incluye tiempos para actividades como disminución de las guías, de los accesorios y/o herramientas, inspección/marcado de herramientas, limpieza de máquinas y áreas adyacentes.
4. Tiempo de Remoción dependiente del trabajo procesado. Incluye tiempos para actividades como la designación de herramientas para el trabajo a procesar.

La ocurrencia de estos tiempos en los talleres se presenta en tres fases:
Fase 1. Tiempos de Cambio independiente del trabajo, llamada partidas separadas
Fase 2. Tiempos de Cambio dependiente del trabajo procesado, llamada fase de procesado.
Fase3. TR llamada fase de tiempos de finalización, (Cheng, 2000).

Los casos más estudiados en los últimos años, sobre el Taller de Flujo (*flowshop)* son aquellos en que los tiempos de partida son separados de los tiempos de proceso. Estos se llaman anticipatorios, entendiendo que el tiempo de partida del siguiente trabajo inicia tan pronto como una máquina se encuentre libre para procesarlo, (Cheng, 2003).

En muchas situaciones reales los Tiempos de Cambio de partida no son anticipatorios y la máquina inicia solo cuando arriba el trabajo a la misma, estando lista para procesarlo. Basados en varias situaciones prácticas los investigadores han clasificado el Taller de Flujo con Tiempos de Cambio de partida en varias categorías, como se muestra en la Figura 2.4, (Cheng, 2000), a continuación se describen cada uno de ellos.

Taller de flujo con trabajos con Tiempos de Cambio de Partida Independientes de la secuencia (TF TCP IS)

Su valor es idéntico sea cual sea el trabajo a secuenciar o el recurso donde se secuencia. En estos casos, el tiempo de cambio se incluye o no dentro del tiempo de procesamiento del trabajo. Un caso particular es considerarlo como valor nulo.

Taller de Flujo con trabajos con Tiempos de Cambio de Partida dependientes de la secuencia (TF TCP)

Cada Tiempo de Cambio tiene valor diferente en función de los trabajos o las máquinas donde se secuencia. Aquí los tiempos de preparación dependen de la secuencia y dependen de los recursos. Los tiempos de preparación dependientes de la secuencia son una característica definitoria de la secuenciación en talleres como el de la industria electrónica y mecánica. En estos se considera como un todo, al conjunto de operaciones de montaje y desmontaje puesto que no admiten fácilmente una separación. También existen tiempos de cambios cuyo valor no sólo depende de la secuencia y de las máquinas sino además de otros recursos auxiliares como la mano de obra involucrada (tipo y cantidad), utillaje o sistemas de transporte (tipo, velocidad, etc.).

Taller de Flujo con una Familia con Tiempos de Cambio de partida independientes/dependientes de la secuencia (TF F TCP IS, TF F TCP)

Los Tiempos de Cambio de partida para este tipo de talleres se asocian todos a trabajos individuales. En casos prácticos todos los trabajos que pertenecen a una misma familia necesitan de un único Tiempo de Cambio de partida cuando son programados continuamente. Este se presenta en las dos situaciones siguientes: i) cuando los trabajos pertenecen a varias familias y los Tiempos de Cambio de partida se presentan solo cuando una máquina cambia del procesamiento un trabajo de una familia al procesamiento de un trabajo de otra familia; ii) cuando los trabajos contiene varias unidades idénticas y se separa en múltiples sublotes con el Tiempo de Cambio de operación se presenta solo

cuando una máquina cambia el procesamiento del sublote de un trabajo al sublote de otro trabajo.

Taller de Flujo con un Grupo con Tiempos de Cambio de partida Independientes o dependientes de la secuencia (TF GTCPIS, TF GTCP)

Cuando sólo hay un lote en cada familia, es decir, la familia no se subdivide en sublotes, el problema de programación se dice que satisface las características de un grupo tecnológico, asumiendo que los trabajos de una misma familia se procesan posteriormente. Si el sublote consiste de múltiples trabajos, los *tiempos de cambio de partida independientes de la secuencia* no se añaden al tiempo de procesamiento de cada uno de estos trabajos, ya que el primer trabajo en la secuencia de un sublote no se conoce hasta haber resuelto el problema de programación. Por otra parte los tiempos de cambio de partida dependientes de la secuencia son separados y se agregan cada vez que se procesa un nuevo sublote si la máquina así lo requiere.

Taller de Flujo con una Serie con tiempos de cambio de partida independientes o dependientes de la secuencia (TF STCPIS, TF STCP).

Este tipo de taller se presenta cuando es necesario que las familias de trabajos a procesar se subdividan en una serie de sublotes interrelacionados para optimizar el tamaño de cada sublote. Para realizarlo, se requiere resolver dos problemas interrelacionados (definir los sublotes y la programación), este es difícil de resolver ya que sólo para definir los grupos de sublotes se resuelve un problema de programación (Monm et al, 1989). Los tiempos de cambio de partida independientes de la secuencia no se añaden al tiempo de procesamiento de cada uno de estos trabajos, ya que el primer trabajo en la secuencia de un sublote aun no se conoce solo hasta haber resuelto el problema de programación, así que estos se descartan. Por otra parte los tiempos de cambio de partida dependientes de la secuencia son separados y se agregan cada vez que se procesa un nuevo sublote si la máquina así lo requiere.

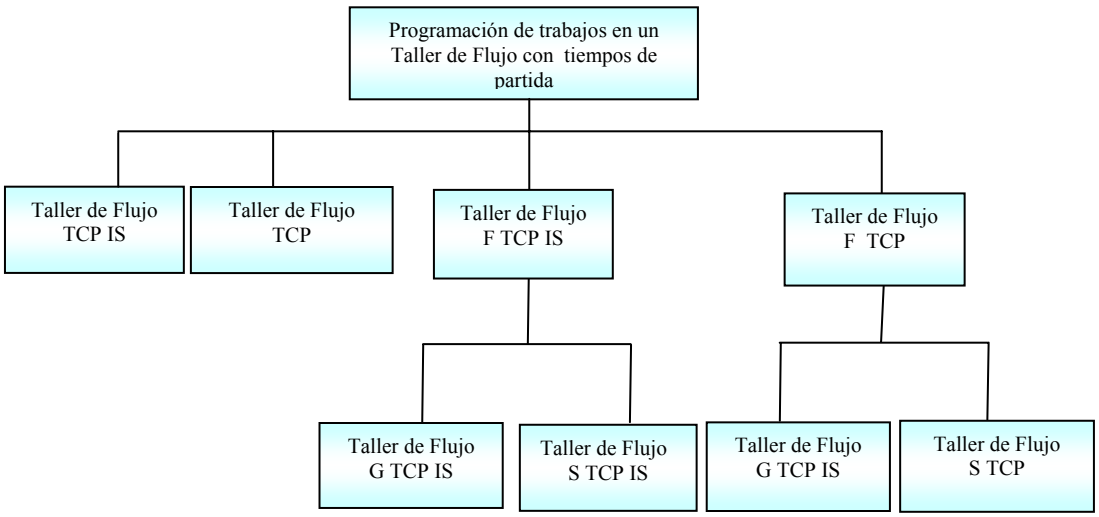


Figura 2.4 Clasificación de Problemas de Programación de trabajos con tiempos de partida

El tiempo de cambio de partida dependiente de la secuencia, denotado como s_{jk}^{i}, es el tiempo que se necesita para cambiar del trabajo *j* al trabajo k en cualquier máquina de la etapa i.

En la presente investigación se enfocará la atención en el problema de Taller de Flujo Híbrido con Tiempos de Cambios de partida dependiente de la secuencia (TFH TCP).

2.2 Complejidad computacional

Muchos problemas concretos, son resueltos por una serie de algoritmos, con los cuales los investigadores tratan de mejorar la(s) solución(es) del problema, sin embargo, en los casos de Talleres de Flujo Híbrido hallar una mejor solución implica mayor complejidad del algoritmo. Se establece que *el orden de complejidad de un problema*, es el *orden de complejidad del algoritmo que lo resuelve de mejor manera*.

En consecuencia para saber la complejidad de un algoritmo, se hace la medición del número de operaciones que se requiere para obtener una solución.

Si el código programado del algoritmo esta dividido por N módulos, donde cada uno de ellos tiene el número $f(n)$ de operaciones que se llevan al cabo, todas las funciones asociadas a los módulos forman un conjunto denotado por $\{f_1(n), f_2(n),\ldots, f_N(n)\}$. Si en

este conjunto alguna de las funciones $f_i(n)$ supera a las restantes en un intervalo dado, se dice que el conjunto esta acotado superiormente; y $O(f_i(n))$ es una cota superior para el conjunto de todas las funciones. En consecuencia, la complejidad computacional de un algoritmo reside en encontrar dicha cota superior. A su vez, los algoritmos según las cotas superiores se distinguen en:

- los que se resuelven en un tiempo polinomial $O(n^b)$,
- los que se resuelven en un tiempo exponencial $O(b^n)$,
- los que se resuelven en un tiempo factorial $O(n!)$.

Estudios realizados han permitido constatar que existen problemas muy difíciles, problemas que desafían al diseñador de algoritmos. Las clases mediante las cuales se clasifica la complejidad de los problemas son:

- Clase P, $O(n^b)$, $b \in Q$, $b \geq 1$**:** Los problemas de complejidad polinomial son tratables en el sentido de que suelen ser abordables en la práctica. Los problemas para los que se conocen algoritmos con esta complejidad se dice que forman la clase *P*.
- Clase *NP***:** Son aquellos problemas para los se comprueba en tiempo polinómico que una solución efectivamente lo es. Los problemas de esta clase se denominan NP ya que provienen de la expresión *N* de no-deterministas y la *P* de polinomiales.
- Clase *NP-completos***:** Gráficamente podemos decir que son los problemas se hayan en la "frontera externa", ya que se reducen a la clase *NP*.
- Clase *NP-duros***:** Son aquellos problemas para los cuales no se ha podido probar que se puedan reducir a *P*. Son los problemas más difíciles de la clase *NP* de los cuales destacan algunos de ellos de extrema complejidad.

En la Figura 2.5 se muestra una clasificación de algoritmos que resuelven problemas de distinta complejidad. A los problemas que se clasifican como de clase ***P,*** se les aplican algoritmos que proporcionan la solución exacta en un tiempo polinomial. En cambio a los problemas *NP-duros*, es decir, de elevada complejidad computacional, los algoritmos de solución a su vez se subdividen en enumerativos exactos (*seudo-polinomiales*), algoritmos

aproximados con la evaluación analítica de la solución y algoritmos heurísticos con análisis probabilístico o con estudios por simulación (Ecker, 2003).

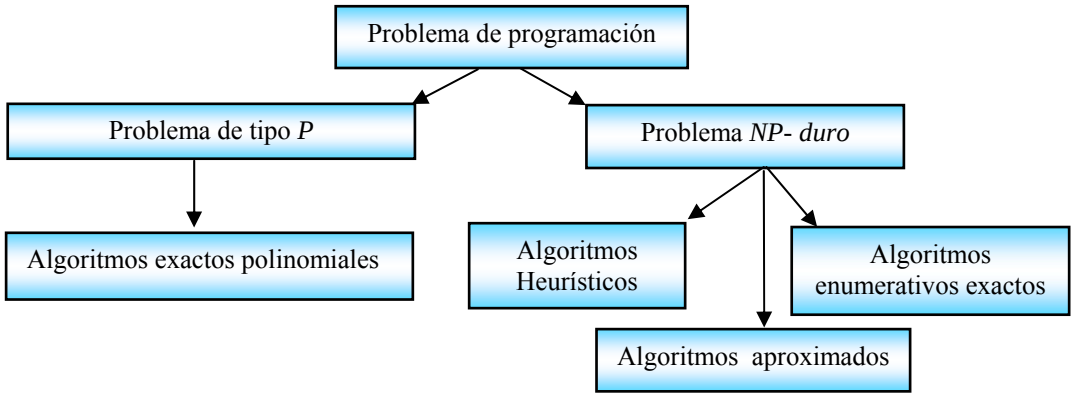


Figura 2.5 Clasificación de algoritmos que dan la solución a problemas de programación de distinta complejidad computacional

Otra medición aplicada a un algoritmo es la complejidad temporal, que se logra midiendo el tiempo de ejecución del código programado que se expresa mediante la función, denotada por $T(n)$, obtenida físicamente, es decir, ejecutando el programa, reloj en mano; o contando sobre el código las instrucciones a ejecutar y multiplicando por el tiempo requerido por cada instrucción. Es importante indicar que debido a que distintas computadoras necesitan tiempos diferentes para realizar las mismas operaciones básicas. Es decir, los ordenadores más rápidos realizan operaciones básicas con bits en 10^{-9} segundos, mientras que un ordenador personal requiere 10^{-6} segundos (un microsegundo), mil veces más, para realizar la misma operación. Además, que es bastante complicado descomponer todas las operaciones que desarrolla un algoritmo en las operaciones elementales que realiza un ordenador (Rosen, 2004). Por tanto es más recomendable utilizar la complejidad de un algoritmo mediante el número de pasos requeridos como se mencionó anteriormente, en lugar de utilizar el tiempo de cálculo real.

El tipo de análisis de complejidad se realiza con el peor caso y el caso promedio, los cuales indican el número de pasos a realizar por el algoritmo con una entrada de datos específica, el peor caso representa las peores condiciones de los datos del problema a resolver, mediante este se tendrá el número máximo de pasos que realizará el algoritmo

para resolverlo. En el caso promedio, se busca el número de pasos para entradas de datos de un tamaño determinado.

Resolver un problema de Taller de Flujo Híbrido implica diseñar un algoritmo que genere el ordenamiento de los trabajos a procesar con un tiempo de finalización mínimo que incluya hasta el procesamiento del último trabajo en la última etapa, es decir, encontrar la solución optima al problema $HFm|\,|C_{\max}$. El desarrollo de la teoría, sobre este problema, emerge con la generalización del Taller de Flujo con Multiples Procesadores (*flow shop multiple procesor's*, FSMP) en los años 70's con la investigación de Arthanari y Ramamurthy quienes resolvieron un TFF con dos etapas (Arthanari et al., 1971). Salvador, quien presenta un procedimiento de separación y evaluación basado en programación dinámica, para la optimización del $C_{\max}$ en un TFH de *k* etapas y un número diferente de máquinas por etapa (Salvador, 1973). Estas investigaciones fueron las primeras donde se definió un problema TFH, y se aplicó el método de ramificación y acotamiento (*branch and bound*) para atacar el problema. Este método es una técnica que garantiza la solución óptima. Sin embargo, sólo se aplica a tamaños pequeños del problema.

Sriskandarajah desarrolló un algoritmo heurístico simple, basado en la regla de Johnson, para el problema de THF de dos etapas e investigó el comportamiento del algoritmo en los casos peor y promedio $C_{\max}$ (Sriskandarajah et al., 1989).

Salvador y Hunsucker analizaron su complejidad clasificándolo como uno de los problemas más difíciles de la producción, debido al gran número de posibilidades para procesar los trabajos (Salvador et al, 1991).

Brah y Hunsucker presentan un algoritmo basado en el método de ramificación y acotamiento. En el artículo fue demostrado que este método se usa para optimizar otros criterios (Brah, et al.,1991).

Elmaghraby y Karnoub mencionan que un TFH consiste de una serie etapas de producción, cada uno de los cuales posee varios servicios en paralelo. Propusieron para resolver el problema una heurística que es una extensión del algoritmo de Johnson (Elmaghraby et al., 1995). Hoogeveen muestra que el caso de TFH con tardanza es

también NP-duro. A pesar de los resultados discutidos teóricamente, los modelos de TFH han capturado la atención de muchos investigadores (Hoogeveen et al, 1996)

Por otra parte, en la literatura se menciona que la aproximación mediante heurísticas es más rápida pero no garantiza una solución óptima. Gupta propuso una técnica para un TFH para minimizar $C_{\max}$ en un taller de dos etapas con una sola máquina en la segunda etapa (Gupta, 1998).

Riane, Artiba y Elmagharaby plantean un estudio de un TFH de tres etapas a partir de un caso en una industria de fabricación de tableros, con una máquina en las etapas uno y tres y dos máquinas en las etapas intermedias en las que la asignación de los trabajos se realiza con antelación. Se proponen dos algoritmos para su resolución. El primero se basa en aplicar la programación dinámica a una secuencia de los trabajos obtenidos. El segundo enfoque se basa en un algoritmo de ramificación y acotación. Para ello, definen cotas superiores del $C_{\max}$. El último procedimiento es mejor que el primero (Raine et al., 1998).

El TFH de dos etapas fue considerado por Gupta como NP-duro (Gupta, 1998) sus investigaciones son en ambientes de manufactura como los textiles y procesos industriales. Anteriormente las investigaciones publicadas se concentraron en la mayoría de los casos en problemas de dos etapas. Sólo pocas publicaciones consideran los modelos de tres etapas. Sin embargo, el problema de múltiples etapas es tratado en la actualidad con mayor frecuencia, como ejemplo esta el trabajo desarrollado por Portmann (1998) que generaron un algoritmo Genético que resuelve el problema múltiples etapas. Linn (1999) y Riane (1999) proporcionaron investigaciones sobre problemas de TFH con múltiples etapas con diferentes criterios a optimizar. M.Gourgand, N. Grangeon y S. Norre, demostraron en su investigación que el número de soluciones de un TFH asciende $n!\left(\prod_{i=1}^{m} m_i\right)^n$ (M. Gourgand et al., 1999). Botta-Genoulaz (2000) resuelve TFH con restricciones de precedencia y demoras en tiempo tomando como criterio la minimización de la máxima tardanza. Sivrikaya (2000) aplicó un algoritmo reconocido simulado para resolver el THF con múltiples etapas. Andrés (2001) aplicó la técnica de metaheurística para resolver un TFH de tres etapas con Tiempos de Cambio de Partida dependientes de la secuencia. Oğuz creó un algoritmo heurístico para resolver el problema TFH con dos etapas (Oğuz et al, 2003).

El problema de TFH con múltiples etapas es considerado en la literatura por diferentes autores como uno de los más difíciles (Ecker, 2003), es semejante al problema de la mochila o al de empaquetamiento binario, los cuales están clasificados como NP-completos, ya que hasta el momento no se han diseñado algoritmos que resuelvan el problema en un tiempo polinomial. Mas tarde Palmer, Alfaro, Andrés y Albarracín, se enfocan en el problema, definido por ellos como un THF de tres etapas, con máquinas idénticas y con tiempos de partida dependientes de la secuencia, para la asignación de un calendario para una empresa que produce azulejos (Palmer et al., 2003). Utilizaron reglas de prioridad y métodos metaheurísticos, como colonias de hormigas y simulado reconocido (también llamado: templado simulado, enfriamiento simulado, *simulate annealing*). Sivrikaya desarrolló un algoritmo genético para minimizar la máxima fecha de finalización en un TFH con múltiples etapas (Sivrikaya et al, 2004).

En el trabajo realizado por Ruíz (2004) llamado "Soluciones para complejos problemas de la producción", se analiza el problema $FHm\left(RM\left(i\right)_{i=1}^{(m)}\right)\left|Ssd, Mj\right|C_{\max}$. En ese mismo año J. Chang, W. Yan y H. Shao (2004) analizaron el problema de un TFH de dos etapas sin espera.

Chen Lu, Xi Li-feng, Cai Jian-guo, Bostel Nathalie, Dejax Pierre, analizaron un TFH con tres etapas, precendencia, tiempos de inicio de partida y bloqueo; su objetivo fue C_{max} ($HFS3\left|prec, s_{ij}, block\right|C_{\max}$). Usaron Búsqueda de Tabu (*tabu search)* para resolver el problema (Chen et al., 2006).

Jungwattanakit y otros proponen una técnica metaheurística para un problema de Taller de Flujo Flexible, donde al menos una de las etapas está compuesta por máquinas no relacionadas (Jungwattanakit et al., 2006).

En este mismo año, M. Zandieh, SMT Fatemi Ghomi y SM Moattar Hisseni diseñaron un algoritmo inmune, el cual simula el sistema de defensas de un organismo vivo, activando o disminuyendo la creación de anticuerpos. El algoritmo proporciona una solución en un tiempo aceptable. El problema tratado es un caso general de THF con Tiempos de Cambio de Partida que dependen de la secuencia (Zandieh et al., 2006). Yaurima hace la comparación de dos algoritmos genéticos para resolver un TFH TCP y

disponibilidad de las máquinas (Yaurima et al., 2008), en la Tabla 2.1 se presenta una lista de las investigaciones mencionadas anteriormente.

Tabla 2.1. Investigaciones realizadas en TFH *TCP* y los algoritmos que se usaron.

