UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA
DIVISION DE ESTUDIOS DE POSTGRADO

MODELO DE SOLUCION PARA EL PROBLEMA DE PROGRAMACION DE TAREAS EN DOS MAQUINAS POR MEDIO DEL METAHEURISTICO BUSQUEDA TABU

POR

ING. JOSE LUIS LIRA DE LA GARZA

TESIS

EN OPCION AL GRADO DE MAESTRO EN CIENCIAS
DE LA ADMINISTRACION CON ESPECIALIDAD
EN SISTEMAS

SAN NICOLAS DE LOS GARZA, N. L. FEBRERO DE 1999

MODELO DE SOLUCION PARA EL PROBLEMA DE PROGRAMACION DE TAREAS EN DOS MAQUINAS POR MEDIO DEL METAHEURISTICO BUSQUEDA TABU

JLLG

UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA
DIVISION DE ESTUDIOS DE POSTGRADO

MODELO DE SOLUCION PARA EL PROBLEMA DE PROGRAMACION DE TAREAS EN DOS MAQUINAS POR MEDIO DEL METAHEURISTICO BUSQUEDA TABU

POR

ING. JOSE LUIS LIRA DE LA GARZA

TESIS

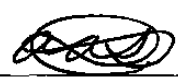
EN OPCION AL GRADO DE MAESTRO EN CIENCIAS
DE LA ADMINISTRACION CON ESPECIALIDAD
EN SISTEMAS

SAN NICOLAS DE LOS GARZA, N. L. FEBRERO DE 1999

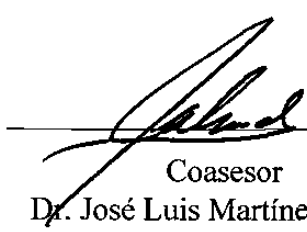
UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA
DIVISION DE ESTUDIOS DE POSTGRADO

Los miembros del comité de tesis recomendamos que la tesis MODELO DE SOLUCION PARA EL PROBLEMA DE PROGRAMACION DE TAREAS EN DOS MAQUINAS POR MEDIO DEL METAHEURISTICO BUSQUEDA TABU, realizada por el Ing. José Luis Lira de la Garza, sea aceptada para su defensa como opción al grado de Maestro en Ciencias de la Administración con especialidad en Sistemas.

El Comité de Tesis



Asesor
Dra. Ada Margarita Álvarez Socarrás



Coasesor
Dr. José Luis Martínez Flores

Coasesor
M. C. José Luis Castillo Ocañas

Vo. Bo.
M.C. Roberto Villarreal Garza
División de Estudios de Postgrado

San Nicolás de los Garza, N.L. a 15 de Febrero de 1999

DEDICATORIAS

A Dios . . .

GRACIAS . . . . . .

A mis Padres . . .

José Luis Lira Martínez y

Rosa María de la Garza de Lira.

A mis Hermanas . . .

María Luisa Lira de la Garza y

Dulce Rocío Lira de la Garza.

A mi Sobrino . . .

Erick Eduardo Sandoval Lira.

A todos los que han compartido algún momento de sus vidas conmigo, por insignificante que parezca, gracias, por que de no ser por ellos, no sería como soy...

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a la Dra. Ada Margarita Álvarez Socarrás, asesora de esta tesis, por sus consejos, paciencia y ayuda, para la terminación de la misma.

Al Dr. José Luis Martínez Flores y al M.C. José Luis Castillo Ocañas coasesores, por su valiosa ayuda e interés para el desarrollo y revisión de esta tesis.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por su apoyo económico para la realización de este proyecto. Al Doctorado en Ingeniería de Sistemas (DIS), a la Facultad de Ingeniería Mecánica y Eléctrica y a la Universidad Autónoma de Nuevo León por permitirme el uso del equipo e instalaciones.

A todos mis maestros que compartieron sus conocimientos y su experiencia, además de su amistad y consejo para mejorar tanto personal como profesionalmente.

Agradezco especialmente a mis compañeros y amigos por su amistad y compañía durante esta etapa de mi vida, y por dejarme compartir con ellos tantos momentos especiales que recordare por siempre.....

RESUMEN

José Luis Lira de la Garza **Fecha de Graduación: Febrero, 1999**

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Título del Estudio: **MODELO DE SOLUCIÓN PARA EL PROBLEMA DE PROGRAMACIÓN DE TAREAS EN DOS MAQUINAS POR MEDIO DEL METAHERÍSTICO BÚSQUEDA TABÚ.**

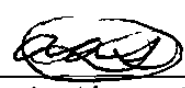
Número de Páginas: 71

Candidato para el grado de Maestría en Ciencias de la Administración con Especialidad en Sistemas

Área de Estudio: Sistemas

Propósito y Método del Estudio: El propósito principal de este estudio es solucionar el Problema de Programación de Tareas (PPT), el cual, en forma general, trata de lo siguiente: Se desea realizar una asignación de un conjunto de trabajos o tareas en una o varias máquinas en una secuencia óptima tal que se minimicen los costos de la secuencia de tareas programadas. El PPT se encuentra comúnmente en las empresas y se han realizados diversos estudios para tratar de resolverlo, pero aún no se ha encontrado un método que proporcione soluciones óptimas para este tipo de problemas, debido a que pertenece a la clase NP-completo. El presente estudio aborda el PPT en una máquina, después trata el de dos máquinas iguales y posteriormente se resuelve el PPT en dos máquinas diferentes. Para ello se utiliza un procedimiento metaheurístico llamado Búsqueda Tabú. Búsqueda Tabú es un procedimiento metaheurísto utilizado para guiar un algoritmo heurístico de búsqueda local para explorar el espacio de soluciones más allá de la simple optimilidad local. Los investigadores lo han utilizado como metaheurístico para optimización, debido a que pueden encontrar soluciones aproximadas a problemas de gran complejidad computacional. Por todo esto, se desarrolló e implementó computacionalmente un modelo para dar solución al PPT, con lo cual se observa la conveniencia de la utilización de búsqueda tabú para resolver este tipo de problemas.

Contribuciones y Conclusiones: El modelo se implementó. Se realizaron simulaciones con el modelo de solución y se obtuvieron resultados que muestran que, en general, las soluciones encontradas por el modelo implementado se consideran buenas con respecto a los resultados reportados en la literatura. Se concluye que es conveniente la utilización de búsqueda tabú para la solución de problemas como el PPT.

FIRMA DEL ASESOR: 

Dra. Ada Margarita Alvarez Socarrás

TABLA DE CONTENIDO

Capítulo **Página**

Capítulo **Página**

CAPÍTULO 1

INTRODUCCIÓN

1.1 Planteamiento del problema

La programación de trabajos o tareas, comúnmente llamada "scheduling", se encuentra fuertemente relacionada con los problemas que enfrentan las compañías manufactureras al querer optimizar la operación de sus líneas de producción, y por consiguiente minimizar sus costos. El proporcionar una solución óptima o muy cercana a la óptima a este problema, representa no solo una mejora en los procesos de producción sino también una utilización más eficiente de sus recursos.

La filosofía Justo a Tiempo (Just in Time, JIT) sugiere que tanto la anticipación como la tardanza deben ser considerados al evaluar la efectividad de un schedule, ya que si un trabajo es terminado antes de su fecha de entrega, se irá almacenando como inventario de producto terminado, hasta que se llegue la fecha acordada para su entrega, lo cual genera costos tales como almacenaje, manejo, etc. Por otra parte, si un trabajo se termina después de su fecha de entrega, podría ocasionar que el cliente suspenda su pedido, o bien, que fuera su último pedido en la compañía.

Por lo tanto, como se ha mencionado anteriormente, podría considerarse un schedule ideal, a aquel en el que todas las fechas de terminación de los trabajos coinciden exactamente con sus fechas de entrega programadas.

El problema que se desea abordar en esta tesis, consiste precisamente en tratar de aproximarse lo más posible a este schedule ideal, evaluando la efectividad del schedule en función del costo total incurrido, penalizando que un trabajo se adelante o se atrase de su fecha de entrega programada.

Debido a que existen una gran número de secuencias o schedules posibles, dado un conjunto de trabajos, el problema se complica y no es fácil de resolver por métodos tradicionales e incluso en ocasiones no es posible encontrar una solución óptima. Además, hay que recordar que se trata de un problema NP-completo. Por todo esto, es conveniente recurrir al uso de algoritmos más simples que proporcionan soluciones buenas y en relativamente poco tiempo, aunque no sean precisamente óptimas. Estos algoritmos son llamados heurísticos. Si el procedimiento utilizado para obtener una buena solución, tiene que estar diseñado especialmente para cada tipo de problema, entonces se trata de un metaheurístico.

Es precisamente un metaheurístico el que se utiliza como base para resolver el problema planteado en esta tesis, se trata de Búsqueda Tabú. Este método requiere de recursos computacionales mínimos, permitiendo de esta manera resolver problemas relativamente grandes en una microcomputadora.

1.2 Objetivo

Esta tesis tiene como objetivos:

- Diseñar e implementar computacionalmente un modelo de solución para el problema de programación de tareas en una máquina.

- Diseñar e implementar computacionalmente un modelo de solución para el problema de programación de tareas cuando se tienen disponibles dos máquinas idénticas.

- Diseñar e implementar computacionalmente un modelo de solución para el problema de programación de tareas cuando se tienen disponibles dos máquinas diferentes.

1.3 Formulación del problema

En su modelo mas sencillo, la programación de tareas (PPT) implica la asignación de trabajos o tareas a una máquina específica en una secuencia óptima, tal que se minimicen los costos de la secuencia de tareas programadas. Este problema puede complicarse tanto como se desee, o tanto como lo exija la situación real que se esté analizando.

El PPT que se aborda en esta tesis, independientemente de la cantidad y características de las máquinas, tiene las siguientes restricciones: Se tiene un conjunto de tareas que requieren ser procesadas, cada una de las tareas tiene una fecha de entrega que cumplir y un tiempo de procesamiento. Existe un tiempo y un costo de preparación de la máquina entre tareas, los cuales no dependen solamente de la tarea que se va a procesar, sino también de la que se procesó anteriormente. Además, como ya fue mencionado, se aplica un enfoque de JIT, en donde se penaliza tanto las entregas tardías como las

tempranas. Para ayudar a que las tareas se terminen lo más cerca posible a su fecha de entrega, se incluye también el manejo de tiempos de ocio entre cada par de tareas en caso de ser necesario.

El objetivo consiste en minimizar el costo total de la secuencia de tareas, el cual se obtiene a partir de los costos por concepto de preparación y por concepto de penalizaciones.

1.3.1 El caso de una máquina

El problema para el caso de una máquina se formula de la siguiente manera:
Se tiene un conjunto de n tareas $\pi_1 , \pi_2 , \ldots , \pi_n$ y se considera que una secuencia tiene la forma: $\prod = \{\pi_{\sigma 1} , \pi_{\sigma 2} , \ldots , \pi_{\sigma n} \}$, donde σi representan una permutación de los símbolos $1,2,\ldots,n$.

Para cada tarea π_i ($i = 1, 2, \ldots, n$) se tiene:

$p_{\pi i}$: tiempo de procesamiento.
$d_{\pi i}$: fecha o tiempo de entrega.
$\alpha_{\pi i}$: castigo por entrega prematura por unidad de tiempo.
$\beta_{\pi i}$: castigo por entrega tardía por unidad de tiempo.

Además también se tiene,

$sc(\pi_i , \pi_j)$: costo de preparación para cada par de tareas π_i , π_j.
$st(\pi_i , \pi_j)$: tiempo de preparación para cada par de tareas π_i , π_j.

Denotemos por:

$f_{\pi i}$: tiempo de entrega real, dado por los tiempos de procesamiento.

$e_{\pi i}$: cantidad de tiempo prematuro.

$t_{\pi i}$: cantidad de tiempo de retraso.

$to(\pi_i, \pi_j)$: tiempo de ocio para cada par de tareas π_i , π_j.

siendo $i, j = 1, 2, ..., n$.

Se asume lo siguiente:

- Todas las tareas están disponibles en el tiempo 0.
- Todas las tareas se pueden procesar sin que dependan del procesamiento previo de otra.
- $to(\pi_i, \pi_j) \geq 0$ para $i \neq j$.
- $sc(\pi_i, \pi_j) = 0$, $st(\pi_i, \pi_j) = 0$ y $to(\pi_i, \pi_j) = 0$ para $i = j$.
- $e_{\pi i} = \max(0, d_{\pi i} - f_{\pi i})$ y $t_{\pi i} = \max(0, f_{\pi i} - d_{\pi i})$.
- $\alpha_{\pi i} > 0$ y $\beta_{\pi i} > 0$.

Se desea encontrar una permutación $\prod$, tal que el costo de la secuencia de tareas sea mínimo, por consiguiente la función objetivo del problema para el caso de una máquina es:

$$\text{minimizar } f(\prod) = \sum_{i=1}^{n} [\, \alpha_{\pi i} * e_{\pi i} + \beta_{\pi i} * t_{\pi i} \,] + \sum_{j=2}^{n} [\, sc(\pi_{(j-1)}, \pi_j) \,]$$

1.3.2 El caso de dos máquinas

En la formulación del problema para el caso de dos máquinas se consideran dos variantes, que las máquinas sean idénticas o que sean diferentes. Ambos casos pueden considerarse una extensión del problema de una máquina.

