UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

DIVISION DE ESTUDIOS DE POSTGRADO

DESARROLLO E IMPLEMENTACION DE UN ALGORITMO GENETICO QUE RESUELVE EL PROBLEMA DE PROGRAMACION DE TAREAS EN DOS MAQUINAS

POR

ING. VICTOR HUGO ORDAZ GAITAN

TESIS

EN OPCION AL GRADO DE

MAESTRO EN CIENCIAS DE LA ADMINISTRACION

CON ESPECIALIDAD EN SISTEMAS

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

DESARROLLO E IMPLEMENTACION DE UN ALGORITMO GENETICO QUE RESUELVE EL PROBLEMA DE PROGRAMACION DE TAREAS EN DOS MAQUINAS

VHOG

UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

DIVISION DE ESTUDIOS DE POSTGRADO

DESARROLLO E IMPLEMENTACION DE UN ALGORITMO GENETICO QUE RESUELVE EL PROBLEMA DE PROGRAMACION DE TAREAS EN DOS MAQUINAS

POR

ING. VICTOR HUGO ORDAZ GAITAN

T E S I S

EN OPCION AL GRADO DE

MAESTRO EN CIENCIAS DE LA ADMINISTRACION

CON ESPECIALIDAD EN SISTEMAS

SAN NICOLAS DE LOS GARZA, N. L. MARZO 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 DESARROLLO E IMPLEMENTACION DE UN ALGORITMO GENETICO QUE RESUELVE EL PROBLEMA DE PROGRAMACION DE TAREAS EN DOS MAQUINAS, realizada por el Ing. Victor Hugo Ordaz Gaitán, 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 Alvarez 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, Nuevo León, a 15 de Febrero de 1999

DEDICATORIAS

A mis padres

José Isaac Ordaz Nava y María Antonieta Gaytán de Ordaz,
por su amor y apoyo incondicional hacia mí

A mis hermanos

Mónica y Raúl, Isaac y Maribel,

gracias por su amor y por sus consejos

A mi sobrino

David,
por la alegría que siempre me transmite

A mi abuelita

María Luisa,
por su amor y fe que superan cualquier obstáculo

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a la Dra. Ada Margarita Alvarez Socarrás, asesora de esta tesis, por su ayuda, correcciones, y recomendaciones para la realización de esta tesis.

También agradezco a los coasesores de esta tesis, Dr. José Luis Martínez Flores y M. C. José Luis Castillo Ocañas, por sus revisiones y correcciones necesarias para el buen término de este trabajo.

De igual forma agradezco al Consejo Nacional de Ciencia y Tecnología por el apoyo económico otorgado para la realización de mis estudios. Así como también agradezco a la Universidad Autónoma de Nuevo León, en especial a la Facultad de Ingeniería Mecánica y Eléctrica, por las facilidades otorgadas para la realización de mis estudios de postgrado.

Agradezco también a todos los maestros que compartieron conmigo sus conocimientos, ayudando con ello a mi formación profesional.

Por último, deseo agradecer de forma especial a mis compañeros y amigos del Doctorado en Ingeniería de Sistemas (DIS), quienes me ayudaron en todo momento y quienes, principalmente, comparten conmigo su valiosa amistad.

RESUMEN

Victor Hugo Ordaz Gaitán **Fecha de Graduación: Marzo, 1999**

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Título del Estudio: **DESARROLLO E IMPLEMENTACIÓN DE UN ALGORITMO GENÉTICO QUE RESUELVE EL PROBLEMA DE PROGRAMACIÓN DE TAREAS EN DOS MÁQUINAS**

Número de Páginas: 61 **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 algoritmo 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 algoritmo heurístico, el Algoritmo Genético (AG). Los algoritmos genéticos son algoritmos de búsqueda basados en los mecanismos de la selección natural y en la genética natural. Los investigadores los han utilizado como metaheurística para optimización, debido a que pueden encontrar soluciones aproximadas a problemas de gran complejidad computacional. Por todo esto, se desarrolló e implementó computacionalmente el AG para dar solución al PPT, con lo cual se observa la conveniencia de la utilización de los algoritmos genéticos para resolver este tipo de problemas.

Contribuciones y Conclusiones: El AG se implementó utilizando una representación genética del PPT mediante la combinación de un alfabeto binario y uno no binario. Se realizaron simulaciones y se obtuvieron resultados que muestran que, en general, las soluciones encontradas por el AG implementado se consideran buenas con respecto a los resultados reportados en la literatura. Se concluye que es conveniente la utilización de los algoritmos genéticos para la solución de problemas como el PPT.

FIRMA DEL ASESOR: _____

Dra. Ada Margarita Alvarez Socarrás

TABLA DE CONTENIDO

Capítulo **Página**

Capítulo **Página**

LISTA DE TABLAS

Tabla **Página**

Tabla **Página**

LISTA DE FIGURAS

Figura **Página**

NOMENCLATURA

AG	Algoritmo Genético
AGs	Algoritmos Genéticos
DNA	Ácido Desoxirribonucleico
JIT	Justo a Tiempo (del inglés *Just in Time*)
OX	Cruzamiento de Orden (del inglés *Order Crossover*)
PMX	Cruzamiento de Emparejamiento Parcial (del inglés *Partially Matched Crossover*)
PPT	Problema de Programación de Tareas
TSP	Problema del Vendedor viajero (del inglés *Traveling Salesman Problem*)

CAPITULO 1

INTRODUCCION

1.1 Planteamiento del problema

Existen variados y numerosos problemas de programación (en inglés *scheduling*) entre los que se encuentra 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. Cada una de las tareas tiene una fecha de entrega que cumplir y un tiempo de procesamiento. También se tiene un tiempo y costo de preparación (en inglés *setup*) de la máquina antes de procesar cada tarea. Se penalizan tanto las entregas tardías como las entregas tempranas o prematuras, esto, debido a que se aplica un enfoque Justo a Tiempo (JIT, *Just In Time*).

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 algoritmo que proporcione soluciones óptimas para este tipo de problemas. Ha sido abordado mediante distintas técnicas, tales como Búsqueda Tabú [Rosas, 91], Algoritmos Genéticos (AGs) [Bautista, 91], o mediante algoritmos desarrollados especialmente para resolverlo [Chen, 97].

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 algoritmo heurístico, el Algoritmo Genético (AG). Los algoritmos genéticos son algoritmos de búsqueda basados en los mecanismos de la selección natural y en la genética natural [Golberg, 89]. Un AG estándar opera sobre una población de individuos o cromosomas (posibles soluciones del problema) y se compone de los siguientes operadores: selección de individuos sobrevivientes, cruzamiento y mutación. Aplicar una sola vez los operadores constituye una generación. La población de individuos evoluciona durante varias generaciones mejorando a los individuos. De este modo se puede encontrar una solución óptima o aproximada a la óptima.

1.2 Objetivo de la investigación

Esta tesis tiene los siguientes objetivos:

1) Desarrollo e implementación computacional de un algoritmo genético para resolver el PPT en una máquina.
2) Desarrollo e implementación computacional de un algoritmo genético para resolver el PPT en dos máquinas iguales.
3) Desarrollo e implementación computacional de un algoritmo genético para resolver el PPT en dos máquinas diferentes.

Además, se observará la conveniencia de la utilización de los algoritmos genéticos para resolver este tipo de problemas.

1.3 Justificación y limitaciones

El PPT es un problema de secuenciación cuyas características lo convierten en un complejo problema combinatorio. Está considerado dentro de la clase NP-completo [Garey y Johnson, 79]. Consecuentemente, no existen algoritmos que garanticen obtener soluciones óptimas para este tipo de problemas en un tiempo de ejecución polinomial.

Resulta interesante aplicar un algoritmo heurístico, tal como el AG, para resolver el PPT. Los investigadores han utilizado los AGs como metaheurística para optimización, debido a que pueden encontrar soluciones aproximadas a problemas de gran complejidad computacional [Díaz et al., 96].

Además, en años recientes, los AGs han sido utilizados por varios investigadores para resolver diversos problemas de *scheduling* [Husbands, 95]. Aún así, no se encontraron trabajos anteriores que aborden el PPT de la misma forma en la que aquí se presenta.

La principal limitación presente en los programas computacionales desarrollados es que no se puede manejar una cantidad de tareas mayor de 70, en el caso de una máquina, y de 50, en el caso de dos máquinas. Quizá esto se pueda solucionar si se maneja memoria dinámica, lo cual no se realizó en ninguno de los programas.

1.4 Metodología

El principal interés de este trabajo es resolver el PPT en dos máquinas diferentes, para ello se comenzó primeramente por solucionar el PPT en una máquina, después en dos máquinas iguales y por último en dos máquinas diferentes.

Se analizó el problema para encontrar una representación del mismo, que pudiera ser utilizada en el AG. Se decidió utilizar una codificación que combinara un alfabeto binario y uno no binario. Con esta representación, se procedió a realizar un programa computacional en lenguaje C para el caso de una máquina. Posteriormente, se realizaron las modificaciones necesarias sobre el programa computacional, con la intención de resolver el problema en dos máquinas iguales y después en dos diferentes.

Se realizaron las pruebas para evaluar el funcionamiento del AG implementado y por último se obtuvieron estadísticas para evaluar su comportamiento.

1.5 Revisión bibliográfica

En la literatura revisada se encontraron varios problemas de *scheduling* que han sido abordados por medio de AGs. Entre los principales problemas abordados por los investigadores se encuentran el problema de *Job Shop* [Nakano, 91] y el problema de *Flow Shop* [Kleveland y Smith, 89]. Ambos encontraron soluciones, consideradas buenas, utilizando AGs. Tanto el problema de *Job Shop* como el de *Flow Shop* son problemas que caen dentro de la clase NP-completo, sin embargo, difieren en gran medida del PPT .

También se encontró que se han realizado investigaciones que abordan problemas similares al tratado en este estudio. En una de ellas [Chen, 97], se 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. Proponen un algoritmo de programación dinámico polinomial para resolver el problema, pero con solo dos lotes de trabajos, en donde todos los trabajos de cada lote tienen las misma fecha de entrega y, por consiguiente, las mismas penalizaciones. También, consideran un caso especial para

ambos problemas donde las fechas de entrega para los diferentes lotes son todas iguales. Bajo este caso especial, proponen 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.

En otra investigación previa [Rosas, 91], se resuelve el PPT en dos máquinas mediante la utilización del metaheurístico Búsqueda Tabú (conocido en inglés como *Tabu Search*). Las diferencias principales respecto al presente trabajo son que maneja un número inferior de tareas y no considera varias de las restricciones tales como costos y tiempos de preparación de las máquinas. Además, solamente llega a solucionar el PPT hasta el caso de dos máquinas iguales.

Otro trabajo previo [Bautista, 91] resuelve el PPT en dos máquinas mediante la utilización de un AG. En dicho trabajo, se utiliza una codificación completamente binaria para resolver el PPT, tanto en el caso de una máquina como en el de dos, aunque en este último se trata de dos máquinas iguales. Como propuesta futura propone la extensión del PPT a n máquinas. También difiere del PPT aquí presentado en que no considera costos de preparación en el caso de dos máquinas. De cualquier manera, el trabajo de Bautista sirve de base para realizar las comparaciones necesarias para evaluar el AG implementado.

1.6 Guía de la tesis

Esta tesis se conduce de la siguiente manera: en el capítulo 2 se da una descripción formal del problema de programación de tareas. También, se establece la complejidad del mismo. En el capítulo 3 se da una descripción de los algoritmos genéticos y también se definen conceptos y términos relacionados con ellos. Además, se explica el

funcionamiento de los operadores genéticos y se dan ejemplos de varios tipos de operadores.

En el capítulo 4 se explicará la forma en que se desarrolló e implementó el algoritmo genético para solucionar el PPT. Se discuten las razones por las cuales se selecciona el tipo de representación, así como también los tipos de operadores genéticos utilizados.

