UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTADAD DE ESTUDIOS SUPERIORES ACATLÁN

"BÚSQUEDA TABÚ Y ALGORITMOS GENÉTICOS: COMPARACIÓN PARA SOLUCIONAR EL PROBLEMA DEL AGENTE VIAJERO"

TESIS

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN MATEMÁTICAS APLICADAS Y COMPUTACION

PRESENTA

MARRÓN LUNA DYANA ERIKA

ASESOR: FRANCISCO JAVIER PATIÑO DONNADIEU

Marzo, 2006

DEDICATORIAS Y AGRADECIMIENTOS

Este trabajo que me permite cumplir uno de mis sueños y la finalización de una etapa en mi vida, se lo dedico a mi madre: Maria Luna Romero, la mujer que en espíritu y recuerdos me alentó a seguir adelante cuando más lo necesitaba.

- *Primero quiero agradecer a Dios por la oportunidad de vivir y llegar hasta aquí, por permitirme conocer a mis padres y por darme unas hermanas que me quieren.,*

- *Gracias a mi madre que me dio la vida y que cuido de mí cuando más la necesitaba. Gracias por dejarme una buena impresión y darme un ejemplo a seguir. Gracias por ser la mujer excepcional que fuiste, pero sobre todo gracias por pensar en mí antes de pensar en ti.*

- *Quiero agradecer a mi padre por brindarme su apoyo económico y por enseñarme que en la vida los conocimientos son muy importantes.*

- *Gracias Yaz por todos esos regaños y por presionarme para terminar mi tesis, pero sobre todo gracias por ser mi hermana y seguir con nosotros. Gracias por darme tu apoyo moral.*

- *Gracias Miguel Ángel por enseñarme a valorar lo que me rodea, por confiar en mí, por tu paciencia, por compartir los mejores momentos de mi vida al igual que los peores y hacerlos más ligeros. Gracias por ser parte de mi vida y por el apoyo brindado.*

- *Gracias abuelita Ángela por tus cuidados, pues sin ellos no estaría aquí.*

- *Gracias a mi Asesor Francisco Javier Patiño Donnadieu, por la confianza brindada, por sus conocimientos compartidos durante el desarrollo de este trabajo y durante toda la carrera, gracias por su ayuda y por su amistad, sin usted no hubiera sido posible finalizar este trabajo.*

- *Gracias a la UNAM y a la FES Acatlán por abrirme sus puertas y llenarme de experiencias maravillosas, así como de conocimientos.*

- *Gracias a cada uno de los profesores de la carrera que me transmitieron sus conocimientos y experiencias.*

- *Gracias Andrés por tu amistad y por darme ánimo.*

- *Gracias Joaco, Adrián, Fermín por el apoyo en las clases y en cada trabajo que hicimos juntos, y por permitirme ser su amiga.*

- *Gracias Caro por ser mi compañera de aventuras.*

- *Gracias a todas las personas que me ayudaron a recopilar la información necesaria para la elaboración de este trabajo. (CIC, CIMAT, IIMAS, Instituto de matemáticas, Facultad de Ingeniería, CINVESTAV).*

ÍNDICE

INTRODUCCIÓN

En la actualidad tanto el sector público como privado se enfrenta a problemas de decisión, en los cuales se parte de una serie de bienes limitados o de requisitos mínimos que se tienen que cumplir para tomar la resolución correcta. Sin embargo, cuando las variables que influyen son demasiadas[1], el llegar a solucionar nuestro problema se vuelve complejo.

Un ejemplo de este tipo de dificultades es el Problema del Agente Viajero, en el cual un individuo o una compañía tienen que visitar varios lugares, ya sea para vender, repartir, comprar o promocionar su mercancía. Cuando el número de puntos que se deben visitar es demasiado grande, el encontrar la secuencia en la que deben ser visitados, de manera que la longitud de la distancia a recorrer sea la mínima, resulta complicado, pues la implementación de los métodos tradicionales, como la Búsqueda Exhaustiva o la Búsqueda Local, no es suficiente ya que las computadoras tardarían años en localizar la solución y esto ocasionaría una pérdida de tiempo la cual implica una disminución monetaria.

Es por esta razón que se decidió abordar este tipo de problemas con técnicas diferentes llamadas metaheurísticas. Las metaheurísticas son algoritmos que fueron diseñados para resolver problemas difíciles de optimización combinatoria.

El objetivo de este trabajo de investigación es realizar una comparación entre la metaheurística *Búsqueda Tabú* y la de *Algoritmos Genéticos* para encontrar cual de éstas resuelve de una manera óptima Problemas del Agente Viajero. La Hipótesis del trabajo de investigación es que ambas metaheurísticas resuelven de

[1] Se considera que el número de variables es grande cuando son mayores a 10.

manea óptima el Problema del Agente Viajero. Esto conllevó a efectuar una búsqueda de los trabajos realizados sobre este tema así como de los antecedentes de las metaheurísticas y el Problema del Agente Viajero.

Considerando los antecedentes encontrados, se hizo una investigación sobre las dos metaheurísticas mencionadas y posteriormente se realizó la confrontación entre éstas.

El presente trabajo se divide en cuatro capítulos. En el primero se citan los orígenes de las metaheurísticas: Búsqueda Tabú y Algoritmos Genéticos, asimismo personas que han trabajado con ellas y las aplicaciones que han hecho, así como las que han sido orientadas a Problemas del tipo del Agente Viajero.

En el segundo capítulo se describe en que consiste el Problema del Agente Viajero y el por qué de su dificultad para ser resuelto con los métodos tradicionales. Otro propósito de este apartado es explicar como funciona la Búsqueda Tabú y los Algoritmos Genéticos y los elementos con los que trabajan.

El capítulo número tres es la parte medular de este trabajo, pues de acuerdo con el marco teórico de cada una de las metaheurísticas se formularon dos modelos, uno para cada metaheurística que representa el problema a resolver.

El capítulo cuatro presenta la elaboración y ejecución de dos programas el primero siguiendo la metodología de la Búsqueda Tabú y el segundo aplicando la de los Algoritmos Genéticos. Se observaron los datos obtenidos y se realizó la comparación entre las metaheurísticas mencionadas.

Prefiero equivocarme creyendo en un Dios que no existe, que equivocarme no creyendo en un Dios que existe. Porque si después no hay nada, evidentemente nunca lo sabré, cuando me hunda en la nada eterna; pero si hay algo, si hay Alguien, tendré que dar cuenta de mi actitud de rechazo.

PASCAL

I. ANTECEDENTES.

A. Problema del Agente Viajero (PAV).

Un problema que ha tenido pensando a varios investigadores por largo tiempo es el PAV, pues a pesar de que es fácil de comprender, el hecho de encontrar un algoritmo que satisfaga los mínimos requerimientos, en tiempos y costos, de eficiencia y que lo resuelva ha sido complicado.

1. Definición.

Una definición, que me parece completa, de este problema es la siguiente:

> En su forma más sencilla, un viajante tiene que hacer una ruta que pasa por cierto número de ciudades, visitándolas todas

> una sola vez, y minimizando la distancia total recorrida. El problema es hallar la secuencia correcta en que hay que visitar las ciudades. Las limitaciones son que hay que visitar todas las ciudades, que se visita cada una sólo una vez, y que el viajante vuelve al punto inicial del viaje. La función de costo que hay que minimizar es la distancia total recorrida durante el viaje.[1]

La ciencia que se ha encargado de estudiar al PAV es la optimización combinatoria. Esta ciencia surgió cuando George Datzing quería encontrar el mínimo de una función lineal de un poliedro. Al principio la Programación Lineal (PL) fue la herramienta principal de la optimización combinatoria.

2. Componentes de un problema.

Existen tres aspectos que se deben considerar en la historia de cualquier problema matemático, los cuales son:

1. ¿Cómo surgió?

2. ¿Cómo la investigación de esto ha influenciado a otros desarrollos matemáticos?

3. ¿Cómo se solucionó el problema?

[1] Freeman y Skapura. Redes neuronales algoritmos, aplicaciones y técnicas de programación, pp. 155.

a. Origen.

Empezaremos por describir como surgió el PAV. El PAV como otros problemas matemáticos surge de la teoría de grafos, y al recorrido que tiene que encontrarse se le denomina circuito hamiltoriano[2].

Los problemas de grafos, en especial los de ciclos hamiltorianos han sido discutidos por varios científicos desde hace años, como Euler y Vandermonde quienes en 1759 se encargaron de estudiar el caso del recorrido del caballo, el cual consistía en visitar los sesenta y cuatro cuadros del tablero de ajedrez con el caballo, exactamente una vez. Otra regla era que el caballo sólo podía moverse en un paso de un cuadro a otro.