En el caso de dos máquinas idénticas no existe variación significativa en comparación con el caso de una máquina, es decir, las características de las máquinas son las mismas, por lo tanto, los parámetros de las tareas no varían de una máquina a otra. En el caso de dos máquinas diferentes, sucede que los costos y tiempos de preparación, así como los tiempos de procesamiento de las tareas, varían de una máquina a otra. Esto se debe a que por diversas razones las máquinas tienen características diferentes entre sí.

El problema para el caso de dos máquinas se formula de la siguiente manera: Se tiene un conjunto de n tareas $\pi_1, \pi_2, \ldots, \pi_n$ y se considera que para cada máquina k la secuencia tiene la forma: $\prod^k = \{\pi^k_{\sigma 1}, \pi^k_{\sigma 2}, \ldots, \pi^k_{\sigma n}\}$, para $k = 1, 2$, donde $\pi_{\sigma i}$ denota la tarea que se realizará en la posición i.

Para cada tarea π_i ($i = 1, 2, \ldots, n$) se tiene:

$p^k_{\pi i}$: tiempo de procesamiento en la máquina k, $k = 1, 2$.
$d_{\pi i}$: fecha o tiempo de entrega.
$\alpha_{\pi i}$: castigo por entrega prematura por unidad de tiempo.
$\beta_{\pi i}$: castigo por entrega tardía por unidad de tiempo.

Además también se tiene,

$sc_k(\pi_i, \pi_j)$: costo de preparación de la máquina k para cada par de tareas π_i, π_j, $k = 1,2$.
$st_k(\pi_i, \pi_j)$: tiempo de preparación de la máquina k para cada par de tareas π_i, π_j, $k = 1,2$.

Denotemos por:

$f_{\pi i}$: tiempo de entrega real, dado por los tiempos de procesamiento.

$e_{\pi i}$: cantidad de tiempo prematuro.

$t_{\pi i}$: cantidad de tiempo de retraso.

$to(\pi_i, \pi_j)$: tiempo de ocio para cada par de tareas π_i , π_j.
siendo $i, j = 1, 2, ..., n$.

$\prod^1$: conjunto de tareas que se procesan en la máquina 1.

$\prod^2$: conjunto de tareas que se procesan en la máquina 2.

n1: cantidad de tareas que se procesan en la máquina 1.

n2: cantidad de tareas que se procesan en la máquina 2.

Se asume lo siguiente:

- Todas las tareas están disponibles en el tiempo 0.
- Todas las tareas se pueden procesar en cualquiera de las dos máquinas.
- Todas las tareas se pueden procesar sin que dependan del procesamiento previo de otra.
- $to(\pi_i, \pi_j) \geq 0$ para $i \neq j$.
- $sc_k(\pi_i, \pi_j) = 0$, $st_k(\pi_i, \pi_j) = 0$ y $to(\pi_i, \pi_j) = 0$ para $i = j$.
- $e_{\pi i} = \max(0, d_{\pi i} - f_{\pi i})$ y $t_{\pi i} = \max(0, f_{\pi i} - d_{\pi i})$.
- $\alpha_{\pi i} > 0$ y $\beta_{\pi i} > 0$.
- $\prod^1 \cup \prod^2 = \prod$.
- $\prod^1 \cap \prod^2 = \varnothing$.
- n1 + n2 = n.

Se desea encontrar una permutación de $\prod^1$ y otra permutación de $\prod^2$, tal que el costo total de la suma de las secuencias de tareas sea mínimo, por consiguiente la función objetivo del problema para el caso de dos máquinas es:

$$\text{minimizar } f(\prod) = \sum_{k=1}^{2} \sum_{i=1}^{nk} [\, \alpha_{\pi i} * e_{\pi i} + \beta_{\pi i} * t_{\pi i} \,] + \sum_{j=2}^{nk} [\, sc_k(\pi_{(j-1)}, \pi_j) \,]$$

Para el caso de dos máquinas idénticas $sc_1(\pi_i, \pi_j) = sc_2(\pi_i, \pi_j)$ y $st_1(\pi_i, \pi_j) = st_2(\pi_i, \pi_j)$, no cumpliéndose esto, para el caso de máquinas diferentes.

1.4 Metodología de solución

La tesis se enfoca en el problema de programación de tareas en dos máquinas, utilizándose como herramienta principal el metaheurístico Búsqueda Tabú. Se analiza el problema para los siguientes casos:

- El caso de una máquina.
- El caso de dos máquinas idénticas.
- El caso de dos máquinas diferentes.

Se toman en cuenta las siguientes restricciones:

- penalización por anticipación.
- penalización por tardanza.
- costo de preparación (setup cost).
- tiempo de preparación (setup time).
- tiempo de ocio (idle time).

También se hace la implementación computacional para la solución de los casos ya mencionados.

1.5 Revisión bibliográfica

En la literatura revisada se encontraron varios problemas de scheduling que han sido abordados. El documento más antiguo encontrado que aborda este tema es el escrito por Sidney [77]. Sidney demuestra en su escrito que el problema de minimizar la máxima penalización de un trabajo (debido a su retraso o a su anticipación), donde todos los trabajos tienen las mismas funciones de costo, tanto para la anticipación como para el retraso, es acotado polinomialmente. Además, considera la posibilidad de insertar tiempos de ocio. Sidney supone que la penalización ocurre cuando un trabajo comienza ya sea después de su fecha de inicio objetivo o bien, si termina después de su fecha de entrega, y se enfoca básicamente a minimizar la máxima penalización sujeta a ciertas suposiciones restrictivas en las fechas de inicio. Él desarrolló un algoritmo de complejidad $O(n^2)$, es decir, cuyo número de pasos es proporcional a n^2, para este problema y lo generalizó para cubrir el caso en el cual la máquina se encuentra disponible solamente por un período de tiempo limitado.

Lakshmanan, Papineau y Rochette [78], proporcionaron más tarde un algoritmo $O(nlogn)$, esto es, cuyo número de pasos es proporcional a $nlogn$, para este problema.

Seidman *et al* [82] y Panwalker *et al* [82] han incluido en sus trabajos la determinación de las fechas de entrega como parte del problema de scheduling. Seidman et al. determinan tanto las fechas de entrega como la secuencia óptima. Ellos buscan minimizar una función de penalización total, definida como la suma de la penalización por tiempo de entrega (lead time penalty), la cual depende de la fecha de entrega asignada; la penalización por anticipación (earliness penalty) y la penalización por tardanza (tardiness penalty). Se muestra además que la regla de secuenciación basada en el tiempo de procesamiento más corto es óptima para este problema y que es fácil calcular las fechas de entrega óptimas.

Panwalker et al. consideran un problema similar, sólo que suponen una fecha de entrega común a todos los trabajos y un penalización lineal. Para dar solución a este problema elaboran un algoritmo óptimo *O(nlogn)*.

Kanet [81] estudió el problema de programar *n* trabajos con una fecha de entrega en común, con el objeto de minimizar el retraso absoluto en una sola máquina. Kanet proporciona un algoritmo polinomialmente acotado para este problema. Además considera la posibilidad de insertar tiempos de ocio en el schedule.

Chen [97] considera un modelo en donde varios lotes de trabajos tienen que ser procesados en una máquina. Se abordan dos problemas, en el primero se minimizan las penalizaciones por entregas tardías y tempranas, y en el segundo se minimizan, además de lo anterior, la penalización provocada por la fecha de entrega global. Propone un algoritmo polinomial de programación dinámica para resolver el problema, con sólo dos lotes de trabajos, en donde los trabajos de cada lote tienen las mismas fechas de entrega y por consiguiente, las mismas penalizaciones. También considera un caso especial para ambos problemas, en donde las fechas de entrega para los diferentes lotes son iguales. Bajo este caso especial, propone un algoritmo de programación dinámico para resolver el primer problema con una fecha de entrega irrestrictamente grande y para resolver el segundo problema. Este algoritmo corre en un tiempo polinomial respecto al número de trabajos, pero en un tiempo exponencial respecto al número de lotes.

El caso más general del problema de minimización de la máxima penalización por retraso y anticipación, es aquél en el cual se consideran fechas de entrega distintas para cada trabajo. Esta característica hace que el problema se vuelva más complicado [Rosas, 91].

Garey, Tarjan y Wilfong, fueron los primeros en demostrar que este problema pertenece a la clase de problemas NP-completo, es decir aquellos que no pueden ser resueltos determinísticamente en tiempo polinomial [Rosas, 91].

El trabajo realizado por Bautista [91], resuelve el PPT en dos máquinas mediante la utilización de un algoritmo genético. En dicho trabajo, se utiliza una codificación binaria para resolver el PPT, tanto para el caso de una máquina como para el de dos idénticas.

Un trabajo previo, el realizado por Rosas [91], resuelve el PPT en dos máquinas mediante la utilización del metaheurístico búsqueda tabú. En dicho trabajo se aborda el PPT para los casos de una máquina como para el de dos máquinas idénticas. Como propuesta futura propone el análisis de situaciones más complejas, como incluir los tiempos y costos de preparación y la extensión del problema para los casos de máquinas diferentes y el caso cuando se tienen n máquinas disponibles. Por esto, el trabajo de Rosas es tomado como punto de partida para la realización de esta tesis.

La relevancia de esta investigación radica en que, aún cuando la búsqueda tabú ha sido aplicada con éxito a algunos problemas de programación de tareas, no existe todavía nada reportado acerca del problema específico que aquí se plantea.

1.6 Guía de la tesis

Como se ha visto hasta ahora, en el capítulo 1, se ha dado una explicación del problema de programación de tareas y también se ha desarrollado el modelo matemático del mismo. En los capítulos posteriores se ven los elementos necesarios para la realización de este trabajo. En el capítulo 2, se hace una descripción del método utilizado para la solución del PPT, llamado Búsqueda Tabú. En el capítulo 3, se hace la aplicación de la metodología de solución para los casos de una máquina, dos máquinas idénticas y dos máquinas diferentes. En el capítulo 4, se hace el análisis de datos y presentación de los resultados obtenidos de las pruebas realizadas en el capítulo 3. En el capítulo 5, se sacan conclusiones y se hacen recomendaciones para trabajos futuros.

CAPÍTULO 2

DESCRIPCIÓN DE

BÚSQUEDA TABÚ

2.1 Búsqueda Tabú

Búsqueda Tabú (tabú search, TS) es un procedimiento metaheurístico utilizado para guiar un algoritmo heurístico de búsqueda local para explorar el espacio de soluciones más allá de la simple optimalidad local [Díaz et al, 96].

Búsqueda Tabú se basa en la premisa de que para poder calificar de inteligente la solución de un problema, debe incorporar *memoria adaptativa y exploración sensible* [Díaz et al, 96].

memoria adaptativa: Selectividad (incluyendo olvido estratégico).
Abstracción y descomposición (a través de memoria explícita y por atributos).

exploración sensible: Imposición estratégica de limitaciones e inducciones (condiciones tabú y niveles de aspiración).

La base para la Búsqueda Tabú puede describirse como sigue:

Dada una función *f(x)* a ser optimizada en un conjunto X, TS empieza de la misma manera que cualquier búsqueda local, procediendo iterativamente de un punto a otro hasta satisfacer un criterio dado de terminación. Cada $x \in X$ tiene un entorno (o vecindad) asociado $N(x) \subseteq X$, y cada solución $x' \in N(x)$ se puede alcanzar desde x mediante una operación llamada movimiento.

TS rebasa la búsqueda local empleando una estrategia de modificación de N(x) a medida que la búsqueda progresa, reemplazándola por otro entorno N*(x) mediante el uso de estructuras especiales de memoria y así organizar la manera en la cual se explora el espacio.

Las soluciones que son admitidas en N*(x) por estas estructuras de memoria se determinan de varias formas. Una de ellas, que da a búsqueda tabú su nombre, identifica soluciones sobre un horizonte especificado, y les prohibe pertenecer a N*(x) clasificándolas como tabú [Díaz et al, 96].

2.2 Memoria

La memoria usada en TS puede ser explícita o basada en atributos. La memoria explícita conserva soluciones completas y consiste típicamente en una élite de soluciones visitadas durante la búsqueda. Estas soluciones especiales se introducen estratégicamente para ampliar N*(x), y así presentar opciones útiles que no se encuentren en N(x).

La memoria en TS se diseña también para producir un efecto más sutil en la búsqueda, a través de una memoria basada en atributos, la cual guarda información sobre atributos de las soluciones que cambian al moverse de una solución a otra[Díaz et al, 96].

2.3 Memoria de corto plazo y sus elementos

La memoria de corto plazo más comúnmente usada, lleva la cuenta de los atributos de solución que han sido cambiados en el pasado reciente, y es llamada memoria basada en recenciates (hechos recientes). Para explotar esta memoria, los atributos seleccionados que se presentan en soluciones recientemente visitadas son designados "tabú-activos", y a las soluciones que contienen elementos tabú activos, o combinaciones particulares de estos atributos, son las que se convierten en "tabú".

El manejo de la memoria basada en la recencia se hace creando una o más listas tabú, las cuales registran los atributos tabú-activos y explícita e implícitamente identifican su estatus actual. Se denomina tenencia tabú a la duración que un atributo permanece tabú-activo [Díaz et al, 96].

2.3.1 El nivel de aspiración

El nivel de aspiración introduce un elemento importante de flexibilidad en la búsqueda tabú. El estatus tabú de una solución (o un movimiento) puede ser ignorado, si ciertas condiciones se cumplen, en la forma de niveles de aspiración. En efecto, estos niveles de aspiración dan umbrales de atracción, los cuales controlan el hecho de que las aspiraciones puedan ser consideradas admisibles a pesar de ser de estar clasificadas como tabú. Es obvio que cualquier solución que sea mejor que cualquiera de las encontradas anteriormente merece ser considerada admisible, incluso aunque para alcanzarla debamos utilizar un movimiento prohibido [Díaz et al, 96].