En el capítulo 5 se analizan y muestran los resultados de las simulaciones realizadas. Por último, en el capítulo 6 se expresan las conclusiones a las que se llegan y se presentan las sugerencias para la realización de trabajos futuros.

CAPITULO 2

EL PROBLEMA DE PROGRAMACION DE TAREAS

2.1 Introducción

En el capítulo anterior, entre otras cosas, se dio una breve explicación del PPT. En este capítulo lo definiremos formalmente. En la sección 2.2 se describe el PPT en forma general. En las secciones 2.2.1 y 2.2.2 se realiza la formulación matemática del PPT en una máquina y dos máquinas respectivamente. Por ultimo, en la sección 2.3 se verá la complejidad del PPT y la clasificación a la que pertenece de acuerdo a esta complejidad.

2.2 Descripción del Problema

El PPT, en su modelo más sencillo, implica la asignación de un conjunto de trabajos o tareas en una máquina en una secuencia óptima tal que se minimicen los costos de la secuencia de tareas programadas.

El PPT que se aborda en esta tesis, independientemente de la cantidad de máquinas que se utilicen, tiene las siguientes características. Supóngase que se tiene un conjunto de tareas que deben se procesadas. Cada una de las tareas tiene una fecha de entrega que cumplir y un tiempo de procesamiento. También existe un tiempo y costo de preparación de la máquina, el cual no depende solamente de la tarea que se va a ejecutar, sino también de la que se ejecutó anteriormente. Esto ocasiona que el costo y tiempo de preparación sea único para cada par de tareas. Además, como se mencionó en el capítulo 1, se aplica un enfoque JIT, esto es, se desea que el procesamiento de cada tarea se termine lo más cercano posible a su fecha de entrega con el propósito de minimizar el inventario existente. Por consiguiente, se penalizan tanto las entregas tardías como las entregas tempranas o prematuras.

El objetivo consiste en minimizar el costo total de la secuencia de tareas, el cual se genera a partir de los costos de preparación, aunados a los costos de las penalizaciones.

2.2.1 El PPT en una máquina

El problema, para el caso de una máquina, se formula de la siguiente manera. Dado un conjunto T de n tareas, para cada tarea i (i=1, 2, ..., n) se tiene:

- p_i: tiempo de procesamiento;
- d_i: fecha o tiempo de entrega;
- C_i: tiempo de entrega real;
- α_i: castigo por entrega prematura por unidad de tiempo;
- β_i: castigo por entrega tardía por unidad de tiempo;

y denotemos por

- E_i: cantidad de tiempo prematuro de la tarea i (i=1, 2, …, n);
- T_i: cantidad de tiempo de retraso de la tarea i (i=1, 2, …, n).

Además, para i,j=1, 2, …, n también se tiene,

- $sc(t_i,t_j)$: costo de preparación de la máquina para cada par de tareas $t_i, t_j \in T$;
- $st(t_i,t_j)$: tiempo de preparación de la máquina para cada par de tareas $t_i, t_j \in T$.

Se asume lo siguiente:

- todas las tareas están disponibles en el tiempo 0,
- todas las tareas pueden procesarse sin que dependan del procesamiento previo de otra,
- $E_i=\max(0,d_i-C_i)$,
- $T_i=\max(0,C_i-d_i)$,
- $sc(t_i,t_j) = 0$ y $st(t_i,t_j) = 0$ cuando i=j,
- $\alpha_i > 0$ y $\beta_i > 0$.

Se desea encontrar una permutación $t_{\sigma(1)}, t_{\sigma(2)}, \ldots, t_{\sigma(n)}$ de T tal que el costo de esa secuencia de tareas sea mínimo, esto es, se desea

$$\text{minimizar } f(\sigma) = \sum_{i=1}^{n-1} sc(t_{\sigma(i)}, t_{\sigma(i+1)}) + \sum_{i=1}^{n} (\alpha_i E_i + \beta_i T_i)$$

2.2.2 El PPT en dos máquinas

En la formulación del PPT en dos máquinas pueden considerarse dos variantes, que las máquinas sean idénticas o que sean diferentes. Ambos casos son una extensión del PPT en una máquina.

Para el caso de dos máquinas idénticas no existe variación significativa en comparación con el caso de una máquina, es decir, las características de las tareas no varían de una máquina a otra. El caso de máquinas diferentes se presenta, comúnmente, cuando alguna de las dos máquinas utilizadas para procesar las tareas es más nueva o con características diferentes. Esto provoca que los costos y tiempos de preparación varíen de una máquina a otra, y que varíen los tiempos de procesamiento de las tareas dependiendo de la máquina en que se procesen.

El problema, para el caso de dos máquinas, se formula de la siguiente manera. Se tiene un conjunto T de n tareas, y para cada tarea i (i=1, 2, ..., n) se tiene:

- p_{ik}: tiempo de procesamiento en la máquina k (k=1,2);
- d_i: fecha o tiempo de entrega;
- C_i: tiempo de entrega real;
- α_i: castigo por entrega prematura por unidad de tiempo;
- β_i: castigo por entrega tardía por unidad de tiempo.

Al igual que el caso anterior denotaremos por E_i, T_i la cantidad de tiempo prematuro y tiempo de retraso respectivamente, de la tarea i.

Además, para i,j=1, 2, …, n también se tiene,

- $sc_k(t_i,t_j)$: costo de preparación de la máquina k para cada par de tareas t_i, $t_j \in T$, k=1,2;
- $st_k(t_i,t_j)$: tiempo de preparación de la máquina k para cada par de tareas t_i, $t_j \in T$, k=1,2.

Y denotaremos por:

- T1: conjunto de tareas que se procesan en la máquina 1;
- T2: conjunto de tareas que se procesan en la máquina 2;
- n_1: cantidad de tareas que se procesan en la máquina 1;
- n_2: cantidad de tareas que se procesan en la máquina 2.

Se asume lo siguiente:

- todas las tareas están disponibles en el tiempo 0;
- las tareas se pueden procesar en cualquiera de las dos máquinas;
- todas las tareas pueden procesarse sin que dependan del procesamiento previo de otra;
- E_i y T_i se calculan de la misma forma que en el caso de una máquina (i= 1, 2, …, n);
- $sc_k(t_i,t_j) = 0$ y $st_k(t_i,t_j) = 0$ cuando i=j (i,j=1, 2, …, n);
- $T1 \cup T2 = T$;
- $T1 \cap T2 = \varnothing$;
- $n_1 + n_2 = n$;
- $\alpha_i > 0$ y $\beta_i > 0$, i = 1, …, n.

Se desea encontrar una secuencia de tareas para cada una de las máquinas, de modo que el costo total sea mínimo. Por lo tanto, se requiere encontrar dos permutaciones,

$$t_{\sigma 1(1)}, t_{\sigma 1(2)}, \ldots, t_{\sigma 1(n1)} \text{ para la máquina 1 y}$$
$$t_{\sigma 2(1)}, t_{\sigma 2(2)}, \ldots, t_{\sigma 2(n2)}, \text{ para la máquina 2,}$$

que minimicen la siguiente función objetivo,

$$f(\sigma) = \sum_{k=1}^{2} \sum_{i=1}^{n_k - 1} sc_k(t_{\sigma k(i)}, t_{\sigma k(i+1)}) + \sum_{i=1}^{n} (\alpha_i E_i + \beta_i T_i)$$

Es obvio que en el caso de dos máquinas iguales, $sc_1(t_i,t_j) = sc_2(t_i,t_j)$ y $st_1(t_i,t_j) = st_2(t_i,t_j)$, lo cual no sigue siendo cierto en el caso de dos máquinas diferentes.

2.3 Complejidad del PPT

Claramente, el PPT es un problema de tipo combinatorio. Incluir la filosofía JIT, obliga a utilizar tiempos muertos en donde la máquina no procese ninguna tarea, debido a que si se termina una tarea antes de su fecha de entrega, existirá un castigo por entrega prematura. Esto provoca el conflicto entre tareas, porque se podría provocar que al mantener la máquina sin hacer nada, otras tareas se retrasen, aumentando su castigo por entrega tardía.

Existen otros tipos de problemas de programación de tareas, algunos no consideran tantas restricciones o difieren en el objetivo del que se presenta aquí. Garey y Johnson [1979] presentan la formulación de algunos de ellos, y les asignan denominaciones tales como Secuenciación de Tareas en un Procesador considerando fechas límites, Secuenciación de Tareas minimizando las Tareas Tardías, Secuenciación de Tareas minimizando el Tiempo Total de Procesamiento, entre otros.

Todos los problemas mencionados en el párrafo anterior, así como el PPT aquí tratado, pertenecen a la clase de problemas NP-completo. Para este tipo de problemas no se han encontrado algoritmos que puedan resolverlos en un tiempo de ejecución polinomial [Garey y Johnson, 79].

2.4 Resumen

En este capítulo se definió formalmente el PPT, se presentaron sus características y se realizó la formulación matemática del mismo tanto para el caso en una máquina,

como para el caso del PPT en dos máquinas. Para este último caso se aclararon las diferencias existentes entre tener dos máquinas iguales o dos diferentes.

Al final del capítulo se estableció que el PPT es un complejo problema combinatorio que cae dentro de la clase NP-completo. En el siguiente capítulo se verán aspectos relacionados con los algoritmos genéticos, que es la metodología que utilizaremos para resolver los problemas formulados en esta tesis.

CAPITULO 3

ALGORITMOS GENETICOS

3.1 Introducción

En el capítulo anterior se estableció la definición formal del PPT. En el presente capítulo se observarán definiciones y aspectos relacionados con los algoritmos genéticos.

El capítulo se conduce como sigue: en la sección 3.2 se da una descripción de los algoritmos genéticos, así como también se definen algunos de los términos relacionados con ellos, en la sección 3.3 se presentan los componentes de los AGs y en la sección 3.4, se explica la representación genética de problemas.

Posteriormente se definen y se dan ejemplos de los operadores genéticos selección, cruzamiento y mutación, en las secciones 3.5, 3.6 y 3.7 respectivamente.

3.2 Descripción de los algoritmos genéticos

Los algoritmos genéticos son algoritmos de búsqueda basados en los mecanismos de la selección natural y en la genética natural [Golberg, 89]. Los AGs fueron desarrollados

por Holland, sus colegas, y sus estudiantes en la Universidad de Michigan en los 60's y 70's. El objetivo original de Holland no fue diseñar algoritmos para resolver problemas específicos, sino para estudiar formalmente el fenómeno de la adaptación tal como ocurre en la naturaleza y para desarrollar maneras en las cuales los mecanismos de la adaptación natural puedan ser importados dentro de sistemas computacionales [Mitchel, 96].

Ha sido probado teóricamente y empíricamente que los AGs proveen una búsqueda robusta en espacios complejos. Se diferencian de la mayoría de los procedimientos de búsqueda y de optimización en cuatro aspectos [Golberg, 89]:

1. Trabajan con una codificación del conjunto de parámetros, no con los parámetros mismos.
2. Realizan una búsqueda a partir de una población de puntos, no únicamente desde un solo punto.
3. Utilizan información de la función objetivo, no información sobre la derivada u otro conocimiento auxiliar.
4. Utilizan reglas de transición probabilísticas, no reglas deterministas.

Los AGs requieren que el conjunto de parámetros del problema de optimización sean codificados como una cadena de longitud finita de algún alfabeto finito.

3.2.1 Terminología

En el contexto de los AGs se utilizan términos biológicos en analogía con la biología real [Mitchel, 96]. Todos los organismos vivientes están constituidos por células, y cada célula contiene el mismo conjunto de uno o más *cromosomas* -cadena de DNA- que sirven como "huella digital" para el organismo. Conceptualmente, un cromosoma puede ser dividido en *genes* -bloques funcionales de DNA-, cada uno de los

cuales codifica una proteína particular. En general, se puede pensar en un gene como la codificación de una característica o rasgo, tal como el color de los ojos. Los diferentes posibles valores para una característica (por ejemplo, azul, café, verde) son llamados *alelos*. Cada gene se localiza en un *sitio* (posición) particular del cromosoma.