Pero la persona que posiblemente consideró a los ciclos hamiltorianos en una forma general fue Kirkman. Los trabajos que realizó no solamente se redujeron a los ciclos del poliedro, sino que creó el famoso rompecabezas de las 15 escuelas de niñas. En el libro *The Traveling salesman problem*[3], el autor refiere que Kirkman fue uno de los primeros en crear la teoría de los diseños combinatorios y de la clasificación del poliedro (figura 1).

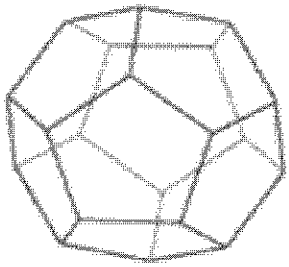


Figura 1.

[2] Se denomina circuito hamiltoriano desde el siglo diecinueve por el matemático irlandés: Sir William Rowan Hamilton.

[3] Lawler, Lenstra, Rinnooy y Shmoys D. *The traveling salesman problem.*

En la misma época en que Kirkman realizaba estas investigaciones, Hamilton inventó un sistema de álgebra no conmutativa y la llamó *Icosian calculus.* La representación gráfica de esta álgebra es considerada como la representación básica del rompecabezas denominado *The Icosian Game.*

Sin embargo un precursor más directo del PAV fue Menger con su trabajo denominado el problema del mensajero, en el cual la distancia de los arcos era de gran importancia, ya que se tenía que encontrar el camino más corto que uniera todos los puntos, donde las distancias eran conocidas.

La primera vez que se usó el PAV en matemáticas fue entre 1931 y 1932, sin embargo varios autores consideran que desde 1932 el concepto del PAV ya era utilizado, ejemplo de ello, se encuentra en el libro alemán "El agente viajero, como y que debe de hacer para lograr obtener comisiones y tener éxito en su negocio, por un agente vendedor experimentado".

En 1939 Mahalanobis presentó en una conferencia al congreso de la ciencia de la India un problema de estimación del área de la tierra de una granja, el objetivo era mantener los costos del estudio de la tierra por debajo de un parámetro mientras se maximizaba la exactitud de la estimación. Él tenía especial interés en encontrar el menor costo, para tal estimación.

Aunque no se sabe quien creó el nombre del PAV se asegura que el encargado de darle difusión dentro de la comunidad de la Investigación de Operaciones (IO) fue Merrill Flood.

En 1948 John Williams convenció a Flood de que popularizaran al PAV en la RAND Corporation

Se cree que el término PAV para problemas de optimización tuvo su origen en un documento publicado en los Estados Unidos en 1949.

En 1955 Morton y Land escribieron que el PAV posiblemente es similar al estadístico de la media de la distancia mínima del problema.

b. Influencia en otras investigaciones.

Para cubrir el punto número dos de los aspectos de la historia de un problema, es suficiente mencionar que desde la aparición del artículo: S*olution of a large-scale traveling-salesman problem* en el *Journal of Operations Research Society of America*, se han desarrollado diferentes trabajos en los cuales se han creado nuevos métodos para solucionar este tipo de problemas; por otro lado han surgido variantes del PAV. Una de las primeras variantes se presentó cuando lo que se tenía que reducir era tiempo ó costos. Sin embargo debido al hecho de la existencia de varios casos reales se han tenido que modificar algunos conceptos del PAV. Algunas de las variantes más comunes son:

- **El PAV máximo:** La diferencia con el PAV reside en encontrar un recorrido, de la lista de ciudades, que cubra la distancia máxima.
- **El cuello de botella del PAV:** el objetivo es encontrar un recorrido, de las ciudades existentes, de tal manera que se encuentre el costo menor de los costos más grandes de los arcos.
- **El PAV con múltiples visitas:** en esta variante se tiene que encontrar una asignación de ruta para el agente viajero, quien empezará en una ciudad dada, visitando cada ciudad por lo menos una vez y regresando a la ciudad

de donde partió, de tal manera que la distancia del camino recorrido sea mínima.

- **El PAV errante:** se dan dos ciudades especificas *a* y *b*, y se tiene que encontrar el mínimo circuito hamiltoriano entre a y b.

- **El PAV agrupado:** en este caso las ciudades colocadas en nuestro grafo se dividen en agrupaciones V_1, V_2,...V_k, el objetivo es encontrar la ruta con el menor costo sujeta a la restricción de que se deben visitar todas las ciudades pertenecientes a una agrupación consecutivamente.

- **El PAV generalizado:** Como en el caso anterior las ciudades se agrupan en V_1, V_2,..., V_k, el objetivo es encontrar el circuito más corto que pase exactamente por una ciudad de cada agrupación.

c. ¿Cómo solucionarlo?

Finalmente no podemos responder el punto tres, pues a pesar de las indagaciones realizadas no se ha encontrado un algoritmo que pueda solucionar al PAV sin importar el número de ciudades que contenga.

El PAV cada día tiene más aplicaciones, estas aplicaciones no solamente son el área de las matemáticas o la IO sino ha abarcado áreas como la ingeniería, la electrónica, la genética, la informática, etc. y si no se tiene un algoritmo que lo resuelva de una manera óptima, entonces hay una pérdida.

B. Metaheurísticas.

En IO existen diferentes técnicas para solucionar problemas, en las cuales se está sujeto a ciertas condiciones, algunos de los métodos más comunes son:

- Programación lineal.
- Programación dinámica.
- Métodos del gradiente.
- Métodos de ramificación y búsqueda.

Sin embargo estas técnicas han demostrado no ser factibles para resolver todo tipo de problemas, debido a que sus características son determinísticas y al enfrentarse con problemas estocásticos se vuelven deficientes.

Las características más comunes por la que algunos problemas no pueden ser resueltos por los métodos tradicionales son las siguientes:

- El número de posibles soluciones en el espacio de búsqueda es muy grande.
- El tiempo para encontrar las posibles soluciones es exponencial.
- La modelación del problema es difícil.
- Para resolver el problema se requiere de información que no está disponible o no es suficiente.

1. Clasificación de los problemas en investigación de operaciones.

En IO los problemas se clasifican de acuerdo a su complejidad en:

- **P (polinomial)**: Al hablar de un problema de complejidad P nos referimos a aquellos problemas que pueden ser resueltos en un tiempo polinomial en una máquina determinística, es decir sin importar lo que se haya hecho en un paso anterior, sólo existe otro paso que se debe hacer a continuación.
- **NP (polinomial no determinístico):** Un problema NP es aquel que puede ser resuelto en un tiempo polinomial pero en una máquina no determinística, esto es, en cada paso o algunos se muestra una serie de opciones para que la maquina pueda seleccionar la correcta.
- **NP-completos (polinomial no determinístico completo)**: En 1971 Cook demostró que existen problemas muy difíciles de resolver y los denominó NP-completos en los cuales se requiere de un tiempo exponencial para resolverlos, una muestra de este tipo de problemas es el problema del agente viajero.

Este último tipo de problemas pertenece a la optimización combinatoria, pues las variables de decisión son discretas. Estos problemas no pueden resolverse con métodos tradicionales, ya que su funcionamiento es lento y aún el hecho de implementarlos en una computadora, implica un tiempo muy grande para encontrar una solución. Es por esta causa que se requiere la creación de nuevas técnicas como lo expuso Glover:

> La clave para tratar con tales problemas es ir un paso más allá de la aplicación directa de la destreza y el conocimiento del

> experto y generar recursos para un procedimiento especial (o marco), las reglas expertas se pueden empatar, permitiendo un punto donde ninguna mejora puede percibirse, a menos que existan alternativas superiores.[4]

Es fácil entender que los problemas P pueden ser solucionados y bueno también es claro que P⊆NP, pero no podemos decir que NP⊆P pues entonces todos los problemas NP tendrían un algoritmo que los pudiera solucionar.

2. Definición.

En los últimos 40 años, los investigadores han tratado de crear métodos más eficientes como las heurísticas y las metaheurísticas.

Existen varias definiciones de heurística y metaheurística, sin embargo las más usuales son:

> Las heurísticas son métodos que se encargan de encontrar soluciones buenas (es decir casi óptimas) a un costo computacional razonable, aunque sin garantizar factibilidad u óptimalidad de las mismas. En algunos casos ni siquiera puede determinar que tan cerca del óptimo se encuentra una solución factible en particular.[5]

> Las metaheurísticas se refieren a una estrategia maestra que guía y modifica otras heurísticas para producir soluciones

[4] Glover y Laguna. *Tabu search, modern heuristic techniques for combinatorial problems,* pp. 70.
[5] Reeves. *Modern heuristic techniques for combinatorial problems,* pp. 22.