2.3.2 Lista de candidatos

Las estrategias para generar la lista de candidatos son usadas para restringir el número de soluciones examinadas en una iteración dada, para los casos en los que N*(x) es grande o la evaluación de sus elementos es costosa [Díaz et al, 96].

Dada la importancia que TS da a la selección juiciosa de elementos para el proceso de la búsqueda, es crítico contar con reglas eficientes para la generación y evaluación de buenos candidatos [Díaz et al, 96].

2.4 Memoria de largo plazo

En algunas aplicaciones, las componentes de la memoria de corto plazo son suficientes para producir soluciones de muy alta calidad. Sin embargo, en general, TS se vuelve significativamente más potente incluyendo memoria de largo plazo y sus estrategias asociadas.

Tipos especiales de memoria basada en frecuencia son fundamentales en consideraciones de largo plazo. Estas operan introduciendo penalizaciones e incentivos determinados por el rango relativo de tiempo durante el que los atributos han permanecido a soluciones visitadas durante la búsqueda, permitiendo diferenciación por regiones. Las frecuencias de transición mantienen un registro de la frecuencia conque cambian los atributos, mientras que las frecuencias de residencia mantienen el registro de las duraciones relativas de los atributos en las soluciones generadas. Este tipo de memorias son acompañadas algunas veces por formas extendidas de memoria basada en recencia [Díaz et al, 96].

2.5 Proceso de memoria de corto plazo

Este proceso es utilizado con el propósito de seleccionar el mejor movimiento de un conjunto de posibles movimientos sujetos a ciertas restricciones. Estas son llamadas restricciones tabú y son utilizadas para que el proceso no se cicle, es decir, impiden el regreso a una solución visitada anteriormente [Rosas, 91]. Esto se logra, gracias a la utilización de la memoria de corto plazo y a sus elementos ya antes mencionados.

Glover, menciona que el principal objetivo de las restricciones tabú es permitir que el método vaya más allá de los puntos de optimalidad local, y continúe haciendo movimientos de alta calidad en cada paso [Rosas, 91].

En la figura 1 se presenta en forma esquemática este proceso.

2.5.1 Solución inicial

El proceso de memoria de corto plazo empieza con una solución inicial factible, la cual denotaremos como S0. Esta solución puede ser obtenida por diferentes métodos o criterios, por ejemplo, para el caso de programación de tareas en una sola máquina, puede utilizarse como solución inicial, una secuencia de trabajos ordenados en forma ascendente en relación a su fecha de entrega. También puede utilizarse alguna otra heurística. Esta solución inicial se evaluará en una función objetivo f y obtendremos un valor f(S0). Esta solución se considera como la mejor obtenida hasta el momento, es decir, nuestro incumbente es igual a f(S0), y la mejor secuencia es igual a S0.

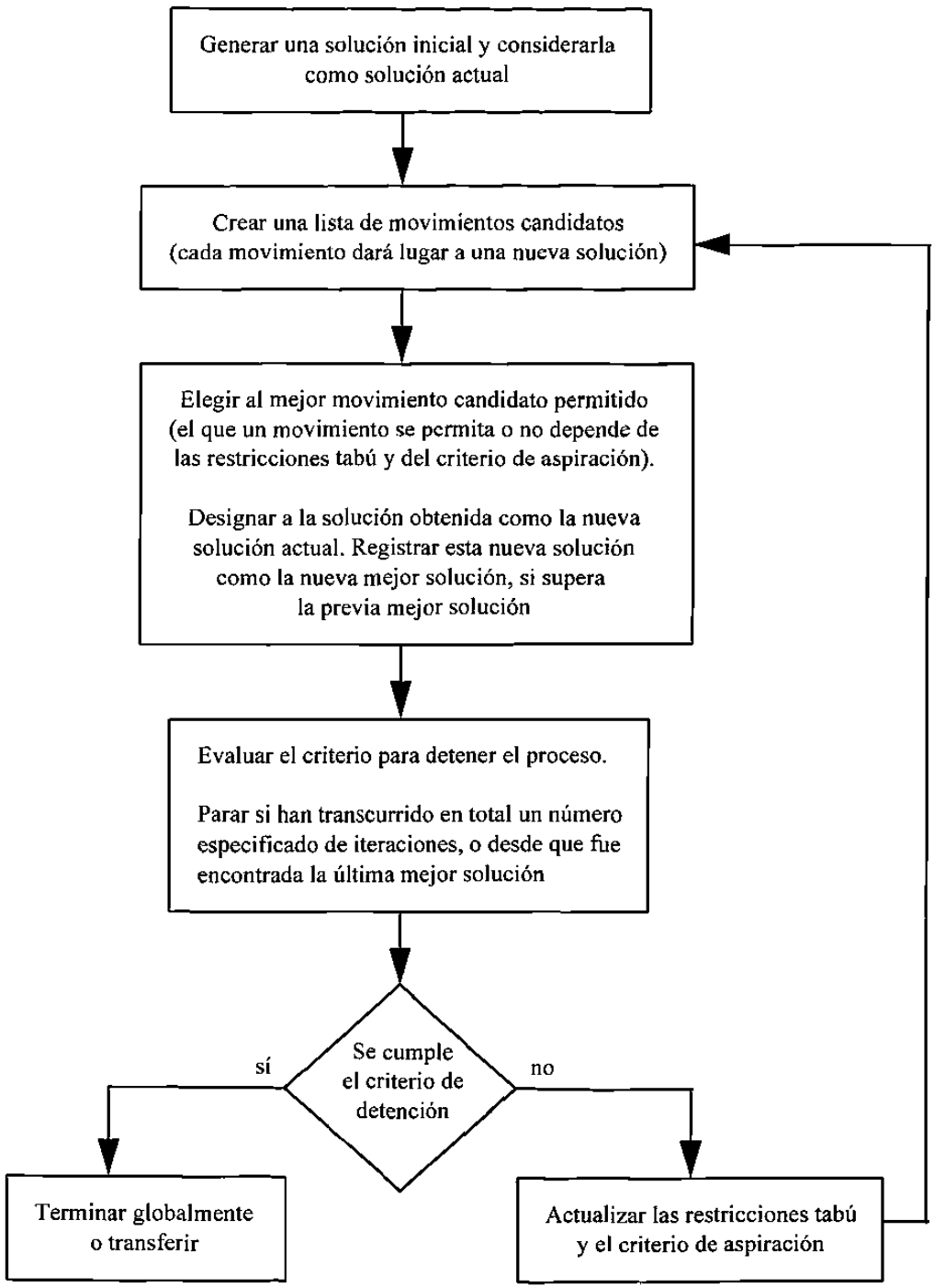


Figura 1. Proceso de memoria de corto plazo.

2.5.2 Crear una lista de movimientos candidatos

Los movimientos candidatos pueden ser de diferente tipo, por ejemplo, podemos definir un movimiento "swap" como un intercambio de posiciones entre una pareja de trabajos en una secuencia dada, o bien un movimiento de tipo "insert", donde un trabajo cambia de posición colocándose entre dos trabajos. Se debe entender que cada movimiento implica una nueva solución, diferente a la obtenida en el punto anterior.

2.5.3 Elegir el mejor candidato permitido

El proceso que se sigue para seleccionar al mejor movimiento candidato es el siguiente: Primeramente habrá que determinar cuáles movimientos de nuestra lista de movimientos candidatos pueden ser admitidos, esto se logra mediante la evaluación de cada uno de ellos. Esta evaluación puede basarse en el cambio que se produce en el valor de la función objetivo, es decir, se comparan los valores de la función objetivo antes y después de llevar a cabo el movimiento. Si el movimiento que está siendo evaluado no produce un mejor valor al encontrado en el último movimiento admitido, entonces se pasa a verificar el siguiente movimiento en la lista, si no es así (que sí produce mejora), entonces se procede a evaluar el estatus tabú del movimiento. Si el movimiento no es tabú, éste ha sido admitido y lo consideramos como nuestro mejor candidato encontrado hasta el momento (S*). En caso contrario (el movimiento sí es tabú), se deberá preguntar por el nivel de aspiración, si el movimiento en cuestión lo satisface, entonces se admite y pasa a ser nuestro mejor candidato, si no satisface el nivel de aspiración, entonces habrá que seleccionar a otro movimiento, repitiendo el proceso. Al terminar de evaluar todos los movimientos, tendremos al fin, nuestro mejor candidato y por consiguiente una nueva solución (S*), que servirá de base para repetir nuevamente el proceso. Si ésta solución es mejor a la obtenida en la iteración anterior, entonces se convierte en nuestra mejor solución hasta el momento (S_best, incumbente).

En la figura 2 se muestra en forma esquemática el proceso que se sigue para seleccionar al mejor candidato.

2.5.4 Criterio para finalizar el proceso

En ocasiones se conoce una cota inferior para la función objetivo que se está evaluando. En estos casos, el proceso iterativo se detiene cuando la mejor solución obtenida esté lo suficientemente cerca de este límite. Desafortunadamente no siempre es posible contar con esta cota, por lo que se debe hacer uso de algún otro criterio. Lo más recomendable es fijar un número de iteraciones a ejecutar, ya sea en total, o bien desde que la última mejor solución fue encontrada.

2.5.5 Actualización de las condiciones de admisión

Si el criterio para finalizar el proceso iterativo no se ha cumplido, es necesario actualizar algunos parámetros. Ya se ha mencionado que para evitar caer en un ciclo, no se permite regresar a una solución visitada anteriormente, mediante el uso de una lista tabú. Esta lista tabú se utiliza a manera de fila de la siguiente manera: siempre que se ejecute algún movimiento que arroje una nueva solución (S*), se debe introducir esta solución al final de la lista y al mismo tiempo, eliminar la solución más antigua presente en la lista. De esta forma, que todos los movimientos que provoquen que se regrese a esta solución (S*) quedan ahora prohibidos para un número de iteraciones dado por la longitud de la lista tabú.

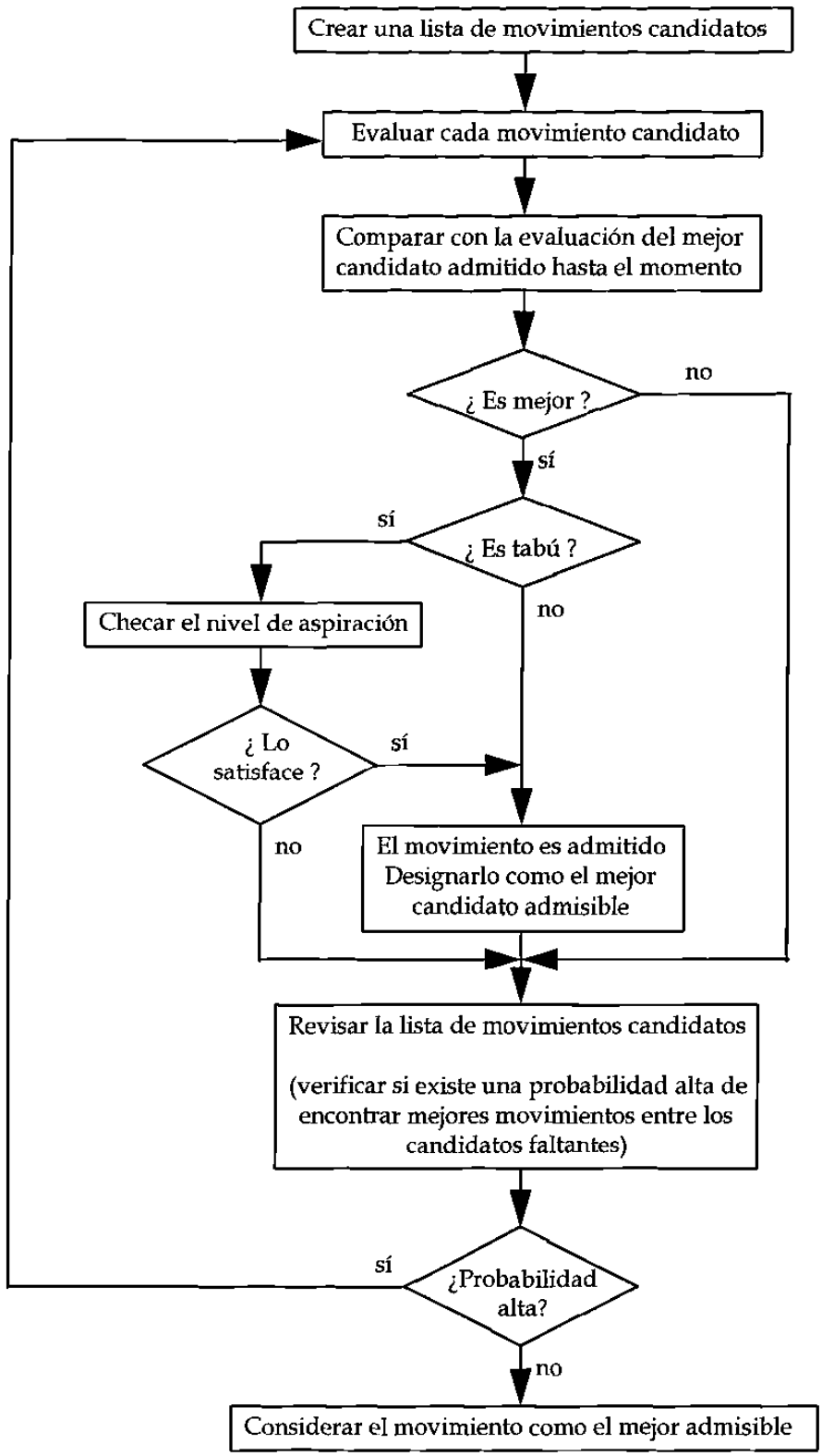


Figura 2. Selección del mejor candidato admitido.