Durante la reproducción sexual, ocurre la *recombinación* (o *cruzamiento*): en cada padre, los genes son intercambiados entre cada par de cromosomas para formar un *gameto* (un cromosoma sencillo). Los hijos son susceptibles a la *mutación*, en la cual núcleos sencillos (pequeños pedazos elementales de DNA) son cambiados de un padre al hijo. La *aptitud* de un organismo es definida, típicamente, como la probabilidad de que el organismo viva para reproducirse (viabilidad) o se define en función del número de hijos que el organismo tenga (fertilidad).

En los AGs, el término *cromosoma* se refiere a una solución candidata del problema, usualmente codificada como una cadena de bits. A los cromosomas se les conoce como *individuos*. Los *genes* son bits sencillos, o bloques cortos de bits adyacentes, que codifican un elemento particular de la solución candidata (por ejemplo, en el contexto de la optimización de una función con múltiples parámetros, los bits que codifican un parámetro en particular deben ser considerados un gene). Un *alelo*, en una cadena de bits, es 0 o 1; para alfabetos más grandes, se permiten más alelos en cada *sitio*. Típicamente, el *cruzamiento* consiste de un intercambio de material genético entre dos cromosomas padres. La *mutación* consiste en cambiar el valor del bit de un sitio seleccionado aleatoriamente (o, para alfabetos grandes, reemplazar el símbolo de un sitio seleccionado aleatoriamente con otro símbolo seleccionado aleatoriamente).

3.3 Componentes de los Algoritmos Genéticos

Un AG para un problema en particular tiene los siguientes cinco componentes [Michalewicz, 92]:

- una representación genética para las soluciones potenciales del problema,
- una forma de crear una población inicial de las soluciones potenciales,
- una función de evaluación que tome el papel del medio ambiente, evaluando las soluciones en términos de su aptitud,
- operadores genéticos que alteren la composición de los hijos,
- valores para varios de los parámetros que los AGs utilizan (tamaño de la población, probabilidades de aplicar los operadores genéticos, etc.)

También se puede decir que la mayoría de los métodos llamados AGs tienen al menos los siguientes elementos en común: poblaciones de cromosomas, selección de acuerdo a la aptitud, cruzamiento para producir nuevos hijos, y mutación aleatoria de los nuevos hijos [Mitchel, 96].

Es útil ver la ejecución del algoritmo genético como un proceso de dos etapas [Whitley, 93]. Inicia con una población actual. La selección se aplica a la población actual para crear una población intermedia. Entonces, el cruzamiento y mutación se aplican a la población intermedia para crear la siguiente población. El proceso de ir de la población actual a la siguiente población constituye una generación en la ejecución del AG.

Por último, un algoritmo genético simple con el cual se obtienen buenos resultados en muchos problemas prácticos se compone de tres operadores [Golberg, 89]:

1. Reproducción (o selección)
2. Cruzamiento
3. Mutación.

3.4 Representación genética de problemas

La representación genética de un problema se refiere a la forma en que se codifican la soluciones candidatas del problema y constituye uno de los factores más importantes en el éxito del desarrollo de un AG [Mitchel, 96]. Veremos dos tipos de codificaciones, la *binaria*, en donde los cromosomas son un cadena de bits (ceros y unos), y la *no binaria*, en donde los cromosomas son una cadena de caracteres pertenecientes a algún alfabeto finito.

3.4.1 Codificación binaria

Las codificaciones binarias son las más comunes por varias razones. Una es la historia: Holland y sus estudiantes se concentraron en tales codificaciones y la práctica de AGs tendió hacia este tipo de codificaciones. Otra, es que mucha de la teoría existente sobre AGs está basada en la suposición de codificaciones binarias de longitud fija. Además, los valores apropiados para los parámetros, tales como probabilidad de cruzamiento y de mutación, generalmente han sido desarrollados en el contexto de codificaciones binarias.

Algunos problemas se adaptan fácilmente a este tipo de codificación, tal es el caso del Problema de la Mochila (en inglés, *Knapsack Problem*). El objetivo de este problema es encontrar el grupo de objetos con diferentes tamaños que se deben incluir en una mochila con el propósito de minimizar el desperdicio de espacio, o con algún otro propósito, como el de llevar la mayor cantidad de objetos que tenga más valor. Si cada objeto se representa con un bit del cromosoma, entonces un valor de 1 significará que ese objeto debe ser incluido, mientras que un cero indicará lo contrario. Por lo tanto, un cromosoma representará una lista de objetos que deben ser incluidos en la mochila y será aquellos cuyos bits tengan un valor de 1.

Por ejemplo, supóngase que se tienen 6 objetos de los cuales se tienen que seleccionar algunos de ellos para ser incluidos en la mochila. Con una codificación binaria, el cromosoma estará formado por 6 bits, de esta forma, el cromosoma **110010**, significa que se llevarán los objetos representados por las posiciones 1, 2 y 5.

3.4.2 Codificación No Binaria

Para muchas aplicaciones, resulta más natural utilizar un alfabeto de muchos caracteres o números reales para formar los cromosomas. Un problema al que se le puede aplicar una codificación no binaria es el Problema del Vendedor Viajero (TSP, por sus siglas en inglés). En el TSP, un vendedor trata de encontrar la ruta más corta al viajar por un grupo de ciudades por las que tiene que pasar solamente una vez. Podemos asignar una letra a cada una de las ciudades y utilizar este alfabeto para formar el cromosoma.

Por ejemplo, supóngase que se tienen que visitar cinco ciudades, y que a cada ciudad le asignamos una letra de la A a la E. El cromosoma **C E A D B**, representa la ruta que seguirá el vendedor. Esto es, el vendedor empezará por la ciudad C, continuará por la ciudad E, y así sucesivamente.

3.5 Selección

La selección, también llamada reproducción [Goldberg, 89], es un proceso en el cual las cadenas de individuos son seleccionadas para formar parte de la población intermedia de acuerdo a sus valores de la función de aptitud, *f*. Esto significa que las cadenas con un mayor valor de *f* tienen una mayor probabilidad de contribuir con uno o más hijos en la siguiente generación.

En este punto es importante hacer notar la diferencia entre función de aptitud y función de evaluación. La función de evaluación, o función objetivo del problema tratado, provee una medida del desempeño del individuo con respecto a una conjunto particular de parámetros del problema. La función de aptitud transforma esa medida de desempeño del individuo hacia una localización de oportunidades de reproducción. La evaluación de un cromosoma o individuo representando a un conjunto de parámetros es independiente de la evaluación de cualquier otro cromosoma. Sin embargo, la aptitud de ese cromosoma siempre será definida con respecto a otros miembros de la población actual [Whitley, 93].

Numerosos esquemas de selección han sido propuestos en la literatura. Aún así, estos esquemas no proveen una guía rigurosa para decidir cuál método debe ser usado para determinado problema [Mitchel, 96]. A continuación se presentan algunos de los métodos más utilizados en la selección.

Selección por ruleta. Es la más sencilla, consiste en lo siguiente: Se asignan probabilidades de reproducción a cada individuo en proporción a su aptitud. Suponga un ejemplo en donde se tiene una población de 4 cromosomas. Para cada individuo x, la aptitud, $f(x)$, y probabilidad de reproducción, $f(x)/\sum_j f(x)$, se muestran en la tabla 3.1.

Tabla 3.1

Aptitudes y probabilidades de reproducción

Individuo	$f(x)$	$f(x)/\sum_j f(x)$
A = 01100	12	0.30
B = 01001	9	0.23
C = 10000	16	0.40
D = 00011	3	0.08
Total	$\sum_j f(x) = 40$	1.00

De acuerdo a la probabilidad obtenida, se asigna a cada individuo un intervalo de probabilidad de ser seleccionado. Se genera un número aleatorio (por cada individuo en la población) y dependiendo del intervalo en que haya quedado, se escoge al individuo para obtener una copia de él, de modo que forme parte de la población sobreviviente (población intermedia). Claramente, se puede observar que el individuo con mayor probabilidad de ser escogido es el C con una probabilidad de 0.40. Y se puede decir que el *valor esperado* de copias para ese individuo es de 1.6 (su probabilidad multiplicada por el número de individuos de la población).

Selección por torneo. Este tipo de selección se lleva a cabo de la siguiente manera: Dos individuos son seleccionados aleatoriamente de la población. Entonces, se selecciona un número aleatorio r entre 0 y 1. Si $r < k$ (donde k es un parámetro, por ejemplo 0.75), el individuo con mayor aptitud es seleccionado para formar parte de la nueva población intermedia; de otra manera se selecciona el individuo con menor aptitud. Es obvio que si k toma el valor de 1, siempre se seleccionará al individuo con mayor aptitud. Después, los dos individuos se retornan a la población original y pueden ser seleccionados nuevamente. Este procedimiento se lleva a cabo hasta tener a la nueva población intermedia completa.

Selección por clasificación (*Rank Selection*). La selección por clasificación es un método alternativo cuyo propósito es prevenir la convergencia prematura. Funciona como sigue: cada individuo en la población se clasifica en orden creciente de acuerdo a su aptitud, desde 1 hasta N. El usuario selecciona el valor esperado Max del individuo en la posición N de la clasificación, con $Max \geq 0$. El valor esperado de cada individuo i en la población en un tiempo t está dado por

$$\text{ExpVal}(i,t) = Min + (Max - Min)\frac{\text{rank}(i,t) - 1}{N - 1},$$

donde Min es el valor esperado del individuo colocado en la posición 1 de la clasificación. Dadas las restricciones $Max \geq 0$ y $\sum_i \text{ExpVal}(i,t) = N$ (si el tamaño de la

población se mantiene constante de generación en generación), se requiere que $1 \leq Max \leq 2$ y $Min = 2 - Max$.

3.6 Cruzamiento

Después de la selección, se lleva acabo el cruzamiento. Como ya se mencionó anteriormente, el cruzamiento es un intercambio de material genético entre dos cromosomas padres. Se debe decidir qué tipo de cruzamiento se utilizará. Esta decisión depende de la estrategia de codificación. Para codificaciones binarias se explicará el cruzamiento de 1-punto y el cruzamiento de 2-puntos. Para codificaciones no binarias se presentan algunos de los tipos de cruzamiento más utilizados.

Cruzamiento de 1-punto. Procede en dos pasos. Primero, los miembros de la nueva población creada por la selección se seleccionan aleatoriamente para que formen parejas. Segundo, cada par de cadenas se cruza como sigue: se selecciona aleatoriamente una posición k entre 1 y la longitud de la cadena $[1, l\text{-}1]$. Dos nuevas cadenas se obtienen mediante el intercambio de todos los caracteres entre las posiciones k+1 y l. Por ejemplo, considere los cromosomas A y B de nuestra población inicial:

A = x x x x | x x

B = y y y y | y y

Suponga que se genera un número aleatorio entre 1 y 5, y se obtiene $k = 4$ (tal como se indica mediante el símbolo separador |). El resultado del cruce produce los dos hijos que se muestran a continuación:

A' = x x x x | y y

B' = y y y y | x x

Cruzamiento de 2-puntos. Funciona de forma similar al de 1-punto. Difiere en que ahora se seleccionan aleatoriamente dos posiciones k_1 y k_2 y los segmentos entre ellas se intercambian. Ejemplo, de nuevo considere los cromosomas A y B de nuestra población

inicial. Suponga que se generan dos números aleatorios entre 1 y 5, y se obtiene $k_1 = 2$ y $k_2 = 4$ (tal como se indica mediante el símbolo separador |).