> más allá de las generadas normalmente. Estas metaheurísticas manejan procedimientos de alto nivel.[6]
>
> Las metaheurísticas son algoritmos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heuristícos clásicos no son ni efectivos ni eficientes. Las metaheurísticas proporcionan un marco para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la Inteligencia Artificial (IA), evolución biológica y mecanismos estadísticos.[7]

El término metaheurística fue incluido en el mismo documento donde fue introducido el término Búsqueda Tabú por Glover en 1986, desde entonces ha sido ampliado y empleado en diferentes documentos de investigación.

En los últimos diez años la evolución de las metaheurísticas ha tenido un desarrollo muy alto. Los investigadores dicen que esto se debe a que estas técnicas utilizan una búsqueda inteligente, debido a que han demostrado tener una habilidad para obtener buenas aproximaciones a soluciones óptimas en problemas NP-completos.

Hay diferentes metaheurísticas tales como: Búsqueda Tabú, Algoritmos Genéticos, Recocido Simulado, Colonia de Hormigas y Redes Neuronales, entre otras.

[6] Glover Fred y Laguna Manuel. *Tabu search,* pp. 17.
[7] Osman y Kelly. *Meta-heuristic: theory and applications,* pp. 39.

Sin embargo todas las metaheurísticas están formadas básicamente por los siguientes elementos:

- Un estado actual, el cual es un punto o un conjunto de puntos del espacio de búsqueda.
- Una vecindad definida por un operador, aplicado al estado actual para obtener la colocación del próximo estado. En algunos casos puede haber más de un operador como en el caso de los Algoritmos Evolutivos.
- Una regla de transición, llamada también estrategia de selección.
- Un criterio para decidir cuando debe parar el algoritmo.

Glover en 1997 en su libro *Tabu search*[8] menciona la principal clasificación de las metaheurísticas de acuerdo a sus características. Esta clasificación se basa en tres conceptos:

1) El uso de memoria adaptativa.
2) El tipo de exploración de vecindad usada.
3) El número de soluciones actuales llevadas de una iteración a la próxima.

Las metaheurísticas se basan en el criterio de búsqueda en una vecindad, este concepto fue introducido por Croes, quien usa esta idea para obtener mejores soluciones al PAV.

Dado que las metaheurísticas se basan en las características mencionadas, cuentan con una flexibilidad para poder modificar algunos de los movimientos en

[8] Glover. *Tabu search.*

el espacio de búsqueda, por ejemplo puede excluir algunos miembros o incluir a otros.

Otras de las características de las metaheurísticas que han llamado la atención es que contienen varios procesos que son similares a los procesos de la naturaleza, en el caso de los Algoritmos Genéticos se observa cuando hablamos de población, mutación y en la Búsqueda Tabú la similitud es que esta metaheurística utiliza una memoria adaptativa que permite almacenar situaciones que se han presentado, para después al tener un caso parecido se sepa que hacer sin tener que empezar desde cero, de igual manera con la utilización de la exploración sensible es posible, no solo recurrir a lo almacenado sino explorar nuevos caminos. Estas cualidades hacen que estas metaheurísticas sean consideradas parte de la Inteligencia Artificial.

Es importante mencionar que la Búsqueda Tabú y los Algoritmos Genéticos fueron evaluados y calificados en 1988, por el Committee on Next Decade of Operations Research (CONDOR) como técnicas extremadamente promisorias para el tratamiento futuro de aplicaciones prácticas.

C. Búsqueda Tabú (BT).

La creación de la metaheurística Búsqueda Tabú (BT) se debe a la necesidad de crear técnicas para resolver problemas que los algoritmos exactos no pueden. Esta técnica se deriva de conceptos de la IA y de la IO, sin embargo varios investigadores han hecho indagaciones por separado en estos campos, sin considerar la estrecha relación que existe entre ellos, pues aunque empezaron a

crecer sin seguir la misma meta, al final su objetivo ha sido desarrollar métodos que resuelvan el mismo tipo de problemas.

De igual manera, el diseño correcto de modelos utilizando BT impulsa a realizar exploraciones, las cuales provocan el desarrollo de nuevos métodos en la IA y la IO. De esta manera las metaheurísticas no solo toman principios de estas áreas, sino proporcionan las bases para crear nuevas técnicas.

Por lo anterior varios artículos han insistido en que la IA y la IO tienen que acercarse para encontrar métodos óptimos, pero a pesar de esas insistencias el alejamiento entre estas dos áreas continuó por varios años, debido a que la IO siguió preocupándose por los resultados matemáticos y la convergencia, mientras que la IA ponía más atención a los análisis cualitativos. Fue así como estas ideas permanecieron en el olvido y no fue hasta mediados de los ochenta cuando algunos investigadores trataron de introducir el uso de memoria y de procedimientos probabilísticos a las técnicas existentes.

Es entonces cuando los investigadores lograron crear el puente entre la IO y la IA y surgió la técnica que hoy conocemos como búsqueda tabú.

1. Definición.

Fred Glover fue la persona que desarrollo la Búsqueda Tabú, su nombre así como su metodología se dieron a conocer en el año de 1986. Las principales ideas en las que se basa este método son: el diseño de procedimientos que permitan cruzar los límites de la factibilidad o la óptimalidad local para que se puedan explorar zonas que no son permitidas en el espacio de búsqueda.

Podemos decir que la BT guía la búsqueda local de tal manera que se supere la óptimalidad local, esto lo hace introduciendo capacidades de aprendizaje (llamada memoria), por lo tanto decimos que la búsqueda es inteligente, el objetivo es obtener un óptimo global.

> La filosofía de la BT es la de manejar y explotar una colección de principios para resolver problemas de manera inteligente. Uno de los elementos fundamentales de la BT es el uso de memoria flexible, desde el punto de vista de la BT, la memoria flexible envuelve el proceso dual de crear y explotar estructuras para tomar ventaja mediante combinar actividades de adquisición, evaluación y mejoramiento de la información de manera histórica.[9]

La BT sigue procedimientos determinísticos pues los investigadores que la crearon decidieron que era mejor tener una mala decisión y saber de donde provenía, que tener una buena solución al azar sin saber cuales habían sido sus orígenes; la explicación es sencilla, la existencia de una buena solución al azar no proporciona información para futuras decisiones, sin embargo una solución que se sabe de donde proviene nos guía hacia las siguientes acciones, en el caso de que sea mala sabremos que la búsqueda debe de orientarse hacia otro lado.

[9] Cobos Silva Sergio. *La técnica de la búsqueda tabú y sus aplicaciones*, pp. 25.

2. Principios de la BT.

Hubo cuatro principales ideas que dieron la pauta para crear la BT:

1. Estrategias que combinaran reglas de decisión basadas en reglas de reestructuración lógica.
2. La violación sistemática y la restauración de la factibilidad.
3. Memoria flexible.
4. Procesos selectivos, para combinar soluciones, aplicados a una población.

La primera de estas ideas surgió del estudio de las reglas de decisión de una empresa con problemas de calendarización. Los métodos que se usaban en los sesenta para este tipo de problemas se basaban en una serie de decisiones que generaban los calendarios, incrustándolos en criterios de decisión local.

En 1963 Fisher y Thompson diseñaron una alternativa en la cual, por medio de estrategias probabilísticas, había múltiples reglas en cada nodo de decisión. Estas estrategias usaban aprendizaje para corregir las elecciones tomadas probabilísticamente y así lograr obtener varias soluciones factibles.

Glover consideró esta idea y utilizó las reglas de decisión para crear un método que las combinara y así crear nuevas reglas, de esta manera el método no paraba en el óptimo local, sino generaba otros procedimientos que crearan soluciones adicionales.

La segunda idea se basa en los métodos que se encargaban de solucionar problemas enteros. En 1969 Glover decidió agregar memoria a cada variable, de

esta manera la memoria crearía restricciones que evitarían la generación de soluciones inservibles. Las restricciones se codifican para guardar los atributos de las soluciones, en lugar de guardar toda la solución. Esto provocó que se introdujera otra estrategia la cual combinaba dos soluciones y generaba las soluciones que podrían ser óptimas.

Nuevamente el método no paraba en un óptimo local. Esta idea, de la codificación de las restricciones, contribuyó al uso de la memoria flexible y de la combinación de las restricciones para controlar las soluciones que se generan.