No. de etapas	Referencia	Año	Problema	Comentarios
2	Arthanary et al.	1971	$HFS2 \mid \mid C_{\max}$	Ramificación y acotamiento
3	Salvador	1973	$HFS3 \mid s_{ij} \mid C_{\max}$	Ramificación y acotamiento
2	Gupta	1988	$HFS2, m, 1 \mid \mid C_{\max}$	Heurística
2	Sriskandarajah et al.	1989	$FFS2 \mid \mid C_{\max}$	Heurística
3	Brah and Hunsucker	1991	$FFS3 \mid \mid C_{\max}$	Ramificación y acotamiento
2	Elmaghraby y Karnoub	1995	$HFS2 \mid \mid C_{\max}$	Heurística
3	Raine et al.,	1998	$HFS, P1^{(1)}, P2^{(2)}, P1^{(3)} \mid \mid C_{\max}$	Programación dinámica, Ramificación y acotamiento
	Wendong, Xiao et all.	2000	HFS	Genético
3	Andrés	2001	$HFS3 \left(P1^{(1)}, P2^{(2)}, P1^{(3)}\right) \mid S_{sd} \mid C_{\max}$	Heurística y Meta-heurística
3	Andrés	2003	$HFS3 \mid S_{ij} \mid C_{\max}$	Meta-heurística
3	Palmer et al.	2003	$HFS3 \mid S_{ij} \mid C_{\max}$	Meta-heurística reconocido simulado y colonia de hormigas
m	Ruíz et al.	2004	$HFSm \left(RM(i)_{i=1}^{(m)}\right) \mid Ssd, Mj \mid C_{\max}$	Búsqueda Tabú
m	Oğuz, Ceyda et all	2005	HFS y MPT**	Genético
	Belkadi K et all	2006	$FH2(P3, P2) \mid \mid C_{\max}$	Genético
3	Chen et al.	2006	$HFS3 \mid prec, s_{ij}, block \mid C_{\max}$	Búsqueda Tabú
m	Chen et all.	2006	$HF, unrelated \mid prec, s_{ik}, block \mid C_{\max}$	Lagrangeano relajado
m	Jungwattanakit J. et al.	2006	$HFSm \mid \mid C_{\max}$	Meta-heurística
m	Ruíz et all	2006	$FHm, \left((RM_{(i)})_{i=1}^{m}\right) \mid S_{sd}, M_j \mid C_{\max}$	Genético
m	Zandieh et al.	2006	$HFSm \mid s_{ij} \mid C_{\max}$	Algoritmo Inmune
m	Jungwattanakita J. et al.	2007	$HFSm \mid s_{ij} \mid C_{\max}$	Programación mezclada entera
	Vazquiez et all.	2007	$FH(P2^{(1)}, P2^{(2)}, Q7^{(3)}, Q4^{(4)}, P2^{(5)}) \mid s_{sd}^{(1)}, s_{sd}^{(4)}, s_{sd}^{(5)} \mid C_{\max}$	Búsqueda Tabú
	Yaurima, Victor et all.	2007	$FHm, \left((RM^{(i)})_{i=1}^{(m)}\right) \mid S_{sd}, M_j \mid C_{\max}$	Genético

Analizando el desarrollo teórico de los problemas de THF se observa que en los años 70's surgió al mismo tiempo el concepto de Tiempos de Cambio de Partida como se

muestra en la Tabla 2.1, los cuales intervienen en la manera de procesar los trabajos en los TF y afecta al TFH aumentando su complejidad.

Los problemas de programación con TCP están clasificados como los problemas más difíciles en programación. El problema de programación en un taller de una máquina con TCP es equivalente a un problema del vendedor viajero (*Traveling Salesman Problem*, TSP) que es NP duro (Pinedo, 1995), en consecuencia un TFH TCP y múltiples etapas es de alta dificultad.

Debido a su complejidad, los progresos reportados sobre la resolución de problemas de programación de trabajos con los TCP dependientes de la secuencia, todavía están lejos de ser completos (Luh et al., 1998). A través de los años los algoritmos que resuelven al TFH TCP se han clasificado según su complejidad como *NP*-duros, Gupta y Darrow muestran que la programación de las permutaciones (cuando la secuencia de todos los trabajos en todos las etapas es idéntica) no siempre es optima en caso de tener dos máquinas, en consecuencia, si las máquinas son diferentes y hay un número mayor de etapas, encontrar la permutación optima es más complicado (Gupta et al, 1986).

La configuración en TFH aparece muy frecuentemente en la realidad industrial asociada a los tiempos de cambio de partida. En los procesos donde existen tiempos de cambio de partida apreciables, es frecuente observar una multiplicidad de los recursos necesarios para hacer cada operación, debido a que con esto es posible ir procesando trabajos mientras que en otros recursos se esta realizando el cambio y por lo tanto no hay interrupciones en la fabricación. Esto, evidentemente exige una inversión adicional, y debe de analizarse a la hora de diseñar el sistema para evitar ponderar los costes de instalar varios recursos o máquinas equivalentes por operación, frente a los beneficios en la reducción del tiempo de fabricación. Por tanto crear algoritmos que generen las permutaciones para resolver los problemas de TFH tomando en cuenta los TCP es muy importante.

2.3 Métodos para la resolución del problema de TFH TCP

Anteriormente se describieron las notaciones formales del TFH así como algunas de sus aplicaciones. Este tema enfoca su atención en algunos de los diversos métodos utilizados por los investigadores para resolver a los TFH. Así tenemos que los más usados para el problema de TFHTCP son la programación dinámica, ramificación y acotamiento, búsqueda de tabú, heurísticas, meta heurísticas como genéticos y algoritmos inmunes. En la Tabla 2.2 se describen brevemente las características principales de los algoritmos antes mencionados. Posteriormente se describe a detalle algunos de estos métodos y sus algoritmos, así como el utilizado en la aplicación de esta investigación.

Tabla 2.2. Métodos aplicados para resolver problemas de Talleres de Flujo Híbrido con tiempos de cambio de partida que dependen de la secuencia.

Método		Descripción
Programación dinámica		Esta técnica consiste en separar un problema en subproblemas; su eficiencia consiste en resolver los subproblemas una sola vez, guardando sus soluciones en una tabla para su futura utilización.
Ramificación y acotamiento *Branch & Bound*		Realiza una enumeración parcial del espacio de soluciones basándose en la generación de un árbol de expansión. Una característica que le hace diferente a este diseño es la posibilidad de generar nodos siguiendo distintas estrategias
Búsqueda de Tabú (BT) *Tabu Search*		La búsqueda tabú es un procedimiento heurístico utilizado para resolver problemas de optimización de gran escala
Heurístico		Es utilizado para resolver problemas NP-duro, es decir de tamaño grande mediante algoritmos exponenciales y proporciona soluciones cercanas a las óptimas dentro de un tiempo polinomial.
Evolutivos	Genético	Los algoritmos genéticos son métodos de búsqueda estocástica para optimizar problemas basados en los mecanismos de selección natural, se llaman genéticos porque usan el concepto de la supervivencia de la teoría de evolución de Darwing.
	Inmune	Imita el proceso de defensa del sistema inmune cuando lo invade un agente externo, este garantiza una rápida convergencia hacia el óptimo global.

2.3.1 Algoritmo de Johnson

El algoritmo de Johnson (1954), o llamado regla de Johnson, resuelve de manera óptima el problema de taller de flujo con dos máquinas denotado por $F2\|C_{\max}$, el cual ha sido reconocido como el origen de la investigación en programación. Este algoritmo se ha adaptado según como ha aumentado la complejidad de los diferentes talleres en producción. Las funciones para las cuales fue creado son:

- Minimizar los tiempos de procesamiento para la secuenciación de un grupo de trabajos a través de dos máquinas.
- Minimizar el total de tiempos muertos en las máquinas.
- Minimizar el tiempo de flujo del inicio del primer trabajo hasta la terminación del último trabajo.

Se requieren las siguientes restricciones para aplicar el método:

- El tiempo de cada trabajo (incluyendo tiempo de cambio de partida y tiempo de procesamiento) deben conocerse y ser constantes para cada trabajo en cada máquina.
- Los tiempos de los trabajos deben de ser independientes de la secuencia de los trabajos.
- La prioridad de los trabajos no existe.

A continuación se presenta el algoritmo clásico y una de sus modificaciones, creada para resolver THF TCP.

Regla de Johnson para programación de *n* trabajos en dos máquinas:

1. Se enlistan los trabajos, y el tiempo requerido para procesarse en cada una de las máquinas.
2. Seleccionar el trabajo con el tiempo de actividad más corto. Si el tiempo más corto coincide con la primera máquina, el trabajo es asignado primero; si con la segunda máquina, el trabajo es asignado al último.
3. Si un trabajo es asignado, eliminarlo de la lista.
4. Aplicar los pasos 2-3 para los trabajos restantes, hacia el centro de la secuencia.

2.3.2 Algoritmo de Johnson m/2, m/2

Sean m el número de etapas y n el número de trabajos que deben ser procesados en cada etapa sucesivamente, y sea m_i el número de máquinas en cada etapa i, $i = \overline{1,m}$. Se asume que las máquinas necesitan un tiempo para preparación antes de procesar el trabajo *j*,

$j = \overline{1,n}$, en cada etapa, llamado el cambio de partida. El trabajo $n+1$ existirá en cada etapa sólo para indicar el fin del proceso, si se necesita. Se tienen las siguientes notaciones:

Sean,

$P = \begin{pmatrix} p_1^1 & p_1^2 & \cdots & p_1^m \\ p_2^1 & p_2^2 & \cdots & p_m^1 \\ \vdots & \vdots & \cdots & \vdots \\ p_n^1 & p_n^2 & \cdots & p_n^m \end{pmatrix}$, la matriz del tiempo de procesado p_j^i del trabajo *j* en la etapa *i*;

$S = \begin{pmatrix} 0 & s_{12}^1 & s_{13}^1 \cdots & s_{1n}^1 \\ s_{21}^2 & 0 & s_{23}^2 \cdots & s_{2n}^2 \\ s_{31}^3 & s_{32}^3 & 0 \cdots & s_{3n}^3 \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1}^m & s_{n2}^m & s_{n3}^m \cdots & 0 \end{pmatrix}$, la matriz del *tiempo de cambio de partida* s_{kj}^i del trabajo *k* al trabajo *j* en la etapa *i*;

$\widetilde{P} = \begin{pmatrix} p_1^1 & p_1^2 + \min_k s_{1k}^2 & \cdots & p_1^m + \min_k s_{1k}^m \\ p_2^1 + \min_k s_{2k}^1 & p_2^2 & \cdots & p_2^m + \min_k s_{2k}^m \\ \vdots & \vdots & \ddots & \vdots \\ p_n^1 + \min_k s_{nk}^m & p_n^2 + \min_k s_{nk}^2 & \cdots & p_n^m \end{pmatrix}$, la matriz del tiempo de modificación para el trabajo *j* en la etapa *i* ($\widetilde{p}_j^m = p_j^m + \min_k s_{jk}^m$);

S_i el conjunto de trabajos en cada etapa *i*.

El tiempo inicial para procesamiento del trabajo *j* se define en cero. El tiempo de cambio de partida de dicho trabajo *j* indica el tiempo para moverlo hacia la siguiente etapa. Se asume que todos los trabajos son completados en cada etapa antes de iniciar su procesamiento en la siguiente etapa.

Asumiendo que el tiempo de terminación del trabajo *j* en cada etapa debe efectuarse lo más pronto posible, antes del tiempo de cambio de partida dado para la siguiente etapa. El

tiempo de cambio de partida para el trabajo j_n es cero en cada etapa y solo existe para indicar el fin del proceso, si es que se necesita. Se incluye la restricción que cada etapa se procesa al menos un número de trabajos igual al número de máquinas que haya en dicha etapa.

Para mejorar la eficiencia de su algoritmo Johnson aplica el concepto antiguo de la dicotomía que proviene de la paradoja Zenón (490 a.C.- 430 a.C.), que desde un punto de vista lógico se transforma en un método. Zenón propone un problema que matemáticamente se describe de la siguiente manera. Asúmase que Aquiles y una Tortuga necesitan desplazarse del punto 0 al 48 de la recta. Cada uno recorre como primer paso la mitad de la distancia, es decir, 24 unidades y el siguiente paso será la mitad de la distancia que falta por recorrer, y así sucesivamente para el siguiente paso. Además, Aquiles permite que la tortuga avance con el primer paso de 24 unidades. ¿Quien llegará primero a la meta?

Para responder a esta pregunta se aplican uno de dos procedimientos:

- El primer procedimiento es finito, sólo se toman en cuenta a los números naturales, lo cual implica que la tortuga llegará primero.
- El segundo procedimiento es infinito, moviéndose ½ unidad = $\frac{48}{2}$, después ¼ unidad = $\frac{48}{4}$, y así sucesivamente, asumiendo la existencia de los números racionales, entonces Aquiles jamás alcanzara a la tortuga, más aun, ninguno logrará el objetivo de llegar al valor de 48. Sólo se tendría una distancia lo suficientemente cercana al final. Este hecho está expresado en la igualdad,

$$48 = 24 + \frac{24}{2} + \frac{24}{4} + \frac{24}{8} + \frac{24}{16} + \frac{24}{32} + \ldots + \frac{24}{2^n} + \ldots, n \in \mathrm{N}$$

la cual Zenón consideraba paradójica.

El asumía a priori que no existía el “infinito actual”, y en consecuencia ningún proceso infinito podría considerarse completo [1]. Es decir, en los tiempos de Zenón la razón humana no aceptaba de una manera natural el concepto de convergencia, el cual es en nuestros tiempos uno de los más importantes para la búsqueda de soluciones que no se determinan mediante un procedimiento analítico (Leyva, 2007).

Johnson aplica este concepto en la búsqueda de mejorar la permutación que genera su algoritmo básico para el cálculo de $C_{\max}$ mínimo, y define el valor $\tilde{p}_j^1$ como la suma de los tiempos de procesamiento modificado de la etapa 1 a la etapa $\left[\frac{m}{2}\right]$ y $\tilde{p}_j^m$ es la suma en las etapas $\left[\frac{m}{2}\right]+1$ a m, el realiza una separación de dos mitades de etapas para asignar los trabajos siguiendo los criterios de asignación de trabajos con *tiempos de cambio de partida* mínimos.

Algoritmo Johnson ($\frac{m}{2},\frac{m}{2}$):

1. Crear el tiempo de proceso modificado $\tilde{p}_i^1$ y $\tilde{p}_i^m$.
2. Sea $U=\left\{j\middle|\tilde{p}_j^1<\tilde{p}_j^m\right\}$ y $V=\left\{j\middle|\tilde{p}_j^1\geq\tilde{p}_j^m\right\}$.
3. Colocando los trabajos en U en orden no decreciente de $\tilde{p}_j^1$ y colocando los trabajos en V en orden no decreciente de $\tilde{p}_j^i$. Añadir el orden de la lista V al final de U.
4. En cada estado $i=1,2,\ldots,m$, inicializar los tiempos de procesamiento en cada máquina en ese estado con valor 0.
5. Para [j] =1,…, n, $j\in S^1$
    a. Para $mc=1\ldots m_1$:

    Desplaza el trabajo [j] al último en la máquina *mc*.

    Si este desplazamiento resulta en el tiempo de terminación mínimo para el trabajo [j], sea $m=mc$

    b. Desplaza el trabajo [j] al último en la máquina m.
6. Para cada etapa $i=2,\ldots,m$:
    a. Actualizar los tiempos de inicio en la etapa i para la terminación de los tiempos en la etapa $i-1$.
    b. Arreglara los trabajos en orden creciente para tiempos de inicio.
    c. Para $[j]=1\ldots n, i\in S_i$

i. Para $mc = 1$ hasta m_i:

Desplaza el trabajo [j] al último de la máquina mc.

Si este desplazamiento resulta en el menor tiempo de terminación para el trabajo [j], sea $m = mc$.

ii. Desplaza el trabajo [j] al último en la máquina m.

Si el algoritmo de Johnson $(\frac{m}{2}, \frac{m}{2})$ no genera la permutación que proporciona un tiempo mínimo de procesamiento, el valor de $C_{\max}$ de su permutación correspondiente se utiliza como un criterio, es decir, como comparativo para mejorar los resultados de otro tipo de algoritmos (Zandieh et al., 2006).

2.3.3 Algoritmos Evolutivos

Los algoritmos evolutivos son técnicas de búsqueda basadas en los principios de evolución natural de las especies. Estas aparecieron a fines de los años 50s (Fraser, 1957). Los algoritmos desarrollados por Holland (1962) y Fogel (1966), son algunos ejemplos que dieron pauta para la publicación del libro "*Genetic Algorithms in Search, Optimization and Machina Learning*", publicado por D.E. Goldberg (1989). Los algoritmos evolutivos tratan varios puntos de un espacio de búsqueda concurrentemente permitiendo que no sean atrapados en un óptimo local. Además, se ajustan para resolver problemas de optimización combinatorios eficientemente (Roger, 1999) (Holland, 1990).

Los algoritmos evolutivos requieren que se les especifique la codificación de los individuos y una función de evaluación (función objetivo) que mida la aptitud de cada individuo. La función de Aptitud es la guía que emplean estos algoritmos para explorar de forma eficiente su amplio espacio de búsqueda de posibles soluciones.

Un algoritmo evolutivo clásico tiene la estructura mostrada en la Figura 2.6 en la cual se observa 5 componentes característicos de un algoritmo evolutivo:

1. Representación genética de soluciones del problema.
2. Una manera para crear la población inicial.

3. Una función de evaluación que pone en juego las reglas de la evolución, proporcionando soluciones en términos de la aptitud.
4. Operadores genéticos que efectúan la creación de la descendencia durante la reproducción.
5. Valores para los parámetros que el algoritmo requiere: tamaño de la población, probabilidades para aplicar los operadores genéticos como cruzamiento, mutación y selección.

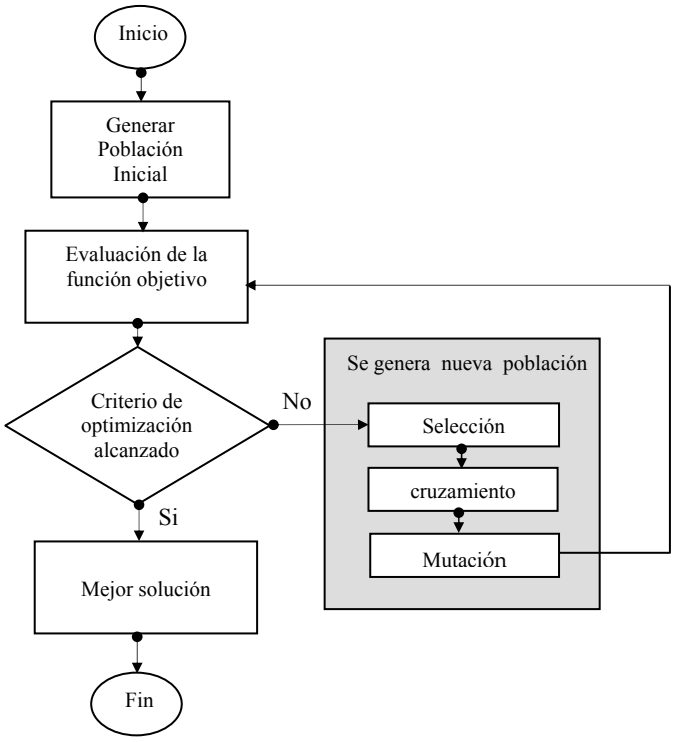


Figura 2.6 Estructura del algoritmo evolutivo

De esta manera, es posible hacer una analogía, en la cual el algoritmo representa el proceso evolutivo de los individuos y la búsqueda local se asocia a elementos culturales a los cuales cada individuo es sometido durante su vida. Es así como un cromosoma antes de pasar a la generación siguiente, toma una forma diferente cuando se le incorpora información aprendida durante su existencia y tiene la posibilidad de traspasar a sus descendientes la información por la vía genética. Esta herramienta ha sido utilizada con

éxito en diferentes tipos de problemas preferentemente del tipo combinatorio, (Jaszkiewicz, 2004). En particular, aplicaciones en el ámbito multiobjetivo combinatorio es posible destacar a Ishibuchi y Kaige con el problema de la mochila (Ishibuchi et al, 2004), cubrimiento de conjuntos (Jaszkiewicz, 2004), vendedor viajero (Buriol et al, 2004), problema de asignación cuadrática (Drezner, 2003). Un vasto número de problemas son tratados con estos algoritmos: Edificación de almacenes en localidades distintas, Programación de Eventos, Distribución de Recursos o de Materiales a las máquinas en talleres de flujo, etc.

Los algoritmos evolutivos, generan resultados satisfactorios, sin embargo, cada problema a resolver requiere configuraciones diferentes, implicando diseñar y programar procedimientos de respuesta para cada problema, con tareas que consumen gran cantidad de tiempo. Entre los algoritmos evolutivos se encuentran los Algoritmos Genéticos (AG) que son una técnica de solución común en problemas que surgen en las operaciones de manufactura. Estos se usaron para determinar la estructura del modelo de sistemas lineales y no-lineales que mejor representen sistemas de entrada y salida de datos (Ahmad, et al., 2004).

Los AG son los más usados, ya que son más flexibles que otras técnicas. Estos se definen como técnicas evolutivas, que mejoran su comportamiento de acuerdo con la experiencia que adquiere a medida que van haciendo la misma tarea en diversas oportunidades, simulando el comportamiento evolutivo de la naturaleza, que hace que los individuos evolucionen de acuerdo con las condiciones que se les plantean en su medio ambiente mediante el uso de técnicas como herencia, mutación, selección natural y cruzamiento (recombinación).

Su aplicación ha servido para encontrar buenas soluciones a problemas cotidianos de búsqueda y optimización en ingeniería, partiendo de una representación abstracta de posibles soluciones, llamada cromosoma, que va evolucionando desde una población inicial aleatoria a la que se le cuantifica su desempeño y se va modificando para crear una nueva población, repitiendo este procedimiento generación tras generación hasta llegar a una adecuada combinación de genes que permiten obtener un buen desempeño.

Los AG funcionan mediante un cromosoma, que es un individuo que representa una solución particular al problema que está siendo resuelto, que no tiene una estructura específica, sino que se adapta a las condiciones de cada problema y al método de análisis que se está utilizando, por lo tanto hay una gran variedad de representaciones implementadas para la solución de un solo problema (Castro et al, 1992).

Otro tipo de algoritmo evolutivo es el Inmune, a diferencia de los AG, son probabilísticos, garantizan una rápida convergencia hacia el óptimo global. Los algoritmos inmunes enriquecen el estado del arte con una rutina fina que se encarga de una diversidad del sistema inmune real. Se ajusta así mismo, mediante la respuesta inmune, pudiendo producir o eliminar anticuerpos, (Zandieh et al., 2006).