A = x x | x x | x x

B = y y | y y | y y

El resultado del cruce produce los dos hijos que se muestran a continuación:

A' = x x | y y | x x

B' = y y | x x | y y

3.6.1 Operadores especiales de cruce

Recordemos la codificación genética realizada para el ejemplo del TSP. A cada una de las ciudades se le asignó una letra de A a la E, suponiendo que se tienen 5 ciudades que visitar. Ahora, supóngase que tenemos los cromosomas X y Y, y se desea cruzarlos utilizando el cruce de 1-punto. Se selecciona aleatoriamente una posición k y se obtiene $k = 3$ (nuevamente se muestra con el símbolo separador |).

X = D C B | A E

Y = B E D | C A

Después de realizar el cruce los cromosomas hijos que se obtienen son los siguientes:

X′ = D C B | C A

Y′ = B E D | A E

Se puede observar, que los hijos generados muestran repeticiones de ciudades, así como la omisión de algunas de ellas, por lo que no representan soluciones factibles para el TSP.

Esto mismo sucede con el PPT si se codifica la secuencia de tareas a realizar como una permutación de símbolos de algún alfabeto no binario. Es debido a lo anterior que se requiere algún tipo de cruzamiento que genere únicamente individuos válidos. Otra alternativa es realizar algún tipo de codificación binaria que pueda servir para los problemas de secuenciación, tales como el TSP o el PPT. Ordoñez y Valenzuela

(referenciados por Bautista [1991]) proponen un método para codificar problemas de ordenamiento con un alfabeto binario.

A continuación se presentan algunos operadores especiales de cruce más utilizados que fueron desarrollados para ser utilizados en problemas de secuenciación.

El operador PMX (*Partially Matched Crossover*). Fue presentado por Golberg y Lingle en la *1985 International Conference on Genetic Algorithms and Their Application* [Golberg, 89]. El operador procede como sigue: se seleccionan dos cadenas para ser cruzadas y se seleccionan aleatoriamente dos puntos de cruzamiento. Estos dos puntos definen una sección de emparejamiento (*matching section*). Para ver esto, considere los dos cromosomas (los puntos de cruce se muestran con el símbolo separador |):

A = 9 8 4 | 5 6 7 | 1 3 2 10

B = 8 7 1 | 2 3 10 | 9 5 4 6

Primero se mapea el cromosoma B hacia el A, intercambian posiciones el 2 y el 5, el 3 y el 6, y el 10 y el 7. Similarmente se mapea el cromosoma A hacia el B, intercambian posiciones el 5 y le 2, el 6 y el 3 y el 7 y el 10. Después de seguir el procedimiento del PMX, obtenemos dos hijos, A´y B´:

A´ = 9 8 4 | 2 3 10 | 1 6 5 7

B´ = 8 10 1 | 5 6 7 | 9 2 4 3

donde ambos cromosomas contienen información de ordenamiento parcial determinada por cada uno de sus padres.

El operador OX (*Order Crossover*). El OX empieza de forma similar al PMX, seleccionando una sección de emparejamiento:

A = 9 8 4 | 5 6 7 | 1 3 2 10

B = 8 7 1 | 2 3 10 | 9 5 4 6

y luego utiliza una sección corrediza para llenar los huecos dejados por la transferencia de las posiciones mapeadas. Por ejemplo, cuando el cromosoma B se mapea hacia el cromosoma A, el 5, 6 y 7 dejarán huecos (marcados con una H) en el cromosoma:

B = 8 H 1 | 2 3 10 | 9 H 4 H

Estos huecos se llenan con un movimiento corredizo que empieza siguiendo el segundo punto de cruzamiento:

B = 2 3 10 H H H 9 4 8 1

Los huecos son llenados con los elementos de la sección de emparejamiento tomados de la pareja. Realizando esta operación y completando el cruzamiento complementario, obtenemos los hijos A′ y B′ como sigue:

A′ = 5 6 7 | 2 3 10 | 1 9 8 4

B′ = 2 3 10 | 5 6 7 | 9 4 8 1

El operador de recombinación de aristas. Originalmente este operador fue desarrollado para ser utilizado en el TSP [Whitley et al., 89], por lo que puede ser utilizado en problemas de secuenciación. Fué desarrollado con la idea de que se deben de transmitir las aristas de los padres a los hijos tan efectivamente como sea posible. Se considera que una arista es la liga entre dos ciudades. El operador utiliza un *mapa de aristas*. Este mapa contiene todas las conexiones o ligas (de los dos padres) que llegan o que salen de cada ciudad. Cada ciudad tiene al menos dos y a lo máximo cuatro aristas asociadas (dos por cada padre). Ejemplo, considere los siguientes cromosomas: [A B C D E F] y [B D C A E F]. El mapa de aristas para este ejemplo es el siguiente:

A tiene aristas hacia: B F C E

B tiene aristas hacia: A C D F

C tiene aristas hacia: B D A

D tiene aristas hacia: C E B

E tiene aristas hacia: D F A

F tiene aristas hacia: A E B

Para empezar, la ruta que representa el hijo se inicializa con una de las dos ciudades iniciales de los padres, la cual tenga el menor numero de aristas. En el ejemplo A y B

tienen el mismo número de aristas, cuatro, por lo tanto, se selecciona aleatoriamente una de ellas. Suponga que B fue escogida. Ahora, los candidatos para la siguiente ciudad son las ciudades relacionadas mediante aristas con B, en este ejemplo son A, C, D, y F. C, D y F tienen dos aristas cada uno (ya no se considera la arista hacia B) y A ahora tiene tres aristas, de modo que se descarta, por lo tanto se selecciona aleatoriamente entre C, D y F. Asuma que C fue seleccionado. Ahora C tiene aristas con A y D. D se selecciona debido a que tiene menor número de aristas que A. D solo tiene una arista hacia E, por lo tanto E es la siguiente selección. E tiene arista hacia A y F, a ambos les resta solo una arista. Aleatoriamente, se selecciona A. Después de A solo queda una opción, sigue F. El hijo resultante es [B C D E A F].

3.7 Mutación

Aunque la selección y el cruzamiento proporcionan una búsqueda efectiva, ocasionalmente pierden algún material genético útil [Golberg, 89]. En los sistemas genéticos artificiales, el operador de mutación protege en contra de pérdidas irrecuperables. Además, ayuda a que la población no se quede fija (converja prematuramente), provocando que se salga de los mínimos y máximos locales.

Este operador cambia aleatoriamente el valor de algunos de los bits en el cromosoma [Mitchel, 96]. Por ejemplo, la cadena 00000100 puede ser mutada en su segunda posición para obtener 01000100. La mutación puede ocurrir en cualquier posición de una cadena con alguna probabilidad, usualmente muy pequeña (ejemplo, 0.001)

3.8 Resumen

Este capítulo sirvió como introducción a los AGs. Se definieron términos relacionados con las AGs, y se explicaron los tipos de codificaciones para realizar representaciones genéticas de problemas. Se definieron los operadores que componen un algoritmo genético: selección, cruce y mutación. Así como también se definieron varios tipos de cada operador y se dieron ejemplos de ellos.

Todo lo anterior servirá como base para una fácil comprensión del siguiente capítulo, en donde se explicará el desarrollo e implementación del AG con el cual damos solución al PPT.

CAPITULO 4

DESARROLLO E IMPLEMENTACION DEL ALGORITMO GENETICO

4.1 Introducción

En este capítulo se describirá la forma en que se desarrolla e implementa el AG para dar solución al PPT. El capítulo se conduce de la siguiente manera: en la sección 4.2 se explica la representación genética del PPT y posteriormente, en la sección 4.3, se establecen los tipos de operadores genéticos que se utilizaron en la implementación.

4.2 Representación genética del PPT

Al implementar el AG la primera cuestión a resolver es: ¿de qué manera se va a representar el PPT?, ¿con un codificación binaria?, ó ¿con una codificación no binaria?. Tratemos con una codificación no binaria. De esta manera, a cada una de las tareas se le asigna un caracter de algún alfabeto. Por ejemplo: suponga que se tiene un conjunto de 5 tareas que necesitan ser procesadas en una máquina. Si tenemos el alfabeto {A,B,C,D,E} podemos representar a cada tarea con una letra de este alfabeto. Así, el cromosoma [B D

E A C] representa la secuencia de tareas que serán procesadas, es decir, la primer tarea en ser procesada es la B, la segunda es la D, y así sucesivamente.

Pero debido a que se maneja una filosofía JIT existe un problema: ¿cómo es posible representar los tiempos de ocio de la máquina con una codificación no binaria?. Además, en el caso de dos máquinas, ¿de qué manera se representa en cuál de las dos máquinas se procesará cada tarea?. Obviamente, con una codificación no binaria, como la anterior, sería muy difícil representar estos aspectos del PPT debido a que los tiempos de ocio sí pueden tener caracteres repetidos, mientras que en la permutación no se permite que se repitan las tareas.

Bautista [1991] hace uso de un AG para resolver el PPT en dos máquinas iguales mediante la utilización de una codificación completamente binaria. Para esto, utiliza un método para codificar problemas de ordenamiento con un alfabeto binario propuesto por Ordoñez y Valenzuela (referenciados por Bautista). El cromosoma lo forma de la siguiente manera: en una sección del cromosoma codifica la secuencia de tareas, en otra parte del cromosoma agrega los bits necesarios para representar tiempos muertos, y en otra parte agrega los bits necesarios para identificar a cada tarea con su máquina. Ejemplo:

10100	0101010010101010100110110	00110101010100
bits para identificar a cada tarea con su máquina	bits para representar secuencia de tareas	bits para representar tiempos de ocio

Como se puede observar, los bits para representar la secuencia de tareas pueden ser sustituidos por una codificación no binaria. Así, se tendría el cromosoma dividido en dos partes, una con codificación binaria (para representar tiempos de ocio y relacionar a las tareas con su máquina) y una con codificación no binaria (para representar secuencia de tareas).

En su estudio, Bautista argumenta que una codificación de este tipo "requiere de dos operadores de cruce haciendo el problema innecesariamente más complicado". Pero entonces nos preguntamos, ¿no se realiza un esfuerzo similar al decodificar cada uno de los cromosomas (respecto a la parte que representa la secuencia de tareas) para poder evaluarlos?. De ahí surge la idea de solucionar el PPT utilizando una representación que combine tanto la codificación binaria como la no binaria.

4.2.1 Codificación para una máquina

Suponga que se tienen 5 tareas que se van a procesar en una máquina. Se realiza la codificación dividiendo el cromosoma en dos partes, una con representación no binaria y la otra con representación binaria. Ejemplo:

No binaria	Binaria
B D A C E	110 100 101 010 010
Secuencia de tareas	Tiempos de ocio

Se puede observar que para representar la secuencia de tareas se utiliza la codificación no binaria, y para representar los tiempos de ocios que existen antes de ejecutar alguna tarea se utiliza la codificación binaria.

En este ejemplo, se asocian tres bits de la parte binaria a cada una de las tareas. Esto es, a la tarea B se le asocian los primeros tres bits (110), a la tarea D se asocian los bits 4, 5 y 6 (100) y así sucesivamente. Esto significa que antes de ejecutar la tarea B, la máquina permanecerá ociosa durante 6 unidades de tiempo ($110_2 = 6_{10}$); después de ejecutar la tarea B, la máquina permanecerá ociosa durante 4 unidades de tiempo ($100_2 = 4_{10}$) y así sucesivamente.

La cantidad *b* de bits necesarios para representar los tiempos de ocio se calcula con la siguiente fórmula:

$$b = n * \lceil \log_2(T+1) \rceil$$

en donde *n* es el número de tareas y T es el máximo tiempo de ocio que puede permanecer la máquina antes de procesar una tarea.

Es claro que la cantidad de bits asociados a cada tarea está dada por $\lceil \log_2(T+1) \rceil$. Así, si suponemos que el máximo tiempo de ocio es T = 5, la cantidad de bits asociados a cada tarea será 3.