La tercera idea se basa de igual manera en los métodos utilizados en la programación entera, solo que ésta hace referencia en especial al método simplex utilizado en la programación lineal. Este método utiliza procedimientos de corte de planos para introducir nuevas restricciones de acuerdo a los requerimientos de las variables enteras. Sin embargo este método no puede utilizarse cuando el área esta desconectada y no es convexa. Fred Glover, en 1968, decide crear un procedimiento al cual llama seudo primal-dual, éste viola y reestructura las condiciones factibles, haciendo posible el corte por varias regiones del espacio de solución.

La convergencia en la óptimalidad se aseguraba por las reglas que siempre permitieron la factibilidad. Esta estrategia permitía la búsqueda tanto en las zonas factibles como no factibles. La habilidad de explorar ambas regiones ha llegado ha convertirse en una de las características principales de la BT, llamada técnica de oscilación.

La cuarta idea se genera en 1965, Glover introduce en ella métodos de restricciones substitutas para la programación entera. Estos métodos se basaron en

la idea de combinar restricciones para formar otras, con la finalidad de producir información que no estuviera contenida en las restricciones principales.

BT ha provocado gran interés entre la comunidad de la IO, y en los últimos años varios científicos han agregado nuevos elementos a la BT, entre éstos se han agregado procedimientos probabilísticos.

Debido a que se han obtenido resultados óptimos con esta técnica, la investigación ha crecido y ha creado campos como son: el Aprendizaje Tabú, Maquina Tabú, Diseño Tabú y la Búsqueda Dispersa entre otros.

D. Algoritmos Genéticos (AGs).

En los últimos 50 años ha crecido el interés en resolver problemas por medio de sistemas basados en los principios de la evolución y la herencia.

Ángel Kuri[10] declara en su libro algoritmos genéticos:

> Si la naturaleza ha sido capaz de generar organismos óptimos para desempeñarse en medios ambientes sumamente complejos, por qué no copiar sus métodos para resolver nuestros propios problemas y tratar de encontrarles soluciones que, de alguna manera, sean óptimas.[11] (SIC) error

Dentro de estas nuevas técnicas encontramos a los Algoritmos genéticos (AGs) considerados como una metaheurística que forma parte de la computación

[10] Kuri Morales Ángel y Galaviz Casas José. *Algoritmos genéticos.*
[11] Kuri Morales Ángel y Galaviz Casas José. *Algoritmos genéticos,* pp. 9.

evolutiva (CE). Los AGs están inspirados en la teoría de la evolución; siguiendo el principio de la supervivencia del más apto (expuesta por Carlos Darwin en 1858).

La teoría de Carlos Darwin, como lo expone en su libro *The origin of the species,* nos dice que las especies sufren cambios adaptativos a lo largo de períodos de tiempo, debido a la influencia del medio ambiente. Darwin se había dado cuenta de que si una especie no sufría cambios se volvía incompatible con su ambiente, pues éste tiende a cambiar con el tiempo. De igual manera se percató de que ciertas características son hereditarias de generación en generación, permitiendo que los organismos que en un principio fueron raros se vuelvan comunes, y ocasionando que los organismos anteriores desaparezcan, de esta manera se habrá dado un paso en el proceso evolutivo. Estos cambios ocurren con la finalidad de hacer a los individuos más aptos para sobrevivir.

1. Nacimiento del Algoritmo Genético.

La evolución empezó a verse como un proceso de aprendizaje desde 1930. W. D. Cannon plantea en su libro *The wisdom of the body* que el proceso evolutivo es similar al aprendizaje obtenido por prueba y error el cual se presenta en los seres humanos. Alan Mathison Turing afirma en su articulo *Mind* publicado en la revista Computing machinery and intelligence *(considerado hoy clásico en la IA)* que existe una conexión entre la evolución y el aprendizaje de máquina.

Pero no fue sino hasta después de los 50's, que las características de los procesos de la evolución natural, empezaron a llamar la atención de varios investigadores y se empezó a simular los procesos evolutivos en computadoras.

A finales de 1950 Alexander S. Fraser, publicó varios trabajos de la evolución de sistemas biológicos en computadoras digitales. En sus trabajos introduce la representación binaria, una operación de cruza probabilística, una población de padres para generar una población de hijos tras recombinarse y un mecanismo de selección.

Casi al mismo tiempo que Fraser, el estadístico inglés George E. P. Box propusó un enfoque evolutivo para la optimización de la producción industrial, la técnica fue llamada Evolutionary Operation (EVOP). La técnica consistía en efectuar pequeños cambios a un conjunto de parámetros de producción, vigilando ciertos datos estadísticos de los procesos para dirigir la búsqueda. Estos cambios eran una analogía con la mutación en la naturaleza. EVOP era un proceso iterativo y sólo podía automatizarse una parte de este proceso.

Entre 1953 y 1956 Nils Aall Barricelli elaboró lo que probablemente fue una de las primeras simulaciones de un sistema evolutivo en una computadora.

R. M. Friegberg es considerado, por los recientes investigadores en CE, como uno de los primeros en intentar evolucionar programas de computadoras; aunque Friegberg no uso la palabra evolución, es evidente que fue el enfoque que siguió. Su trabajo fue publicado en 1959, en éste se logró superar una búsqueda aleatoria, pero no se pudo resolver el problema del estancamiento local.

En 1959 George J. Friedman propone en su tesis de maestría, la aplicación de las técnicas evolutivas a la robótica, a pesar de que su trabajo no se puso en practica dio la pauta que conduciría a otros investigadores a diseñar maquinas pensantes.

En 1962 Hans Joachim Bremermann vio a la evolución como un proceso de optimización, realizó una de las primeras evoluciones usando cadenas binarias que se procesaban por medio de reproducción (podía ser sexual o asexual), selección y mutación. Introdujo la noción de función de aptitud.

Bremermann utilizó esta técnica para problemas de optimización con restricciones lineales, la idea de su propuesta consistía en usar a un individuo factible que se cambiaba a través de un operador de mutación, de igual manera se creaba un conjunto de posibles direcciones de movimiento. Además utilizó operadores de recombinación y fue uno de los primeros en utilizar el concepto de población.

En ese mismo año John H. Holland, junto con otros colegas y alumnos de la Universidad de Michigan, después de haber estudiado por dos años los procesos lógicos involucrados en la adaptación, se dieron cuenta de que el uso de reglas simples podía generar comportamientos flexibles y vieron la oportunidad de estudiar la evolución de comportamientos en un sistema complejo. Holland aseguraba que se podían encontrar soluciones aproximadas a problemas complejos mediante un proceso de evolución.

Holland analizó el proceso de adaptación como programas de una población que interactúan y mejoran sobre la base del ambiente el cual se encargará de determinar su comportamiento.

Holland desarrolló esta técnica con la intención de elaborar un estudio formal acerca de la adaptación en sistemas naturales y artificiales, es decir realizar un modelo para la adaptación.

Fue en 1975 cuando la investigación realizada por Holland se dio a conocer con el nombre de *Planes Reproductivos y Adaptativos*, después esta técnica se popularizó con el nombre de Algoritmo Genético.

A pesar de que los AGs fueron creados como métodos para el aprendizaje de máquina y procesos de adaptación, en la actualidad se han utilizado como herramientas de optimización.

En los siguientes años surgieron otras técnicas similares a los AGs como las Estrategias Evolutivas[12] y la Programación Genética[13].

2. Definición.

Los Algoritmos Genéticos utilizan un operador principal denominado cruza sobre un operador secundario llamado mutación y una selección estocástica.

Una definición de AGs dada por John Koza es la siguiente:

> El algoritmo genético es un algoritmo matemático altamente paralelo que transforma un conjunto (población) de objetos matemáticos individuales (típicamente de cadenas de caracteres de longitud fija que se ajustan al modelo de las cadenas de cromosomas), cada uno de los cuales se asocia con una aptitud, en una población nueva (la siguiente generación) usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y sobrevivencia del más apto y

[12] Creadas por estudiantes de posgrado en la Universidad Técnica de Berlín, Alemania en 1963.
[13] Elaborada por John R. Koza en la década de los 80's.

> tras haberse presentado de forma natural una serie de operaciones genéticas (notablemente la recombinación sexual).[14]

En los últimos años los AGs se han vuelto una de las metaheurísticas más usadas, debido a que mediante estos se ha conseguido obtener resultados satisfactorios para diferentes tipos de problemas, a los cuales no era posible abordar con métodos matemáticos tradicionales.

Muchos investigadores han reconocido que la similitud de la manera en que trabajan los AGs con la naturaleza es el motivo por el cual esta técnica ha obtenido resultados favorables.