Jones, Mirrazavi y Tamiz mencionan que el 70% de los artículos se utilizan algoritmos genéticos, el 24% reconocido simulado, y sólo el 6% búsqueda tabú (Jones et al., 2000).

2.4 Conclusiones del capítulo

Las investigaciones en Teoría de Programación durante los últimos 50 años observan una separación entre teoría y práctica (Ruíz et al, 2006). Esto se debe ha que los problemas estudiados son producidos por distintas situaciones reales. Así que la generalización de la teoría, de los métodos, de las técnicas y los procedimientos a seguir para resolver los problemas de TFH aun no se consolidan. Como este es un tema muy basto en conceptos, métodos, técnicas, algoritmos y procedimientos, hay que iniciar por enfocar los esfuerzos en un solo tema e ir paso a paso, hasta consolidar toda la teoría.

Los antecedentes muestran que las primeras investigaciones utilizaron la técnica de ramificación y acotamiento para resolver los problemas de TF, sin embargo, las actuales y las nuevas se dirigen a la aplicación de algoritmos evolutivos especialmente los genéticos e inmunes.

No obstante los esfuerzos por generar nuevos algoritmos, se observa la existencia de un vacío entre las decisiones con las cuales se aplican los algoritmos, los tipos de distribuciones de los talleres y la manera en como se ataca su resolución o el análisis de los resultados. Es importante descartar o asumir si esto tiene relación directa con los

componentes de cada algoritmo y/o la forma en como se analizan los resultados actualmente.

Así mismo, no se observa en la literatura estudios respecto a la contribución de cada componente de los algoritmos de forma aislada sobre la obtención de las soluciones, por ejemplo, sería deseable analizar como contribuyen los componentes de los algoritmos genéticos a mejorar la solución, es decir, analizar por separado la función objetivo, los cruzamientos, las mutaciones y la selección, los cuales forman la estructura básica de un AG, en el siguiente capítulo se describen sus componentes.

CAPITULO 3

APLICACIÓN DE ALGORITMOS EVOLUTIVOS PARA RESOLUCIÓN DEL PROBLEMA

3.1 Parámetros de algoritmos

Como se mencionó anteriormente los AG pertenecen a la familia de algoritmos evolutivos. Se consideran como algoritmos de búsqueda estocástica de soluciones que imitan en su funcionamiento el principio de evolución y selección natural de las especies de Darwin. Fueron propuestos por Holland (1975) y han sido aplicados en diferentes campos de estudio. El funcionamiento de estos algoritmos describe de forma general la evolución de una *población* de individuos que representa distintas soluciones de un problema a resolver. La aptitud (*fitness*) de cada individuo es el grado de efectividad de la solución, es decir, que tan buena es esa solución. El algoritmo imita el proceso de selección natural, los mejores individuos se cruzan más veces que los peores, creando una nueva generación de individuos previsiblemente mejores. Sin embargo, los nuevos individuos sufren mutaciones que alteran las propiedades heredadas de los padres. Cada nueva generación se mejora, es decir, se obtienen mejores soluciones para el problema, conforme evolucione la población Figura 3.1.

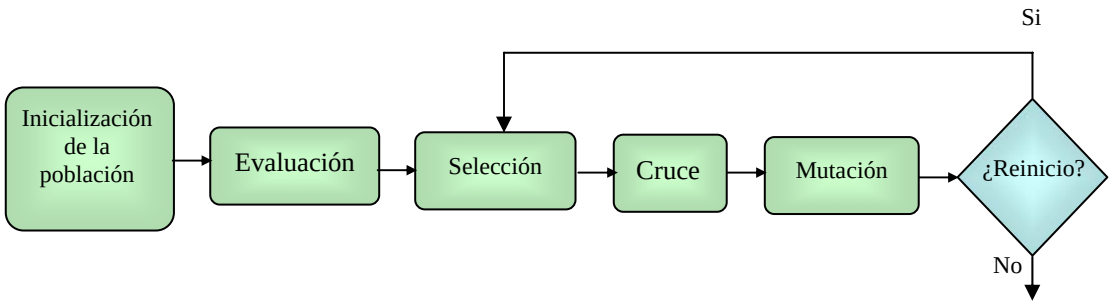


Figura 3.1 Esquema general del funcionamiento de los algoritmos genéticos

En los AG el cruce de genes es semejante a la reproducción de los seres vivos. Una vez seleccionados, dos individuos buenos de la población se emparejan y se reducen de acuerdo a una probabilidad P_c, la cual es un parámetro de control de los algoritmos genéticos. Si la pareja de padres no se cruza, se aplica un procedimiento para generar nuevos individuos. De manera general, el cruce combina los individuos progenitores para generar nuevos individuos llamados descendientes. El objetivo del cruzamiento es crear nuevas soluciones del problema, de manera que tenga las mejores características de los padres. Se han propuesto numerosos tipos de cruce, la evaluación de su capacidad para mejorar las soluciones requiere analizar individualmente su evolución.

No hay evidencia teórica o experimental que indique la existencia de un operador de cruzamiento universal que obtenga los mejores resultados para todo problema, siendo necesario determinar qué operador se ajusta a las características particulares de cada problema y de cada codificación para obtener buenos resultados.

El cruzamiento realiza el trabajo más delicado, así que seleccionar un buen método de cruzamiento es muy importante, en consecuencia analizar su evolución dentro del AG es muy importante.

Las notaciones y definiciones necesarias para los operadores de cruzamiento, asociadas a los problemas de TFH son:

- Representación genética (*genetic representation*): es la representación de una permutación del conjunto $J = \{j_1, j_2, \ldots, j_n\}$ en forma genética.
- Individuo (*individual*, *genotype*, *chomosome* o *string*): Es la representación codificada en serie de una solución del problema (la permutación).
- Gen (*feature,carácter o detector*): es una parte indivisible del individuo.
- Alelo (*allele*): son los valores posibles que toma cada individuo.
- Lugar (*locus*): es la posición que ocupa cada gen en el individuo.
- Población (*population*): es el conjunto de individuos, denotado por P.
- Fenotipo (*phenotype*): es el significado de cada individuo en términos del problema a resolver. Generalmente el fenotipo es el valor de la función objetivo asociado al individuo.

- Valor de adecuación (*fitness value*): es la medida de optimalidad de cada individuo dentro del ámbito del algoritmo genético. En algunos casos coincide con el fenotipo y en otros no.
- Generación: es la población en un instante concreto del proceso evolutivo, en una iteración dada del AG. Se denota por P_t a la población en la generación t.
- Descendencia (*offspring*): es la combinación de individuos progenitores para generar nuevos individuos.
- Mutación: es el proceso de introducir nueva información en la población o recuperar información perdida en el cruzamiento, mediante la variación de uno o más genes dentro de los individuos.
- Selección: es el proceso que se sigue para escoger los individuos con un mayor valor de adecuación, mediante los cuales se construye la siguiente generación.

En esta investigación, se analizaron los operadores de cruzamiento apropiados para TFH TCP como el factor de más impacto en la solución.

Los componentes restantes del AG se dejan fijos. Para la selección de padres, se aplicó el torneo binario (*binary tournament*) en el cual se procede a tomar de la población aleatoriamente dos individuos y se selecciona el primero con mayor aptitud.

Para evitar la convergencia a un óptimo local, se incorporó un operador de mutación, el cual permite introducir material genético perdido y variabilidad en la población. Se considera el operador de mutación por inserción (*insert)*, es uno de los más frecuentemente en la literatura (Michalewicz, 1996), (Gen, 1997), (Ruiz et al, 2006). El operador de mutación por inserción selecciona aleatoriamente dos puntos. Si punto1 < punto2, entonces el gen correspondiente a punto1 se coloca en punto2 y todos los genes desde punto1 hasta punto2 se recorren una posición hacia punto1. Si punto1 > punto2, entonces la operación de corrimiento se realiza hacia punto2 (Figura 3.2).

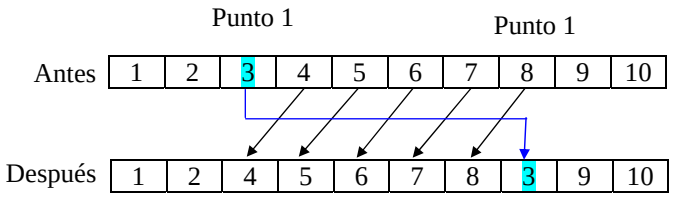


Figura 3.2. Ejemplo de mutación por inserción.

3.2 Operadores de cruzamiento

En esta investigación, se analizan los operadores de cruzamiento (OC) apropiados para TFH TCP como el factor de más impacto, dentro del AG, para la búsqueda de la solución. Enseguida se presentan los siete operadores de cruzamiento considerados en este trabajo de investigación, sus algoritmos así como ejemplos de su funcionamiento:

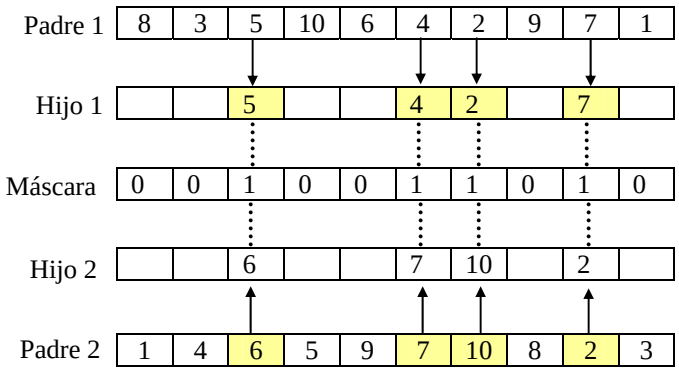
1. Orden Básico (*Order Based Crossover*) con Tiempos de Cambio de Partida dependientes de da secuencia (OBX_TCP).
2. Precedencia Conservativa (*Precedence Preservative Crossover*) con Tiempos de Cambio de Partida dependientes de da secuencia (PPX_TCP).
3. Un Segmento De Cruce (*One Segment Crossover*) con Tiempos de Cambio de Partida dependientes de da secuencia (OSX_TCP)
4. Cruzamiento Entre Dos Puntos (2 *Point Crossover*) con Tiempos de Cambio de Partida dependientes de da secuencia (2PX_TCP).
5. Cruzamiento Entre *K* Puntos De Corte (*K Point Crossover* con Tiempos de Cambio de Partida dependientes de da secuencia (KPX_TCP).
6. Cruzamiento Parcialmente Emparejado (*Partially Matched Crossover*) con Tiempos de Cambio de Partida dependientes de da secuencia (PMX_TCP).
7. Cruzamiento Por Orden (*Order Crossover*) con Tiempos de Cambio de Partida dependientes de da secuencia (OX_TCP).

3.2.1 OBX_TCP

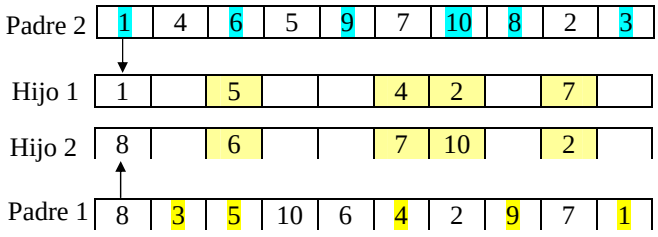
Método. Proviene del cruzamiento OBX considerando los TCP dependientes de la secuencia con una representación binaria que funciona con una máscara de bits de la misma longitud que el individuo de tamaño n. En cada posición de esta máscara se encuentran los valores 1 ó 0, donde 1 indica que esa posición se hereda del padre directo,

mientras que un valor 0 indica que esa posición se hereda del otro padre, es decir, se copia la secuencia de acuerdo con el mínimo valor de *tiempo de cambio de partida*. OBX usa múltiples puntos de corte que vienen dados por la máscara.

El cromosoma representa la posición en que se ejecutan los trabajos, por ejemplo en la Figura 3.3 (a) se muestran 2 cromosomas cada uno cuenta con 10 lugares que son las posición en que se ejecutan los trabajos, el valor de cada posición o alelo es el trabajo a procesar. Al crear la máscara, los lugares con valor 1 heredan los alelos del padre directo y los lugares con valor 0, quedan en espera de ser asignados Figura 3.3 (b).



(a) Se seleccionan del padre directo los alelos en las posiciones donde la máscara tiene el valor 1



(b) Se heredan los alelos en las posiciones donde la máscara tiene el valor 1

Figura 3.3. Funcionamiento del operador antes e aplicar TCP

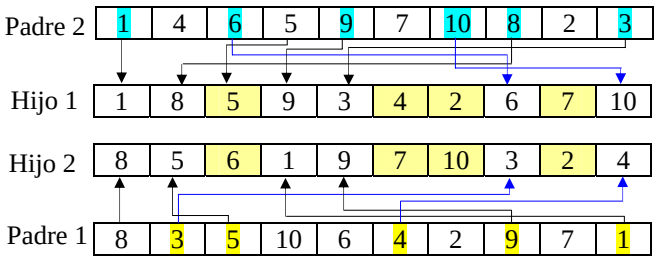
La Figura 3.4 (a) muestra los TCP para cada trabajo. Por ejemplo en el hijo 2 han quedado sin asignar los alelos {8, 3, 5, 4, 9, 1}, es el padre 1 quien heredará dichos alelos. El padre 1 tiene en la primera posición al alelo 8, este se asigna directamente al hijo 2, los alelos restantes se asignan si su TCP es el menor. Así después del trabajo 8 se considera un TCP para cada trabajo que falta por procesar, son los valores de la columna 8 de la

Figura3.4.(a). Se crea la secuencia de trabajo y TCP como {3(51), 5(34), 4(38), 9(38), 1(50)}, el trabajo 5 posee el menor TCP y se asigna al hijo 2; después de 5 sigue 6 pues forma parte fija del Hijo 2. Después de 6, se crea una secuencia semejante {3(49), 4(53), 9(48), 1(47)}, donde se asigna el trabajo 1; después de 1 según la secuencia {3(56), 4(45), 9(43)}, se asigna el trabajo 9; después le sigue 7 y 10; después de 10 según la secuencia {3(42), 4(52)} se asigna el trabajo 3, sigue 2 y por último 4.

Trabajo por procesar

Trabajo que ya fue procesado	1	2	3	4	5	6	7	8	9	10
1	0	33	54	38	32	47	38	50	49	35
2	36	0	53	57	31	54	37	48	52	33
3	56	32	0	52	43	49	54	51	38	47
4	45	56	33	0	30	53	42	38	47	52
5	43	59	51	31	0	41	52	34	44	58
6	41	35	49	53	45	0	33	52	38	37
7	37	31	48	55	46	42	0	36	46	54
8	34	43	52	42	34	55	50	0	53	45
9	43	44	55	37	41	48	45	38	0	32
10	38	48	39	31	55	33	36	43	51	0

(a) TCP



(e) asignación final considerando los TCP

Figura 3.4. Operador de cruzamiento OBX_TCP

Algoritmo OBX_TCP.

Entrada: dos padre A^t y B^t del banco de padres y matriz de tiempos de cambio de partida.

Salida: dos descendientes C^{t+1} y D^{t+1}

1. seleccionar dos padre A^t y B^t del banco de padres
2. crear máscara E^{t+1}
3. crear dos descendientes C^{t+1} y D^{t+1} como sigue:
4. para $i = 1$ a n

4.1 si $e_i^{t+1} = 1$

4.1.1 $c_i^{t+1} = a_i^t$

4.1.2 $d_i^{t+1} = b_i^t$

4.2 si $e_i^{t+1} = 0$

4.2.1 $c_i^{t+1} = \min\{sdst(b_i^t)\}$

4.2.2 $d_i^{t+1} = \min\{sdst(a_i^t)\}$

1. fin ciclo.

3.2.2 PPX_TCP

Método. Proviene del cruzamiento de PPX considerando los TCP. El operador realiza pasos semejantes a OBX, genera también una representación binara que funciona como una máscara de bits con la misma longitud que el individuo de tamaño n. En cada posición de esta máscara se encuentran los valores 1 ó 0 considerando los TCP. A diferencia de OBX, el valor 0 ahora indica los elementos copiados del padre 1 al Hijo 1 y el valor 1 indica los elementos copiados del padre 2 al hijo 1

Algoritmo PPX_TCP.

Entrada: dos padre A^t y B^t del banco de padres y matriz de de tiempos de cambio de partida.
Salida: dos descendientes C^{t+1} y D^{t+1}

1 seleccionar dos padre A^t y B^t del banco de padres

2 crear máscara E^{t+1}

3 crear dos descendientes C^{t+1} y D^{t+1} como sigue:

4 para $i = 1$ a n

4.1 si $e_i^{t+1} = 0$

4.1.1 $c_i^{t+1} = b_i^t$

4.1.2 $d_i^{t+1} = a_i^t$

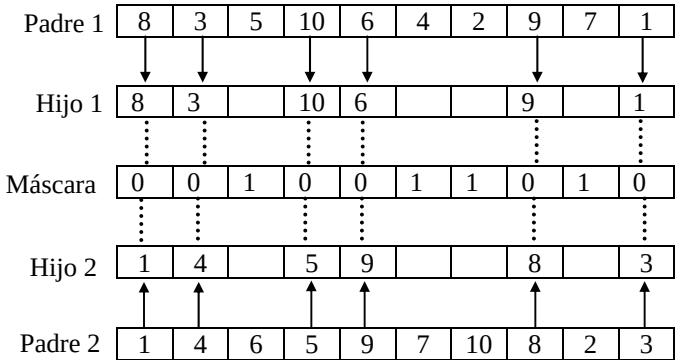
4.2 si $e_i^{t+1} = 1$

4.2.1 $c_i^{t+1} = \min\{sdst(a_i^t)\}$

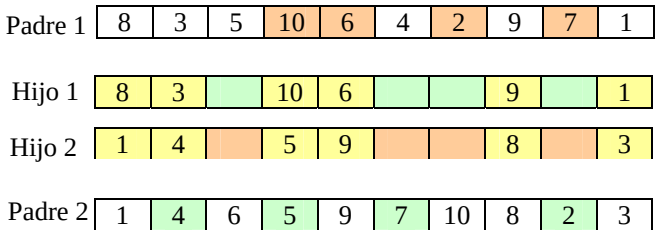
4.2.2 $d_i^{t+1} = \min\{sdst(b_i^t)\}$

5 fin ciclo.

En la Figura 3.5 (a) se muestra el procedimiento de este operador, el cual es muy semejante al anterior, la máscara hereda los valores de las posiciones con 0 del padre directo, mientas que los valores que poseen 1 esperan por ser asignadas según su valor mínimo de TCP Figura 3.5 (b). Por ejemplo el Hijo 1 hereda del padre 1 los alelos {8,3,10,6,9,1} faltan por heredar del padre 2 los alelos {4,5,7,3}.



(a) El padre directo hereda los alelos en las posiciones donde la máscara tiene el valor 0



(b) Cada hijo hereda los alelos faltantes del otro padre con el TCP de menor valor

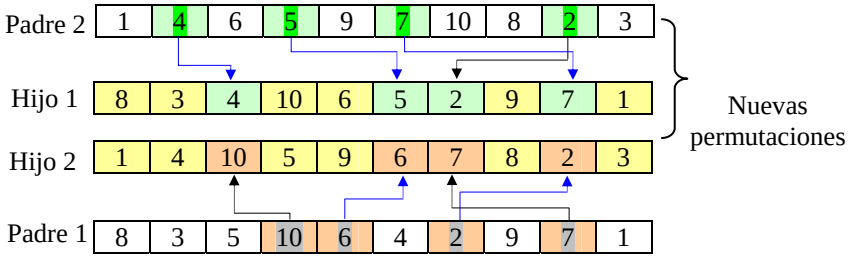
Figura 3.5. Funcionamiento antes operador antes de aplicar TCP

La Figura3.6 (a) muestra los TCP, el procedimiento para asignar los alelos faltantes es igual al descrito anteriormente descrito. Se crean dos nuevos cromosomas con características semejantes a las de sus padres Figura3.6 (b).

Trabajo por procesar

Trabajo que ya fue procesado	1	2	3	4	5	6	7	8	9	10
1	0	33	54	38	32	47	38	50	49	35
2	36	0	53	57	31	54	37	48	52	33
3	56	32	0	52	43	49	54	51	38	47
4	45	56	33	0	30	53	42	38	47	52
5	43	59	51	31	0	41	52	34	44	58
6	41	35	49	53	45	0	33	52	38	37
7	37	31	48	55	46	42	0	36	46	54
8	34	43	52	42	34	55	50	0	53	45
9	43	44	55	37	41	48	45	38	0	32
10	38	48	39	31	55	33	36	43	51	0

(a) TCP



(b) asignación final considerando los TCP

Figura 3.6. Operador de cruzamiento PPX_TCP

3.2.3 OSX_TCP

Método. A partir de dos padres (llamados padre 1 y padre 2), se establecen al azar dos puntos de corte aleatoriamente. Se generan los hijos (Hijo 1 e Hijo2), por ejemplo el Hijo 1 hereda los genes antes del punto de corte del Padre 1; después del primer punto de corte hereda los genes del Padre 2 hasta el segundo punto de corte y por último después del segundo punto de corte hereda los genes del Padre 1 hasta la última posición, considerando solo los elementos no copiados con el valor mínimo de TCP.