Si se forman grupos de tres bits asociados a cada tarea se desperdician los valores 6 y 7 (110 y 111). Para evitar que estos valores no válidos se desperdicien se obtiene el residuo de la división del valor en decimal de los bits entre el máximo tiempo de ocio deseado. Para el ejemplo anterior, esto se muestra en la tabla 4.1.

Tabla 4.1

Representación de tiempos de ocio mediante bits

Bits	Valor en decimal	Tiempo de ocio que representan
000	0	0 mod 5 = 0
001	1	1 mod 5 = 1
010	2	2 mod 5 = 2
011	3	3 mod 5 = 3
100	4	4 mod 5 = 4
101	5	5 mod 5 = 5
110	6	6 mod 5 = 1
111	7	7 mod 5 = 2

4.2.2 Codificación para dos máquinas

Para realizar la codificación del PPT en dos máquinas, es necesario encontrar la forma de representar en qué máquina se va a procesar cada tarea. Para esto podemos agregar a la codificación un bit por cada una de las tareas. De esta manera podemos asociar a todas las tareas que tengan un bit con valor de 0 a la primera máquina, mientras que las demás tareas se procesarán en la segunda máquina.

A continuación se presenta un ejemplo en donde se tienen 6 tareas que se van a procesar en dos máquinas y se tiene tiempo de ocio máximo T = 3.

No Binaria	Binaria	
B D F A C E	011010	10 00 01 11 10 00
Secuencia de tareas	Máquinas	Tiempos de ocio

En este cromosoma el primer bit de la sección *máquinas* de la parte binaria (0) se asocia con la primer tarea de la secuencia (B), el segundo bit (1) se asocia con la segunda tarea de la secuencia (D) y así sucesivamente. De esa forma se obtiene que en la máquina 1 se van a procesar las tareas B, A y E. En la máquina 2 se procesan las tareas D, F y C.

Respecto a los tiempos de ocio, se asocian a cada una de las tareas de la misma forma en que se explicó anteriormente. Así, los dos primeros bits de la sección *tiempos de ocio* de la parte binaria (10) se asocian con la tarea B, de la misma forma se realiza con la demás tareas.

Continuando con el ejemplo, en la tabla 4.2 se presentan los tiempos de procesamiento de las tareas en cada una de las máquinas.

Tabla 4.2

Ejemplos de tiempos de procesamiento

	Tiempos de procesamiento	
Tareas	en la máquina 1	en la máquina 2
A	3	4
B	4	4
C	2	3
D	5	4
E	1	2
F	3	3

También, supongamos los tiempos de preparación de las máquinas que se presentan en la tabla 4.3 (solo se muestran los necesarios):

Tabla 4.3

Ejemplos de tiempos de preparación

En la máquina 1:	En la maquina 2:
de B hacia A, 1 unidad de tiempo	de D hacia F, 2 unidades de tiempo
de A hacia E, 2 unidades de tiempo	de F hacia C, 2 unidades de tiempo

Entonces, la solución que representa el cromosoma del ejemplo, se puede ver de la siguiente manera:

Programación de tareas en la máquina 1

Unidades de tiempo

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
to	*to*	B	B	B	B	*to*	*to*	*to*	*st*	A	A	A	*st*	*st*	E	

Programación de tareas en la máquina 2

Unidades de tiempo

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
D	D	D	D	*to*	*st*	*st*	F	F	F	*to*	*to*	*st*	*st*	C	C	C

en donde *to* significa tiempo de ocio y *st* tiempo de preparación.

Como ya se sabe, el cromosoma codifica una posible solución del problema, así que este ejemplo solo nos sirve para observar de forma más clara cómo es que el cromosoma representa esa solución. Para evaluar la solución se necesita conocer los costos de preparación, las fechas de entrega de las tareas, así como las penalizaciones por entrega temprana y tardía, lo cual no ha sido incluido en el ejemplo tratado.

4.3 Tipos de operadores genéticos utilizados

Después de haber definido una representación genética para el PPT, ahora es necesario definir los tipos de operadores que se utilizaron en el desarrollo del AG. Esto es, definir qué tipo de selección se utiliza, cuál tipo de cruce se seleccionó y cómo se llevó a cabo la mutación.

Cabe mencionar que la inicialización de la población se realizó de forma aleatoria.

4.3.1 Selección

El método de selección utilizado fue el llamado "selección por ruleta" . Como se explicó anteriormente, en este tipo de selección, a cada individuo de la población se le asignan probabilidades de reproducción (ser seleccionado) en proporción a su aptitud. En este método existe la desventaja de que algún individuo reciba un valor de la función de evaluación (o función objetivo) muy alto en comparación con el resto de la población. Por consiguiente, la probabilidad para reproducir a ese individuo será tan alta, que probablemente domine a la población en unas cuantas generaciones, provocando que el AG converja rápidamente y se quede estancado en un mínimo o máximo local.

También existe otro inconveniente: la función de aptitud (*f.a.*) trata con valores positivos y considera a los valores más alejados del cero como los mejores, mientras que la función objetivo (*f.o.*) del PPT considera a los valores positivos más próximos a cero como los mejores, debido a que se desea minimizar los costos. Para solucionar estos problemas, se realizó un mapeo de los valores de la función de evaluación hacia valores de la función de aptitud.

En la figura 4.1 se presenta el mapeo realizado. Ahí, se puede observar que el máximo de la función objetivo, Max *f.o.*, se convierte en el mínimo de la función de aptitud, Min *f.a.*, así como también el mínimo de la función objetivo, Min *f.o.*, se convierte en el máximo de la función de aptitud, Max *f.a.*. Los demás valores intermedios se mapean para obtener su respectivo valor. Se decidió utilizar valores fijos para Max *f.a.* y Min *f.a.*, siendo de 100 y 1 respectivamente. Estos valores se obtuvieron mediante simulaciones y probablemente se tengan que cambiar dependiendo de la cantidad de individuos de la población.

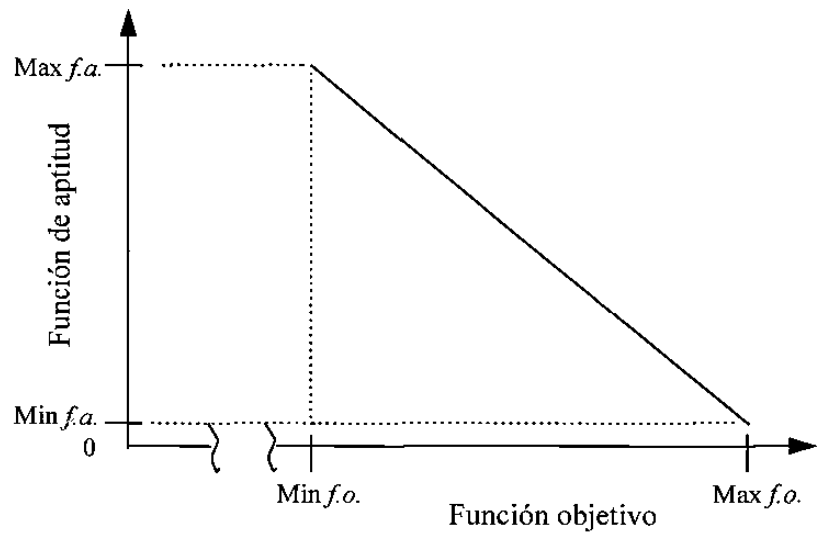

Figura 4.1. Mapeo de la función objetivo a la función de evaluación.

4.3.2 Cruzamiento y mutación

Respecto a los operadores de cruzamiento se decidió utilizar el operador PMX para la parte no binaria del cromosoma. Esta decisión se basó en que se han realizado estudios para determinar qué tipo de operador especial de cruce es mejor y los resultados reportados no han sido suficientes para asegurar que algún operador especial de cruce sea mejor que los demás [Cleveland y Smith, 89], [Oliver et al., 87], [Fox y McMahon, 91] (los últimos dos referenciados por Bautista [1991]).

Para la parte binaria de cromosoma se utilizó el cruzamiento de 2-puntos. La probabilidad de cruce se obtuvo mediante simulaciones y se estableció en 0.70.

Para la parte no binaria, la mutación se realizó de la siguiente manera: se seleccionan aleatoriamente dos posiciones y se intercambian las tareas que se encuentran en esas posiciones. Para la parte binaria del cromosoma se realizó una mutación tradicional en donde se selecciona aleatoriamente una posición y el valor del bit de la

posición seleccionada se cambia. Mediante simulaciones se estableció una probabilidad de mutación de 0.30.

4.4 Resumen

En este capítulo se discutió la forma de representar genéticamente el PPT. Se decidió utilizar una representación que combina un alfabeto binario y uno no binario. Esto, con el fin comparar esta nueva alternativa con la codificación completamente binaria presentada en la literatura consultada. También, se definieron los tipos de operadores utilizados en la implementación del AG.

CAPITULO 5

RESULTADOS PARA EL PROBLEMA DE PROGRAMACION DE TAREAS UTILIZANDO UN ALGORITMO GENETICO

5.1 Introducción

En el capítulo anterior se explicó la manera en que se realizó la implementación del algoritmo genético y la forma de representar genéticamente el PPT. En este capítulo se presentan los resultados encontrados mediante las simulaciones realizadas y se comparan con los reportados en la literatura.

Primero, se describen las pruebas realizadas, y posteriormente se analizan los resultados encontrados. Lo anterior, tanto para el caso del PPT en una máquina como para el de dos máquinas.

Al final de capítulo presentamos los tiempos de ejecución del AG implementado.

5.2 Especificación de las pruebas realizadas para el PPT en una máquina

Debido a la escasez de literatura que reporte trabajos que se igualen al presente estudio, los parámetros de las siguientes pruebas se tomaron únicamente del trabajo realizado por Bautista [1991], con el propósito de comparar los resultados encontrados.

Para el caso del PPT en una máquina con 50 tareas por procesar, se muestran a continuación los parámetros del AG utilizados en las simulaciones de la presente investigación y los utilizados por Bautista.

Tabla 5.1

Parámetros del AG para el PPT en una máquina con 50 tareas

	Ordaz (1999)	Bautista (1991)
Tipo de codificación	Combinación de binaria y no binaria	Completamente binaria
Tamaño de población	20	20
Longitud del cromosoma	50 caracteres	237 bits
Bits para tiempos de ocio	100	200
Longitud total	150	437
Número de generaciones	5,000	5,000
Probabilidad de cruce	0.7	0.8
Probabilidad de mutación	0.3 p/individuo (sección no binaria) 0.003 p/bit (sección binaria)	0.00069 p/bit
Número de evaluaciones	100,000	100,000

Las diferencias en los parámetros del AG de la presente investigación y los presentados por Bautista radica en que fueron seleccionados mediante simulaciones para encontrar los valores de los mismos que generaran los mejores resultados.

Es por lo anterior que el tiempo de ocio máximo permitido para la máquina antes de procesar una tarea fué de 3 unidades de tiempo, esto es lo que genera que la cantidad de bits utilizados para el tiempo de ocio sea 100, es decir, 2 bits por cada una de las tareas. Esto difiere de la cantidad de bits utilizada por Bautista para el mismo propósito, él utiliza 200 bits. Esto provoca que antes de procesar cada tarea se permita hasta 15 unidades de tiempo de ocio (4 bits por tarea).

Al igual que Bautista, los valores de los problemas fueron generados aleatoriamente a partir de las siguientes distribuciones:

Tabla 5.2

Distribuciones para el PPT en una máquina

PPT en una máquina	
Tiempos de procesamiento	$N(8,2)$
Castigo por entrega tardía por unidad de tiempo	$U(1,10)$
Castigo por entrega prematura por unidad de tiempo	$U(1,7)$
Tiempos de preparación	$U(1,10)$
Fechas de entrega	$U(5,800)$

En donde $U(\alpha,\beta)$ representa una distribución uniforme con rango de $[\alpha,\beta]$ y $N(\mu,\sigma)$ es una distribución normal con media μ y desviación estándar σ.