En la actualidad los AGs han llegado a ser utilizados preferentemente como técnicas de búsqueda de soluciones óptimas y han sido aplicados con éxito en importantes áreas como: en ingeniería, teoría de juegos, aprendizaje de máquina, reconocimiento de patrones, generación de gramáticas, consultas en bases de datos y sobre todo en optimización combinatoria.

Dentro de las aplicaciones en optimización combinatoria encontramos que los AGs han podido resolver Problemas del tipo del Agente Viajero, esto es debido a que los operadores de cruza y mutación permiten realizar una búsqueda en todas las combinaciones de las ciudades, que son más factibles de acuerdo a una función de aptitud. Debido a lo anterior importa que el número de ciudades sea grande.

Una característica importante de esta metaheurística se debe a que logra superar el estancamiento local.

---

[14] John R. Koza. *Genetic programming. On the programming of computers by means of natural selection,* pp. 33.

Los AGs al igual que las otras metaheurísticas siguen siendo estudiadas y varios investigadores han propuestos diferentes tipos de cruzamiento, mutación, selección y han dado otros tipos de representación aparte de la binaria. De igual manera estos estudios han proporcionado la base para crear nuevos métodos híbridos como los Algoritmos Meméticos o el Algoritmo Genético Messy.

La evolución no es una fuerza, sino un progreso; no una causa, sino una ley.

JOHN MORLEY

II. CONCEPTOS BÁSICOS.

A. Componentes del PAV.

El PAV ha llegado a ser uno de los problemas más estudiados en la optimización combinatoria, debido al gran número de aplicaciones que tiene y al trabajo computacional que implica resolverlo. Por este motivo, recientemente se han realizado investigaciones dirigidas a la búsqueda de nuevos métodos que sean capaces de resolver al PAV de una manera óptima.

La definición conceptual del PAV, como se mencionó en el capítulo anterior, es encontrar la ruta mínima que el agente debe seguir por todo el conjunto de ciudades que tiene en su mapa, recordando que sólo se debe visitar una vez cada ciudad y además al final del recorrido se debe regresar a la ciudad de donde partió. Por lo tanto la ruta que se debe encontrar es cíclica y corresponde a un

circuito hamiltoriano, en donde cada nodo representa a una ciudad y los arcos el costo que implica viajar a esa ciudad.

1. Definición matemática.

La definición matemática del PAV es la siguiente:

Sea G=(V, E) un grafo dirigido o no y F la familia de todos los circuitos hamiltorianos posibles. Y para cada arco e $\in$ E, le corresponde un costo C_e dado. Se debe encontrar f $\in$ F tal que la suma de los costos de los arcos sea la mas pequeña.

La letra V representa los nodos o vértices y V={1,2,3,...,n}. El costo o la distancia de los arcos se puede representar en una matriz de adyacencia llamada matriz de costo simbolizada por C, en donde $C=(c_{ij})_{nxn}$, donde c_{ij} representan el costo entre el nodo i con el nodo j en G.

Abraham Punnen escribió en su artículo *The traveling salesman problem: applications, formulations and variations*[1] que el PAV puede ser visto como un problema de permutación y de acuerdo a ésta la representación puede ser lineal o cuadrática.

En este trabajo nos ocuparemos de la representación lineal en donde únicamente las permutaciones cíclicas son consideradas, y cuyo objetivo es encontrar X_{ij} que:

[1] Gutin y Punnen. *The traveling salesman problem and its variations.*

$$Min\ z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij}$$

sujeto a

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1,2,..,n.$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1,2,..,n.$$

$$v_i - v_j + n x_{ij} \leq n-1, \quad 1 \leq i \neq j \leq n$$

Donde:

x_{ij}=1 significa que el agente viajero va desde la ciudad i a la ciudad j.

x_{ij}=0 significa que el agente viajero no va de la ciudad i a la ciudad j.

c_{ij} es la distancia o el costo entre la ciudad i y j.

v_i y v_j son números reales arbitrarios.

La primera igualdad, perteneciente al grupo de las restricciones, asegura que el agente viajero visite una vez cada ciudad. La segunda igualdad asegura que el agente viajero recorra todas las ciudades. La última desigualdad permite cerciorarnos de que cada circuito contenga una ciudad con el valor de cero. La formulación anterior del PAV fue hecha por Miller.

La matriz de costo nos sirve para visualizar de que tipo es el PAV, si C es una matriz simétrica, nos indica que el grafo es no dirigido y que $c_{ij}=c_{ji}$ para toda i, j y se denomina PAV simétrico. Pero si C es una matriz no simétrica entonces se trata de un grafo dirigido en donde la $c_{ij}\neq c_{ji}$ para cualquier i, j y se le denomina PAV asimétrico. En las siguientes figuras se muestra un ejemplo posible de cada

uno de los tipos del PAV con sus respectivas matrices, en donde los costos de las redes fueron escogidos aleatoriamente.

La siguiente red corresponde a un grafo no dirigido (ejemplo del caso en donde el PAV es simétrico)

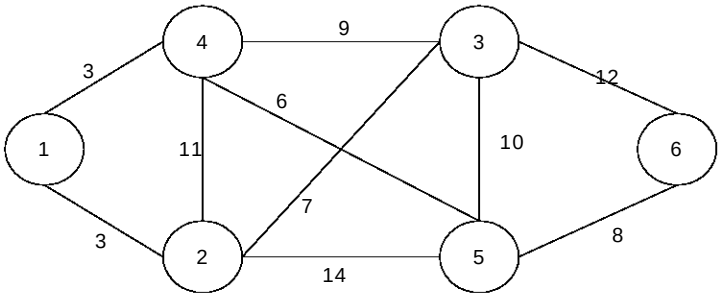


Figura 1

Matriz de costo de la Figura 1 (La matriz es simétrica)

$$c = \begin{bmatrix} 0 & 3 & 0 & 3 & 0 & 0 \\ 3 & 0 & 7 & 11 & 14 & 0 \\ 0 & 7 & 0 & 9 & 10 & 12 \\ 3 & 11 & 9 & 0 & 6 & 0 \\ 0 & 14 & 10 & 6 & 0 & 8 \\ 0 & 0 & 12 & 0 & 8 & 0 \end{bmatrix}$$

La siguiente red corresponde a un grafo dirigido (Ejemplo en donde el PAV es asimétrico)

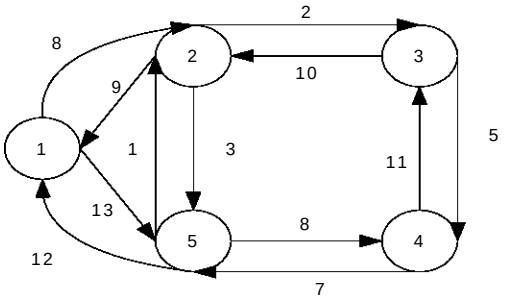


Figura 2

Matriz de costo de la figura 2 (La matriz es asimétrica)

$$c = \begin{bmatrix} 0 & 8 & 0 & 0 & 13 \\ 9 & 0 & 2 & 0 & 3 \\ 0 & 10 & 0 & 5 & 0 \\ 0 & 0 & 11 & 0 & 7 \\ 12 & 1 & 0 & 8 & 0 \end{bmatrix}$$

El PAV simétrico es un caso especial del PAV asimétrico. En esta investigación nos ocuparemos del PAV simétrico debido a que todo PAV asimétrico se puede representar como un PAV simétrico duplicando el número de nodos.

2. Complejidad computacional.

La dificultad computacional reside en que el espacio de búsqueda es muy grande, y el tiempo que se necesita para explorarlo, es también grande. Existen diferentes tipos de complejidad computacional, en donde el tiempo de ejecución depende del tamaño del espacio de búsqueda. En la tabla 1, tomada del libro

Genetic algorithms and grouping problem[2], se observa como crece el tiempo de ejecución de un algoritmo en diferentes tipos de complejidades.

	Tamaño del espacio de búsqueda					
Complejidad	10	20	30	40	50	60
O(x)	$1x10^{-5}$ seg.	$2x10^{-5}$ seg.	$3x10^{-5}$ seg.	$4x10^{-5}$ seg.	$5x10^{-5}$ seg.	$6x10^{-5}$ seg.
$O(x^2)$	$1x10^{-4}$ seg.	$4x10^{-4}$ seg.	$9x10^{-4}$ seg.	$16x10^{-4}$ seg.	$25x10^{-4}$ seg.	$36x10^{-4}$ seg.
$O(x^3)$	0.001 seg.	0.008 seg.	0.027 seg.	0.064 seg.	0.125 seg.	0.216 seg.
$O(x^5)$	0.1 seg.	0.32 seg.	24.3 seg.	1.7 min.	5.2 min.	13.0 min.
$O(2^x)$	0.001 seg.	1.0 seg.	17.9 min.	12.7 días	35.7 años	366 siglos
$O(3^x)$	0.059 seg.	58.0 min.	6.5 años	3855 siglos	$2x10^8$ siglos	$13x10^{12}$ siglos

Tabla 1

El PAV pertenece a los problemas con dificultad computacional $O(2^x)$.