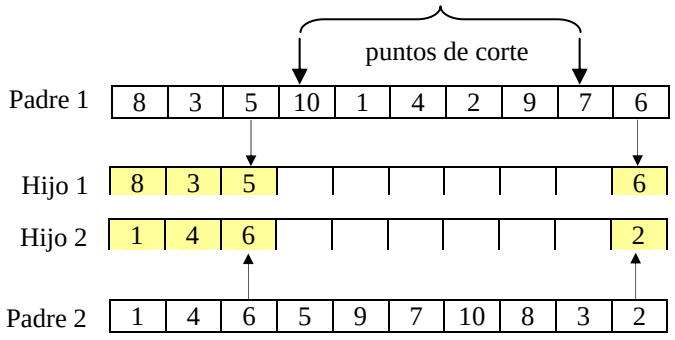
Algoritmo OSX_TCP

Entrada: dos padre A^t y B^t del banco (*pool*) de padres y matriz de de tiempos de cambio de partida.

Salida: dos descendientes C^{t+1} y D^{t+1}

1. seleccionar dos padre A^t y B^t del banco (*pool*) de padres
2. crear dos descendientes C^{t+1} y D^{t+1} como sigue:
3. elige aleatoriamente 2 puntos de cruce $cp_1, cp_2 \in \{1, \ldots, n-1\}$
   - 3.1 para $i = 1$ a $cp_1 - 1$ haz
     - 3.1.1. $c_i^{t+1} = a_i^t$
     - 3.1.2. $d_i^{t+1} = b_i^t$
   - 3.2 fin ciclo
   - 3.3 para $j = cp_2 + 1$ a n haz
     - 3.3.1 $c_i^{t+1} = \min\{sdst(a_i^t)\}$
     - 3.3.2 $d_i^{t+1} = \min\{sdst(b_i^t)\}$
   - 3.4 fin ciclo
   - 3.5 para $i = cp_1$ hasta cp_2 haz
     - 3.5.1 $c_i^{t+1} = \min\{sdst(b_i^t)\}$
     - 3.5.2 $d_i^{t+1} = \min\{sdst(a_i^t)\}$
   - 3.6 fin ciclo

La Figura 3.5 muestra como actúa OSX, este asigna los alelos antes y después del punto de corte del padre directo, así el hijo 1 hereda los alelos {8, 3, 5, 6}.



(a) El padre directo hereda los alelos en las posiciones donde la máscara tiene el valor 1.

Trabajo por procesar

Trabajo que ya fue procesado	1	2	3	4	5	6	7	8	9	10
1	0	33	54	38	32	47	38	50	49	35
2	36	0	53	57	31	54	37	48	52	33
3	56	32	0	52	43	49	54	51	38	47
4	45	56	33	0	30	53	42	38	47	52
5	43	59	51	31	0	41	52	34	44	58
6	41	35	49	53	45	0	33	52	38	37
7	37	31	48	55	46	42	0	36	46	54
8	34	43	52	42	34	55	50	0	53	45
9	43	44	55	37	41	48	45	38	0	32
10	38	48	39	31	55	33	36	43	51	0

(b) valor de los tiempos de cambio de partida dependientes de la secuencia (TCP).

Figura 3.7. Funcionamiento del operador antes de aplicar TCP

Faltan por asignar los alelos {1, 4, 9, 7, 10, 2} el padre 2 los heredara según el valor mínimo de los TCP y la secuencia asignación Figura 3.7(b). El procedimiento a seguir es igual que en los anteriores obteniendo dos nuevos cromosomas Figura 3.8.

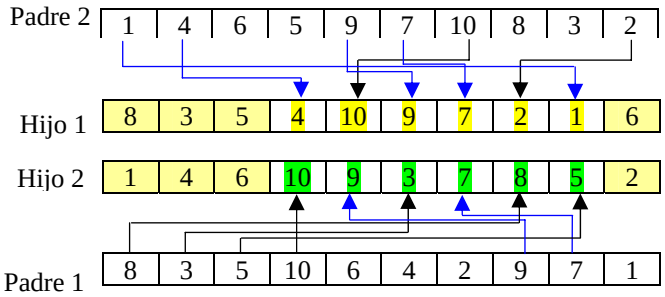


Figura 3.8. Operador de cruzamiento OSX_TCP

3.2.4 2PX_TCP

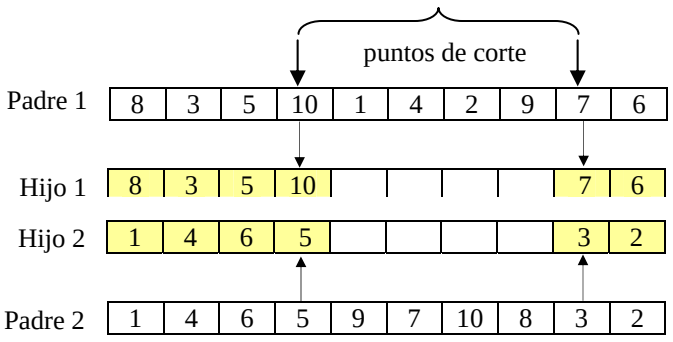
Método. Este operador escoge dos puntos de cruce al azar. Los elementos ubicados hasta el primer punto de corte se copian directamente del padre 1. De igual forma, los elementos a partir del segundo punto de corte hasta el último se copian del padre 1. Los elementos entre el primer punto de corte y el segundo se copian del padre 2 de acuerdo con el valor mínimo del TCP.

Algoritmo 2PX_TCP

Entrada: dos padres A^t y B^t del banco (*pool*) de padres, matriz de tiempos de cambio de partida.
Salida: dos nuevos padres C^{t+1} y D^{t+1};
Sea n el número de genes de un cromosoma

1. crea dos descendientes C^{t+1} y D^{t+1} como sigue:
   1.1. elige aleatoriamente un punto dos cruce $cp_1, cp_2 \in \{1,\ldots,n-1\}$
2. para $i = 1$ a cp_1 haz
   2.1. $c_i^{t+1} = a_i^t$
   2.2. $d_i^{t+1} = b_i^t$
3. para $i = cp_1 + 1$ a cp_2 haz
   3.1. $c_i^{t+1} = \min\{sdst(b_i^t)\}$
   3.2. $d_i^{t+1} = \min\{sdst(a_i^t)\}$
   3.3.
4. para $i = cp_2$ a n
   4.1. $c_i^{t+1} = a_i^t$
   4.2. $d_i^{t+1} = b_i^t$

La Figura 3.9 muestra cómo se realiza esta operación. El primer punto esta en la posición 4, y el segundo punto en la posición 9 Figura 3.9 (a). Los alelos son copiados de la posición 1 a la posición 4 del padre 1 al hijo2 y del padre 2 al hijo 1. Los alelos de la posición 5 a la 8, se copian del padre 1 al hijo 1, de manera que estos tengan el menor TCP Figura 3.9 (b).



(a) El padre directo hereda los alelos en las posiciones hasta el primer punto de corte y del segundo hasta la última posición

Trabajo por procesar

Trabajo que ya fue procesado	1	2	3	4	5	6	7	8	9	10
1	0	33	54	38	32	47	38	50	49	35
2	36	0	53	57	31	54	37	48	52	33
3	56	32	0	52	43	49	54	51	38	47
4	45	56	33	0	30	53	42	38	47	52
5	43	59	51	31	0	41	52	34	44	58
6	41	35	49	53	45	0	33	52	38	37
7	37	31	48	55	46	42	0	36	46	54
8	34	43	52	42	34	55	50	0	53	45
9	43	44	55	37	41	48	45	38	0	32
10	38	48	39	31	55	33	36	43	51	0

(b) TCP

Figura 3.9. Funcionamiento del operador antes de aplicar TCP

La posición 4 del hijo 1 tiene un alelo igual a 10, este se usa para asignar las demás posiciones, comparando los TCP de los 4 genes restantes que por asignar de igual manera que en los casos anteriores, obteniendo dos cromosomas nuevos Figura 3.10.

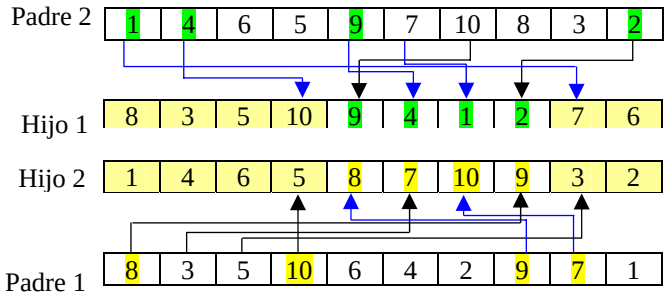


Figura 3.10. Operador de cruzamiento 2PX_TCP

3.2.5 KPX_TCP

Método. Los pasos a seguir son una generalización de 2PX. A partir de dos padres se realizan k puntos de corte en cada uno, quedando cada padre separado en $k+1$ partes. Las partes $\{1,3,5,...\}$ se copian directamente de uno de los padres. Las partes $\{2,4,6,...\}$ se copian del otro padre según el TCP con menor valor.

Algoritmo KPX_TCP.

Entrada: dos padres A^t y B^t del banco (*pool*) de padres y matriz de tiempos de cambio de partida.

Salida: dos descendientes C^{t+1} y D^{t+1}

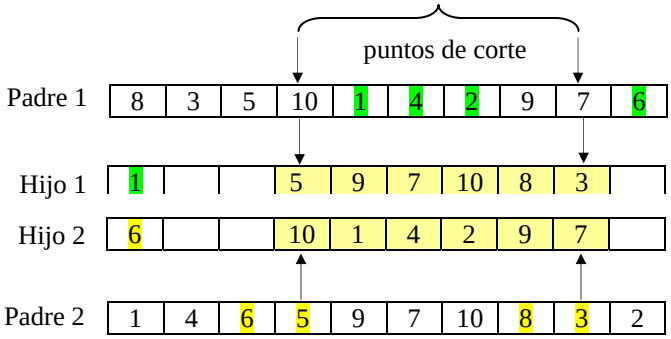
1. Crear dos descendientes C^{t+1} y D^{t+1} como sigue:
2. elige aleatoriamente k puntos de cruce $cp_1,\ldots,cp_k,\in\{1,\ldots,n-1\}$
3. para $i=1$ a cp_1 haz
   - 3.1.1. $c_i^{t+1}=a_i^t$
   - 3.1.2. $d_i^{t+1}=b_i^t$
4. fin ciclo
5. *Interrumpir*=0
6. para $j=2$ a k haz
   - 6.1. si *interrumpir* = 0 entonces
     - 6.1.1. para $i=cp_{j-1}+1$ hasta cp_j haz
       - 6.1.1.1. $c_i^{t+1}=b_i^t$
       - 6.1.1.2. $d_i^{t+1}=a_i^t$
     - 6.1.2. Interrumpir = 1
   - 6.2. en otro caso
     - 6.2.1. para $i=cp_{j-1}+1$ hasta cp_j haz
       - 6.2.1.1. $c_i^{t+1}=\min\{sdst(a_i^t)\}$
       - 6.2.1.2. $d_i^{t+1}=\min\{sdst(b_i^t)\}$
     - 6.2.2. interrumpir = 0
     - 6.2.3. fin condición
7. fin ciclo
8. si interrumpir =0 entonces
   - 8.1. para $i=cp_k+1$ hasta n haz
     - 8.1.1. $c_i^{t+1}=b_i^t$
     - 8.1.2. $d_i^{t+1}=b_i^t$
   - 8.2. en otro caso
     - 8.2.1. $c_i^{t+1}=a_i^t$
     - 8.2.2. $d_i^{t+1}=b_i^t$
9. fin condición

3.2.6 PMX_TCP

Método. Como primer paso se escogen dos puntos de corte que determinan una región que se copia inalteradamente del otro padre a cada hijo como se muestra en la Figura 3.11(a).

Esta región establece lo que se conoce como parejas de mapeo entre ambos padres. En el segundo paso, los hijos heredan las posiciones faltantes, antes y después de los puntos de cruce del padre directo, aplicando el mapeo en las posiciones que ya se copiaron en la primera fase con el menor tiempo de TCP.

En la Figura 3.11 (a) se muestra como el padre contrario hereda las posiciones del primer punto de corte hasta el segundo punto de corte, así el hijo 1 hereda del padre 2 los alelos {5, 9, 7, 10, 8, 3}, faltando por asignar el padre directo los alelos {10, 1, 4, 2, 6} según poseen el menor TCP en la secuencia Figura 3.11 (b).



(a) El padre directo hereda los alelos en las posiciones hasta el primer punto de corte y del segundo hasta la última posición

Trabajo por procesar

Trabajo que ya fue procesado	1	2	3	4	5	6	7	8	9	10
1	0	33	54	38	32	47	38	50	49	35
2	36	0	53	57	31	54	37	48	52	33
3	56	32	0	52	43	49	54	51	38	47
4	45	56	33	0	30	53	42	38	47	52
5	43	59	51	31	0	41	52	34	44	58
6	41	35	49	53	45	0	33	52	38	37
7	37	31	48	55	46	42	0	36	46	54
8	34	43	52	42	34	55	50	0	53	45
9	43	44	55	37	41	48	45	38	0	32
10	38	48	39	31	55	33	36	43	51	0

(b) Valor de los *tiempos de cambio de partida dependientes de la secuencia (TCP)*

Figura 3.11. Funcionamiento del operador antes de aplicar TCP

En la figura 3.12 se muestran los nuevos cromosomas que se obtienen al aplicar los TCP, con el mismo procedimiento que los casos anteriores.

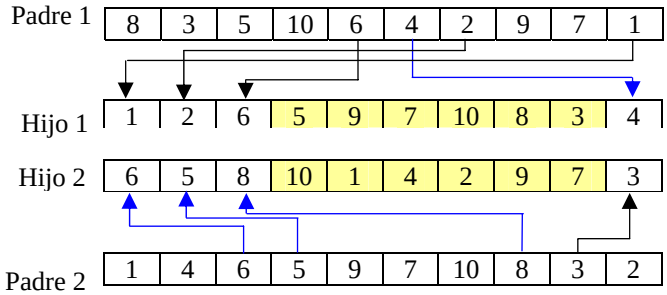


Figura 3.12. Operador de cruzamiento PMX_TCP

Algoritmo PMX_TCP:

Entrada: dos padre A^t y B^t del banco (*pool*) de padres y matriz de de tiempos de cambio de partida.
Salida: dos descendientes C^{t+1} y D^{t+1}

1 seleccionar dos padre A^t y B^t del banco (*pool*) de padres
2 crear dos descendientes C^{t+1} y D^{t+1} como sigue:
3 elige aleatoriamente 2 puntos de cruce $cp_1, cp_2 \in \{1,\ldots,n-1\}$
4 para $i = cp_1$ a cp_2 haz
  4.1 $c_i^{t+1} = b_i^t$
  4.2 $d_i^{t+1} = a_i^t$
5 fin ciclo
6 para $j = 1$ a $cp_1 - 1$ haz
  6.1 $c_i^{t+1} = \min\{sdst(a_i^t)\}$
  6.2 $d_i^{t+1} = \min\{sdst(b_i^t)\}$
7 fin ciclo
8 para $i = cp_2 + 1$ hasta n haz
  8.1 $c_i^{t+1} = \min\{sdst(a_i^t)\}$
  8.2 $d_i^{t+1} = \min\{sdst(b_i^t)\}$
9 fin ciclo

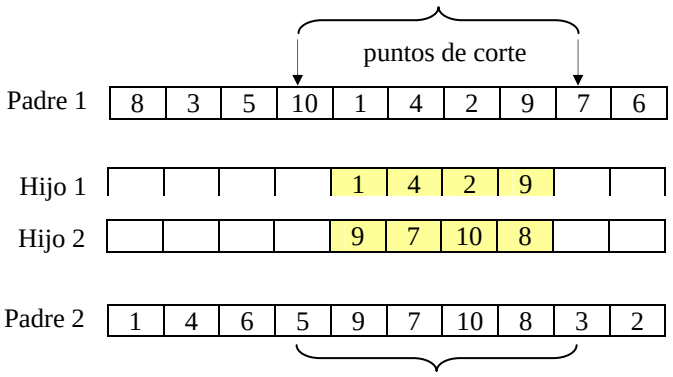
3.2.7 OX_TCP

Se utilizan dos puntos de cruce y la parte interior entre estos puntos se heredan del padre directo. Las posiciones libres en los hijos se rellenan a partir del segundo punto de cruce en el orden relativo del otro padre.

Algoritmo OSX_TCP.

1 seleccionar dos padre A^t y B^t del banco de padres
2 crear dos descendientes C^{t+1} y D^{t+1} como sigue:
3 elige aleatoriamente 2 puntos de cruce $cp_1, cp_2 \in \{1,\ldots,n-1\}$
4 para $i = cp_1$ a cp_2 haz
  4.1 $c_i^{t+1} = a_i^t$
  4.2 $d_i^{t+1} = b_i^t$
5 fin ciclo
6 para $j = 1$ a $cp_1 - 1$ haz
  6.1 $c_i^{t+1} = \min\{sdst(b_i^t)\}$
  6.2 $d_i^{t+1} = \min\{sdst(a_i^t)\}$
7 fin ciclo
8 para $i = cp_2 + 1$ hasta n haz
    8.1.1 $c_i^{t+1} = \min\{sdst(b_i^t)\}$
    8.1.2 $d_i^{t+1} = \min\{sdst(a_i^t)\}$
9 fin ciclo

En la Figura 3.13 (a) se muestra un ejemplo de este ordenador, donde el hijo 1 hereda del padre directo los alelos después del primer y antes del segundo punto de corte {1, 4, 2, 9}. El padre contrario asigna los alelos restantes considerando los TCP para la secuencia de asignación semejante a los casos anteriores Figura 13 (b).



(a) El padre directo hereda los alelos en las posiciones entre los puntos de corte

Trabajo por procesar

Trabajo que ya fue procesado	1	2	3	4	5	6	7	8	9	10
1	0	33	54	38	32	47	38	50	49	35
2	36	0	53	57	31	54	37	48	52	33
3	56	32	0	52	43	49	54	51	38	47
4	45	56	33	0	30	53	42	38	47	52
5	43	59	51	31	0	41	52	34	44	58
6	41	35	49	53	45	0	33	52	38	37
7	37	31	48	55	46	42	0	36	46	54
8	34	43	52	42	34	55	50	0	53	45
9	43	44	55	37	41	48	45	38	0	32
10	38	48	39	31	55	33	36	43	51	0

(b) TCP

Figura 3.13. Funcionamiento del operador antes de aplicar TCP

La asignación final se observa en la Figura 3.14 creando dos nuevos cromosomas, cuya secuencia considera los TCP de menor valor.

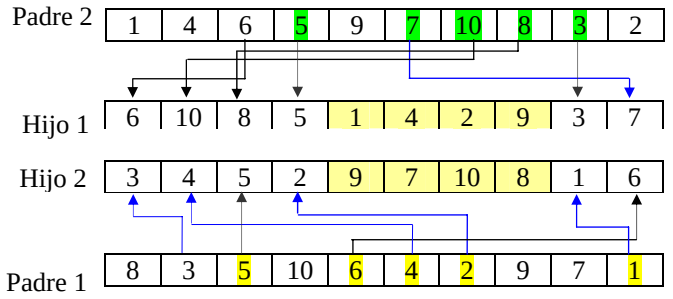


Figura 3.14. Operador de cruzamiento OX_TCP

Se eligieron estos operadores de cruzamiento por las distintas recombinaciones de los alelos que realizan en cada permutación, es decir, se cruzan todos los alelos del cromosoma (permutación), tanto los genes extremos como los intermedios, generando así variabilidad en las nuevas secuencias de las permutaciones.

3.3 Conclusiones del capítulo

Al hacer la revisión bibliográfica para la búsqueda de operadores de cruzamiento aplicables a TFH TCP, observamos que los investigadores no siguen un criterio para seleccionarlos, aplican los disponibles en la literatura. Además la incorporación de los TCP a los OC es algo nuevo, no se cuenta con un análisis del comportamiento de esta clase de operadores. Así como el análisis de su contribución sobre el AG para mejorar los valores $C_{\max}$.

Se concluye que es necesario describir el comportamiento teórico y experimental de los OC, para indicar cuales son los más adecuados para cada problema. No hay evidencia de la existencia de un operador de cruzamiento universal que obtenga los mejores resultados para todo tipo de problema. En consecuencia se necesita determinar qué operador se ajusta a las características particulares de cada problema para obtener buenos resultados.

El cruzamiento realiza el trabajo más delicado, así que seleccionar un método de cruzamiento para cada problema, así como el análisis de su evolución dentro del AG es muy importante.

CAPÍTULO 4

EXPERIMENTO COMPUTACIONAL PARA ANÁLISIS COMPARATIVO DE OPERADORES DE CRUZAMIENTO

4.1 Sistema PLARETF

El sistema PLARETF (Romero, 2007), como se muestra en la Figura 4.1, incluye los siguientes módulos: Interfaz, Entrada, Contenedor de Algoritmos, de Referencias y Salida. El módulo Contenedor de Algoritmos, como su nombre lo indica, es asignado para una colección de procedimientos, facilitando al usuario el desarrollo y la implementación de nuevos algoritmos para resolver problemas de Talleres de Flujo sin afectar su entrada y salida del sistema. Los algoritmos se agregan en forma independiente, es decir, dentro del módulo respetando los estándares de codificación establecidos en el sistema. Cabe indicar que el módulo Contenedor de Algoritmos es el único utilizado en esta investigación, quedando fuera de alcance en este trabajo los demás módulos.