Como se puede observar, en la tabla 5.2 no se hace mención de costos de preparación, esto es debido a que Bautista no los utiliza. De cualquier forma, se

realizaron simulaciones sin costos de preparación y con ellos. Estos costos de preparación se generaron aleatoriamente con una distribución $U(1,8)$.

5.3 Resultados para el PPT en una máquina

En la figura 5.1 se observa la convergencia del AG para el PPT en una máquina sin incluir costos de preparación. Los puntos en la gráfica representan el promedio (μ) de los valores mínimos cada 10 generaciones en 10 corridas del problema con las distribuciones anteriormente descritas. En la gráfica también se representa $\mu+\sigma$ (media más desviación estándar) mediante la curva superior y la curva inferior representa $\mu-\sigma$.

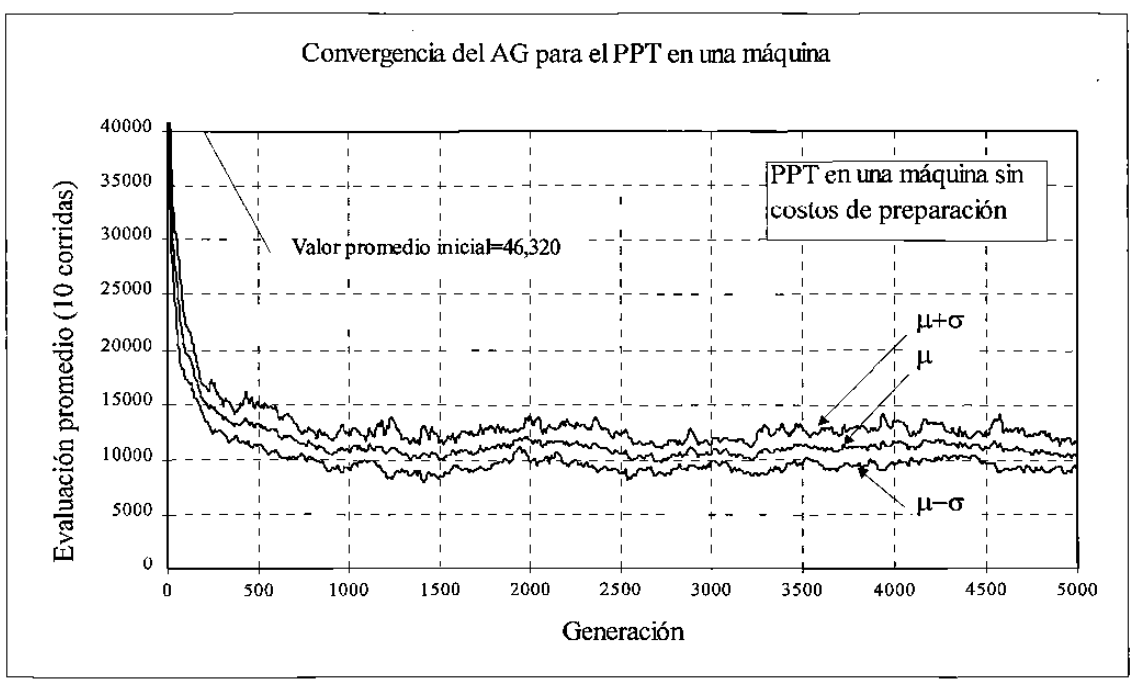

Figura 5.1. Convergencia del AG para el PPT en una máquina con 50 tareas, sin costos de preparación.

En la tabla 5.3 se presentan los mejores valores encontrados en cada una de las 10

corridas del PPT en una máquina a las que hace referencia la figura 5.1. Se puede observar que el mejor valor encontrado es 6,948. Sin embargo, este valor es más grande que el encontrado por Bautista [1991], quien encontró un costo mínimo de 3,319 unidades de costo. Inclusive, el AG desarrollado por Bautista, presenta una convergencia similar pero en promedio disminuye los costos hasta un valor cercano a las 5,000 unidades.

De aquí, se puede concluir que en este caso del PPT (en una máquina), el AG de Bautista, el cual hace uso de una codificación completamente binaria, tiene un mejor desempeño que el presentado en esta tesis.

Tabla 5.3

Mejores valores encontrados para el PPT en una máquina con 50 tareas, sin costos de preparación

Corrida	Mejor valor encontrado (costo mínimo)	Generación en que se encontró
1	6948	2538
2	7895	1473
3	7397	881
4	8390	4556
5	7526	3343
6	7425	3805
7	7453	1406
8	8437	3614
9	8644	3437
10	7775	1243
Promedio	7789	2629.6

En la figura 5.2 se observan los resultados para el PPT en una máquina con 50 tareas incluyendo costos de preparación. Los resultados encontrados son muy similares a los

encontrados cuando no se incluyen estos costos. Sin embargo, no existen investigaciones previas para poder comparar estos resultados. Aún así, se puede decir que el desempeño del AG no fué afectado al incluir estos costos.

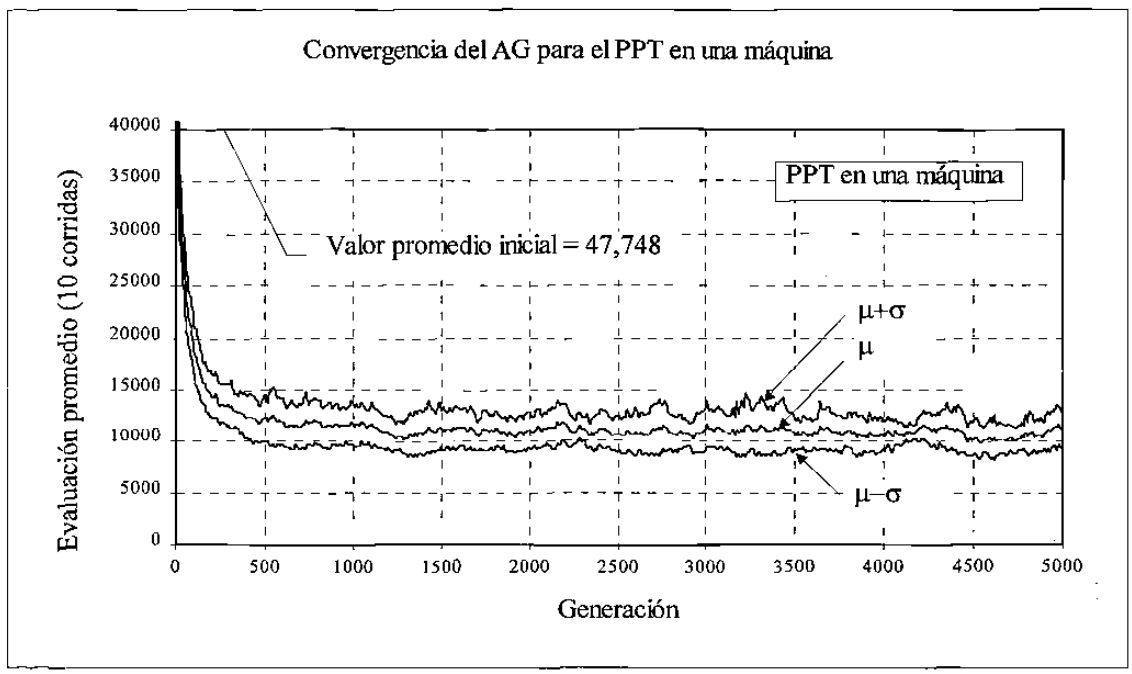

Figura 5.2. Convergencia para el PPT en una máquina con 50 tareas incluyendo costos de preparación.

En la tabla 5.4 se presentan los valores mínimos encontrados en cada una de las 10 corridas a las que hace referencia la figura 5.2. El mejor valor encontrado fué de 6,773 unidades de costo. Este costo es menor que el encontrado cuando se incluyeron costos de preparación, lo que confirma que el AG tiene un desempeño similar al aumentar restricciones.

Tabla 5.4

Mejores valores encontrados para el PPT en una máquina con 50 tareas

Corrida	Mejor valor encontrado (costo mínimo)	Generación en que se encontró
1	7393	2752
2	7956	3106
3	7419	1334
4	6773	4911
5	7432	4044
6	7665	4518
7	9143	2515
8	7098	3295
9	8150	4600
10	7943	4368
Promedio	7697.2	3544.3

5.4 Especificación de las pruebas realizadas para el PPT en dos máquinas iguales.

Nuevamente, los parámetros de las pruebas se tomaron del trabajo realizado por Bautista [1991], con el propósito de comparar los resultados encontrados.

Para el caso del PPT en dos máquinas iguales con 50 tareas por procesar, los parámetros del AG utilizados en las simulaciones de la presente investigación y los utilizados por Bautista se muestran en la tabla 5.5.

Tabla 5.5

Parámetros del AG para el PPT en dos máquinas iguales

	Ordaz (1999)	Bautista (1991)
Tipo de codificación	Combinación de binaria y no binaria	Completamente binaria
Tamaño de población	20	20
Longitud del cromosoma	50 caracteres	237 bits
Bits para tiempos de ocio	150	200
Bits para asociar tareas/máquinas	50	50
Longitud total	250	487
Número de generaciones	5,000	5,000
Probabilidad de cruce	0.7	0.8
Probabilidad de mutación	0.3 p/individuo (sección no binaria) 0.0012 p/bit (sección binaria)	0.00061 p/bit
Número de evaluaciones	100,000	100,000

El tiempo de ocio máximo permitido para la máquina antes de procesar una tarea fué de 5 unidades de tiempo, es decir se asignaron 3 bits por cada una de las tareas. Debido a esto, el total de bits asignados para representar tiempos de ocio fué 150.

Los valores de los problemas fueron generados aleatoriamente a partir de las distribuciones mostradas en la tabla 5.6.

Tabla 5.6

Distribuciones para el PPT en dos máquinas iguales

PPT en dos maquina iguales	
Tiempos de procesamiento	$N(10,2)$
Castigo por entrega tardía por unidad de tiempo	$U(1,10)$
Castigo por entrega prematura por unidad de tiempo	$U(1,7)$
Tiempos de preparación	$U(1,10)$
Fechas de entrega	$U(5,400)$

Las distribuciones difieren de las utilizadas en el caso de una máquina en los tiempos de procesamiento y en las fechas de entrega, ya que se tiene la misma cantidad de tareas, pero ahora se pueden procesar en cualquiera de las dos máquinas.

De nuevo, se realizaron simulaciones sin costos de preparación -para comparar contra Bautista- y con ellos. Los costos de preparación se generaron aleatoriamente con una distribución $U(1,8)$.

5.5 Resultados para el PPT en dos máquinas iguales

En la figura 5.3 se observa la convergencia del AG para el PPT en dos máquinas iguales sin incluir costos de preparación. Los puntos en la gráfica representan el promedio (μ) de los valores mínimos cada 10 generaciones en 10 corridas del problema.