Para explicar como el espacio de búsqueda crece en el PAV tomaremos como referencia la siguiente red:

[2] Falkenauer Emanuel. *Genetic algorithms and grouping problem*, pp. 194

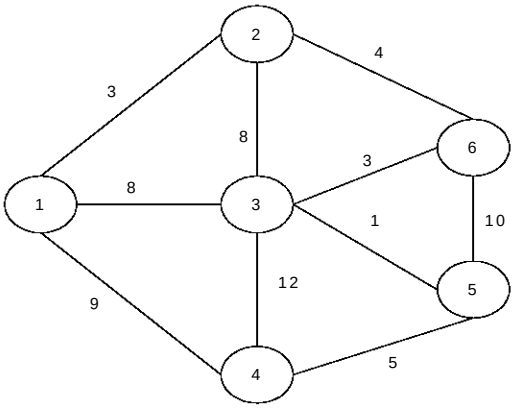


Figura 3

Los nodos de la red anterior representan ciudades y los arcos las distancias o los costos de un nodo a otro.

Observamos que algunas de las rutas posibles son:

1-4-3-5-6-2-1

5-6-2-1-4-3-5

3-5-6-2-1-4-3

Con estas rutas podemos ver que todos los circuitos son iguales a pesar de que se empieza en un nodo diferente, de aquí vemos que existen n recorridos para cualquier nodo n de la red. Después, considerando los cambios que se pueden hacer entre los nodos de cada ruta posible, obtenemos que existen $n!$ posibles soluciones, claro podemos hacer más pequeño este número pero no lo suficiente, si consideramos la misma red de la figura 3, encontramos que unas rutas posibles son:

1-2-6-5-4-3-1 con un costo de 42

1-3-4-5-6-2-1 con un costo de 42

Como se puede observar el costo en ambas rutas es el mismo, pues sólo se invirtió la ruta, entonces cada ruta será igual a su inversa y el espacio de búsqueda se reduce a $\frac{n!}{2}$, pero aún así el espacio resulta demasiado grande.

Ahora tomando en cuenta que el viajero visita todas las ciudades de la red una vez y regresa a donde partió, podemos ver que lo que hay que encontrar son las permutaciones circulares de las n ciudades y elegir la permutación que recorra la distancia mínima. Llamaremos al espacio de búsqueda S por lo tanto $S = \frac{(n-1)!}{2}$**.** A pesar de esta reducción el espacio sigue siendo grande si n es mayor a 6, por ejemplo:

Si n=6 entonces S=60,

Si n=7 entonces S=360,

Ahora si n=10 entonces S=181,000,

Si n=20 entonces S=10, 000, 000, 000, 000, 000

Si n=50 entonces S=100,000,000,000,000,000,000,000,000,000,000,000,000,
000,000,000,000, 000,000,000,000,000

Para tener una idea de la inmensidad de ese número como lo cita el Dr. Carlos Coello basta saber que existen 1, 000, 000, 000, 000, 000, 000, 000 litros de agua en el planeta.

3. Métodos existentes para solucionar el PAV.

En años recientes, cuando n es pequeña, muchos algoritmos exactos se han aproximado a la solución óptima global como:

- Algoritmo del gradiente
- Árboles de expansión
- Inserción cercana y lejana

Sin embargo uno de los tres métodos más utilizados para resolver el PAV, ha sido la búsqueda exhaustiva, esta técnica consiste en numerar todas las soluciones factibles y evaluarlas en la función objetivo para después seleccionar la que resultó óptima. Desafortunadamente la búsqueda exhaustiva sólo se puede utilizar cuando *n* no es mayor a 10, por ejemplo considerando el caso en donde *n* es igual a 50, el obtener el resultado tardaría años y en IO es preferible encontrar rápidamente una solución buena en lugar de la mejor, si ésta se encuentra demasiado tarde para resultar útil.

El segundo método que se ha usado es la formulación del PAV como un problema de programación lineal, donde la pertenencia o no de una variable se representa de forma binaria; es decir la variable toma el valor de uno si se sigue por esa ciudad y de cero si se evita esa ciudad. Sin embargo al igual que los otros métodos, ésta se vuelve inaplicable cuando n es demasiado grande.

El tercer método que ha logrado obtener buenas soluciones para el PAV, en los últimos cuarenta años es la búsqueda de vecindad.

En los años sesenta diferentes investigadores dejaron abierta la siguiente pregunta: ¿Existe un algoritmo polinomial para resolver el PAV? a tal pregunta contestó en 1970 Karp: que si existierá un algoritmo para resolver el PAV, se podrían resolver otros, pero hasta ahora, como lo dice Ángel Kuri: "...no se conoce un algoritmo determinístico que resuelva el problema en un tiempo que dependa polinomialmente de n..."[3].

B. Elementos de la Búsqueda Tabú.

La BT es una técnica que permite explorar el espacio de búsqueda de tal manera que se aproxima a un óptimo global y no se queda atrapado en un óptimo local. El principio de la BT es que se debe utilizar una memoria para encontrar soluciones superiores a las encontradas con los métodos tradicionales.

1. Definición matemática.

> Dada una función f(x), a ser optimizada en un conjunto X, la BT empieza de la misma manera que cualquier búsqueda local, procediendo iterativamente de un punto (solución) 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.[4]

[3] Kuri Morales Ángel y Galaviz Casas José. *Algoritmos genéticos,* pp. 88.

[4] Díaz, Glover, Ghaziri, González, Laguna, Moscato y Tseng. *Optimización heurística y redes neuronales,* pp. 108.

En forma conceptual para minimizar f(x) es necesario moverse paso a paso desde una solución inicial factible a soluciones vecinas y guiar estos movimientos utilizando el conocimiento previo acerca de las iteraciones previas.

Al procedimiento de transición de un punto a otro se le denomina movimientos y puede ser descrito por uno o más atributos.

Las características importantes de la BT son:

- Necesita una solución previa del problema, para resolverlo o encontrar una solución cercana al óptimo global.
- La selección de los elementos para realizar la búsqueda necesita del establecimiento de reglas eficientes para la generación y evaluación de movimientos.

El procedimiento que utiliza la BT supera a la búsqueda local utilizando una estrategia que reemplaza a N(x) por $N^*(x)$ de acuerdo al avance de la búsqueda. La memoria que utiliza la BT es la encargada de determinar hacia donde se dirige la exploración del espacio de búsqueda. La forma de establecer cuales soluciones se permiten en $N^*(x)$ y cuales no, son varias, una de las más utilizadas lo hace mediante un criterio establecido, el cual generalmente es una penalización, la cual les prohíbe pertenecer a $N^*(x)$, clasificándolas como tabú, es decir soluciones que estarán prohibidas.

Para guardar las soluciones tabú se crea una lista, en la implementación es un arreglo circular con el proceso FIFO (el primero en entrar es el primero en salir). El tamaño de esta lista es de siete, el por qué de este número, no tiene ningún

motivo, por otro lado en otras implementaciones su tamaño esta dado por $t = \sqrt{n}$ donde *t* es el tamaño y *n* representa el tamaño del problema.

Para evitar caer en ciclos, la BT revisa si la solución analizada ha sido visitada anteriormente. Y esto lo hace por medio de la lista tabú. Considerando el tamaño de memoria del que se tendría que disponer, una opción que podría ayudar a resolver este problema sería no volver a visitar las soluciones revisadas en la última iteración, o de otra manera no analizar las soluciones visitadas durante las últimas T_s iteraciones, siendo ésta última más eficiente al prevenir ciclos.