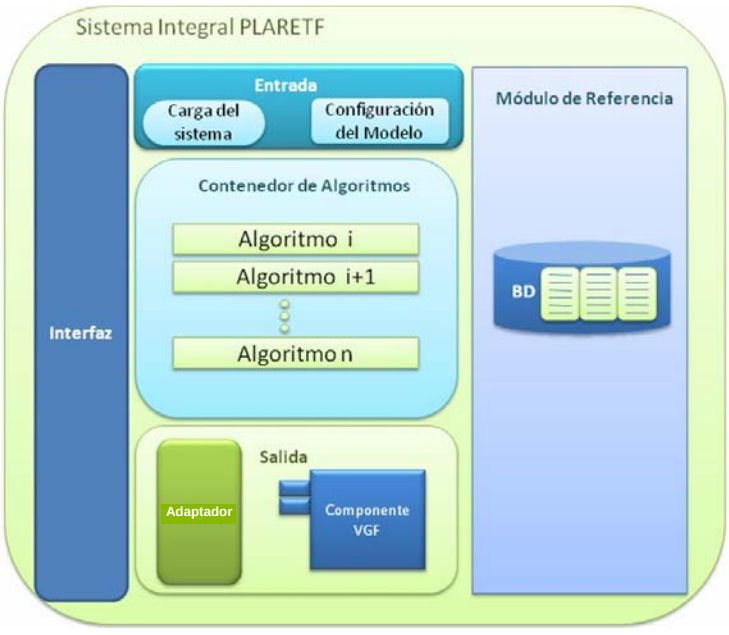


Figura 4.1. Sistema PLARETF

4.2 Implementación de operadores de cruzamiento en el Sistema PLARETF

En la presente investigación se han utilizado los siguientes archivos del sistema PLARETF que pertenecen al módulo Contenedor de Algoritmos:

1. *PLARETF.exe* archivo ejecutable del sistema de simulación, este aplica el algoritmo genético a los datos de los archivos de la carpeta LOGS.
2. *config.ini* archivo de configuración del sistema donde se declaran todas las variables del sistema de simulación y la configuración de los talleres a resolver.
3. *listacarga.ini* archivo de configuración del sistema para la lectura de los distintos talleres a resolver.
4. *GA_HFS_SDST.ini* archivo de configuración para generar la programación del taller que consta del AG, que contempla los tiempos de cambio de partida dependientes de la secuencia e incluye todos los algoritmos de cruzamiento, mutación y selección.
5. *LOGS* carpeta de archivos de carga para el sistema donde se guardan los distintos talleres a resolver.

El archivo *GA_HFS_SDST.ini* cuenta actualmente con siete algoritmos de cruzamiento (OBX_TCP, PPX_TCP, OSX_TCP, 2PX_TCP, KPX_TCP, PMX_TCP y OX_TCP). El algoritmo KPX tiene la posibilidad de generar k diferentes tipos de cruzamiento, aumentando la cantidad de OC.

Se agrega la rutina que generaliza el cálculo del tiempo de procesamiento de todos los trabajos ($C_{\max}$) para TFH TCP de m etapas.

4.3 Diseño del experimento

A continuación se mencionan las especificaciones de cada parámetro y las características de los problemas tratados en el experimento de esta investigación de acuerdo a la tripleta $\alpha|\beta|\gamma$, así como el análisis de los resultados obtenidos.

4.3.1 Parámetros del experimento

Características de los recursos, parámetro α:

Se tienen TFH con dos o tres etapas, $i=\{2,3\}$, donde cada etapa tiene un conjunto $M_i \in \{1,2,3\}$ máquinas no relacionadas, con $|M_i| \geq 1$. En la Figura 4.2 se muestran algunos de los modelos de talleres que se consideraron en esta investigación.

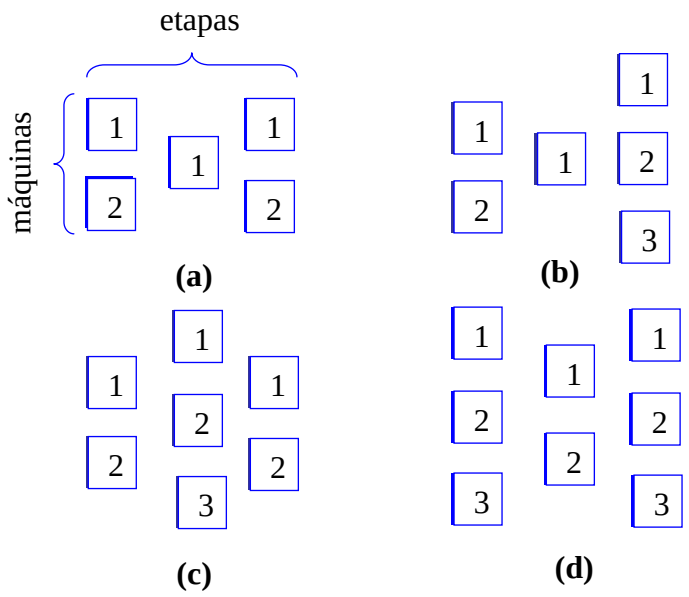


Figura 4.2. Ejemplos de los talleres de tres etapas resueltos, el inciso (a) corresponde a Taller_3_2_1_2, (b) Taller_3_2_1_3, (c) Taller_3_2_3_2 y (d) Taller_3_3_2_3

Características de los trabajos parámetro β:

Se tiene un conjunto $N=\{1,\ldots,n\}$ de n trabajos, $n=20,50,100$ disponibles al tiempo inicial 0. Estos tienen que ser procesados consecutivamente en cada etapa i en los

recursos antes mencionados. Cada trabajo debe ser procesado exactamente por una máquina en cada etapa. Sea p_{ijl} el tiempo de procesamiento del trabajo $j \in N$, en la máquina $l \in M_i$, dentro de la etapa *i*. Se supone que el tiempo de ajuste de cada máquina depende de la secuencia de los trabajos. Sea S_{iljk} el tiempo de ajuste en la máquina *l*, en la etapa *i*, para procesar el trabajo $k \in N$, después de haber procesado el trabajo *j*.

Para obtener de manera aleatoria los valores de procesamiento de cada trabajo así como los TCP se utilizó el generador lineal congruente de números pseudoaleatorios basado en aritmética modular (Taillard, 1989). Los tiempos de procesamiento y TCP tienen la distribución uniforme en el intervalo [50, 99]. El generador lineal con el cual se obtienen los valores mencionados de esta dado por la fórmula recursiva:

$$X_{i+1} = (16807 X_i) \bmod (2^{31} - 1)$$

Esta implementación usa enteros de 32 bits y provee una secuencia de valores distribuidos uniformemente el rango (0,1):

1. Semilla inicial y constantes: $X_0 \left(0 < X_0 < 2^{31} - 1\right)$
   $\mathrm{a} = 16807, \mathrm{b} = 127773, \mathrm{c} = 2836, \mathrm{m} = 2^{31} - 1$

$$K := \left\lfloor \frac{X_i}{b} \right\rfloor$$

2. Modificación de la semilla: $X_{i+1} := a(X_i \bmod b) - kc$
   Si $X_{i+1} < 0$ entonces $X_{i+1} := X_i + 1 + m$

3. Nuevo valor de la semilla: X_{i+1}

   Valor actual del generador: $\frac{X_{i+1}}{m}$

Características del criterio parámetro γ **:**

Se realizó un análisis comparativo de los siete algoritmos de cruzamiento, para saber cuál de ellos obtiene el mejor valor de $C_{\max}$.

Usando la notación de tres campos, el problema considerado es:

$$HF(2,3),(Pm^{(i)})_{i=1}^{3} \left| s_{ijlk} \right| C_{\max}$$

La cuestión para efectos de diseño del experimento sería ¿Sí afecta el operador de cruzamiento el valor de $C_{\max}$ en TFH TCP? con lo cual se crea la siguiente hipótesis nula:

$$H_0 : \mu_1 = \mu_2 = \ldots = \mu_7$$

Esto implica que los operadores de cruzamiento no tienen efecto directo en $C_{\max}$, y los valores promedios obtenidos para $C_{\max}$, serían iguales para todos los operadores de cruzamiento.

De no ser así, la hipótesis alternativa es:

$$H_1 : \text{al menos un } \mu_i, i = 1,\ldots,7, \text{ es diferente de las demás.}$$

El planteamiento de la hipótesis alternativa permite responder a la pregunta ¿cuál operador es el mejor? Es decir, cual de ellos proporciona en medio un valor menor de $C_{\max}$.

El experimento a realizar es unifactorial con el operador de cruzamiento como el factor. Se tienen siete niveles del factor:

1. OBX_TCP
2. PPX_TCP
3. OSX_TCP
4. 2PX_TCP
5. KPX_TCP
6. PMX_TCP
7. OX. _TCP

La variable respuesta del experimento es $C_{\max}$.

4.3.2 Generación de datos de entrada

Los parámetros de entrada se generan aleatoriamente. Se consideran TFH de 2 y 3 etapas. El número de máquinas por cada etapa es de 1 a 3. Para cada configuración se consideran

20, 50 y 100 trabajos cuyos tiempos de procesamiento son distribuidos uniformemente en el rango de $[50,99]$. En consecuencia se tienen $2 \times 7 \times 3 = 42$ problemas distintos considerando 7 niveles del factor OC y tres grupos según el número de trabajos.

En la Figura 4.3 se muestra el contenido del archivo taller3_2_1_2_20_01.txt con los datos de entrada al sistema PLARETF, como su nombre los indica es un taller de 3 etapas con 2 máquinas en la primera etapa, 1 en la segunda y 2 en la tercera, con 20 trabajos a procesar, y el 01 corresponde al primer taller con esta configuración. Este taller se compone de varias matrices de datos. La primera matriz contiene en la primera línea, tres cifras, el número de trabajos, número de máquinas y numero de etapas, respectivamente. Las cifras de la segunda línea corresponde al número de máquinas por cada etapa; los datos restantes corresponde a los tiempos de procesamiento, los valores de las columnas, con valor idéntico, representan el número de cada máquina y las columnas a su derecha con valores diferentes, corresponden a los tiempos de procesamiento de cada trabajo en la máquina.

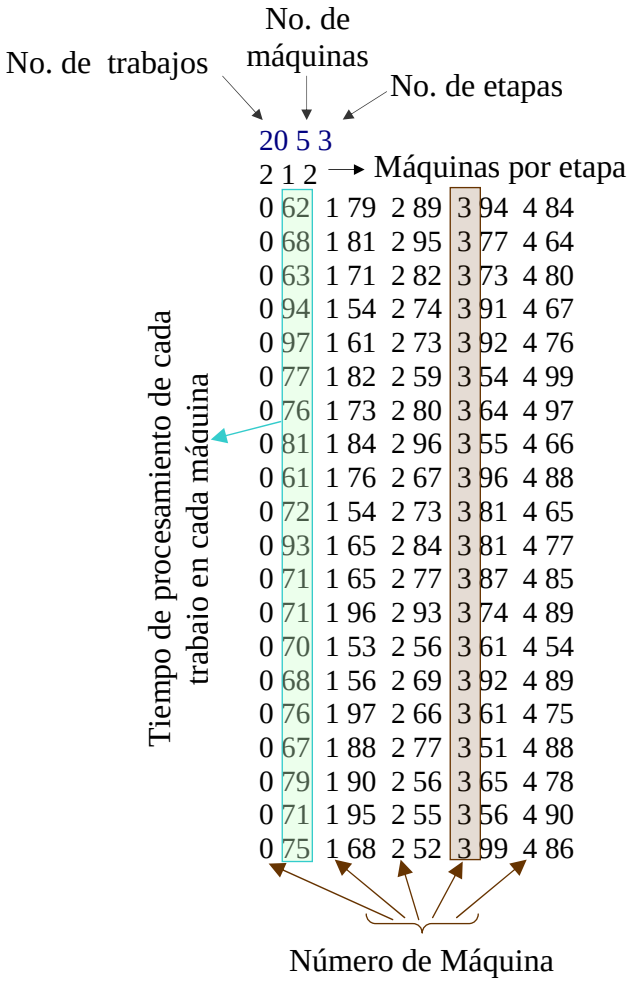


Figura 4.3 Ejemplo de cómo entran los datos de los recursos y tiempos de procesamiento de un TFH TCP de tres etapas

La segunda parte del archivo contiene la matriz de valores de los TCP donde la línea con SSD, indica el inicio de los TCP; la línea con la notación M0 indica el número de la máquina, y los datos restantes son los TCP correspondientes a cada trabajo para esa máquina Figura 4.4 .

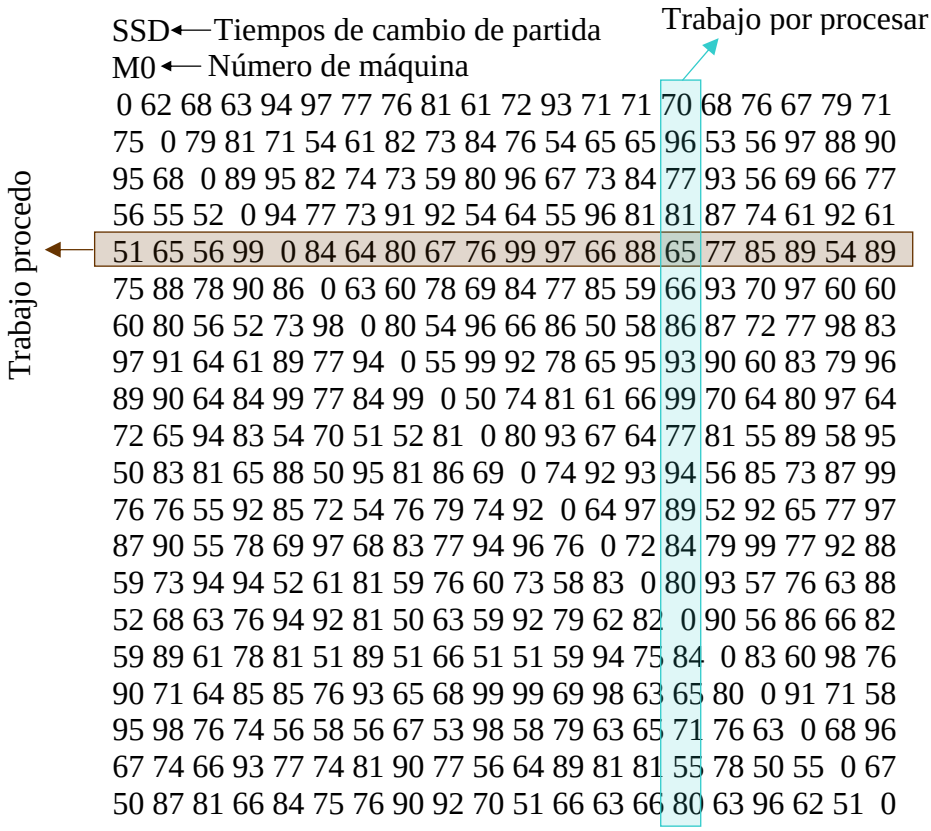


Figura 4.4 Ejemplo de cómo entran los datos de los TCP de un TFH TCP de tres etapas.

4.4 Análisis de resultados

Para obtener los datos que verifican la hipótesis nula H_0, se implementa y añade al sistema PLARETF, el algoritmo que calcula el $C_{\max}$ para cada permutación obtenida del AG. Este incluye los TCP para THF de m etapas, su diagrama de flujo se encuentra en el anexo D. También se generan los archivos que contienen todos los valores $C_{\max}$ para cada experimento con el formato de salida descrito en la Tabla 4.1.

Los experimentos se realizan en una computadora personal con procesador Pentium 4 con CPU de 3.00 GHz y 504 MB de memoria principal, esta cuenta con un sistema operativo Windows XP (V. 5.1.2600).

Tabla 4.1. Datos crudos $C_{\max}$ ordenados por nivel de tratamiento para Completion_time3_2_1_2_20.txt

	$C_{\max}$ para cada OC						
Observaciones	OBX_TCP	PPX_TCP	OSX_TCP	2PX_TCP	KPX_TCP	PMX_TCP	OX_TCP
1	3066	3217	3109	3118	2995	3201	3173
2	3060	3060	3083	3077	3089	3103	3151
3	2991	2907	2864	2855	2883	2855	2855
4	3111	3212	3136	3107	3289	3092	3163
5	3090	3116	3157	3157	2947	3133	3079
6	3188	3202	3051	3116	3194	3172	3209
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
17	3065	3088	3065	3004	2969	3983	2943
18	3214	3196	3293	3180	3303	3120	3217
19	3029	2889	2995	3053	2939	3012	3016
20	3161	3157	3223	3244	3254	3193	3101
Total	62444	62594	62324	62053	61965	62094	62196
Promedio	3122,20	3129,70	3116,20	3102,65	3098,25	3104,70	3109,80

Por las características aleatorias del experimento, se trabaja con los residuos de los resultados, a los que se les aplica un análisis estadístico. Estos requieren la condición de homogeneidad del ambiente del experimento. Para verificar dicha homogeneidad los residuos de los resultados se validan por métodos gráficos y se analizan numéricamente mediante el método de varianza ANOVA (*ANalysis Of VAriance*).

Los resultados del experimento que se muestran en la Tabla 4.1, corresponden a la matriz de valores C_{max},, que pertenecen al archivo Completion_time3_2_1_2_20.txt, que es un TFH TCP de 3 etapas con 2 máquinas en la primera, 1 en la segunda, 2 en la tercera y 20 trabajos a procesar. Cada valor de $C_{\max}$ se denota como C_{ij}, para este taller $i = 1,\ldots 20$ y $j = 1,\ldots,7$, donde i es el número de observaciones y j es el tipo de OC.

Por ejemplo: $C_{43} = 3136$ es el valor de $C_{\max}$ de la observación 4 obtenida de aplicar OSX.

En las siguientes dos secciones se presenta el método aplicado para la obtención de los residuos, su validación mediante métodos gráficos y su análisis numérico mediante la varianza ANOVA (*Analysis Of VAriance*).

4.4.1 Análisis residual

Antes de obtener los residuos de los resultados, se necesita homogenizarlos, es decir, cada dato crudo del experimento se transforma de acuerdo a la siguiente fórmula:

$$C_{ij}^{*} = \frac{\left(MIN\{C_{ij},\, j=1,\ldots,7\} - C_{ij}\right)}{MIN\{C_{ij},\, j=1,\ldots,7\}} \times 100,\ i = 1\ldots 20 \qquad (4.1)$$

Entonces $C_{\max}$ se transforma en $C_{\max}^{*}$.

Así, por ejemplo, los datos anteriores se homogenizan, obsérvese en la Tabla 4.1 el valor de $C_{43} = 3136$ al que le corresponde un $C_{43}^{*} = 1{,}4230$ cómo se muestra en la Tabla 4.2.

Tabla 4.2. Datos tratados de $C_{\max}^{*}$, ordenados por nivel de tratamiento.

Observaciones	$C_{\max}^{*}$ para cada OC						
	OBX_TCP	PPX_TCP	OSX_TCP	2PX_TCP	KPX_TCP	PMX_TCP	OX_TCP
1	23,706	74,124	38,063	41,068	0,0000	68,781	59,432
2	0,0000	0,0000	0,7516	0,5556	0,9477	14,052	29,739
3	47,636	18,214	0,3152	0,0000	0,9807	0,0000	0,0000
4	0,6145	38,810	14,230	0,4851	63,713	0,0000	22,962
5	48,524	57,346	71,259	71,259	0,0000	63,115	44,791
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
17	41,454	49,269	41,454	20,727	0,8835	13,592	0,0000
18	30,128	24,359	55,449	19,231	58,654	0,0000	31,090
19	48,460	0,0000	36,691	56,767	17,307	42,575	43,960
20	19,349	18,059	39,342	46,114	49,339	29,668	0,0000
Total	670,071	713,982	623,127	537,318	500,720	551,402	579,591
Promedio	33,504	35,699	31,156	26,866	25,036	27,570	28,980

El procedimiento anterior se aplica en la mayoría de las investigaciones. Sin embargo, en el transcurso de esta tesis al analizar el comportamiento estadístico de los C_{ij}^{*}, se

observa que tienden a sesgarse hacia el valor mínimo ya que, debido a la transformación para todo par de ij los $C_{ij}^{*} \geq 0$. Esto implica que se concentran al cero por la derecha con el comportamiento gráfico mostrado en la Figura 4.5, el cual no es normal como se asume en las investigaciones, más bien tiene la forma de χ^2 o F de Fisher.

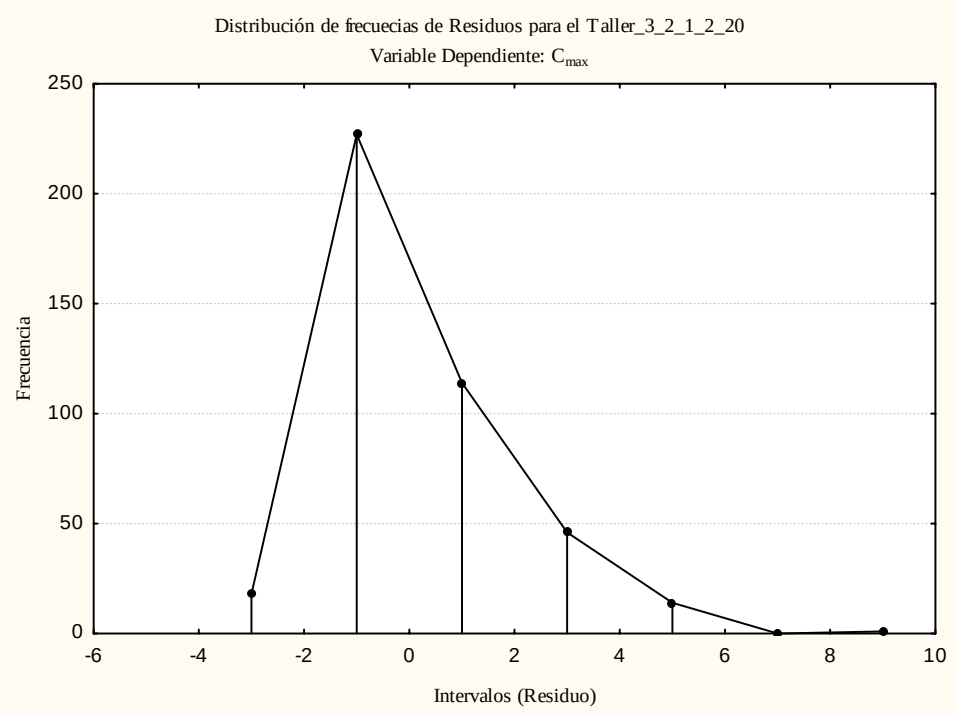


Figura 4.5 Distribución de datos "normalizados" para Taller_3_2_1_2_20.