2.6 Ejemplo

Supongamos que se tienen que procesar cinco trabajos en una sola máquina. Se desea saber cuál es la mejor secuencia en la cual se pueden procesar, de tal manera que se minimice la penalización total incurrida, ya sea por anticipación o por retraso de las fechas de terminación de los trabajos con respecto a las fechas de entrega programadas. A continuación se presentan los tiempos de procesamiento (t), fechas de entrega (d), penalizaciones por anticipación por día (p) y las penalizaciones por retraso por día (u) para cada trabajo:

Trabajo	t	d	p	u
1	5	11	7	4
2	3	9	2	8
3	5	11	7	7
4	2	10	10	5
5	5	7	3	1

Para resolver este problema consideramos una longitud de lista tabú igual a 3 (equivalente a nuestra memoria de corto plazo), y un número máximo de iteraciones a ejecutar de 4. Evaluaremos como movimientos, exclusivamente intercambios de parejas de trabajos consecutivos.

Nuestro nivel de aspiración quedará definido de la siguiente manera: si la penalización total correspondiente a la solución derivada de un movimiento tabú es menor que el incumbente, entonces el movimiento tabú queda liberado. En cuanto a la restricción tabú, diremos que después que se ha hecho un movimiento swap (i,j), el trabajo que se encontraba en la posición *i* no puede ser colocado nuevamente en esta posición en los schedules siguientes, hasta que haya expirado la memoria de corto plazo (en este caso igual a 3), o a menos de que se cumpla con el nivel de aspiración.

Como primer paso tenemos que generar una solución inicial de la cual partir. Para ello, secuenciamos los trabajos en orden ascendente en relación a su fecha de entrega. De esta manera nuestra solución inicial quedará de la forma siguiente:

So = (5,2,4,1,3) representa la secuencia inicial, con
f(So) = 87 como la penalización total, la cual consideramos como nuestro incumbente.

incumbente = 87

Iteración 1.

Inicializamos la variable mej_sol (mejor solución de la iteración) con un valor grande
mej_sol = 100,000

Debemos formar ahora nuestra lista de movimientos candidatos:
1.- swap (5,2)
2.- swap (2,4)
3.- swap (4,1)
4.- swap (1,3)

Evaluamos ahora cada uno de estos movimientos:

Movimiento 1: swap (5,2)
S1 = (2,5,4,1,3) f(S1) = 92

Como nos encontramos en la primera iteración, no tenemos todavía ningún movimiento tabú, por lo tanto, esta primera solución pasa a ser nuestra mejor solución encontrada en esta iteración por el momento.

mej_sol = f(S1)

mej_sec = S1

Como f(S1) no es menor que nuestro incumbente no se provoca ningún cambio y proseguimos a evaluar el siguiente movimiento.

Movimiento 2: swap (2,4)

S2 = (5,4,2,1,3) f(S2) = 123

Como f(S2) > mej_sol entonces procedemos a evaluar el siguiente movimiento.

Movimiento 3: swap (4,1)

S3 = (5,2,1,4,3) f(S3) = 104

Como f(S3) > mej_sol entonces procedemos a evaluar el siguiente movimiento.

Movimiento 4: swap (1,3)

S4 = (5,2,4,3,1) f(S4) = 72

Como f(S4) < mej_sol entonces verificamos su estatus tabú. Como no es un movimiento tabú, entonces se convierte en nuestra mejor solución encontrada en esta iteración.

mej_sol = f(S4)

mej_sec = S4

Hemos concluido la iteración. En este momento comparamos el valor incumbente con la mejor solución encontrada en la iteración.

mej_sol < incumbente (72 < 87). Como la mejor solución encontrada en la iteración es menor al incumbente, entonces actualizamos el incumbente con el valor de esa mejor solución.

incumbente = 72

Como no deseamos regresar a la solución de la que partimos, el swap (3,1) es ahora un movimiento tabú durante las próximas 3 (longitud de la lista tabú) iteraciones, a menos que el nivel de aspiración elimine su estatus tabú. Actualizamos entonces nuestra lista tabú.

Lista tabú

(3,1)

Iteración 2.

Partimos ahora de la secuencia solución: S = (5,2,4,3,1)

encontrada en la iteración anterior

y actualizamos el valor de la variable mej_sol: mej_sol = 100,000

Nuestra lista de movimientos candidatos es:

1.- swap (5,2)

2.- swap (2,4)

3.- swap (4,3)

4.- swap (3,1)

Evaluamos ahora cada uno de estos movimientos:

i	swap	Si	f(Si)	f(Si) < mej_sol	tabú	mej_sol	mej_sec
1	(5,2)	(2,5,4,3,1)	77	sí	no	77	(2,5,4,3,1)
2	(2,4)	(5,4,2,3,1)	108	no			
3	(4,3)	(5,2,3,4,1)	83	no			
4	(3,1)	(5,2,4,1,3)	87	no			

Como se puede observar la mejor solución se obtuvo en el movimiento 1, correspondiente al swap (5,2), sin embargo esta solución no es mejor que nuestro incumbente, por lo tanto no actualizamos éste valor. Lo que sí tenemos que actualizar es nuestra lista tabú de la siguente manera:

Lista tabú

(3,1)
(2,5)

Iteración 3.

Partimos ahora de la secuencia solución: S = (2,5,4,3,1)

y actualizamos el valor de la variable mej_sol: mej_sol = 100,000

Nuestra lista de movimientos candidatos es:

1.- swap (2,5)

2.- swap (5,4)

3.- swap (4,3)

4.- swap (3,1)

Evaluamos ahora cada uno de estos movimientos:

i	swap	Si	f(Si)	f(Si) < mej_sol	tabú	f(Si) < incumbente	mej_sol	mej_sec
1	(2,5)	(5,2,4,3,1)	72	sí	sí	no	100,000	
2	(5,4)	(2,4,5,3,1)	129	sí	no		129	(2,4,5,3,1)
3	(4,3)	(2,5,3,4,1)	88	sí	no		88	(2,5,3,4,1)
4	(3,1)	(2,5,4,1,3)	92	no				

Como se puede observar, al primer movimiento evaluado le corresponde el mejor valor de la función objetivo, sin embargo, como podemos verificar en nuestra lista tabú, se trata de un movimiento tabú, por lo cual, procedimos a verificar si cumplía o no con el nivel de aspiración, como no se satisfizo, simplemente se pasó a evaluar el siguiente movimiento. Finalmente, la mejor solución corresponde al movimiento 3 correspondiente al swap (4,3), no obstante no se mejoró el valor incumbente. Actualizamos ahora nuestra lista tabú.

Lista tabú
(3,1)
(2,5)
(3,4)

Iteración 4.

Partimos ahora de la secuencia solución: S = (2,5,3,4,1)

y actualizamos el valor de la variable mej_sol: mej_sol = 100,000

Nuestra lista de movimientos candidatos es:

1.- swap (2,5)

2.- swap (5,3)

3.- swap (3,4)

4.- swap (4,1)

Evaluamos ahora cada uno de estos movimientos:

i	swap	Si	f(Si)	f(Si) < mej_sol	tabú	f(Si) < incumbente	mej_sol	mej_sec
1	(2,5)	(5,2,3,4,1)	83	sí	sí	no	100,000	
2	(5,3)	(2,3,5,4,1)	100	sí	no		100	(2,3,5,4,1)
3	(3,4)	(2,5,4,3,1)	77	sí	sí	no		
4	(4,1)	(2,5,3,1,4)	105	no				

Como podemos observar, la mejor solución corresponde al swap (5,3), desafortunadamente esta solución no es mejor que el incumbente. Actualizamos ahora nuestra lista tabú, el movimiento que ingresa a la lista es el swap (3,5), nótese que como ya transcurrieron 3 iteraciones el movimiento (3,1) ha abandonado la lista tabú.

Lista tabú
(2,5)
(3,4)
(3,5)

Finalmente hemos terminado, pues se ha cumplido con las 4 iteraciones que se definieron al inicio del ejemplo como criterio de parada.

Revisando los resultados obtenidos durante las cuatro iteraciones, podemos ver que la mejor solución obtenida fue la encontrada en la primera iteración, con una penalización total de 72.

Como se puede apreciar el método es muy sencillo en su comprensión y fácilmente se puede traducir a un lenguaje computacional.

CAPÍTULO 3

APLICACIÓN DE LA METODOLOGÍA DE SOLUCIÓN

3.1 Introducción

En el presente capítulo se explica cuál es la metodología que se sigue para dar solución al problema de programación de tareas planteado en el capítulo 1, para los casos de una máquina, dos máquinas idénticas y dos máquinas diferentes. Cabe recordar que la herramienta en la cual se basa ésta metodología es el metaheurístico búsqueda tabú.

3.2 El caso de una máquina

Para dar solución a éste problema se utiliza como herramienta principal el metaheurístico búsqueda tabú. Como se vio en el capítulo 1, las restricciones que se toman en cuenta son las penalizaciones por tardanza y anticipo, costos y tiempos de preparación, así como tiempos de ocio. A continuación se presenta la metodología de solución para este caso.

3.2.1 Solución inicial

Como primer paso generamos una solución inicial factible, para esto, secuenciamos las tareas en orden ascendente de acuerdo a su fecha de entrega. Una vez que se genera la primera secuencia, se insertan los tiempos y costos de preparación que le corresponden a cada par de tareas. En ocasiones es conveniente insertar tiempos de ocio (tiempo en que la máquina está detenida) entre cada par de tareas, para mejorar el valor de la función objetivo. La secuencia resultante, así como su costo, se consideran como la mejor solución obtenida hasta el momento.

3.2.1.1 Tiempos y costos de preparación

La manera en que se lleva a cabo el control de los tiempos y costos de preparación, es mediante dos matrices de n x n, siendo n el número de trabajos a procesar. En la primera matriz se tiene almacenado el tiempo de preparación que le corresponde a cada par de tareas y en la segunda, se tiene el costo de preparación que le corresponde a cada par de tareas. Una vez que se tienen estas matrices, es fácil calcular el tiempo y el costo en que incurre una secuencia de trabajos dada, por conceptos de preparación de la máquina.

3.2.1.2 Tiempo de ocio

La manera en que se controla la inserción de tiempos de ocio es diferente a como se hace con los tiempos y costos de preparación. La inserción de tiempos de ocio para obtener una secuencia de trabajos retrasada óptima, puede obtenerse resolviendo un problema de programación lineal, en donde el objetivo es minimizar la desviación

absoluta ponderada. Para esto se eligió aplicar el algoritmo propuesto por Fry, Armstrong y Blackstone [87].

3.2.2 Crear una lista de movimientos candidatos

La lista de movimientos candidatos se genera a partir de la solución inicial o de la mejor solución encontrada en cada iteración, cuando se está iniciando el proceso o cuando ya se lleva más de una. La creación de esta lista está restringida a que solo se permiten movimientos entre parejas de trabajos consecutivos, empezando por el primer trabajo de la secuencia. La cantidad de movimientos candidatos en cada iteración será de $n-1$, siendo n la cantidad de trabajos a procesar.

3.2.3 Restricciones tabú

Para dar solución al problema planteado en esta tesis utilizando el método búsqueda tabú, es importante determinar un estatus tabú que impida que se ejecute un movimiento inverso a uno previamente seleccionado durante el tiempo que dure la memoria de corto plazo.

Para ello, se definen las siguientes restricciones tabú: Después de que se lleva a cabo un intercambio entre los trabajos de las posiciones i, j, donde $j > i$.:

- El trabajo $\pi_{\sigma i}$ no puede ser colocado nuevamente en la posición i ni en ninguna posición anterior a ella, hasta que expire la memoria de corto plazo o se satisfaga el nivel de aspiración.
- Por su parte el trabajo $\pi_{\sigma j}$ no puede ser colocado en la posición j ni en ninguna posición posterior a ella, hasta que expire la memoria de corto plazo o se satisfaga el nivel de aspiración.

Para controlar esto, es necesario guardar en memoria los trabajos π_i, π_j que fueron intercambiados, y definir por cuánto tiempo (longitud de la memoria) será un movimiento tabú.

3.2.4 Nivel de aspiración

Como se mencionó anteriormente, puede darse el caso de que un movimiento que es considerado tabú, proporcione una mejor solución que cualquier otro movimiento admitido. En este caso, es conveniente liberar a este movimiento de su estatus tabú.

El nivel de aspiración se define como sigue: Si la solución que se propone al efectuar un movimiento tabú, evaluada en la función objetivo es mejor que nuestra mejor solución hasta el momento, entonces se elimina su condición de tabú y se permite llevarlo a cabo.