2. Algoritmo de la Búsqueda Tabú.

De una manera general, el procedimiento tabú puede ser implementado tomando como referencia el siguiente algoritmo, tomado de libro de Ángel Kuri[5]

PROCESS (TABÚ)

Begin

x=selecc(X)

x^*=x

H=∅

max=b(H,x)

calcular (NC(H,x))

elegir x′ tal que b(H,x′)=max {b(H,y) | y ∈ NC(H,x)}

repeat

x=x′

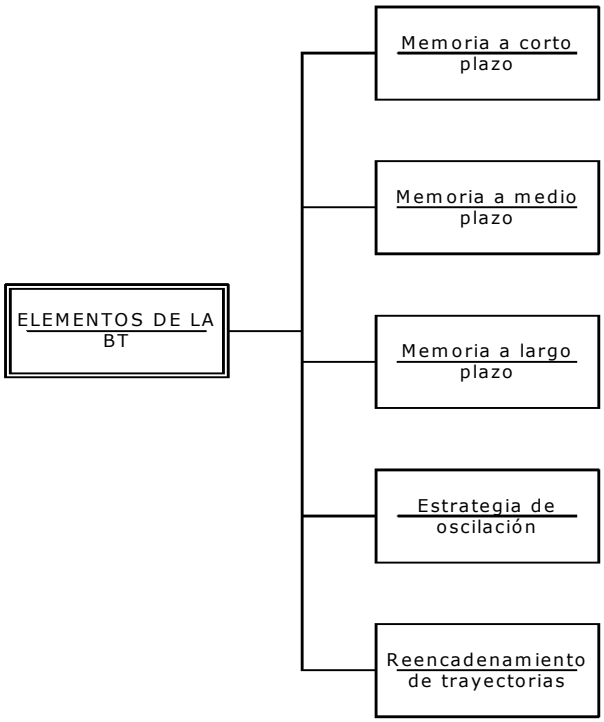
[5] Kuri y Galaviz. *Algoritmos genéticos,* pp. 43.

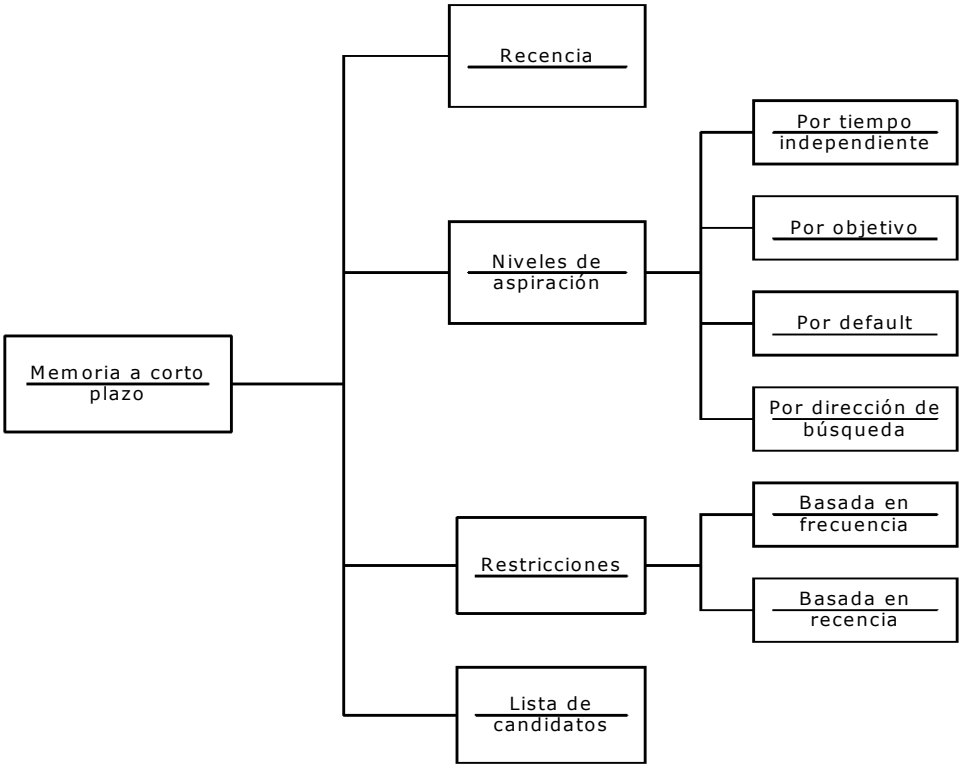
```
    if(b(H,x)>b(x*))
    then
        max=b(H,x)
        x*=x
    end if
    actualizar(H)
    actualizar(NC(H,x))
    elegir x' tal que b(H, x')=max {b(H,y) | y ∈ NC(H,x)}
  until(condición termina)
  return(x*)
end
```

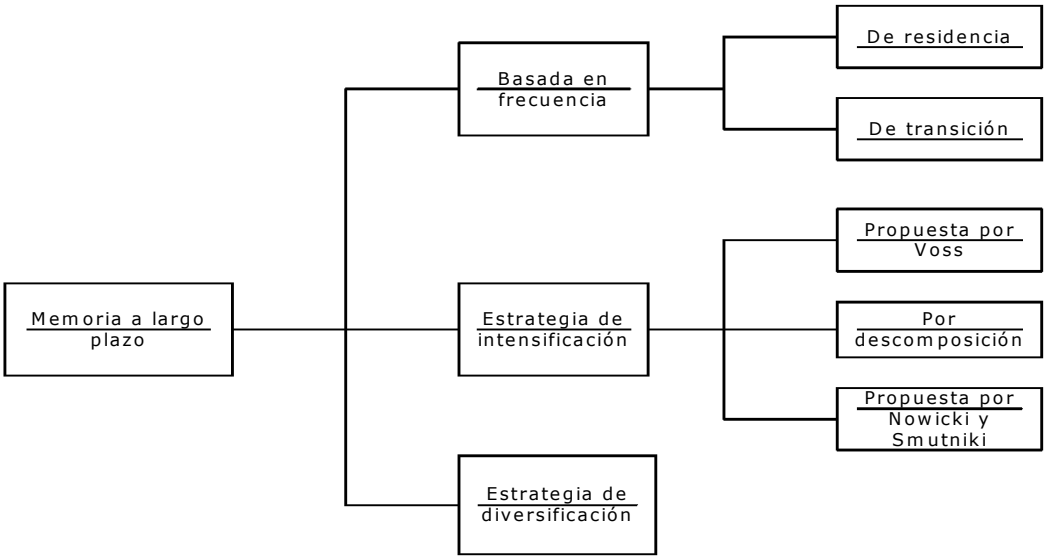
En donde X representa al dominio de búsqueda y x es una posible solución que pertenece a X. H simboliza las soluciones visitadas, es decir la historia de la búsqueda, NC(H,x) es una muestra representativa de la vecindad perteneciente a N(H,x) y b(H,x) representa a la función modificada, que se encarga de evaluar si una solución es posible o no.

3. Diagrama de los elementos de la BT.

Dado que la BT ha sido estudiada desde su nacimiento, se han hecho contribuciones a esta técnica, por tal motivo existen varios elementos que han sido implementados, pero los más sobresalientes se muestran en los siguientes diagramas.


ELEMENTOS DE LA BT
Memoria a corto plazo
Memoria a medio plazo
Memoria a largo plazo
Estrategia de oscilación
Reencadenamiento de trayectorias


Memoria a corto plazo
Recencia
Niveles de aspiración
Por tiempo independiente
Por objetivo
Por default
Por dirección de búsqueda
Restricciones
Basada en frecuencia
Basada en recencia
Lista de candidatos



El uso de memorias en la BT se considera como estrategias de aprendizaje ya que se encargan de recolectar información durante la ejecución del algoritmo de la BT y esta información la usan para dirigir las búsquedas siguientes.

La memoria utilizada en la BT puede ser explícita o basada en atributos. La primera conserva soluciones completas y sólo guarda las mejores soluciones analizadas durante la búsqueda. La segunda guarda la información de los atributos de las soluciones que cambian de una solución a otra.

Algunos ejemplos de atributos son los siguientes:

1. Cambio de una variable seleccionada x_k de 0 a 1.
2. Cambio de una variable seleccionada x_j de 1 a 0.
3. El cambio combinado de los dos anteriores tomados juntos.

4. Memoria a corto plazo.

Esta memoria se implementa con mayor frecuencia que la memoria a largo plazo y es la encargada de contar los atributos de las soluciones. Para sacar la mayor ventaja posible de esta memoria a los atributos seleccionados de las recientes visitas se les asigna el nombre de *"tabú-activos"* y a las soluciones que contengan este tipo de atributos se les convierte en soluciones *"tabú"*. De esta manera se evita el tener que inspeccionar dos veces las soluciones recientemente revisadas, pues estas no pertenecerán a $N^*(x)$.

Esta memoria está basada en la premisa de que las soluciones con altas evaluaciones tienen mayor posibilidad de conducir a una solución cercana a la óptima global en pocos pasos. El proceso termina cuando un cierto número de iteraciones (establecido previamente) se ha ejecutado o hasta que encuentre la mejor solución.

Como se puede observar en los diagramas anteriores, la memoria a corto plazo consta de cuatro elementos, los cuales se explican a continuación.

a. Recencia.