La justificación que realizan los investigadores en la literatura, al aplicar esta transformación, es que los residuos de diferentes muestras aleatorias deben estar normalmente distribuidos para aplicarles el análisis de varianza. Porque en caso contrario este método de análisis no es aplicable. Por ejemplo al generar los valores de $C_{\max}$ de talleres con un número fijo de etapas y distintos modelos de recursos, al aplicar la transformación los residuos las muestras no son normalmente distribuidos. En consecuencia el análisis de varianza no es aplicable. En la presente investigación se determinó que la aplicación de este método es posible solo para casos discretos cuando se

analiza un solo taller con la misma cantidad de trabajos, es decir, un mismo taller con j trabajos pero distintos tiempos de procesamiento y TCP.

Se propone el siguiente procedimiento para normalizar los valores de $C_{\max}$ de un TFH TCP con mismas etapas y distintos modelos de recursos. Se considera la matriz C_{na} como el conjunto de todos los valores de $C_{\max}$ obtenidos de un algoritmo, donde n es el número de observaciones y a es el número de niveles del experimento. Se calcula el valor medio MED_i entre los niveles de cada observación mediante la siguiente formula:

$$MED_i = \frac{\left(MIN\left\{C_{ij},\, j=1,\ldots,a\right\} + MAX\left\{C_{ij},\, j=1,\ldots,a\right\}\right)}{2}.$$

La matriz de datos normalizados se calcula de acuerdo a la siguiente fórmula:

$$\overline{C}_{ij}^{*} = \frac{\left(C_{ij} - MED_i\right)}{MED_i} \times 100,\ i=1,\ldots,n \tag{4.2}$$

Este procedimiento permite normalizar los datos de las muestras y aplicar análisis de varianza para todo tipo de taller con el mismo número de etapas y distintas distribuciones de recursos dentro de cada etapa.

Los datos anteriores se homogenizan con $\overline{C}_{ij}^{*}$, obsérvese en la Tabla 4.1 que el valor de $C_{34} = 3136$ le corresponde un $\overline{C}_{34}^{*} = 3{,}4404$ cómo se muestra en la Tabla 4.3.

Tabla 4.3. Datos tratados de $\overline{C}^{*}_{\max}$, ordenados por nivel de tratamiento.

	$\overline{C}^{*}_{\max}$ para cada OC						
Observaciones	OBX_TCP	PPX_TCP	OSX_TCP	2PX_TCP	KPX_TCP	PMX_TCP	OX_TCP
1	-1,2878	3,5737	0,0966	0,3863	-3,5737	3,0586	2,1571
2	-1,4493	-1,4493	-0,7085	-0,9018	-0,5153	-0,0644	1,4815
3	2,3264	-0,5474	-2,0185	-2,3264	-1,3685	-2,3264	-2,3264
4	-2,4765	0,6897	-1,6928	-2,6019	3,1034	-3,0721	-0,8464
5	1,2451	2,0970	3,4404	3,4404	-3,4404	2,6540	0,8847
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
17	1,6584	2,4212	1,6584	-0,3648	-1,5257	-1,0614	-2,3881
18	0,0934	-0,4671	2,5537	-0,9654	2,8652	-2,8340	0,1869
19	1,9522	-2,7600	0,8078	2,7600	-1,0771	1,3800	1,5146
20	-0,5036	-0,6295	1,4479	2,1089	2,4237	0,5036	-2,3922
Total	9,5717	13,7177	5,0466	-3,3087	-6,8694	-2,0024	0,8803
Promedio	0,4785	0,68588	0,25232	-0,16543	-0,34347	-0,10012	0,04401

Para efecto del análisis los datos se presentan según el formato de la Tabla 4.4 (Montgomery,1991).

Tabla 4.4. Datos tratados de representación de datos para análisis estadístico

Tratamientos	**Observaciones**						**Totales**	**Medias**
	1	2	⋯	*J*	⋯	*n*		
1	y_{11}	y_{21}		y_{1j}	⋯	y_{1n}	$y_{1\bullet}$	$\bar{y}_{1\bullet}$
2	y_{21}	y_{22}		y_{2j}	⋯	y_{2n}	$y_{2\bullet}$	$\bar{y}_{2\bullet}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
i	y_{i1}	y_{i2}		y_{ij}	⋯	y_{in}	$y_{i\bullet}$	$\bar{y}_{i\bullet}$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
a	y_{a1}	y_{a2}		y_{ai}	⋯	y_{an}	$y_{a\bullet}$	$\bar{y}_{a\bullet}$
							$y_{\bullet\bullet}$	$\bar{y}_{\bullet\bullet}$

En la tabla:

$$y_{i\bullet} = \sum_{j=1}^{n} y_{ij}$$

$$\bar{y}_{i\bullet} = \frac{y_{i\bullet}}{n}$$

$$y_{\bullet\bullet} = \sum_{i=1}^{a}\sum_{j=1}^{n} y_{ij}$$

$\bar{y}_{\bullet\bullet} = \frac{y_{\bullet\bullet}}{N}$, $i = 1,\ldots a,\ j = 1,\ldots,k,\ N = an$ es el número global de observaciones. El modeo estadístico lineal es:

$$Y_{ij} = \mu_i + \varepsilon_{ij} \begin{cases} i = 1,\ldots,a \\ j = 1,\ldots,n \end{cases}, \text{ donde } \mu_i = \mu + \tau_i \qquad (4.3)$$

El término Y_{ij} representa a una variable aleatoria que denota a la observación (i, j); μ es la media global, τ_i es el efecto del tratamiento i; ε_{ij} es la componente del error aleatorio; μ_i es la media del tratamiento i. Entonces cada tratamiento define una población con una media μ_i, que consiste de la media global μ más un efecto τ_i. Se supone que los errores ε_{ij} están distribuidos de manera normal e independiente, con $\mathrm{N}(0, \sigma^2)$.

Una vez normalizados los resultados, se calculan sus residuos de acuerdo a la siguiente fórmula (Montgomery & Runger, 1994)

$$e_i = y_i - \bar{y}_i,\ \ i = 1,\ldots,n \qquad (4.4)$$

donde y_i es la variable aleatoria observada para la corrida i y $\bar{y}_i$ es la media del tratamiento i.

El conjunto de resultados e_i se utiliza para verificar que las observaciones están distribuidas de manera normal e independiente, con la misma varianza para cada nivel de factor, es decir, comprobar la suposición de idoneidad del modelo, a través de la normalidad, homocedasticidad e independencia de residuos. Verificándose mediante la construcción de las siguientes gráficas,

a) probabilidad normalidad de los residuos (normalidad),
b) residuos contra niveles del factor, comparando la dispersión de los residuos (varianzas iguales),
c) residuos contra $\bar{y}_i$, la variabilidad de los residuos no debe depender de $\bar{y}_i$.

Además del análisis residual se aplica el análisis de varianza para corroborar la hipótesis H_0, el cual se describe a continuación.

4.4.2 Análisis de varianza

Este análisis permite comprobar la hipótesis nula H_0, es decir, verificar si las medias de los 7 niveles del OC son iguales entre sí, mediante la siguiente fórmula:

$$f_0 = \frac{\frac{SS_{Tratamientos}}{a-1}}{\frac{SS_E}{a(n-1)}} = \frac{MS_{Tratamientos}}{MS_E}, \tag{4.5}$$

donde y_{ij} es la observación j, tomada bajo el tratamiento i $j = 1,\ldots,n,\ i = 1,\ldots,a$, a es el número de niveles, n es el número de observaciones, $N = an$ es el número global de observaciones.

Se calculan:

$$SS_{Tratamientos} = \sum_{i=1}^{a} \frac{y_i^2 \cdot}{n} - \frac{y^2 \cdot \cdot}{N} \tag{4.6}$$

$$SS_E = SS_T - SS_{Tratamientos} \tag{4.7}$$

$$SS_T = SS_{Tratamientos} + SS_E \tag{4.8}$$

$$MS_{Tratamientos} = \frac{SS_{Tratamientos}}{a-1} \tag{4.9}$$

El cociente f_0 tiene una distribución F de Fisher con $(a-1)$ y $a(n-1)$ grados de libertad. Se procede a verificar la hipótesis nula mediante la comparación de f_0 y el valor obtenido por tablas de $f_{\alpha,a-1,a(n-1)}$. De cumplirse $f_0 > f_{\alpha,a-1,a(n-1)}$, H_0 se rechaza, es decir, que al menos una de las medias de los niveles no es igual a las restantes.

4.5 Análisis de Talleres Flujo Híbrido de dos y tres etapas con Tiempos de Cambio de Partida

En las siguientes dos secciones se presenta el análisis de los resultados obtenidos al aplicar el AG a 60 talleres diferentes de 2 y 3 etapas para cada una de las siguientes configuraciones:

a) Taller_2_1_3
b) Taller_2_2_2
c) Taller_2_2_3
d) Taller_3_2_1_2
e) Taller_3_2_1_3
f) Taller_3_2_3_2
g) Taller_3_3_2_3

Con un total de 180 talleres de 2 etapas, y 240 talleres de 3 etapas generando un total de 2940 valores diferentes de $C_{\max}$, estos resultados se discuten a continuación.

4.5.1 Análisis de Talleres de dos etapas

En las Figura 4.6-4.10 se muestran del análisis estadístico de los resultados para TFH TCP de dos etapas. En la Figura 4.6 se observa que los residuos se aproximan en su parte central hacia la recta lineal, implicando la suposición de normalidad.

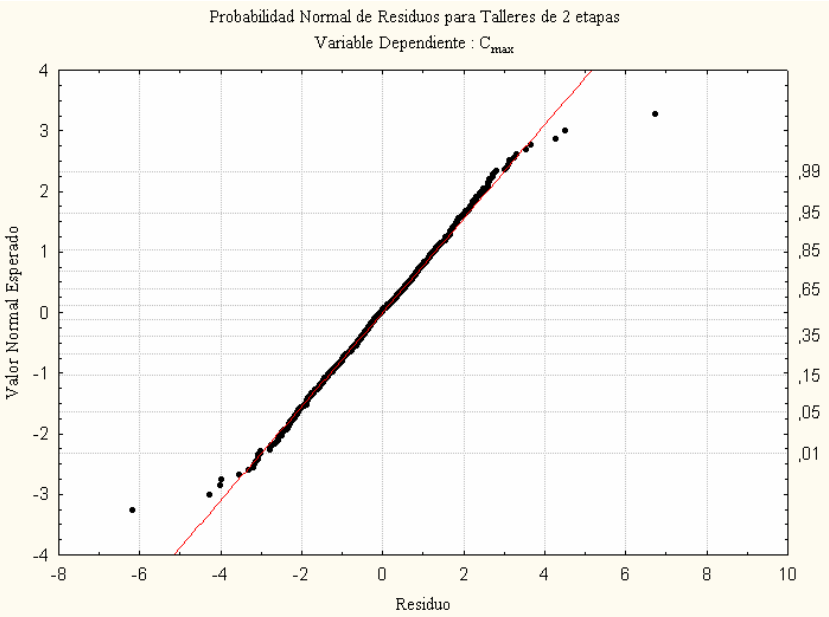


Figura 4.6 Probabilidad normal de residuos para Talleres de dos etapas

En el Figura 4.7 se muestra que los residuos fluctúan en ambos lados con respecto a $C_{\max}$, y la desviación no es excesivamente grande.

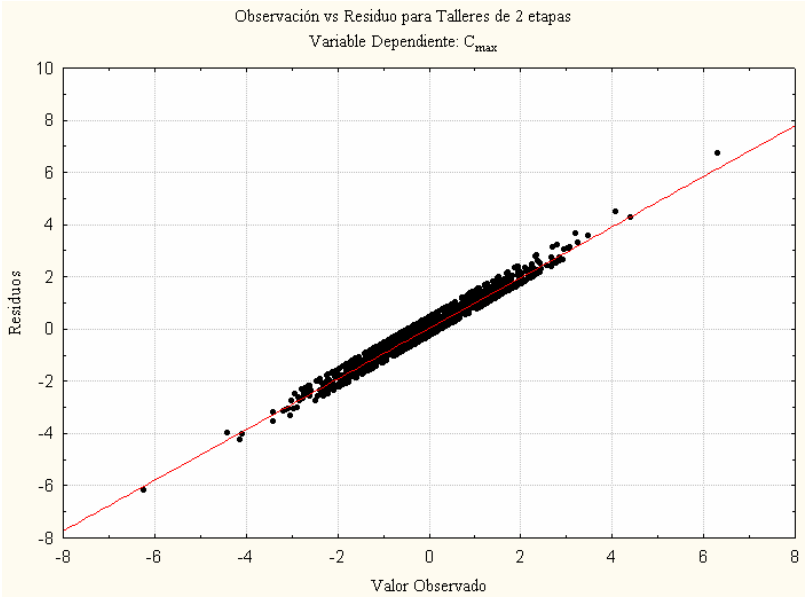


Figura 4.7 Valores observados contra los residuos para Talleres de dos etapas

En la Figura 4.8 se observa la independencia de los residuos ya que la gráfica no muestra un patrón definido de la distribución de los mismos, fluctuando entre valores positivos y negativos a lo largo de la gráfica.

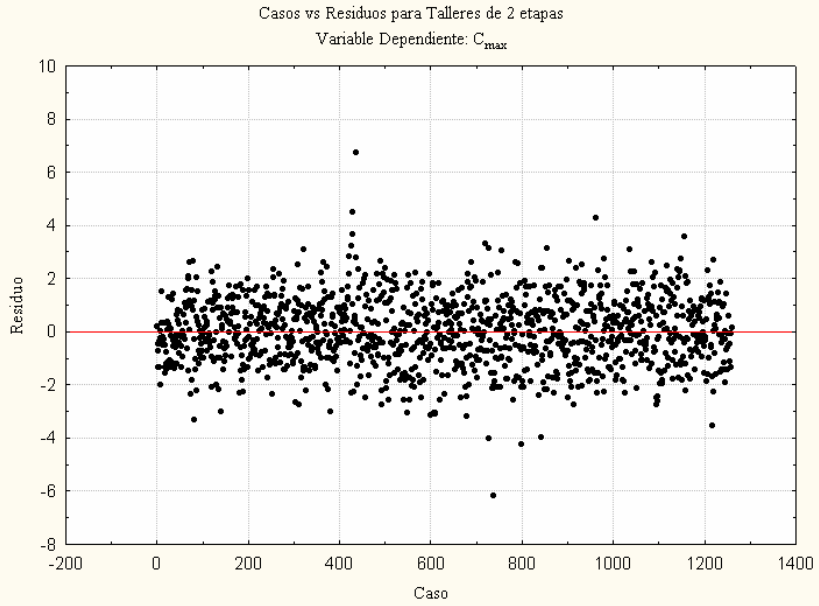


Figura 4.8 Número de casos contra residuos para Talleres de dos etapas

La Figura 4.9 muestra la variabilidad de los residuos con respecto al único factor OC. Se visualiza que no existe un patrón definido de dicha variabilidad con respecto a cada nivel del factor OC.

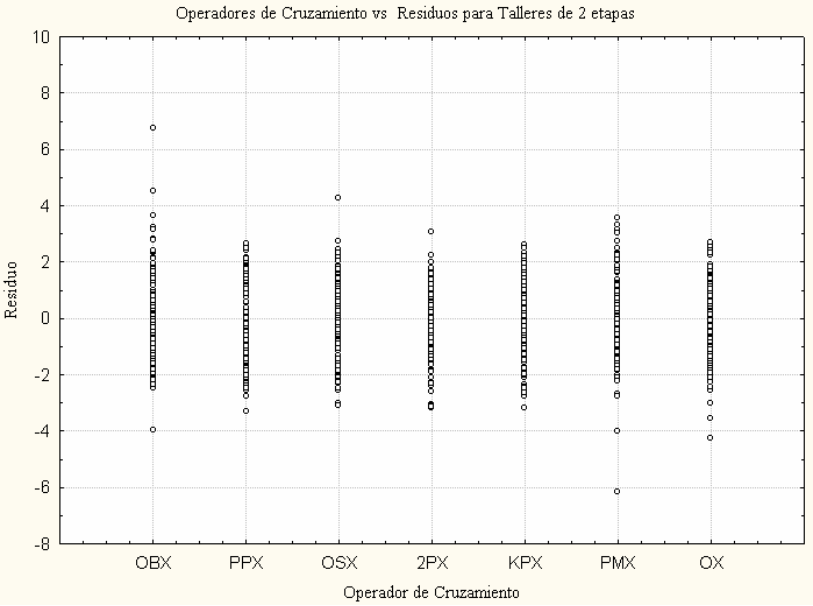


Figura 4.9 Nivel de factor contra residuos para Talleres de dos etapas.

Así mismo en la Figura 4.10 se muestra la dispersión de los residuos con respecto a su valor medio. Se reafirma que no existe un patrón definido entre media y residuos.

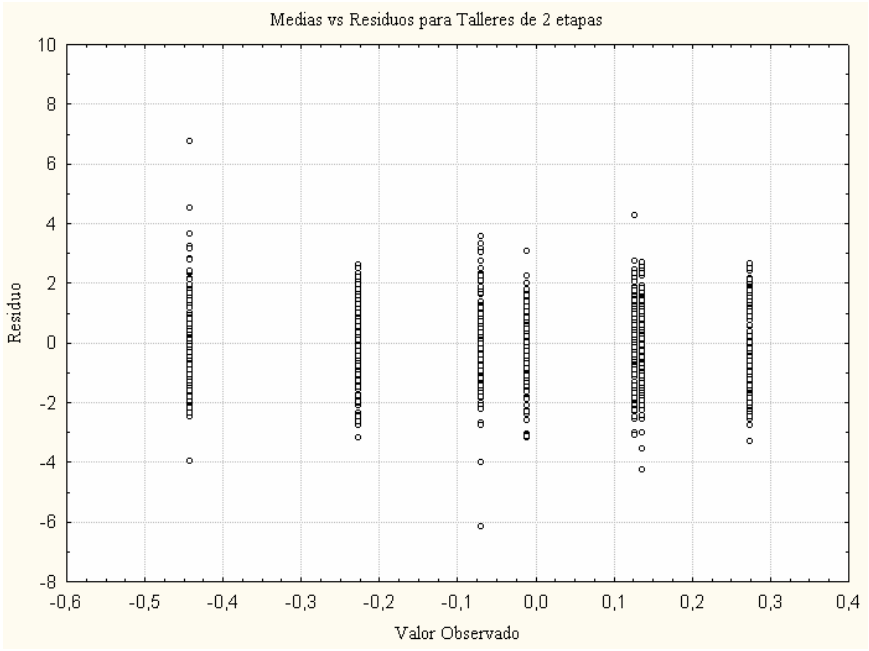


Figura 4.10 Medias de tratamiento contra residuo para Talleres de dos etapas.

Los residuos de TFH TCP de dos etapas, cumplen con las tres condiciones de idoneidad del experimento con un solo factor, esto permite aplicar el análisis de varianza.

Los datos de la Tabla 4.5 corresponden al análisis de varianza ANOVA para el experimento con THF TCP de 2 etapas con $a = 7$ niveles, $a - 1 = (7 - 1) = 6$ grados de libertad, $N = 1260$ observaciones, 95 grados de libertad para el error y un $\alpha = 0.05$ (95% de nivel de confianza).

Tabla 4.5. Valores del Análisis de varianza ANOVA con un nivel de confianza del 95% para THF TCP de 2 etapas

Fuente de Variación	Grados de Libertad	$C_{\max}$ SS	$C_{\max}$ MS	$C_{\max}$ f_0	$C_{\max}$ p
Intersección	1	1,187	1,18682	0,717545	0,397112
OC	6	64,016	10,66929	6,450591	0,000001
Error	1253	2072,464	1,65400		
	1260	2136,480			

El valor que resulto de aplicar ANOVA fue $f_0 = 6{,}450591$, mientras el valor $f_{0.05,6,1253} = 2.10$, como $f_0 > f_{0.05,6,28218953}$, por lo tanto se rechaza la hipótesis nula, es decir, que los operadores de cruzamiento si afectan la obtención de $C_{\max}$.

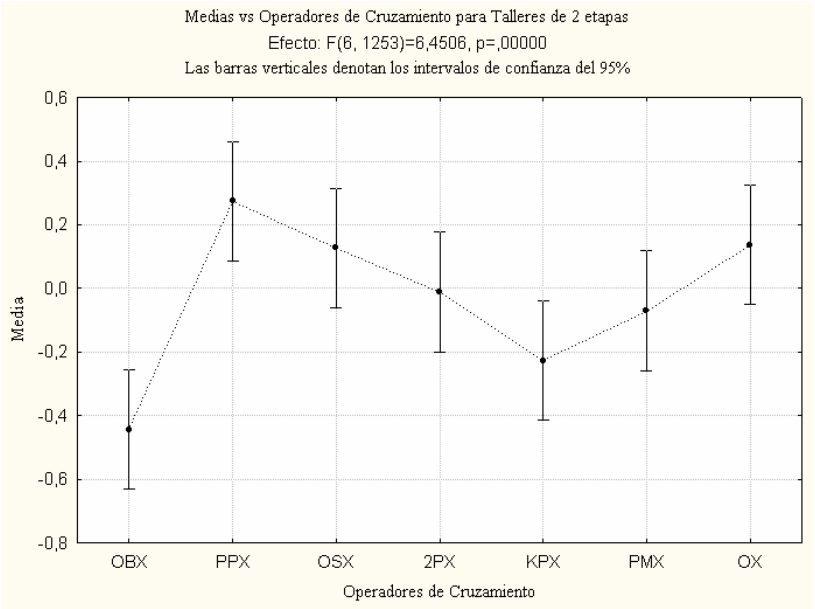


Figura 4.11 Medias de los niveles de tratamiento para Talleres de dos etapas.