3.2.5 Implementación de Búsqueda Tabú

Una vez que se tiene una solución inicial de la cual partir, que se han definido las restricciones tabú, el nivel de aspiración y la memoria a corto plazo, se prosigue a un proceso iterativo. En cada iteración un conjunto de movimientos candidatos es evaluado, con el fin de seleccionar, al final de cada iteración, el mejor movimiento, es decir, aquel que implique un menor costo. Este movimiento conducirá a una nueva secuencia de trabajos y a unos nuevos tiempos de terminación. Esta solución, además, servirá de punto de partida para la siguiente iteración, generando a partir de ella un conjunto nuevo de movimientos.

El proceso que se sigue en cada iteración es el siguiente: Teniendo la lista de movimientos candidatos, se elige el primero y evaluamos la solución que se genera al hacer este movimiento en la función objetivo, considerando el tiempo y costo de preparación así como el tiempo de ocio. Como se trata del primer movimiento, lo consideramos como el mejor admitido hasta el momento en esta iteración y proseguimos a verificar su estatus tabú. Si no es un movimiento tabú, lo designamos por el momento como la mejor solución en esta iteración. Si acaso es un movimiento tabú, entonces se verifica el nivel de aspiración, si lo cumple, se olvida su estatus tabú, en caso contrario, no puede ser un movimiento permitido y proseguimos a evaluar el siguiente movimiento candidato.

Para los movimientos restantes en la lista de candidatos, es necesario verificar primeramente si la solución que proponen implica un menor costo que el costo de la mejor solución encontrada hasta el momento en la iteración actual. Si es así, entonces se continúa con el proceso, tal como se explica en el párrafo anterior, si no es así, se prosigue a evaluar el siguiente movimiento candidato.

Si al término de una iteración, la mejor solución encontrada en ella implica un menor costo que nuestro incumbente, entonces esta solución es considerada como nuestro nuevo incumbente.

Para determinar cuántas iteraciones efectuar, se toma en cuenta el siguiente criterio: El proceso se detiene después de que se ha cumplido con un número de iteraciones previamente establecido.

En la figura 3 se presenta un pseudocódigo del método descrito en los párrafos anteriores que corresponde al caso de una máquina.

```
Generar una solución inicial Π0
Insertar tiempos de preparación y de ocio a Π0
Evaluar f (Π0)
mejor_secuencia = Π0
incumbente = f (Π0)          (mejor valor encontrado hasta el momento)
lista_tabú = memoria de corto plazo
Repeat
    Generar lista de movimientos candidatos a partir de Π0
    mejor_valor_movimientos = ∞
    For (todos los movimientos candidatos)
        valor_movimiento = f (Π)
        secuencia_movimiento = Π
        If (valor_movimiento < mejor_valor_movimientos)
            If (el movimiento no es tabú)
                mejor_valor_movimientos = valor_movimiento
                mejor_secuencia_movimientos = secuencia_movimiento
            else (el movimiento es tabú)
                If (valor_movimiento < incumbente)
                mejor_valor_movimientos = valor_movimiento
                mejor_secuencia_movimientos = secuencia_movimiento
    Π0 = mejor_secuencia_movimientos
    lista_tabú = lista_tabú + movimiento de Π0
    If (mejor_valor_movimientos < incumbente)
        mejor_secuencia = mejor_secuencia_movimientos
        incumbente = mejor_valor_movimientos
Until (se cumpla con el número de iteraciones dado)
```

Figura 3. Pseudocódigo para el problema de una máquina.

3.3 El caso de dos máquinas idénticas

Para dar solución a éste problema también se utiliza como herramienta principal el metaheurístico búsqueda tabú. Además, se utiliza el método que se describe en la sección anterior para optimizar las secuencias de cada máquina en forma individual.

El problema que se resuelve en este caso puede dividirse en dos subproblemas: el primero consiste en asignar cada trabajo a una máquina en particular, tratando de optimizar el costo de las secuencias en conjunto y el segundo, consiste en tratar de optimizar la secuencia de cada máquina por separado. A continuación se presenta la metodología de solución para este caso.

3.3.1 Solución inicial

Al igual que en el caso de una sola máquina, es necesario contar con una solución factible de la cual partir. Para esto, se sigue el siguiente procedimiento: como primer paso, secuenciamos las tareas en orden ascendente de acuerdo a su fecha de entrega, después, se asigna un trabajo a cada máquina en forma alternada, de tal manera que la primera máquina tenga asignados $\lceil n/2 \rceil$ trabajos y la segunda $\lfloor n/2 \rfloor$ trabajos, siendo n el número total de tareas a programar. Una vez que se generan las secuencias para cada máquina, se insertan los tiempos y costos de preparación que le corresponden a cada par de tareas en cada una de las secuencias. Tal como en el caso anterior, en ocasiones es conveniente insertar tiempos de ocio (tiempo en que la máquina está detenida) entre cada par de tareas, para mejorar el valor de la función objetivo. Las secuencias resultantes, así como su costo total (la suma de los costos de las dos secuencias), se consideran como la mejor solución obtenida hasta el momento.

3.3.1.1 Tiempos y costos de preparación

La manera en que se lleva a cabo el control de los tiempos y costos de preparación es mediante dos matrices de n x n, siendo n el número de trabajos a procesar. En la primera matriz se tiene almacenado el tiempo de preparación que le corresponde a cada par de tareas y en la segunda, se tiene el costo de preparación que le corresponde a cada par de tareas. Como se puede observar no existe ninguna diferencia con el caso de una sola máquina, esto se debe a que como se trata de dos máquinas idénticas el tiempo y el costo de preparación no varían de una maquina a otra, pues como se menciona en el capítulo 1, tanto el tiempo como el costo de preparación entre cada par de tareas están dados por las características de la máquina. Una vez que se tienen las matrices de tiempo y costo de preparación, es fácil calcular el tiempo y el costo en que incurre una secuencia de trabajos dada, por conceptos de preparación de la máquina.

3.3.1.2 Tiempo de ocio

La manera en que se controla la inserción de tiempos de ocio no difiere de como se hace con el caso de una sola máquina. La inserción de tiempos de ocio para obtener una secuencia de trabajos retrasada óptima, puede obtenerse resolviendo un problema de programación lineal, en donde el objetivo es minimizar la desviación absoluta ponderada. Para esto se aplica el algoritmo propuesto por Fry, Armstrong y Blackstone [87] a cada una de las secuencias.

3.3.2 Crear una lista de movimientos candidatos

La lista de movimientos candidatos se genera a partir de la solución inicial o de la mejor solución encontrada en cada iteración, para cuando se esta iniciando el proceso o

cuando ya se lleva más de una iteración, según sea el caso. Los movimientos candidatos se consideran nuevamente como intercambios entre las posiciones de las tareas, solo que esta vez, estos intercambios se hacen de una máquina a otra. Considerando este procedimiento, la cantidad de movimientos candidatos en cada iteración no es de n-1, como en el caso de una máquina, sino de $\lceil n/2 \rceil$ x $\lfloor n/2 \rfloor$. Para delimitar un poco el número de movimientos candidatos a evaluar en cada iteración, se sigue el mismo procedimiento presentado por Rosas [91] en su trabajo. Aquí se menciona que el procedimiento es similar al que se presenta en un artículo de Manuel Laguna y José Luis González Velarde en el año de 1991.

Haciendo uso de este criterio, se define un movimiento candidato como un intercambio entre un trabajo de la máquina 1 y otro de la máquina 2, de forma que la diferencia absoluta entre sus fechas de entrega sea menor o igual a una variable *limsup*.

Suponiendo que tenemos un movimiento entre los trabajos $\pi_{\sigma i}$ de la máquina 1 y $\pi_{\sigma j}$ de la máquina 2, este movimiento se considera candidato si cumple con lo siguiente:

$$| d\pi_{1(i)} - d\pi_{2(j)} | \leq limsup$$

donde

$d\pi_{1(i)}$: fecha de entrega del trabajo en la posición i en la máquina 1.

$d\pi_{2(j)}$: fecha de entrega del trabajo en la posición j en la máquina 2.

limsup : max (1 - ρ , 0.25) X (máximo - mínimo)

máximo: fecha de entrega mayor entre todos los trabajos.

mínimo: fecha de entrega menor entre todos los trabajos.

ρ : $\max \left(\frac{i-1}{n1} - \frac{j-1}{n2} \right)$

n1 : cantidad de trabajos asignados a la máquina 1.

n2 : cantidad de trabajos asignados a la máquina 2.

La variable *limsup* se utiliza con el propósito de eliminar aquellos movimientos que empeoren la solución en lugar de mejorarla. Por ejemplo, se puede decir que no tiene caso efectuar un intercambio entre el trabajo que se encuentra en la primera posición de la máquina 1 con el trabajo que se encuentra en la última posición de la máquina 2, pues el costo obviamente se incrementaría. La máxima diferencia posible entre dos fechas de entrega es aquella entre la mayor y la menor. El valor de *limsup* es un porcentaje de dicha diferencia o rango, que depende a su vez del valor de ρ. Si ρ tiene un valor de cero, entonces cualquier diferencia entre fechas de entrega puede ser aceptada.

3.3.3 Restricciones tabú

Como se mencionó anteriormente, es importante determinar un estatus tabú que impida que se ejecute un movimiento inverso a uno previamente seleccionado durante el tiempo que dure la memoria de corto plazo.

Para el caso de dos máquinas idénticas, las restricciones tabú se definen como sigue: Después de que se lleva a cabo un intercambio entre las posiciones de los trabajos $\pi_{\sigma i}$ de la máquina 1 y $\pi_{\sigma j}$ de la máquina 2.

- El trabajo $\pi_{\sigma i}$ no puede regresar a ninguna posición en la máquina 1, hasta que expire la memoria de corto plazo o se satisfaga el nivel de aspiración.
- Por su parte el trabajo $\pi_{\sigma j}$ no puede regresar a ninguna posición en la máquina 2, hasta que expire la memoria de corto plazo o se satisfaga el nivel de aspiración.

Para controlar esto, es necesario guardar en memoria los trabajos $\pi_{1(i)}$ y $\pi_{2(j)}$ que fueron intercambiados, y definir por cuánto tiempo (longitud de la memoria) será un movimiento tabú.

3.3.4 Nivel de aspiración

Cabe recordar que puede darse el caso de que un movimiento que es considerado tabú, proporcione una mejor solución que cualquier otro movimiento admitido. En este caso, es conveniente liberar a este movimiento de su estatus tabú.

El nivel de aspiración para el caso de dos máquinas idénticas se define como sigue: Si la solución que se propone al efectuar un movimiento tabú, evaluada en la función objetivo es mejor que nuestra mejor solución hasta el momento, entonces se elimina su condición de tabú y se permite llevarlo a cabo.

3.3.5 Implementación de Búsqueda Tabú

Al igual que en el caso de una sola máquina, una vez que se tiene una solución inicial de la cual partir, que se han definido las restricciones tabú, el nivel de aspiración y la memoria de corto plazo, se prosigue a un proceso iterativo. Una serie de movimientos candidatos son evaluados en cada iteración, seleccionando de ésta, el mejor movimiento, es decir, aquel que implique un menor costo. Cada vez que finaliza una iteración, se encuentra una nueva solución o secuencias para cada una de las máquinas, siendo este par de secuencias, las mejores encontradas entre el conjunto de soluciones permitidas en esa iteración. Esta nueva solución servirá de punto de partida para la siguiente iteración, generando a partir de ella un conjunto nuevo de movimientos. Cabe recordar que en el momento en que un movimiento ha sido seleccionado como el mejor de la iteración, los trabajos que pertenecen a dicho movimiento adquieren un estatus tabú, prohibiendoles regresar a su posición en la máquina en la que estaban anteriormente, a menos que expire la memoria de corto plazo o se cumpla con el nivel de aspiración.

El proceso que se sigue en cada iteración para el caso de dos máquinas idénticas es el mismo que para el caso de una sola máquina (sección 3.2.5), sólo que ahora, los intercambios se realizan entre máquinas y las restricciones tabú hacen referencia a ello. Para determinar cuántas iteraciones efectuar, se toma en cuenta el mismo criterio, el proceso se detiene después de que se ha cumplido con un número de iteraciones previamente establecido. Una vez que esto se ha cumplido, se procede a optimizar cada máquina por separado, aplicando la metodología descrita anteriormente para el caso de una sola máquina.

En la figura 4 se presenta un pseudocódigo del método propuesto para resolver el problema para el caso de dos máquinas idénticas.

3.4 El caso de dos máquinas diferentes

La metodología que se propone para el caso de dos máquinas diferentes es similar a la que se utiliza para el caso de dos máquinas idénticas. Las diferencias que existen entre estos dos casos se presentan en los tiempos y costos de preparación. Esto se debe a que como se trata de dos máquinas diferentes, el tiempo y el costo de preparación varían de una maquina a otra, pues como se menciona en el capítulo 1, tanto el tiempo como el costo de preparación entre cada par de tareas están dados por las características de la máquina.

En la siguiente sección se ve más a detalle la forma en que se presentan los tiempos y costos de preparación para el caso de dos máquinas diferentes.

```
Generar la solución inicial Π01 y Π02
Insertar tiempos de preparación y de ocio a Π01 y Π02
Evaluar f (Π01) y f (Π02)
mejor_secuencia1 = Π01
mejor_secuencia2 = Π02
incumbente = f (Π01) + f (Π02)        (mejor valor encontrado hasta el momento)
lista_tabú = memoria de corto plazo
Repeat
    Generar lista de movimientos candidatos a partir de Π01 y Π02
    mejor_valor_movimientos = ∞
    For (todos los movimientos candidatos)
        insertar tiempos de preparación y de ocio a Π1 y Π2
        valor_movimiento = f (Π1) + f (Π2)
        secuencia1_movimiento = Π1
        secuencia2_movimiento = Π2
        If (valor_movimiento < mejor_valor_movimientos)
            If (el movimiento no es tabú)
                mejor_valor_movimientos = valor_movimiento
                mejor_secuencia1_movimientos = secuencia1_movimiento
                mejor_secuencia2_movimientos = secuencia2_movimiento
            else (el movimiento es tabú)
                If (valor_movimiento < incumbente)
                mejor_valor_movimientos = valor_movimiento
                mejor_secuencia1_movimientos = secuencia1_movimiento
                mejor_secuencia2_movimientos = secuencia2_movimiento
```

Figura 4. Pseudocódigo para el problema de dos máquinas.