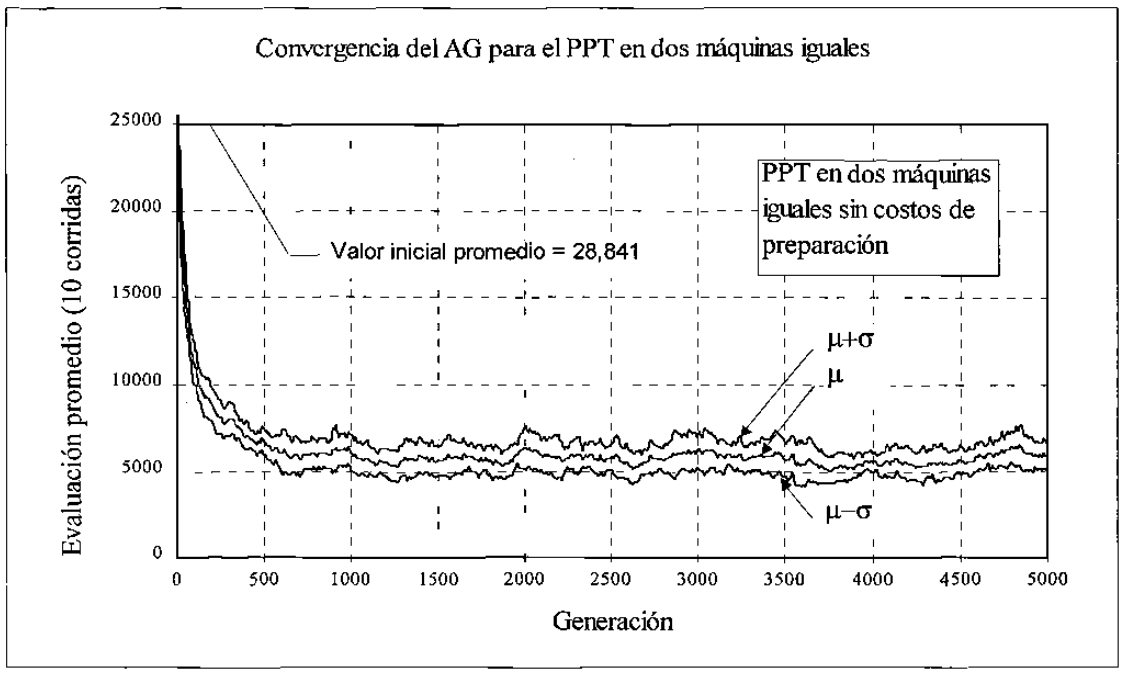

Figura 5.3. Convergencia del AG para el PPT en dos máquinas iguales con 50 tareas, sin costos de preparación.

En la gráfica de la figura 5.3 se observa que el AG tiene un comportamiento similar al que se tiene cuando se trata de una máquina. Esto muestra que al AG desarrollado no le afecta el aumento de una a dos máquinas, inclusive se encontraron mejores resultados.

En la tabla 5.7 se presentan los mejores valores encontrados en cada una de las 10 corridas del PPT en dos máquinas iguales a las que hace referencia la figura 5.3. En la tabla 5.8 se muestra una comparación entre el mejor valor encontrado en esta investigación contra el mejor valor encontrado en la investigación de Bautista. También, en la tabla 5.8, se muestran los promedios de los valores finales de las 10 corridas.

Tabla 5.7

Mejores valores encontrados para el PPT en dos máquinas iguales con 50 tareas, sin costos de preparación

Corrida	Mejor valor encontrado (costo mínimo)	Generación en que se encontró
1	3753	1166
2	3565	3590
3	4299	4285
4	3323	3690
5	3623	1507
6	4181	3051
7	3911	2656
8	3185	617
9	4851	3616
10	3915	878
Promedio	3860.6	2505.6

Como se puede observar en la tabla 5.8, el AG aquí implementado genera un mejor resultado que el encontrado por Bautista. Es por esto que para este caso, el PPT en dos máquinas iguales, se puede decir que es más conveniente una representación genética del PPT mediante la combinación de las codificaciones binaria y no binaria, como se describió anteriormente.

Tabla 5.8

Comparación de los valores mínimos encontrados para el PPT en dos máquinas iguales

	Bautista (1991)	Ordaz (1999)
Mejor valor encontrado	5,134	3,185
Promedio	7,153	5,882

Como se mencionó anteriormente, también se realizaron pruebas para el PPT en dos máquinas iguales incluyendo costos de preparación. En la figura 5.4 se muestra la convergencia del AG para este problema.

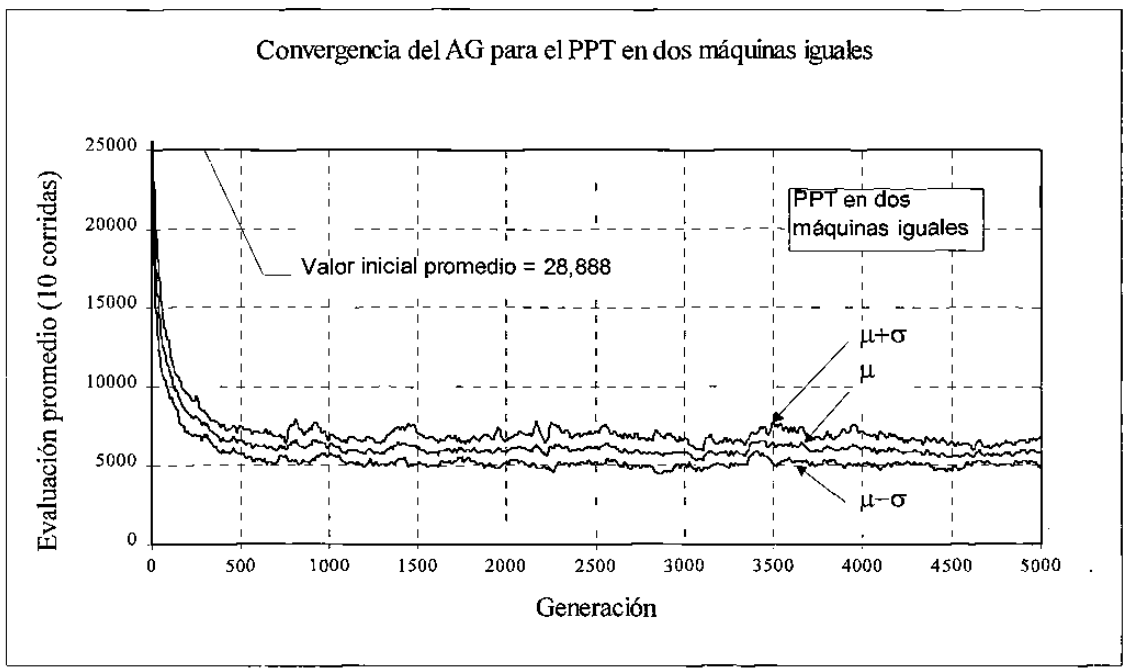

Figura 5.4. Convergencia del AG para el PPT en dos máquinas iguales con 50 tareas incluyendo costos de preparación.

Podemos observar en la figura 5.4 que la convergencia del AG es muy similar a la presentada cuando no se tienen costos de preparación. En la tabla 5.9 se presentan los mejores valores encontrados en cada una de las 10 corridas del PPT en dos máquinas iguales a las que hace referencia la figura 5.4.

El mejor valor encontrado fué de 3,506 unidades de costo, el cual es muy aproximado al mejor valor encontrado (3,185 unidades de costo) cuando no se incluyen costos de preparación. Esto nos muestra que el AG se comporta de la misma manera al considerar estos costos y era de esperarse que los mejores valores para este caso sean ligeramente mayores a los encontrados en el caso sin costos de preparación.

Tabla 5.9

Mejores valores encontrados para el PPT en dos máquinas iguales con 50 tareas incluyendo costos de preparación

Problema	Mejor valor encontrado (costo mínimo)	Generación en que se encontró
1	3967	2458
2	4433	4501
3	4695	4767
4	4290	4075
5	3506	2888
6	4150	3997
7	4802	3051
8	4306	2139
9	3991	1450
10	4750	4454
Promedio	4289	3378

5.6 Especificación de las pruebas realizadas para el PPT en dos máquinas diferentes

En la literatura consultada no se encontró ninguna investigación que trate el PPT en dos máquinas diferentes. Debido a esto, los parámetros utilizados en el AG y las distribuciones del problema para las siguientes simulaciones son los mismos que se utilizaron en el caso de dos máquinas iguales, con el propósito de comparar el desempeño del AG.

Una excepción a lo anterior es que solo se permitió un tiempo de ocio máximo antes de procesar una tarea de 3 unidades de tiempo. Por lo que el total de bits utilizados para el manejo de tiempos de ocio fué de 100.

Obviamente, ahora se consideran dos matrices de costos y tiempos de preparación, así como dos vectores de tiempos de procesamiento, uno para cada una de las máquinas. Todo esto hace que el problema se vuelva más realista pero a la vez más complejo.

5.7 Resultados para el PPT en dos máquinas diferentes

En la figura 5.5 se muestra la convergencia del AG para el PPT en dos máquinas diferentes, aclarando que en estas simulaciones se incluyen costos de preparación.

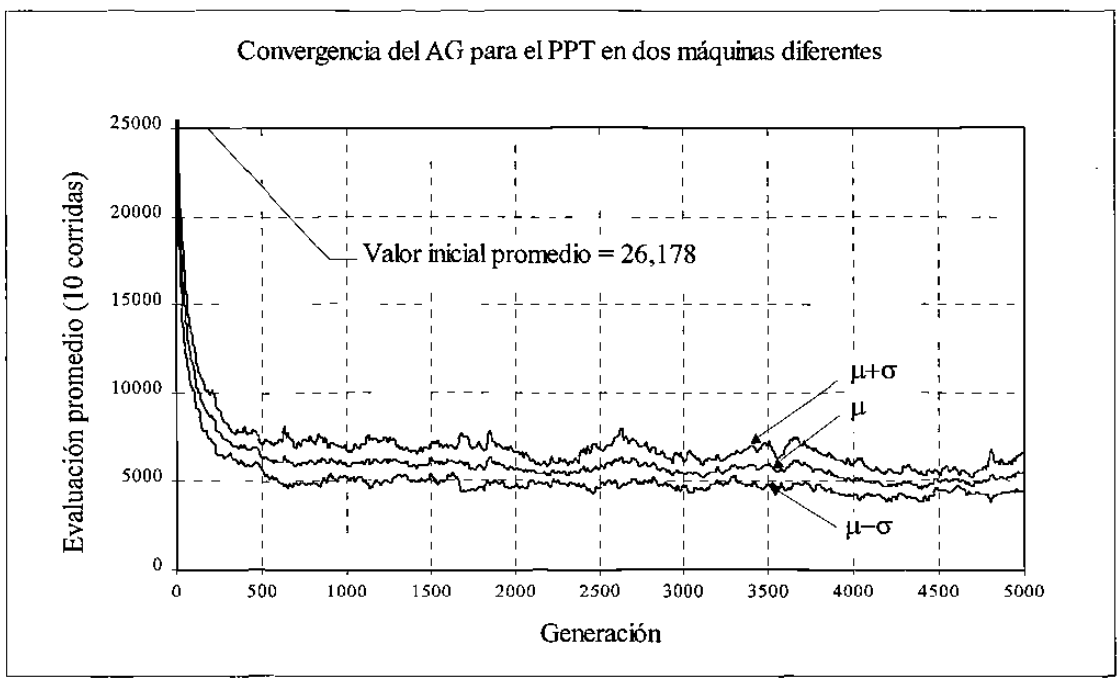

Figura 5.5. Convergencia del AG para el PPT en dos máquinas diferentes con 50 tareas.

En la tabla 5.10 se presentan los mejores valores encontrados en cada una de las 10 corridas del PPT en dos máquinas diferentes a las que hace referencia la figura 5.5.

En la tabla 5.11 se realiza una comparación del mejor valor encontrado por el AG para el PPT en dos máquinas iguales contra el mejor valor encontrado por el AG para el PPT en dos máquinas diferentes. En esta tabla también se presenta el promedio de los valores finales de las 10 corridas del problema.

Tabla 5.10

Mejores valores encontrados para el PPT en dos máquinas diferentes con 50 tareas

Problema	Mejor valor encontrado (costo mínimo)	Generación en que se encontró
1	3347	4440
2	3430	4200
3	3163	4836
4	3941	4403
5	4487	4239
6	4134	4195
7	4075	4779
8	3933	3601
9	2974	4205
10	4787	2355
Promedio	3827.1	4125.3

Como se puede observar en la tabla 5.11, el AG encontró mejores resultados en el PPT en dos máquinas diferentes a pesar de que este caso presenta mayor complejidad. Esto nos indica que el desempeño del AG mejoró cuando el problema a resolver fué más complejo.