Para saber cual de los OC es el que genera valores más pequeños de $C_{\max}$, se calculan las medias como se observa en la Figura 4.11, OBX es el operador que posee el valor medio más pequeño, es decir, este es el mejor para THF TCP con 2 etapas.

4.5.2 Análisis de Talleres de tres etapas

Se obtuvieron los siguientes resultados del análisis estadístico de los valores de $C_{\max}$ arrogados por el AG para TFH TCP de tres etapas. Las Figuras 4.12-4.16 muestran las gráficas del análisis. Como se observa en la Figura 4.12, los residuos se aproximan en su parte central hacia la recta lineal, implicando con ello la suposición de normalidad

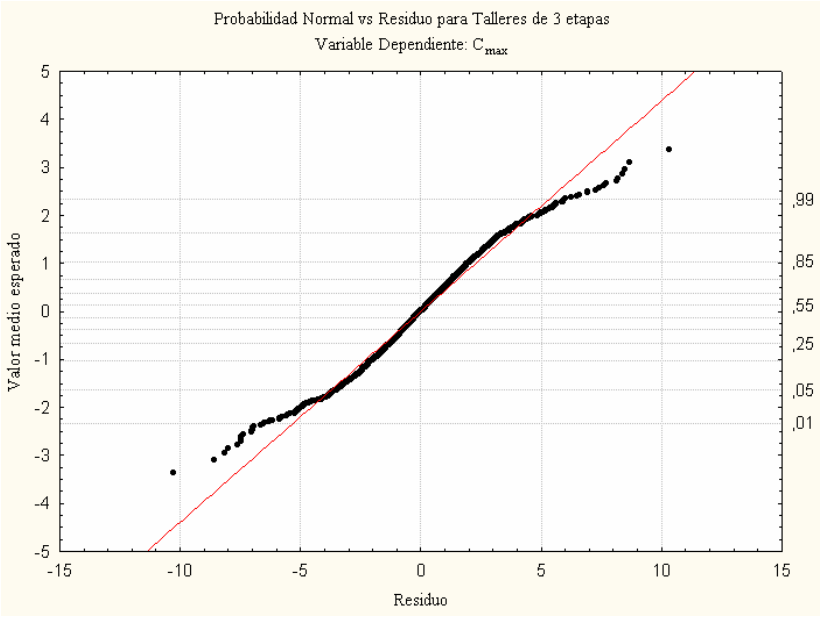


Figura 4.12 Gráfica Probabilidad normal de residuos para Talleres tres etapas

La gráfica de la Figura 4.13 se observa que los residuos están fluctuando en ambos lados con respecto a $C_{\max}$, y la desviación no es excesivamente grande.

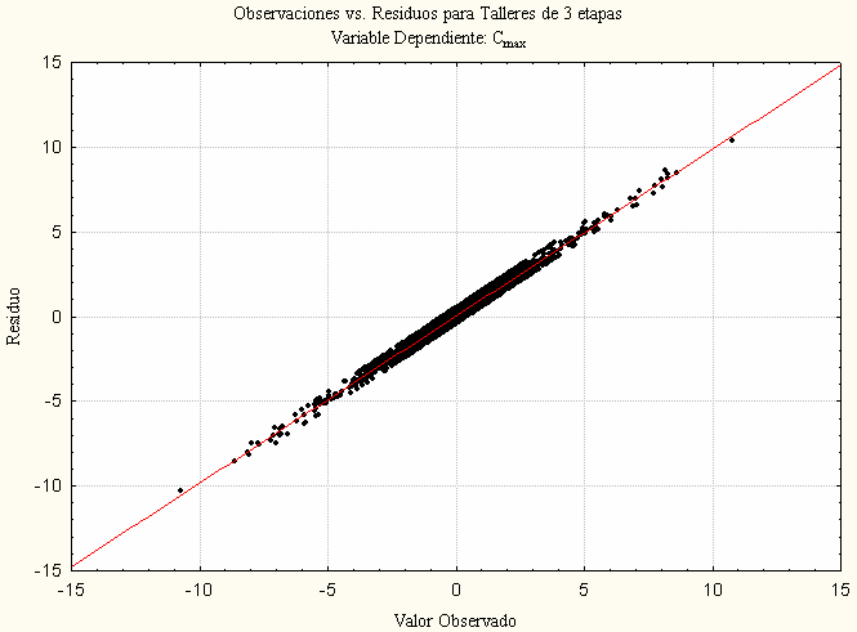


Figura 4.13 Valores observados contra los residuos para Talleres de tres etapas

En la Figura 4.14 se observa la existencia de la independencia de los residuos, ya que la gráfica no presenta un patrón definido en la distribución de los mismos, fluctuando entre valores positivos y negativos a lo largo de la gráfica.

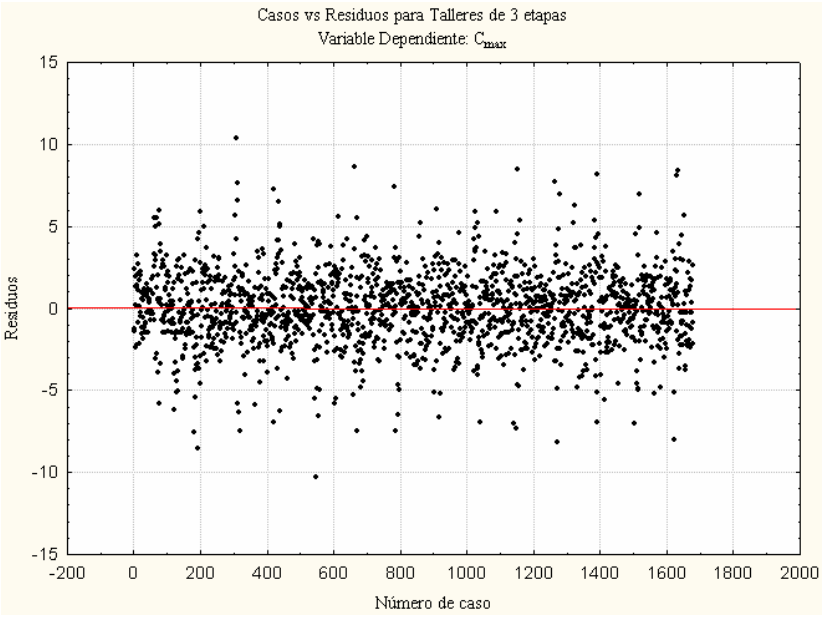


Figura 4.14 Número de casos contra residuos para Talleres de tres etapas

En la Figura 4.15 se observa la variabilidad de los residuos con respecto al único factor OC, se visualiza que no existe un patrón definido de dicha variabilidad con respecto a cada nivel del factor.

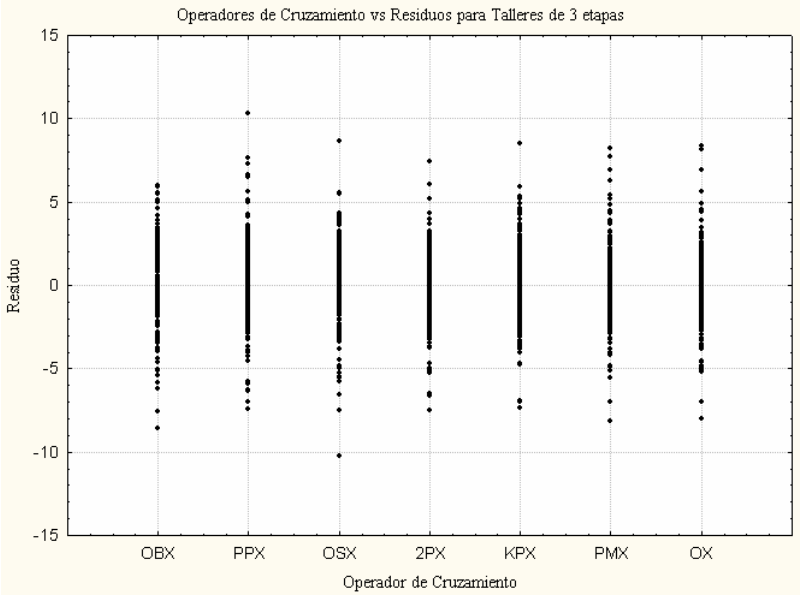


Figura 4.15 Nivel de factor contra residuos para Talleres de tres etapas

Así mismo en Figura 16 se muestra la dispersión de los residuos con respecto a su valor medio, se reafirma que no existe un patrón definido entre media y residuos

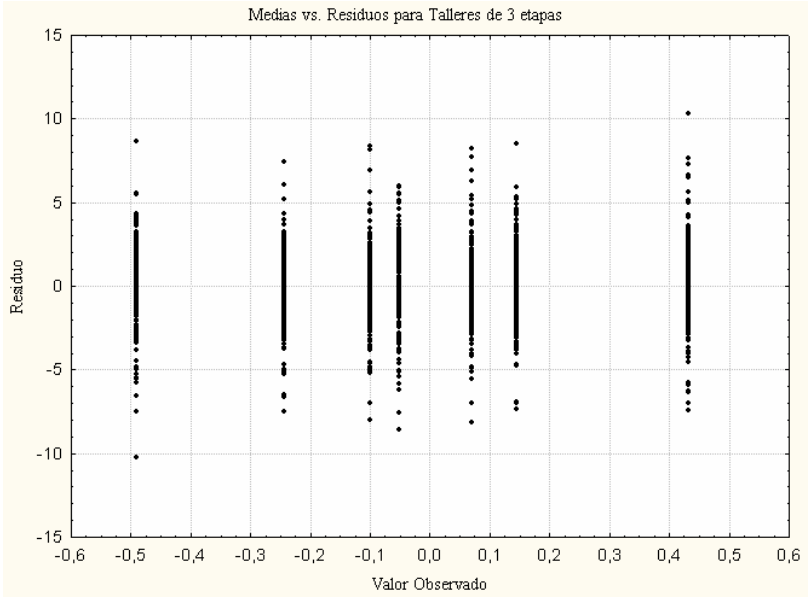


Figura 4.16. Medias de tratamiento contra residuo para Talleres de tres etapas.

Los residuos de TFH TCP de 3 etapas, cumplen con las tres condiciones de idoneidad del experimento con un solo factor, esto permite aplicar el análisis de varianza.

Los datos de la Tabla 4.6 corresponden al análisis de varianza ANOVA para el experimento con THF de 3 etapas con $a = 7$ niveles, $a - 1 = (7 - 1) = 6$ grados de libertad, $N = 1680$ observaciones, 95 grados de libertad para el error y un $\alpha = 0.05$ (95% de nivel de confianza).

El valor que resulto de aplicar ANOVA fue $f_0 = 4.053102$, mientras el valor $f_{0.05,6,1671} = 2.10$, como $f_0 > f_{0.05,6,28218953}$, por lo tanto se rechaza la hipótesis nula, es decir, que los operadores de cruzamiento si afectan la obtención de $C_{\max}$.

Tabla 4.6 Valores del análisis ANOVA con un nivel de confianza del 95% para THF TCP de 3 etapas

Fuente de Variación	Grados de Libertad	C_{max} SS	C_{max} MS	F	p
Intersección	1	1,919	1,91882	0,376726	0,539444
OC	6	123,865	20,64413	4,053102	0,000486
Error	1671	8511,095	5,09341		
Total	1678	8634,950			

El análisis de las medias de los OC, se observa en la Figura 4.17 que OSX_TCP es el operador que permite generar valores menores de $C_{\max}$.

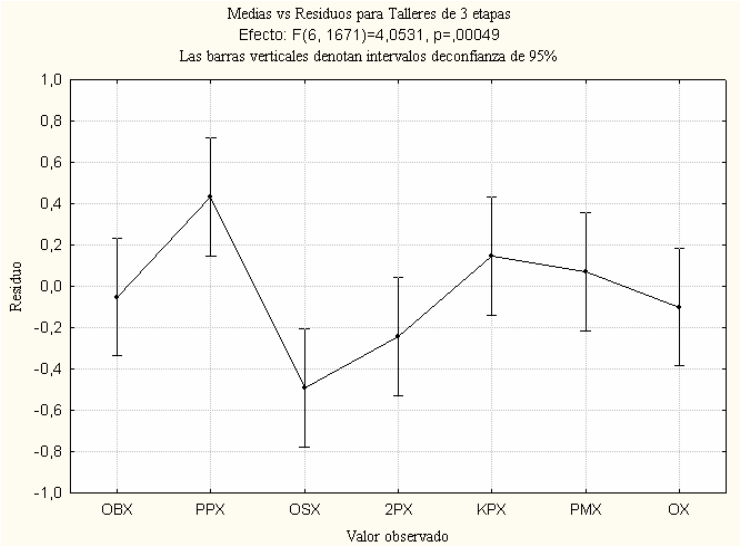


Figura 4.17. Gráfica de medias de los OC para THF TCP de 3 etapas

4.6. Conclusiones del capítulo

El análisis del método que tradicionalmente se aplica para el tratamiento de los valores de $C_{\max}$ para TFH TCP, indica que no es el adecuado para las distribuciones de las muestras de este taller.

La experiencia adquirida con esta investigación, permite concluir que fijando el número de etapas y asignando distintas distribuciones de los recursos, los valores de $C_{\max}$ para cada una de las muestras son heterogéneos.

Con el propósito de aplicar un tratamiento adecuado para estas muestras, se crea un nuevo método para homogenizar los valores de las muestras, el cual permite normalizarlos y aplicar los análisis estadísticos, residual y varianza ANOVA.

Este método se ha denotado como $\overline{C}^{*}_{\max}$, el cual indica el tratamiento de los datos bajo el criterio de usar el valor medio para cada observación y niveles de tratamiento. Este

hecho es muy importante, ya que permite agrupar muestras con datos heterogéneos y analizar como afectan los componentes de un AG a los valores de $C_{\max}$.

Particularmente los resultados del análisis con este nuevo tratamiento, permite analizar los valores de $C_{\max}$ para el problema $FH(3),(RM^{(i)})_{i=1}^{3}|s_{ij}|C_{\max}$, obtenidos a través de un AG con el propósito de verificar cual OC permite obtener un valor menor de $C_{\max}$. Los resultados indican que el factor OC de un AG, si afecta a la obtención del valor mínimo, corroborando que los OC son una parte delicada de los AG. Específicamente se encontró que para TFH TCP de dos etapas, el operador OBX resulta ser el más eficiente. Por el contrario para TFH TCP de tres etapas, el operador OSX resulta ser mejor.

CAPÍTULO 5

CONCLUSIONES Y TRABAJOS FUTUROS

El problema que se aborda en la presente investigación es considerado como uno de los más complejos de la programación de la producción, se presenta frecuentemente en industrias como la electrónica, textil, cerámica entre otras.

En la revisión del estado del arte, aparece la publicación de S.M. Johnson (1954) como el primer investigador que propone una solución óptima al problema de procesar un conjunto de trabajos en dos máquinas sucesivas. Cuando el número de máquinas es igual o mayor a 3, su complejidad se clasifica como tipo NP-duro. A partir de esta publicación se desarrollan diversas investigaciones para resolver el problema. Sin embargo, debido a su complejidad se han creado métodos que solo aproximan la solución óptima.

Los problemas han evolucionado en complejidad, respecto a la disposición física de los recursos, las restricciones de procesamiento de los trabajos y el establecimiento de la programación de los mismos. En consecuencia los métodos y técnicas para su resolución también han evolucionado. Esto implica que no todos los métodos creados sirven para atacar los problemas actuales o los que surjan en el futuro, habiendo un campo fértil de investigación teórica para la búsqueda y creación de nuevos métodos de resolución.

Algunos autores afirman que existe un desarrollo limitado en cuanto a la unificación de la teoría y la práctica en la programación de la producción.

A través del trabajo realizado en búsqueda bibliográfica para alcanzar el objetivo de esta investigación, que es analizar los métodos y algoritmos de resuelven problemas de programación de trabajos para distintos modelos de THF TCP, se encontró en la literatura que los métodos que se aplican para resolver este problema son del tipo evolutivo tales como algoritmos genéticos e inmunes. Así pues, para la realización del experimento dentro de esta investigación se considera un algoritmo genético, al estudiar los componentes de dicho algoritmo se encuentra que cada componente influye en la obtención de las

soluciones, dichas soluciones se presentan como el tiempo que tarda una determinada secuencia (permutación) de trabajos en ser procesada en un taller tal como THF TCP.

El algoritmo genético contenía los métodos de selección, de operadores de cruzamiento y de métodos de mutación, mediante los cuales las permutaciones son modificadas. En la literatura se considera que los operadores de cruzamiento son los que afectan directamente a la obtención de las soluciones, ya que trabajan sobre las posiciones de la permutación, realizando distintos reacomodos con el fin de encontrar aquella que cumpla con el criterio de optimización especificado con antelación como el tiempo mínimo de duración de procesamiento de todos los trabajos en un taller, $C_{\max}$ óptimo.

Por tanto se consideró el análisis del comportamiento de los operadores de cruzamiento dentro de un algoritmo genético, cabe indicar que este tipo de análisis no existe en la literatura hasta el momento.

Específicamente se analizó el comportamiento de siete operadores de cruzamiento, dentro del algoritmo genético, mediante la resolución el problema $FH(3),(RM^{(i)})_{i=1}^{3}\left|s_{ij}\right|C_{\max}$, donde los restantes componentes del algoritmo genético, como mutación y selección se dejaron fijos. La probabilidad de mutación que se consideró fue del 1% usando el método de inserción, el método de selección fue aleatorio. La probabilidad de cruzamiento fue del 90% para cada uno de los siete operadores OBX_TCP, PPX_TCP, OSX_TCP, 2PX_TCP, KPX_TCP, PMX_TCP y OX_TCP.

Los resultados de $C_{\max}$ por el algoritmo genético con los distintos operadores seleccionados, llevaron a concluir que:

- Los operadores si afectan a la obtención de la solución.
- El algoritmo OBX_TPC es el que mejor aproxima a la solución para $FH(2),(RM^{(i)})_{i=1}^{3}\left|s_{ij}\right|C_{\max}$ (dos etapas).
- El algoritmo OSX_TPC es el que mejor aproxima a la solución para $FH(3),(RM^{(i)})_{i=1}^{3}\left|s_{ij}\right|C_{\max}$ (tres etapas).
- El algoritmo que menos aproxima a la solución es PPX_TPC hecho que coincide en ambos problemas.

- Los restantes algoritmos tienen una solución aceptable, es decir, no se alejan mucho de los valores de $C_{\max}$ mínimos que brindan los mejores operadores en cada problema.
- Este análisis brinda información sobre los operadores de cruzamiento que lleven al algoritmo genético a óptimos locales.

Las conclusiones anteriores se obtuvieron una vez realizado el experimento, cabe indicar que estás se consideraron después del siguiente hallazgo.

Al trabajar con los resultados de $C_{\max}$ que se obtuvieron de la aplicación del algoritmo a los distintos talleres, se observó que los datos resultantes eran heterogéneos, es decir, si se cambia la configuración de los talleres de dos a tres etapas, los valores de $C_{\max}$ que se obtienen no poseen un comportamiento normal, además no guardan un comportamiento semejante en las distintas distribución de los valores de $C_{\max}$ e incluso, al agrupar los datos de los talleres con distintas configuraciones, estos no poseen un comportamiento normal. Con lo cual se confirma la necesidad de crear teoría nueva sobre el tratamiento de los datos obtenidos, teoría que sea aplicable a los nuevos problemas que surgen de la programación de la producción.

Por lo antes expuesto, se concluye que la técnica de tratamiento de los datos para TFH con distintas configuraciones, que tradicionalmente se aplica, no es la adecuada para el comportamiento estocástico que estos presentan. En consecuencia se crea un nuevo tratamiento denotado por $\overline{C}^{*}_{\max}$, el cual permite homogenizar los valores de muestras aleatorias heterogéneas utilizando el valor medio como se expuso anteriormente.

Como resultado de esta investigación, se recomienda continuar con el análisis de los métodos actuales, para contribuir a mejorar su funcionamiento y lograr el propósito más común en la programación de la producción tener un calendario optimo según los criterios establecidos en los problemas. En consecuencias, se consideran como investigaciones futuras:

- el análisis de la contribución que tienen sobre el algoritmo genético los componentes mutación y selección, de manera individual como un solo factor.

- El análisis de cruzamiento, mutación y selección como un problema de 3 factores.

Ambas investigaciones tienen el objetivo de mejorar las aproximaciones a la solución de problemas como $FH(3),(RM^{(i)})_{i=1}^{3}|s_{ij}|C_{\max}$.

Se recomienda como un tópico de importancia el desarrollo de investigaciones sobre las técnicas para tratamientos de los resultados obtenidos de los algoritmos.

REFERENCIAS

Alcaraz, J., Maroto, C., & Ruiz, R. (2003). Solving the Multi-Mode Resource-Constrained Project Scheduling Problem with Genetic Algorithms. *Journal ofthe Operational Research Society*, 54, 614-626.