Π_{01} = mejor_secuencia1_movimientos

Π_{02} = mejor_secuencia2_movimientos

lista_tabú = lista_tabú + movimiento de Π_{01} y Π_{02}

If (mejor_valor_movimientos < incumbente)

mejor_secuencia1 = mejor_secuencia1_movimientos

mejor_secuencia2 = mejor_secuencia2_movimientos

incumbente = mejor_valor_movimientos

Until (se cumpla con el número de iteraciones dado)

Π_0 = mejor_secuencia1

Se aplica la metología para el caso de una sola máquina.

Π_0 = mejor_secuencia2

Se aplica la metología para el caso de una sola máquina.

Figura 4. (Continúa).

3.4.1 Tiempos y costos de preparación

La forma en que se lleva a cabo el control de los tiempos y costos de preparación, es mediante cuatro matrices de n x n, siendo n el número de trabajos a procesar. En las primeras dos matrices se tiene almacenado el tiempo de preparación que le corresponde a cada par de tareas, la primera de estas matrices representa los tiempos de preparación correspondientes a la máquina 1 y la segunda de estas matrices representa los tiempos de preparación correspondientes a la máquina 2. En las matrices restantes se tiene almacenado el costo de preparación que le corresponde a cada par de tareas, una de estas matrices representa los costos de preparación que le corresponden a la máquina 1 y la otra matriz representa los costos de preparación que le corresponden a la máquina 2.

Como se puede observar, sí existe diferencia con el caso de dos máquinas idénticas, esto se debe a que el tiempo y el costo de preparación entre cada par de tareas están dados por las características de la máquina. Entonces, se puede decir, que debido a sus características, cada máquina tiene una matriz de tiempos y otra de costos de preparación. Una vez que se tienen las cuatro matrices, dos de tiempos y dos de costos de preparación y sabiendo en qué máquina va a ser procesada, es fácil calcular el tiempo y el costo en que incurre una secuencia de trabajos dada, por conceptos de preparación de la máquina.

CAPÍTULO 4

ANÁLISIS DE DATOS Y RESULTADOS

4.1 Introducción

En el presente capítulo se analizan los datos y se muestran los resultados obtenidos al probar las metodologías de solución propuestos en el capítulo 3, para los casos de una sola máquina, dos máquinas idénticas y dos máquinas diferentes. Para cada caso se realizan pruebas variando n (número de tareas a procesar), siendo 20, 30 y 40 los tres valores que toma. Para cada valor de n se evalúan 10 problemas diferentes. Los datos que corresponden a cada problema son generados de manera aleatoria. En la siguiente sección se presenta a detalle cómo son generados los datos.

4.2 Generación de los datos

Los datos necesarios para formular un problema se generan mediante dos distribuciones de probabilidad, una de ellas es la distribución normal y la otra es una

distribución uniforme. Los tiempos de procesamiento se obtienen de acuerdo a la distribución normal, con una media de 10 y una desviación estándar de 3. Estos valores de media y desviación estándar se toman para los casos de una sola máquina y dos máquinas idénticas. Para el caso de dos máquinas diferentes cada máquina tiene diferentes tiempos de procesamiento para el mismo trabajo, pues como ya se ha dicho, los tiempos de procesamiento así como los tiempos y costos de preparación dependen de las características de la máquina. Por ello, los valores de media y desviación estándar para la máquina 1, son de 10 y 3 y para la máquina 2, son de 12 y 3, respectivamente. Las fechas de entrega se generan con una distribución uniforme con parámetros *(a,b)*, donde *a* es igual al tiempo de procesamiento mínimo y *b* es igual a la suma de los tiempos de procesamiento multiplicado por un factor. Para el caso de una sola máquina se toma un valor de 1.02 y para el caso de dos máquinas idénticas de 1.05/2. Para el caso de dos máquinas diferentes *b* es igual a la suma de los tiempos de procesamiento de la máquina 2 multiplicado por un factor de 1.02/2. El propósito de usar este factor es el de obtener problemas representativos, es decir, obtener un balance entre los trabajos atrasados y adelantados. Las penalizaciones por entregas tardías y tempranas se generan con una distribución uniforme con valores de *a* igual a 1 y *b* igual a 10.

Para los valores de tiempos y costos de preparación se usa una distribución uniforme. Para los tiempos de preparación se utiliza un valor de *a* igual a 1 y *b* igual a 4, y para los costos de preparación se utiliza un valor de *a* igual a 3 y *b* igual a 7. Estos parámetros se toman para los casos de una sola máquina y dos máquinas idénticas. Para el caso de dos máquinas diferentes, se utilizan los mismos parámetros ya antes mencionados para la máquina 1. Para la máquina 2 estos parámetros cambian, para los tiempos de preparación se utiliza un valor de *a* igual a 2 y *b* igual a 6, y para los costos de preparación se utiliza un valor de *a* igual a 5 y *b* igual a 9.

4.3 Resultados para el caso de una sola máquina

Para resolver este tipo de problemas se utilizaron diferentes longitudes para la lista tabú, se probaron los valores de 3,4,5,6,7,8 y 9, siendo los valores 5, 6 y 7 los que arrojaban mejores resultados. En la tabla 1 se muestran los resultados obtenidos para el caso de una sola máquina utilizando 20 tareas a programar. En la tabla 2 se muestran los resultados obtenidos utilizando 30 tareas a programar y en la tabla 3 los obtenidos utilizando 40 tareas a programar.

Estos problemas fueron ejecutados mediante programas hechos en lenguaje C. El número de iteraciones utilizadas para la obtención de los resultados fue de 500, siendo los resultados obtenidos los que a continuación se presentan.

Tabla 1

Resultados del problema para el caso de una sola máquina con $n = 20$

Problema	longitud de la lista tabú	Iteración en que se encontró la mejor solución	Porcentaje de mejora con respecto al primer mínimo local
1	5	68	33.97
2	5	55	8.70
3	5	34	10.75
4	5	26	2.29
5	5	115	16.61
6	5	29	5.10
7	5	52	5.78
8	5	45	12.27
9	5	26	0.21
10	5	100	10.76
		Promedio	10.64

Tabla 2

Resultados del problema para el caso de una sola máquina con $n = 30$

Problema	longitud de la lista tabú	Iteración en que se encontró la mejor solución	Porcentaje de mejora con respecto al primer mínimo local
1	6	40	1.74
2	6	21	2.79
3	6	39	6.74
4	6	34	36.78
5	6	89	13.73
6	6	28	3.72
7	6	36	1.29
8	6	41	1.43
9	6	13	0
10	6	65	14.85
		Promedio	8.31

Tabla 3

Resultados del problema para el caso de una sola máquina con $n = 40$

Problema	longitud de la lista tabú	Iteración en que se encontró la mejor solución	Porcentaje de mejora con respecto al primer mínimo local
1	7	26	1.22
2	7	51	9.51
3	7	190	9.62
4	7	163	11.97
5	7	55	11.37
6	7	140	4.26
7	7	71	1.46
8	7	33	8.10
9	7	51	8.21
10	7	57	10.52
		Promedio	7.62

Como se observa en los resultados mostrados en las tablas anteriores, el método de solución que se propone para el problema de una sola máquina obtiene mejores soluciones en un 10.64% en promedio para los problemas de 20 trabajos, con respecto al primer mínimo local. Sucede lo mismo para los problemas de 30 trabajos con un 8.31% de mejora en promedio y de un 7.62% en promedio de mejora para los problemas de 40 trabajos.

En el apéndice A, se presenta una gráfica que muestra el comportamiento típico del método de solución para el problema de una sola máquina en uno de los problemas que fueron evaluados.

4.4 Resultados para el caso de dos máquinas idénticas

Para resolver este tipo de problemas también se utilizaron diferentes longitudes para la lista tabú, siendo los valores 9, 10 y 11 los que arrojaban mejores resultados. En la tabla 4 se muestran los resultados obtenidos para el caso de una sola máquina utilizando 20 tareas a programar. En la tabla 5 se muestran los resultados obtenidos utilizando 30 tareas a programar y en la tabla 6 se muestran los obtenidos utilizando 40 tareas a programar.

El número de iteraciones utilizadas fue de 500, una vez terminadas éstas, se procede a tratar de optimizar la secuencia de cada máquina por separado, utilizándose 100 iteraciones, siendo los resultados obtenidos los que a continuación se presentan.

Tabla 4

Resultados del problema para el caso de dos máquinas idénticas con $n = 20$

Problema	longitud de la lista tabú	Iteración en que se encontró la mejor solución	Porcentaje de mejora con respecto al primer mínimo local
1	9	183	40.34
2	9	95	20.37
3	9	55	34.45
4	9	69	22.00
5	9	137	15.10
6	9	112	25.37
7	9	152	32.10
8	9	141	1.45
9	9	22	9.43
10	9	62	9.96
		Promedio	21.06

Tabla 5

Resultados del problema para el caso de dos máquinas idénticas con $n = 30$

Problema	longitud de la lista tabú	Iteración en que se encontró la mejor solución	Porcentaje de mejora con respecto al primer mínimo local
1	10	35	13.17
2	10	103	20.68
3	10	76	12.23
4	10	185	17.66
5	10	105	30.82
6	10	124	14.12
7	10	118	14.98
8	10	55	26.98
9	10	100	11.52
10	10	64	21.07
		Promedio	18.32

Tabla 6

Resultados del problema para el caso de dos máquinas idénticas con $n = 40$

Problema	longitud de la lista tabú	Iteración en que se encontró la mejor solución	Porcentaje de mejora con respecto al primer mínimo local
1	11	114	12.95
2	11	175	23.57
3	11	173	17.55
4	11	198	3.86
5	11	196	13.43
6	11	149	12.55
7	11	180	24.21
8	11	152	22.16
9	11	17	0
10	11	67	6.48
		Promedio	13.68

Como puede observarse en las tablas anteriores, el método de solución que se propone para el problema de dos máquinas idénticas obtiene mejores soluciones en un 21.06% en promedio para los problemas de 20 trabajos, con respecto al primer mínimo local. Sucede lo mismo para los problemas de 30 trabajos con un 18.32% de mejora en promedio y de un 13.68% en promedio de mejora para los problemas de 40 trabajos.

En el apéndice A, se presenta una gráfica que muestra el comportamiento típico del método de solución para el problema de dos máquinas idénticas en uno de los problemas que fueron evaluados.

4.5 Resultados para el caso de dos máquinas diferentes

Para resolver este tipo de problemas también se utilizaron las mismas longitudes para la lista tabú que se usaron para el caso de dos máquinas idénticas, es decir los valores de 9, 10 y 11. En la tabla 7 aparecen los resultados obtenidos para el caso de una sola máquina utilizando 20 tareas a programar. En la tabla 8 se muestran los resultados obtenidos utilizando 30 tareas a programar y en la tabla 9 se muestran los resultados obtenidos utilizando 40 tareas a programar.

El número de iteraciones utilizadas fue de 500, una vez terminadas éstas, se procede a tratar de optimizar la secuencia de cada máquina por separado, utilizándose 100 iteraciones, siendo los resultados obtenidos los que a continuación se presentan.

Tabla 7

Resultados del problema para el caso de dos máquinas diferentes con $n = 20$

Problema	longitud de la lista tabú	Iteración en que se encontró la mejor solución	Porcentaje de mejora con respecto al primer mínimo local
1	9	10	23.38
2	9	177	23.59
3	9	79	40.74
4	9	218	32.31
5	9	178	45.90
6	9	242	15.73
7	9	199	19.75
8	9	113	31.76
9	9	14	6.72
10	9	338	15.45
		Promedio	25.53

Tabla 8

Resultados del problema para el caso de dos máquinas diferentes con $n = 30$

Problema	longitud de la lista tabú	Iteración en que se encontró la mejor solución	Porcentaje de mejora con respecto al primer mínimo local
1	10	9	36.87
2	10	427	31.43
3	10	441	26.41
4	10	114	29.49
5	10	164	30.88
6	10	458	27.78
7	10	391	19.31
8	10	450	18.43
9	10	213	17.66
10	10	481	44.89
		Promedio	28.31

Tabla 9

Resultados del problema para el caso de dos máquinas diferentes con $n = 40$

Problema	longitud de la lista tabú	Iteración en que se encontró la mejor solución	Porcentaje de mejora con respecto al primer mínimo local
1	11	449	28.83
2	11	339	61.71
3	11	353	29.38
4	11	468	38.79
5	11	360	55.39
6	11	260	18.78
7	11	329	26.33
8	11	135	29.61
9	11	421	19.30
10	11	138	31.83
		Promedio	34.00

Como puede observarse, el método de solución que se propone para el problema de dos máquinas diferentes obtiene mejores soluciones en un 25.53% en promedio para los problemas de 20 trabajos, con respecto al primer mínimo local. Sucede lo mismo para los problemas de 30 trabajos con un 28.31% de mejora en promedio y de un 34.00% en promedio de mejora para los problemas de 40 trabajos.