Tabla 5.11

Costo mínimo del PPT en dos máquinas iguales vs. costo mínimo del PPT en dos máquinas diferentes (50 tareas)

	PPT con dos máquinas iguales	PPT con dos máquinas diferentes
Mejor valor encontrado	3,506	2,974
Promedio	5,894	5,524

5.8 Tiempos de ejecución del AG

Los resultados presentados en las secciones anteriores fueron obtenidos mediante simulaciones realizadas utilizando una computadora personal compatible con IBM con procesador Pentium con velocidad de 133 MHz, y 32 MB de memoria RAM. El algoritmo se implementó utilizando el lenguaje de programación C para DOS. Bajo estas condiciones se obtuvieron los tiempos de ejecución del algoritmo genético.

En la tabla 5.12 se presentan los tiempos promedio de ejecución del algoritmo genético implementado para cada uno de los casos del PPT mencionados anteriormente.

Podemos observar en la tabla 5.12, que el tiempo de ejecución del algoritmo genético implementado fué muy similar en todos los casos abordados del PPT. Inclusive, el mejor tiempo promedio de ejecución se obtuvo para el caso del PPT en dos máquinas diferentes. Además, estos tiempos de ejecución son relativamente bajos si consideramos que el problema solucionado pertenece a la clase NP-completo.

Tabla 5.12

Tiempos promedio de ejecución del AG

PPT	Tiempo promedio de ejecución en minutos (10 corridas)
En una máquina sin costos de preparación	3.569
En una máquina incluyendo costos de preparación	3.554
En dos máquinas iguales sin costos de preparación	3.053
En dos máquinas iguales incluyendo costos de preparación	3.049
En dos máquinas diferentes	2.866

5.9 Resumen

En este capítulo se presentaron los resultados obtenidos mediante las simulaciones realizadas. Se analizaron los datos y se realizaron comparaciones con resultados reportados en la literatura para el caso de una máquina y para el caso de dos máquinas iguales. Los resultados muestran que el desempeño del algoritmo genético fué similar al reportado en la literatura, mejorando su desempeño cuando se trató el caso de dos máquinas iguales.

Los resultados para el caso de dos máquinas diferentes fueron comparados con los obtenidos con dos máquinas iguales. Los resultados obtenidos por el AG para el PPT en dos máquinas diferentes fueron los mejores y se observó que aumentar la complejidad del problema no afecta el desempeño del AG.

CAPITULO 6

CONCLUSIONES

6.1 Introducción

En este capítulo se discuten los resultados de los datos analizados en el capítulo anterior. En la sección 6.2 se recordarán los objetivos de esta tesis. En la sección 6.3 se presentarán las discusiones y conclusiones de los objetivos mencionados. Y por último, en la sección 6.4 se presentan las recomendaciones para trabajos futuros relacionados con esta investigación.

6.2 Objetivos de la investigación

En la presente investigación se utilizó un algoritmo heurístico, el algoritmo genético, para resolver el problema de programación de tareas. Los objetivos buscados en esta tesis fueron los siguientes:

1) Desarrollo e implementación computacional de un algoritmo genético para resolver el PPT en una máquina.

2) Desarrollo e implementación computacional de un algoritmo genético para resolver el PPT en dos máquinas iguales.
3) Desarrollo e implementación computacional de un algoritmo genético para resolver el PPT en dos máquinas diferentes.

6.3 Discusiones y conclusiones

En esta tesis se explicó el problema de programación de tareas, incluyendo un enfoque JIT, el cual se encuentra presente comúnmente en las empresas. Se discutió la complejidad del problema y se decidió desarrollar un algoritmo genético para resolverlo. El AG se implementó utilizando una representación genética del PPT mediante la combinación de un alfabeto binario y uno no binario. Y de acuerdo a los resultados presentados en el capítulo anterior se llegó a las siguientes conclusiones.

En general, las soluciones encontradas por el AG implementado se consideran buenas con respecto a los resultados reportados en la literatura. Debido a lo anterior, podemos concluir que es conveniente la utilización de los algoritmos genéticos para la solución de problemas como el PPT.

6.3.1 Discusiones y conclusiones para el PPT en una máquina

Las soluciones encontradas por el AG para el PPT en una máquina fueron comparadas con las reportadas en la literatura, en la cual se realiza la utilización de un AG con una codificación completamente binaria del PPT. Se observó que el AG no mejoró las soluciones encontradas anteriormente. La convergencia del algoritmo presentó un buen comportamiento pero no fué suficiente para encontrar mejores soluciones.

Podemos concluir que para el PPT en una máquina no fue conveniente la utilización de una codificación del PPT que combine un alfabeto binario y uno no binario, aunque esto no es lo único que describe al AG. Quizá, para mejorar el desempeño del AG, se deban de realizar nuevas modificaciones en los parámetros del AG, así como también, utilizar algunos otros tipos de operadores genéticos.

6.3.2 Discusiones y conclusiones para el PPT en dos máquinas iguales

Los resultados presentados en el capítulo anterior muestran una mejoría notable en las soluciones encontradas por el AG implementado para el PPT en dos máquinas iguales, con respecto al caso de una máquina. Se observó que al aumentar el número de máquinas a dos, la convergencia del AG no fue afectada, fué muy similar a la presentada cuando se trató de una sola máquina. Esto muestra la flexibilidad de modificación del AG para tratar el PPT con mayor número de máquinas o con diferentes restricciones, por ejemplo, incluir o no, costos de preparación.

Las complicaciones implicadas por el tipo de representación del PPT (combinación de codificación binaria y no binaria) fueron compensadas por los mejores soluciones encontradas al comparar estas con las reportadas en la literatura.

6.3.3 Discusiones y conclusiones para el PPT en dos máquinas diferentes

En este caso, se compararon los resultados obtenidos contra los del AG para el PPT con dos máquinas iguales y se observó que fueron mejores las soluciones encontradas cuando se trató del PPT con dos maquinas diferentes.

A pesar de que al tratar con dos máquinas diferentes el espacio de búsqueda de soluciones aumenta, el AG convergió hacia mejores soluciones. Nuevamente, observamos que el comportamiento del AG mejoró.

Aún así, no se pudieron comparar los resultados con algunos otros que traten el PPT en dos máquinas diferentes. Sin embargo, concluimos que aunque el problema se tornó más complejo al incluir dos máquinas diferentes, el utilizar una combinación de codificaciones binaria y no binaria para representar el PPT redujo significativamente los costos de la programación de tareas.

6.4 Recomendaciones para investigaciones futuras

Se observó la manera en que el AG mejoró las soluciones notablemente cuando trató el PPT en dos máquinas diferentes. Como se mencionó anteriormente, el caso de dos máquinas diferentes se podría presentar con más frecuencia en las empresas que el caso de tener dos máquinas iguales. También, podemos suponer que en algunas ocasiones se tendrán más de dos máquinas para procesar las tareas. Inclusive, se pueden presentar variaciones del PPT cuya función objetivo sea diferente. Por todo esto, como una extensión al trabajo realizado en esta tesis se propone lo siguiente:

- Tratar de generalizar resultados realizando más simulaciones con diferentes cantidades de tareas.
- Modificar la representación genética del PPT para permitir el manejo de más de dos máquinas.
- Realizar cambios en la función objetivo, tales como:
  - ⇒ minimizar los costos junto con el tiempo total de procesamiento,
  - ⇒ minimizar los costos junto con el tiempo total de retraso en la entrega de tareas,
  - ⇒ minimizar los costos junto con el número de tareas que se terminan después de su fecha de entrega, entre otros.
- Incluir más restricciones, tales como:
  - ⇒ restringir para cada tarea el intervalo de tiempo disponible para procesarse,
  - ⇒ en el caso de varias máquinas, igualar la cantidad de tareas en cada una de las máquinas, entre otras.

REFERENCIAS

[Bautista, 91] Bautista Vera, Erick. "Utilización de un algoritmo genético para resolver el problema de programación de tareas en dos máquinas". Tesis de Maestría en Ciencias. Instituto Tecnológico y de Estudios Superiores de Monterrey. Monterrey, N. L., México. (Diciembre de 1991).

[Chen, 97] Chen, Zhi-Long. "Scheduling with Batch Setup Times and Earliness-Tardiness Penalties". European Journal of Operation Research 96. (1997).

[Cleveland y Smith, 89] Cleveland, Gary A. y Smith, Stephen. F. "Using Genetic Algorithms to Schedule Flow Shops Releases". J. D. Schaffer (Ed.), Proceedings of the Third International Conference on Genetic Algorithms, (pp. 160-169). (1989).

[Díaz et al., 96] Díaz, A., Glover, F., Ghaziri, H. M., Gonzalez, J. L., Laguna, M., Moscato, P., Tseng, F. T. Optimización Heurística y Redes Neuronales. Editorial Parainfo. (1996).

[Fox y McMahon, 91] Fox, B. R., y McMahon, M. B. "Genetic Operators for Sequencing Problems". G. Rawlins, (Ed.), Foundations of genetic algorithms (pp. 284-300). Morgan Kaufmann. (1991).

[Garey y Johnson, 79] Garey, Michael R. y Johnson, David S. Computers and Intractability. A Guide to theory of NP-Completeness. Bell Laboratories. (1979).

[Golberg, 89] Goldberg, David E. Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley Publishing Company, Inc. (1989).

[Husbands, 95] Husbands, Philip. "Genetic Algorithm for Scheduling". AISB Quarterly, No. 89. (1995).

[Michalewicz, 92] Michalewicz, Zbigniew. Genetic Algorithms + Data Structures = Evolution Programs. Third, revised and extended edition. Springer.(1992).

[Mitchel, 96] Mitchell, Melanie. An introduction to genetic algorithms. The MIT Press. Cambridge, Massachusetts. London, England. (1996).

[Nakano, 91] Nakano, Ryohei. "Conventional Genetic Algorithm for Job Shop Scheduling". R. K. Belew y L. B. Brooker, (Ed.), Proceedings of the Fourth International Conference on Genetic Algorithms, (pp. 474-479). San Mateo: Morgan Kaufman. (1991).

[Oliver et al., 87] Oliver, I. M., Smith, D. J y Holland, J. R. C. "A study of permutation crossover operators on the traveling salesman problem". Proceedings of the Second International Conference on Genetic Algorithms and their Applications, (pp. 224-230). Cambridge, MA.(1987).

[Rosas, 91] Rosas Vega, Rosario. "Programación de tareas en dos máquinas por medio de Tabu Search". Tesis de Maestría en Ciencias. Instituto Tecnológico y de Estudios Superiores de Monterrey. Monterrey, N. L., México. (Diciembre de 1991).

[Whitley, 93] Whitley, Darrell. "A Genetic Algorithm Tutorial". Technical Report, Computer Science Department, Colorado State University. (1993).

[Whitley et al., 89] Whitley, Darrell, Starkweather, Timothy y Fuquay, D'Ann. "Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator". J. D. Schaffer (Ed.), Proceedings of the Third International Conference on Genetic Algorithms, (pp. 133-140). (1989).

RESUMEN AUTOBIOGRAFICO

Victor Hugo Ordaz Gaitán

Candidato para el Grado de

Maestro en Ciencias de la Administración con Especialidad en Sistemas

Tesis: DESARROLLO E IMPLEMENTACION DE UN ALGORITMO GENETICO QUE RESUELVE EL PROBLEMA DE PROGRAMACION DE TAREAS EN DOS MAQUINAS.

Campo de Estudio: Sistemas

Biografía:

Nacido en Saltillo, Coahuila, el 15 de Diciembre de 1974, hijo de José Isaac Ordaz Nava y María Antonieta Gaytán de Ordaz.

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 1997.

Experiencia Profesional:

Maestro por horas de la Facultad de Ingeniería Mecánica y Eléctrica de la UANL y Maestro Instructor de la Coordinación de Educación Continua de la Facultad de Ingeniería Mecánica y Eléctrica de la UANL.