Adler, L., Fraiman, N., Kobacker, E., Pinedo, M., Plotnicoff, J. C., & Wu, T. P. (1993). BPSS: A Scheduling Support System for the Packaging Industry. *Operations Research*, 41(4), 641–648.

Aghezzaf, E. H., Artiba, A., Moursli, O., & Tahon C. (1995). Hybrid flowshop problems, a decomposition based heuristic approach. *En Proceedings of the International Conference on Industrial Engineering and Production Management, IEPM'95* (pp. 43-56) Marrakech. FUCAM - INRIA.

Alcaraz, J., Maroto, C., & Ruiz, R. (2003). Solving the Multi-Mode Resource-Constrained Project Scheduling Problem with Genetic Algorithms. *Journal ofthe Operational Research Society*, 54, 614-626.

Allahverdi, A., Gupta, J. N. D., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *OMEGA, The international Journal of Management Science*, 27, 219–239.

Andrés, C. (2001). *Programación de la Producción en Talleres de Flujo Híbridos con Tiempos de Cambio de Partida Dependientes de la Secuencia. Modelos, Métodos y Algoritmos de Resolución. Aplicación a Empresas del Sector Cerámico*. Tesis Doctoral, Departamento de Organización de Empresas. Universidad Politécnica de Valencia.

Andrés, C, Gómez, P. (2003) Análisis comparativo del rendimiento de reglas de despacho tradicionales en un taller de flujo híbrido con tiempos de cambios de partida dependientes de la secuencia en un entorno dinámico aplicado a la industria cerámica. 27 Congreso Nacional de Estadística e Investigación Operativa Lleida: 8-11

Andrés, C., Albarracín, J.M., Tormo, G., Vicens, E., & García-Sabater, J.P. (2005). Group technology in a hybrid flowshop environment: A case study. E*uropean Journal of Operational Research*, 167, 272-281.

Arthanary, L.S., & Ramaswamy, K.G. (1971). An Extension of Two Machine Sequencing Problem. *OPSEARCH, The Journal of the Operational Research Society of India*, 8(4),10-22.

Belkadi K., Gourgand M., and Benyettou M. (1990). Parallel genetic algorithms with Migration for the hybrid flow shop scheduling problem. Journal of Applied Mathematics and Decision Sciences, Article ID 65746, 17 pags.

Bellman R. (1957). Dynamic Programming. Princeton, New Jersey: Princeton University Press, 366p

Bellman R. Esogbue AO and Nabeshima I (1982). Mathematical Aspects of Scheduling and Applications, Pergamon Press, New York.

Brah SA, Hunsucker JL (1991). Branch and bound algorithm for the flow shop with multiple processors. Eur J Oper Res 51(1): 88–99.11.

Brah, S. A., & Loo, L. L. (1999). Heuristics for scheduling in a flow shop with multiple processors. *European Journal of Operational Research*, 113, 113-122.

Botta-Genoulaz V (2000). Hybrid flow shop scheduling with precedence constraints and time lags to minimize maximum lateness. Int J Production Econom 64: 101–111.

Chen Lu, XI Li-feng, CAI Jian-guo, BOSTEL Nathalie, DEJAX Pierre (2006). An integrated approach for modeling and solving the scheduling problem of container handling systems. Journal of Zhejiang University SCIENCE 7(2):234-239

Cheng R, Gen M, Tozawa T (1995). Minmax earliness/tardiness scheduling in identical parallel machine system using genetic algorithms. Comput Ind Eng 29(1–4): 513–517.

Cheng TCE, Gupta JND, Wang G (200). Production and Operations Management, 2000. A Review of Flowshop Scheduling Research with Setup Times. IIE Transactions, 36(1):11 - 17

Cheng TCE, Z Liu - Operations Research Letters, (2003). Elsevier Approximability of two-machine no-wait flowshop scheduling with availability constraints Operations Research Letters. 31(4) :319-322.

KH Ecker, JND Gupta (2003) Scheduling tasks on a flexible manufacturing machine to minimize tool change delays. European Journal of Operational Research, Elsevier, 164(3): 627-638

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Reading, Massachusetts: Addison-Wesley, 432p.

Gourgand, M., Grangeon, N., & Norre, S. (1999). Metaheuristics for the deterministic hybrid flow shop problem. Proceedings of the International Conference on Industrial Engineering and Production Management, IEPM'99: 136–145

Gupta, J. N. D. (1971). M-stage scheduling problem—a critical appraisal . International Journal of Production Research, 9(2): 267 - 281

Gupta, J. N. D. (1972). Heuristic algorithms for multistage flowshop scheduling problem. AIIE Transactions, 4 (1), 11-18.

Gupta, J. N. D. (1975). A Search Algorithm for the Generalizad Flowshop Schedulig Problem. Computers and Operations Research, 2:83-90.

Gupta. J. N. D. (1986) y Darrow, W.O.(1986). The two-machines sequence dependent flowshop scheduling problem. European Journal of Operational Reserach, 24:243-446

Gupta, J. N. D. (1988). Two-Stage, Hybrid Flowshop Scheduling Problem. *Journal of the Operational Research Society*, 39(4): 359-364.

Gupta, J. N. D., & Tunc, E. A. (1991). Schedules for a two-stage hybrid flowshop with parallel machines at the second stage. *International Journal of Production Research*, 29(7):1489–1502.

Gupta JND, Tunc EA (1994). Scheduling a two-stage hybrid flowshop with separable setup and removal times. Eur J Oper Res 77(3): 415–428.

Gupta, J. N. D., Hariri, A. M. A., & Potts, C. N. (1997). Scheduling a two-stage hybrid flow shop with parallel machines at the first stage. *Annals of Operations Research*, 69: 171-191.

Gupta, J. N. D., & Tunc, E. A. (1998). Minimizing tardy jobs in a two-stage hybrid flowshop. *International Journal of Production Research*, 36(9), 2397-2417.

Gupta, J. N. D., Kruger, K., Lauff, V., Werner, F., & Sotskov, Y. N. (2002). Heuristics for hybrid flow shops with controllable processing times and assignable due dates.*Computers & Operations Research*, 29, 1417-1439.

Gen M, Cheng R (1997). Genetic algorithms and engineering design. Willy, New York.

Gen M, Cheng R (2000). Genetic algorithm & engineering optimization. Willy, New York.

Szwarc W. y Gupta J. N. D. (1987): "A flor shop problems with sequence dependent additive setup times", Naval Research Quarterly, 23, 619-627.

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor: Universityof Michigan Press, 228p.

Hoogeveen JA, Lenstra JK and Veltman B (1996). Pre-emptive scheduling in a two-stage multiprocessor flow-shop is NP-hard. Eur J Opl Res 89: 172–175.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly, 1, 61-68.

Jones DF, Mirrazavi SK, Tamiz M (2002). Multi-objective metaheuristics: An overview of the current state-of-art. Eur J Oper Res 137(1): 1-9.

Lawler,E.L., Lenstra,J.K. y Rinnoy Kan, A.H.G. (1993). Sequencing and Scheduling: Algorithms and Complexity. En Graves, S.C. Rinnony Kan, A.H.G., y Zipkin, P.H., editores. Logistics of Production and Inventory, Vol 4 de Handsbooks in Operations Reseach and Management Science. Amsterdan. Elsevier Science Publishers, B.V.

Leyva, E., Burtseva, L., & Yaurima, V. (2007). El Concepto de Dicotomía y su uso en Matemáticas. IV Foro Nacional "La problemática en el aprendizaje de las ciencias básicas". Instituto Tecnológico de Mexicali, SEP.

Linn R and Zhang W (1999). Hybrid flow shop scheduling: a survey. Comput Ind Eng 37: 57–61.

Moursli O, Pochet Y (2000). Branch and bound algorithm for the hybrid flowshop. Int J Prod Econ 64(1–3): 113–125.

Oğuz C, ErcanMF, Cheng TCE y Fung YF (2003). Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow-shop. Eur J Opl Res 149: 390–403.

Oğuz C, Zinder Y, Ha Do V, Janiak A, M (2004). Hybrid flow-shop scheduling problems with multiprocessor task systems European Journal of Operational Research, Elsevier.

Palmer, D. S. (1965). Sequencing Jobs Through a Multi-Stage Process in the MinimumTotal Time - A Quick Method of Obtaining a Near Optimum. Operational Research Quarterly, 16, 101-107.

Pinedo, M., 2002. Scheduling Theory, Algorithms, and Systems, 2nd Edition. Prentice Hall, New Jersey.

Portmann MC, Vignier A, Dardilhac D, Dezalay D (1998). Branch and bound crossed with GA to solve hybrid flowshops. Eur J Oper Res 107(2):389–400.

Parthasaraty S. y Rajendran C. (1997): "A simulated annealing heuristic for scheduling to minimize mean weighted tardiness in a flowshop with sequence dependent setup times of jobs: a case study", Production Planning and Control, 8: 475-483.

Riane, F., Artiba, A., & Elmaghraby, S. E. (1998). A hybrid three-stage flowshop problem: Efficient heuristics to minimize makespan. European Journal of Operational Research, 109, 321-329.

Riane F and Artiba A (1999). Scheduling multistage flow-shop problem: a brief review. In: Proceedings of the International Conference on Industrial Engineering and Production Management, Glasgow, ISBN 2-930294-02-7, Volume 2. Faculte´ s Universitaires Catholiques de Mons, Mons, Belgium, pp 323–335.

Reeves CR (1995). A genetic algorithm for flowshop sequencing. Comput Oper Res 22(1): 5-13.

Ríos-Mercado, R. Z. y Bard, J. (1998a). Computational Experience with a Branchand- Cut Algorithm for Flowshop Scheduling with Setups. Computers and Operations Research, 25(5):351–366.

Romero, R. (2007). Sistema Computacional para la evaluación de algoritmos de Planificación de Trabajos en un Taller de Flujo Híbrido. Tesis de Maestría, Facultad de Ingeniería. Universidad Autónoma de Baja California. México.

Romero, R. (2007). *Sistema Computacional para la evaluación de algoritmos de Planificación de Trabajos en un Taller de Flujo Híbrido*. Tesis de Maestría, Facultad de Ingeniería. Universidad Autónoma de Baja California. México.

Ruiz, R. (2003). Técnicas Metaheurísticas para la Programación Flexible de la Producción. Tesis doctoral. Departamento de Estadística e Investigación Operativa Aplicadas y Calidad. Universidad Politécnica de Valencia.

Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. European Journal of Operational Research, 169, 781-800.

Ruiz, R., Serifoğlu, F. S., & Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. Computer Operations Research, 35(4), 1151-1175.

Ruíz R, Maroto C, Alcaraz J (2005) Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. Eur J Oper Res 165(1): 34–54.

Salvador MS (1973). A solution to a special case of flow shop scheduling problems. in: Elmaghraby SE (ed.), Symposium of the Theory of Scheduling and Applications. Springer, New York: 83–91.

Santos DL, Hunsucker JL, Deal DE (1996). An evaluation of sequencing heuristics in flow shops with multiple processors. Comput Ind Eng 30(4): 681-691.

Sivrikaya Serifoğlu F and Tiryaki IU (2002). Multiprocessor task scheduling in multistage hybrid flow-shops: A simulated annealing approach. In: Baykasoglu A and Develi T (eds). Proceedings of the 2nd International Conference on Responsive Manufacturing. University of Gaziantep Printing Office, Gaziantep, Turkey, pp 270–274.

Sivrikaya F, Serifoğlu and G Ulus (2004). Multiprocessor task scheduling in multistage hybrid flow-shops: a genetic algorithm approach. Journal of the Operational Research Society 55, 504–512

Sriskandarajah C, Sethi SP (1989) Scheduling algorithms for flexible flowshops: worst case and average case performance. Eur J Oper Res 43(2): 143–160.

Szwarc W. y Gupta J. N. D. (1987): "A flow-shop problems with sequence dependent additive setup times", Naval Research Quarterly, 23, 619-627.

Taillard E. Benchmarks for basic scheduling problems, ORWP89/21 Dec. 1989.

Voß Stefan (1993). Steiner's problem in graphs: heuristic methods. *Discrete Applied Mathematics*, *Volume 40, Issue 1*, *18 November 1992*, *Pages 45-72*

VoB, S., & Witt, A. (2005). Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: A real-world application. International Journal Production Economics. In Press.

Yaurima V., Romero R., Burtseva L., & Valle Y. (2006). Aplicación del Método de Dicotomía para Optimización Combinatoria. XXVIII Congreso Internacional de Ingeniería Electrónica - ELECTRO. IT Chihuahua, Creel, Chih., México: 283-288

Yaurima, V., Burtseva, L., & Tchernykh, A. (2007). Hybrid Flowshop with Unrelated Machines, Sequence Dependent Setup Time and Availability Constraints: An Enhanced Crossover Operator for a Genetic Algorithm. In: Wyrzykowski et al. (Eds.), Parallel Processing and Applied Mathematics, LNCS 4967, Springer-Verlag, (in press).

Zandieh M., Fatemi Ghomi S.M.T., Moattar Hisseni S.M.(2006). An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. Elsiver Applied Mathematics and Computation: 111-127.

PRODUCTOS GENERADOS
EN EL DESARROLLO DE LA TESIS

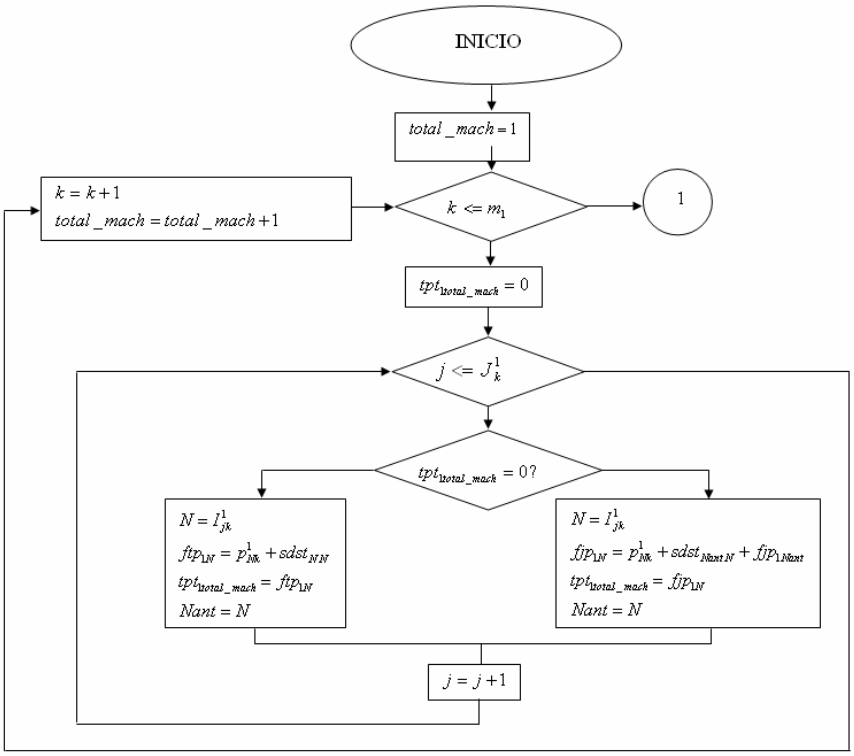
- Leyva, E., Burtseva, L., & Yaurima, V. (2007). El Concepto de Dicotomía y su uso en Matemáticas. IV Foro Nacional “La problemática en el aprendizaje de las ciencias básicas”. Instituto Tecnológico de Mexicali, SEP.

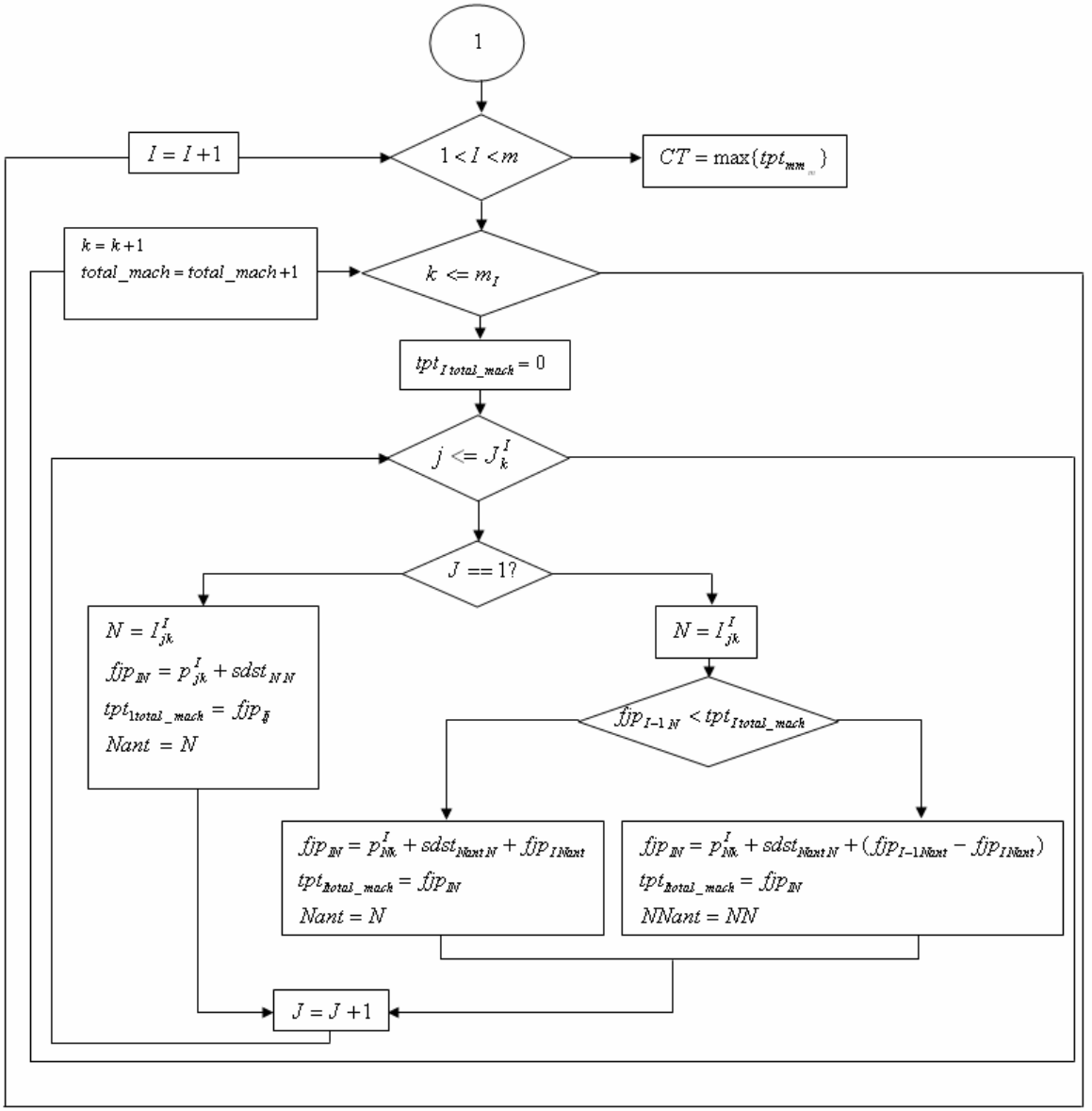
ANEXOS

ANEXO 1 Diagrama de flujo del algoritmo que calcula el tiempo de terminación de todos los trabajos para Talleres de Flujo Híbrido de múltiples etapas y que incluyen los Tiempos de Cambio de Partida.

El siguiente diagrama de flujo esquematiza el algoritmo que realiza el cálculo del tiempo de terminación de los *n* trabajos de un taller de flujo híbrido de *m* etapas. Se requieren las siguientes variables de entrada:

m número de etapas
n número de trabajos
m_i número de máquinas por etapa
p^i_{jk} tiempo de procesamiento de cada trabajo j en la máquina k de la etapa i.
I^i_{jk} índice de cada trabajo j en la máquina k de la etapa i
$sdst_{ij}$ tiempo de cambio de partida del trabajo i al trabajo j
J^i_k número de trabajos en cada máquina de la etapa m_i
tpt_{ik} tiempo de procesamiento acumulado en cada máquina k de la etapa i
fjp_{ij} tiempo de finalización de cada trabajo j en cada etapa i
$total\_mach$ número total de máquinas




1
$I = I + 1$
$1 < I < m$
$CT = \max\{tpt_{mm}\}$
$k = k + 1$
$total\_mach = total\_mach + 1$
$k <= m_I$
$tpt_{I\,total\_mach} = 0$
$j <= J_k^I$
$J == 1?$
$N = I_{jk}^I$
$fjp_{IN} = p_{jk}^I + sdst_{NN}$
$tpt_{1total\_mach} = fjp_{ij}$
$Nant = N$
$N = I_{jk}^I$
$fjp_{I-1\,N} < tpt_{I\,total\_mach}$
$fjp_{IN} = p_{Mk}^I + sdst_{Nant\,N} + fjp_{I\,Nant}$
$tpt_{Ilotal\_mach} = fjp_{IN}$
$Nant = N$
$fjp_{IN} = p_{Mk}^I + sdst_{Nant\,N} + (fjp_{I-1\,Nant} - fjp_{I\,Nant})$
$tpt_{Ilotal\_mach} = fjp_{IN}$
$NNant = NN$
$J = J + 1$

ANÉXO B. Ejemplo de datos de recursos y tiempos de procesamiento para un taller de 3 etapas y 50 trabajos, archivo Taller_3_2_1_2_50.

Vea CD Tesis: Archivo_ Taller_3_2_1_2_50.pdf

ANÉXO C. Ejemplo de valores de C_{max} para el taller de tres etapasy 50 trabajos, archivo Completion_Time_3_2_1_2_50

Vea CD Tesis: Archivo_ Completion_Time_3_2_1_2_50.pdf