En el apéndice A, se presenta una gráfica que muestra el comportamiento típico del método de solución para el problema de dos máquinas diferentes en uno de los problemas que fueron evaluados.

En el apéndice B, se presenta a detalle, uno de los problemas que fueron evaluados por el método de solución propuesto.

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

En esta tesis se presenta una metodología de solución para el problema de programación de tareas, considerando los casos cuando se dispone de una sola máquina, dos máquinas idénticas y dos máquinas diferentes, de tal manera que se minimice la penalización total incurrida, debido a las entregas tardías o tempranas con respecto a su fecha de entrega.

La relevancia de la investigación presentada en esta tesis, radica en que existen muy pocos documentos en que se considera este problema en particular, especialmente para el problema de dos máquinas idénticas y diferentes. Como se mencionó anteriormente, el trabajo realizado por Rosas [91] es similar al presentado en esta tesis, con la diferencia en que Lira toma en cuenta los tiempos y costos de preparación, además de considerar dos máquinas diferentes, siendo éstas algunas de las recomendaciones

hechas por Rosas en su trabajo. Es por esto que no seria válido realizar comparaciones entre este par de trabajos.

El problema presentado en esta tesis pertenece a los problemas de tipo *NP-completo*, por lo tanto no es posible obtener una solución óptima en tiempo polinomial, razón por la cual se recurre a métodos heurísticos, como es el procedimiento metaheurístico de búsqueda tabú, el cual se describe en el capítulo 2. Puede decirse que la ventaja de utilizar búsqueda tabú radica en que evita quedar atrapado en un mínimo local, continuando la búsqueda hacia mejores soluciones.

Como se puede observar, búsqueda tabú es un procedimiento sencillo, por lo mismo es fácil de traducir a un lenguaje computacional.

El lenguaje computacional que se utilizó para implementar el método de solución propuesto en esta tesis fue el lenguaje C para DOS. La computadora que se utilizó fue una 486 con 4 megas en RAM. En lo que se refiere al tiempo de duración del algoritmo, se puede decir que varia dependiendo de las características de los problemas, por ejemplo, la cantidad de iteraciones o de tareas que se evalúan.

A continuación se presenta una tabla presentando los tiempos aproximados de duración del algoritmo de acuerdo a sus caracteristicas.

	Una sola máquina	Dos máquinas idénticas	Dos máquinas diferentes
20 tareas	5 seg.	18 seg.	1 min. 55 seg.
30 tareas	8 seg.	19 seg.	1 min. 57 seg.
40 tareas	18 seg.	23 seg.	2 min. 14 seg.

Se debe aclarar que estos tiempos fueron obtenidos considerando para cada tipo de problema 500 iteraciones.

5.2 Recomendaciones y trabajos futuros

Como se puede observar en el capítulo de resultados, la longitud de la lista tabú o memoria de corto tiempo, es fija para cada tipo de problema, siendo estos valores los que en promedio presentaban mejores resultados. Con esto se quiere decir, que se pueden tomar otros valores, ya que hasta el momento no existe una forma que nos indique como calcular la longitud de la memoria de corto plazo. El número de iteraciones que se utiliza para evaluar a todos los tipos de problemas, también es fijo, este valor fue tomado debido a que la mayoría de los problemas presentaban sus mejores resultados mucho antes de cumplir con este valor. Con esto se quiere decir, que se pueden tomar valores mucho más grandes de acuerdo con las características del problema que se presente.

Con lo dicho anteriormente, se quiere dar a entender que con cada problema se pueden tomar diferentes valores, tanto para la memoria de corto plazo como para el número de iteraciones a evaluar, y con esto tratar de encontrar la mejor solución para ese problema en particular.

Considerando como punto de partida el trabajo presentado en esta tesis, entre las recomendaciones que se proponen para ampliar el estudio son:

- Considerar otras técnicas para la elaboración de movimientos candidatos.
- Considerar otro criterio para la repartición de tareas a programar en cada máquina.
- Considerar más iteraciones para el algoritmo u otro criterio de parada.
- Considerar la extensión del problema para cuando se tengan más de dos máquinas idénticas.
- Considerar la extensión del problema para cuando se tengan más de dos máquinas diferentes.

BIBLIOGRAFÍA

[Díaz, et al., 96] Díaz, Glover, Ghaziri, González, Laguna, Moscano, Tseng. Optimización Heurística y Redes Neuronales, 1996, Ed. Paraninfo.

[Chen, 97] Zhi-Long Chen. "Scheduling with Batch Setup Times and Earliness-Tardiness Penalties", European Journal of Operational Research, Vol. 96, 518-537, 1997.

[Fry, et al., 87] Fry, T.,R. Armstrong, J. Blackstone, "Minimazing Weighted Absolute Deviation in Single Machine Scheduling", IEEE Transactions, 19, 1987. pp. 445-450.

[Rosas, 91] Rosas V., R.,"Diseño de un modelo de solución por medio del metaheuristico tabu search para el problema de programación de tareas en dos maquinas", Tesis de Maestría, ITESM. 1991.

[Bautista, 91] E. Bautista Vera., "Utilización de un Algoritmo Genético para resolver el problema de programación de tareas de dos máquinas", Tesis de Maestría, ITESM. 1991.

[Sidney, 77] Sidney, J. "Optimal Single-Machine Scheduling with Earlyness and Tardiness penalties", Operations Research, Vol. 25, 62-69, 1977.

[Lakshmanan, et al., 78] Lakshmanan L., R. L. Papineau y R. Rochette. "Optimal Single-machine Scheduling with Earliness and Tardiness Penalties", Operations Research, Vol.26, 1079-1082, 1978.

[Seidmann, et al., 82] Seidmann A., S.S. Panwalkar y M.L. Smith. "Optimal assignment of due-dates for a single processor scheduling

problem", International Journal of Production Research, Vol. 19, 393-399, 1982.

[Panwalkar, et al., 82] Panwalkar, S.S., M.L. Smith y A. Seidmann, "Common Due Date Assignment to Minimize Total Penalty for One Machine Scheduling Problem", Operations Research, Vol.30, 391-399, 1982.

[Kanet, 81] Kanet, J., "Minimizing the Avarage Deviation of Job Completion Times about a Common Due Date", Naval Research Logistic Quaterly, Vol. 28, 643-651, 1981.

LISTA DE TABLAS

Tabla **Página**

LISTA DE FIGURAS

Figura **Página**

APÉNDICES

APÉNDICE A

GRÁFICAS DE COMPORTAMIENTO

APÉNDICE A

GRÁFICA DE COMPORTAMIENTO

(Una sola máquina)

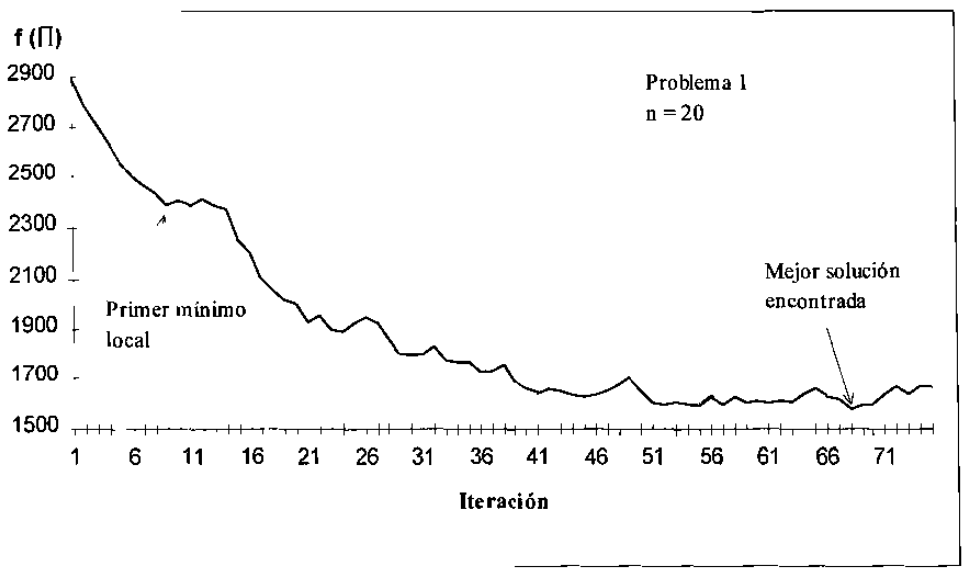


Figura 5. Comportamiento típico del método de solución para el problema de una sola máquina.

Esta gráfica nos muestra el comportamiento del método de solución que se propone para la solución del problema de una sola máquina. Como se puede observar, el método extiende la búsqueda de soluciones más allá del primer mínimo local encontrado. En este problema en particular, la solución fue encontrada en la iteración 68 y hubo un decremento en la solución de un 33.97% con respecto al primer mínimo local encontrado.

APÉNDICE A

GRÁFICA DE COMPORTAMIENTO

(Dos máquinas idénticas)

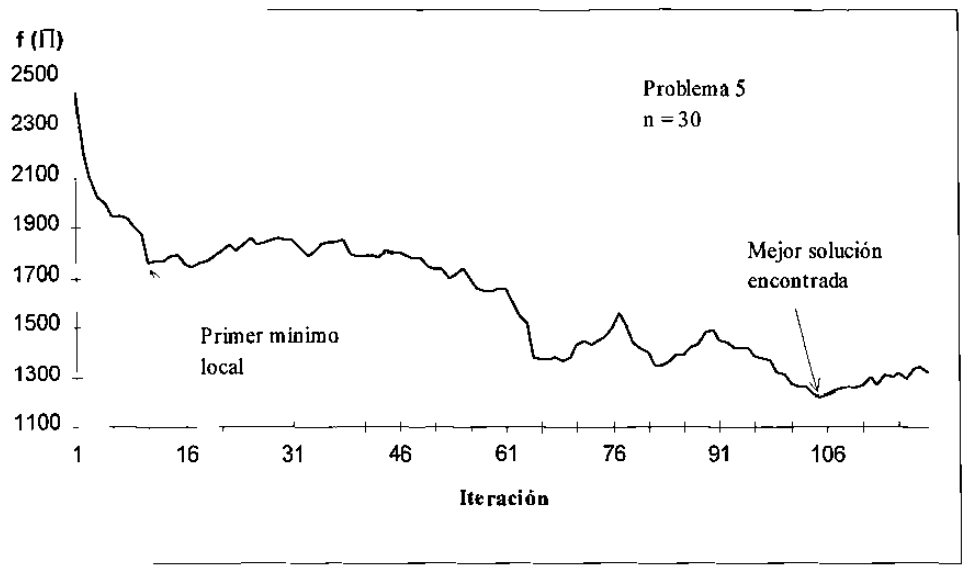


Figura 6. Comportamiento típico del método de solución para el problema de dos máquinas idénticas.

Esta gráfica nos muestra el comportamiento del método de solución que se propone para la solución del problema de dos máquinas idénticas. Como se puede observar nuevamente, el método extiende la búsqueda de soluciones más allá del primer mínimo local encontrado. En este problema en particular, la solución fue encontrada en la iteración 105 y hubo un decremento en la solución de un 30.82 % con respecto al primer mínimo local encontrado.

APÉNDICE A

GRÁFICA DE COMPORTAMIENTO

(Dos máquinas diferentes)

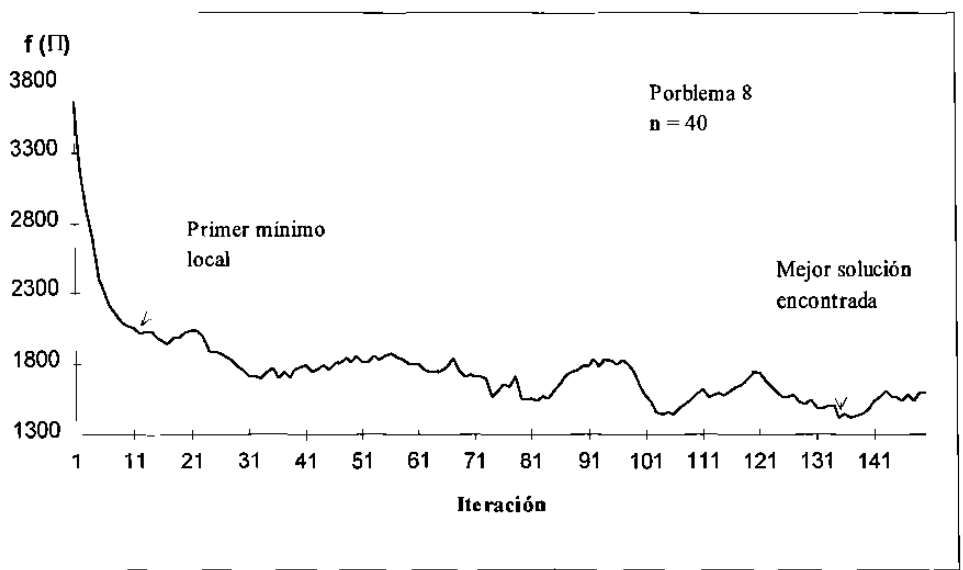


Figura 7. Comportamiento típico del método de solución para el problema de dos máquinas diferentes.

Esta gráfica nos muestra el comportamiento del método de solución que se propone para la solución del problema de dos máquinas diferentes. Como se puede observar nuevamente, el método extiende la búsqueda de soluciones más allá del primer mínimo local encontrado. En este problema en particular, la solución fue encontrada en la iteración 135 y hubo un decremento en la solución de un 29.61% con respecto al primer mínimo local encontrado.

APÉNDICE B

EJEMPLO

APÉNDICE B

EJEMPLO

A continuación se presenta en forma detallada uno de los problemas evaluados con el método de solución propuesto para el caso de dos máquinas diferentes, así como los resultados obtenidos.

El problema que se evalúa es el número 3, que corresponde a un problema con 20 tareas.

Los datos iniciales son los siguientes:

Trabajo	Tiempo de procesamiento en la máquina 1	Tiempo de procesamiento en la máquina 2	Fecha de entrega	Penalización por anticipación por unidad de tiempo	Penalización por retraso por unidad de tiempo
1	8	15	38	3	3
2	7	9	34	2	6
3	11	15	23	7	9
4	12	16	75	9	1
5	10	12	89	6	4
6	11	7	110	1	4
7	10	10	117	7	3
8	10	8	10	5	1
9	10	15	78	7	9
10	9	10	45	6	1
11	11	14	10	3	2
12	9	17	51	7	9
13	2	13	82	8	10
14	12	11	39	1	4
15	10	15	61	5	6
16	10	13	11	6	7
17	12	11	86	6	2
18	10	11	117	6	9
19	15	14	75	7	3
20	1	8	15	5	8

Costos de preparación de la máquina 1.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	3	4	4	5	4	4	3	3	7	7	**6**	5	3	3	6	3	7	7	6
2	4	0	6	3	5	4	7	3	3	3	5	7	5	6	5	7	5	7	4	3
3	**6**	7	0	3	6	3	5	3	5	6	6	3	7	7	7	4	3	6	4	3
4	7	4	5	0	3	6	7	5	6	5	7	6	6	5	5	6	7	7	7	6
5	7	6	4	3	0	5	3	4	6	6	5	7	7	4	6	4	4	**5**	6	6
6	6	3	7	3	3	0	5	5	4	5	6	5	4	5	6	7	4	5	3	6
7	4	5	6	6	7	3	0	4	3	6	3	7	7	3	3	3	6	7	6	3
8	4	5	3	3	3	6	7	0	7	5	7	3	4	4	3	6	5	4	5	6
9	3	3	5	4	4	5	5	5	0	3	4	4	**6**	5	6	5	3	6	4	7
10	5	5	4	6	3	4	3	4	5	0	3	7	3	6	4	7	3	3	5	4
11	4	4	5	7	6	6	6	6	3	7	0	6	6	3	4	3	3	3	6	4
12	5	7	6	6	6	3	5	5	3	3	5	0	6	5	**6**	5	4	7	7	4
13	5	5	6	4	**6**	3	7	6	3	7	3	4	0	4	7	5	5	4	3	7
14	6	3	4	5	5	4	4	6	5	7	4	4	7	0	5	7	4	6	3	6
15	7	3	7	5	3	6	5	3	**3**	5	7	3	5	7	0	4	4	6	3	5
16	7	4	3	5	4	5	3	7	3	5	7	6	3	7	3	0	4	5	3	**6**
17	6	6	7	3	3	3	3	7	6	3	7	5	7	7	4	4	0	3	7	5
18	6	5	4	4	7	4	4	5	7	3	6	5	3	3	3	5	7	0	6	7
19	4	4	7	7	7	7	7	4	7	3	5	6	5	5	5	4	5	3	0	7
20	7	6	**3**	4	4	3	4	5	4	5	5	7	5	5	5	7	4	7	4	0

Tiempos de preparación de la máquina 1.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	1	2	2	2	2	2	1	1	4	4	**4**	2	1	1	3	1	4	4	3
2	2	0	3	1	2	1	4	1	1	1	2	4	2	3	3	4	3	4	1	1
3	**4**	4	0	1	4	1	3	1	3	3	3	1	4	4	4	1	1	4	1	1
4	4	2	2	0	1	3	4	3	3	2	4	3	4	2	3	3	4	4	4	3
5	4	3	2	1	0	3	1	1	4	3	2	4	4	2	4	2	2	**2**	3	3
6	3	1	4	1	1	0	2	2	2	3	3	3	2	2	4	4	1	2	1	3
7	2	3	3	3	4	1	0	2	1	3	1	4	4	1	1	1	4	4	3	1
8	2	2	1	1	1	3	4	0	4	2	4	1	1	2	1	3	3	2	2	4
9	1	1	3	2	2	2	3	3	0	1	1	2	**4**	2	3	3	1	3	2	4
10	3	3	2	3	1	2	1	1	2	0	1	4	1	3	1	4	1	1	3	2
11	1	2	3	4	3	3	3	3	1	4	0	3	4	1	2	1	1	1	3	1
12	2	4	4	3	3	1	3	2	1	1	2	0	3	2	**4**	3	2	4	4	2
13	3	2	3	2	**3**	1	4	3	1	4	1	2	0	1	4	3	3	1	1	4
14	4	1	1	2	3	2	1	4	2	4	2	1	4	0	2	4	1	3	1	3
15	4	1	4	2	1	3	2	1	**1**	2	4	1	2	4	0	2	1	3	1	3
16	4	2	1	3	1	3	1	4	1	2	4	3	1	4	1	0	1	3	1	**3**
17	3	4	4	1	1	1	1	4	4	1	4	2	4	4	2	2	0	1	4	3
18	3	3	2	2	4	1	2	2	4	1	3	2	1	1	1	3	4	0	3	4
19	2	2	4	4	4	4	4	2	4	1	2	3	2	2	3	2	2	1	0	4
20	4	3	**1**	1	2	1	2	3	1	2	3	4	2	2	3	4	2	4	2	0

Costos de preparación de la máquina 2.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	6	5	7	8	9	9	5	7	6	5	6	8	6	7	5	8	9	7	5
2	6	0	8	9	8	9	5	6	7	7	7	8	7	**5**	7	9	8	6	7	8
3	7	7	0	7	7	8	5	7	8	8	7	8	7	9	7	7	5	6	5	7
4	7	7	8	0	8	8	**5**	7	8	8	9	6	7	7	7	6	5	6	8	6
5	8	9	6	5	0	6	9	7	6	7	8	9	9	9	5	6	8	6	9	9
6	7	6	7	**9**	5	0	5	7	8	8	6	7	9	6	7	6	7	5	9	9
7	5	8	5	7	8	8	0	8	9	5	9	5	5	8	6	8	9	7	7	7
8	6	7	7	9	6	**5**	5	0	5	8	8	8	6	5	8	7	9	8	7	7
9	9	7	6	6	8	9	6	6	0	9	6	8	9	9	9	9	7	9	9	6
10	9	6	7	9	8	7	7	7	9	0	9	5	6	5	9	7	7	6	**5**	9
11	5	**5**	9	6	6	6	6	7	8	8	0	9	5	7	8	5	8	5	5	5
12	5	7	9	6	7	5	5	5	8	9	9	0	7	9	6	9	8	9	5	7
13	7	9	5	8	9	6	6	9	8	8	8	7	0	9	7	9	7	6	9	9
14	6	8	9	8	8	6	9	8	9	**8**	6	6	9	0	6	8	5	9	5	8
15	8	6	9	5	5	6	5	5	6	8	7	5	7	7	0	9	6	7	7	5
16	6	8	6	9	7	9	6	7	9	7	8	7	9	5	5	0	9	5	9	7
17	8	7	9	4	9	7	8	**8**	5	7	8	8	5	9	5	5	0	7	8	5
18	7	8	6	6	7	6	8	7	8	8	5	9	9	8	8	8	8	0	7	9
19	9	8	6	6	9	5	5	5	8	6	5	6	6	6	8	6	**6**	9	0	8
20	6	7	5	6	7	6	8	8	8	8	5	8	9	5	9	8	6	5	8	0

Tiempos de preparación de la máquina 2.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	3	2	4	5	6	6	2	4	3	2	3	5	3	4	2	5	6	4	2
2	3	0	5	6	5	6	2	3	4	4	4	5	4	**2**	4	6	5	3	4	5
3	4	4	0	4	4	5	2	4	5	5	4	5	4	6	4	4	2	3	2	4
4	4	4	5	0	5	5	**2**	4	5	5	6	3	4	4	4	3	2	3	5	3
5	5	6	3	2	0	3	6	4	3	4	5	6	6	6	2	3	5	3	6	6
6	4	3	4	**6**	2	0	2	4	5	5	3	4	6	3	4	3	4	2	6	6
7	2	5	2	4	5	5	0	5	6	2	6	2	2	5	3	5	6	4	4	4
8	3	4	4	6	3	**2**	2	0	2	5	5	5	3	2	5	4	6	5	4	4
9	6	4	3	3	5	6	3	3	0	6	3	5	6	6	6	6	4	6	6	3
10	6	3	4	6	5	4	4	4	6	0	6	2	3	2	6	4	4	3	**2**	6
11	2	**2**	6	3	3	3	3	4	5	5	0	6	2	4	5	2	5	2	2	2
12	2	4	6	3	4	2	2	2	5	6	6	0	4	6	3	6	5	6	2	4
13	4	6	2	5	6	3	3	6	5	5	5	4	0	6	4	6	4	3	6	6
14	3	5	6	5	5	3	6	5	6	**5**	3	3	6	0	3	5	2	6	2	5
15	5	3	6	2	2	3	2	2	3	5	4	2	4	4	0	6	3	4	4	2
16	3	5	3	6	4	6	3	4	6	4	5	4	6	2	2	0	6	2	6	4
17	5	4	6	1	6	4	5	**5**	2	4	5	5	2	6	2	2	0	4	5	2
18	4	5	3	3	4	3	5	4	5	5	2	6	6	5	5	5	5	0	4	6
19	6	5	3	3	6	2	2	2	5	3	2	3	3	3	5	3	**3**	6	0	5
20	3	4	2	3	4	3	5	5	5	5	2	5	6	2	6	5	3	2	5	0

A continuación se presenta en forma detallada la solución obtenida para el problema número 3.

La solución fue encontrada en la iteración 79, utilizándose una longitud de lista tabú igual a 9 y hubo un decremento en la solución de un 40.74% con respecto al primer mínimo local encontrado.

El costo total de las secuencias 1 y 2 por concepto de penalizaciones y costos de preparación es de 483, siendo 147 el costo en la máquina 1 y 336 el costo en la máquina 2. Es decir, en la máquina 1, la secuencia de trabajos 16, 20, 3, 1, 12, 15, 9, 13, 5 y 18 tiene un costo por conceptos de penalizaciones de 100 y un costo por concepto de preparación de 47, mientras que en la máquina 2, la secuencia de trabajos 11, 2, 14, 10, 19, 17, 8, 6, 4 y 7 tiene un costo por conceptos de penalizaciones de 280 y un costo por concepto de preparación de 56. En las tablas 10 y 11 se presenta en forma detallada los resultados obtenidos para las mejores secuencias encontradas.

Tabla 10

Resultados de la secuencia de trabajos asignados a la máquina 1

Trabajo	Tiempo de preparación	Tiempo de ocio	Tiempo de procesamiento en la máquina 1	Terminación	Fecha de entrega	Penalización
16	0	0	10	10	11	6
20	3	0	1	14	15	5
3	1	0	11	26	23	27
1	4	0	8	38	38	0
12	4	0	9	51	51	0
15	4	0	10	65	61	24
9	1	0	10	76	78	14
13	4	0	2	82	82	0
5	3	0	10	95	89	24
18	2	10	10	117	117	0
					Total	100

Tabla 11

Resultados de la secuencia de trabajos asignados a la máquina 2

Trabajo	Tiempo de preparación	Tiempo de ocio	Tiempo de procesamiento en la máquina 2	Terminación	Fecha de entrega	Penalización
11	0	2	14	16	10	12
2	2	2	9	27	34	14
14	2	2	11	40	39	4
10	5	2	10	55	45	10
19	2	5	14	74	75	7
17	3	5	11	88	86	4
8	5	5	8	101	10	91
6	2	5	7	110	110	0
4	6	5	16	132	75	57
7	2	5	10	144	117	81
					Total	280

GLOSARIO

JIT	Justo a Tiempo (del inglés *Just in Time*).
Max	Máximo.
Min	Mínimo.
NP-completo	No Polinomial - completo.
PPT	Problema de Programación de Tareas.
Schedule	Programa, horario, programar, poner en una lista o establecer un horario.
TS	Búsqueda Tabú (del inglés *Tabu Search*).

RESUMEN AUTOBIOGRÁFICO

José Luis Lira de la Garza

Candidato para el Grado de

Maestro en Ciencias de la Administración

con Especialidad en Sistemas

Tesis: MODELO DE SOLUCION PARA EL PROBLEMA DE PROGRAMACION DE TAREAS EN DOS MAQUINAS POR MEDIO DEL METAHEURISTICO BUSQUEDA TABU.

Campo de Estudio: Sistemas.

Biografía:

Nacido en Monterrey, Nuevo León, el 15 de Marzo de 1975, hijo de José Luis Lira Martínez y Rosa María de la Garza de Lira.

Educación:

Egresado de la Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León; grado obtenido de Ingeniero Administrador de Sistemas en 1996, con título honorífico.

Experiencia Profesional:

Maestro por horas de la Facultad de Contaduría Pública y Administración de la U.A.N.L. y Maestro Instructor de la Coordinación de Educación Continua de la Facultad de Ingeniería Mecánica y Eléctrica de la U.A.N.L.