En la BT se prohíben ciertos movimientos, los denominados tabú, sin embargo el prohibir absolutamente todos nos conduciría a un límite en nuestra búsqueda, es por ello que es más conveniente prohibir los movimientos más recientes.

La recencia es un proceso en el que se basa el manejo de la memoria, radica en establecer por lo menos una lista tabú, en la cual se deben guardar los atributos denominados "tabú-activos". El tiempo que un atributo permanece tabú-activo es

llamado *"tenencia-tabú"* (en la implementación computacional es medido en iteraciones). La tenencia tabú puede ser fija para todas las iteraciones o variar de una a otra, también puede variar de acuerdo a los atributos.

Para implementar la recencia se necesita crear una lista con todas las soluciones que se han visitado recientemente, claro que esto implica un costo computacional grande por el espacio requerido en memoria, así que dependiendo del tamaño del problema, algunas veces será mejor guardar solo los atributos de las soluciones.

b. Niveles de aspiración.

Este criterio permite cierta flexibilidad en la BT, su objetivo es eliminar el estado tabú de una solución para que ésta pueda ser admisible en un movimiento. Por ejemplo, si encontramos una solución que sea mejor que las anteriores que hayamos encontrado esta debe ser considerada, aunque para alcanzarla se tenga que utilizar un movimiento prohibido.

La implementación adecuada de este criterio ayuda a que la BT pueda encontrar mejores soluciones.

Existen cuatro tipos de niveles de aspiración:

1. Por tiempo independiente: en el cual se remueve una clasificación tabú desde los procesos de solución, cuando la solución ha mostrado mejor desarrollo que la mejor obtenida hasta el momento.

2. Por objetivo: como su nombre lo indica se relaciona con la función objetivo, en donde un movimiento tabú será permitido si $f(x) <$ *al mejor costo* obtenido hasta ese momento (tratándose de un problema de

minimización, en un problema de maximización f(x) tendría que ser mayor al mejor costo).

3. Por default: si todos los movimientos disponibles son clasificados como tabú y por lo tanto no son admisibles ni siquiera por otro criterio de aspiración, entonces la solución menos tabú es seleccionada.

4. Por dirección de búsqueda: un movimiento tabú se vuelve admisible si la dirección en la que hay que moverse para alcanzarlo genera una mejora.

c. Restricciones.

Estas restricciones se encargan de limitar el espacio de búsqueda, y su meta es lograr traspasar la optimalidad local. También ayuda a evitar repeticiones, es decir el regresar a soluciones visitadas previamente. Los movimientos que permite este elemento son solamente aquellos que nos son tabú. Una solución es aceptable si satisface estas restricciones.

Las restricciones tabú pueden ser de dos tipos:

Basada en recencia: se presenta cuando los atributos ocurren dentro de un número limitado de iteraciones anteriores a la presente iteración.

Basada en frecuencias: se presenta cuando un atributo ocurre con cierta frecuencia sobre un número grande de iteraciones.

Algunos ejemplos de restricciones son:

Un movimiento es tabú si:

1. x_k cambia de 0 a 1, donde x_k había cambiado de 1 a 0 previamente.

2. x_j cambia de 1 a 0, donde x_j había cambiado de 0 a 1 previamente.

d. Lista de candidatos.

Es una sublista de posibles movimientos y son usadas para restringir el número de soluciones examinadas en una iteración para el caso en que $N^*(x)$ es grande o la evaluación de sus elementos es costosa, de esta forma solo se examina al subconjunto de la vecindad en lugar de explorar todo la vecindad.

Dado que estamos tratando un problema de minimización el mejor candidato será el que posea el menor valor algebraico.

Algunas veces este elemento no se usa de una manera explícita, pero el uso de estructuras de memoria que dan las actualizaciones de las evaluaciones de un movimiento a otro son parte de este elemento.

5. Memoria a medio plazo.

No es implementada frecuentemente sus objetivos son contrastar, inspeccionar y guardar las características de las mejores soluciones obtenidas durante un tiempo definido en la búsqueda. Después las características que son similares se consideran un atributo de región, gracias a esto el método trata de guiar la búsqueda hacia soluciones que tengan las mismas características de región.

6. Memoria a largo plazo.

De acuerdo al tipo de aplicaciones, la BT genera soluciones cercanas al óptimo, sólo con implementar la memoria a corto plazo, pero si aparte de efectuar

ésta agregamos la memoria a largo plazo, aumentaremos la probabilidad de obtener soluciones cercanas a la óptima global.

Esta tipo de memoria se utiliza con la finalidad de obtener una solución factible en otras regiones.

La memoria a largo plazo al igual que la memoria a corto plazo contiene elementos que hacen posible su éxito. Los elementos que tiene son tres (basada en frecuencia, estrategia de intensificación y estrategia de diversificación), pero a pesar de esto, la efectividad depende de que se equilibren la estrategia de intensificación y la de diversificación.

a. Basada en frecuencias.

Se encarga de introducir penalizaciones e incentivos determinados por el tiempo durante el que los atributos han pertenecido a soluciones visitadas durante la búsqueda, permitiendo la diferenciación de las diferentes regiones.

Generalmente la frecuencia es representada por razones en donde el numerador representa cuentas del número de ocurrencias de un evento en particular y el denominador representa cantidades como las siguientes:

1. El numero total de ocurrencias de todos los eventos representados por el numerador.
2. La suma de los numeradores.
3. El máximo valor del numerador.
4. El promedio del valor de numerador.

Esta memoria de frecuencia puede ser de dos tipos:

1. De residencia: se encarga de registrar las duraciones relativas de los atributos en las soluciones generadas.

2. De transición: se encarga de mantener el registro de las frecuencias con que cambian los atributos.

En otros casos la frecuencia es usada para generar penalizaciones que modifiquen la función objetivo.

b. Estrategia de intensificación.

Se basa en la modificación de reglas de elección para que se beneficien las combinaciones de movimientos y características de las soluciones que históricamente hayan sido buenas, por otro lado pueden iniciar un regreso a regiones que son atractivas para buscar en ellas de una forma más extensa.

Han surgido tres tipos de intensificación que han mostrado ayudar a las obtenciones de soluciones óptimas.

1. Propuesta por Voss: se introduce una medida de diversificación para asegurar que las soluciones registradas sean diferentes en un grado deseado una de otra y elimina los registros de la memoria a corto plazo antes de iniciar el proceso desde la mejor de las soluciones registradas.

2. Por descomposición: en este tipo de intensificación se pueden imponer restricciones para generar una forma de descomposición que permita un enfoque más concentrado en otras partes de la estructura de solución.

3. Propuesta por Nowicki y Smutniki: aquí se mantiene una lista secuencial de longitud limitada que agregue al final una nueva solución sólo si es mejor que cualquiera otra obtenida.

c. Estrategia de diversificación.

Esta estrategia se diseñó con la finalidad de realizar las búsquedas en regiones no exploradas. La mayoría se basa en modificar las reglas de elección para llevar a la solución atributos que no hayan sido usados con frecuencia.

Esta estrategia puede reiniciar total o parcialmente el proceso de búsqueda, pero la ventaja de hacerlo es que se pueden introducir los atributos que no hayan sido utilizados con mucha frecuencia.

La característica de reiniciar ya sea de una forma total o parcial es de gran ayuda para los problemas y estructuras de entorno, donde una trayectoria de soluciones puede ser sólo aislada de entre alternativas nuevas y valiosas por medio de la introducción de un cambio radical.

7. Estrategia de oscilación.

Esta estrategia tiene que ver con los orígenes de la Búsqueda Tabú y proporciona un puente entre la estrategia de intensificación y diversificación con el objetivo de lograr una interacción ya sea a corto o largo plazo.

Opera encaminando los movimientos hacia un nivel crítico, identificándolo por medio de una etapa de construcción o un intervalo dado de valores para una función.

Este nivel crítico representa un punto donde el método generalmente se detendría, sin embargo, en lugar de detenerse cambia las reglas para elegir los movimientos de tal manera que se pueda traspasar la región definida por el punto crítico.

Después de traspasar hasta un punto que se establezca, regresa y trata de volver a alcanzar el punto crítico y traspasarlo nuevamente, pero en una dirección contraria a la anterior y después se vuelve a dirigir a un punto de retorno nuevamente.

El proceso de aproximarse y traspasar el punto crítico en direcciones diferentes genera un comportamiento oscilatorio, del cual proviene su nombre.

Existen dos pasos en la estrategia de oscilación:

1. Constructivo: en donde se añaden elementos o se les da el valor de uno a las variables.

2. Destructivo: en donde se eliminan elementos o se les da el valor de cero a las variables.

Las estrategias de oscilación pueden ser anidadas, es decir se permite la generación de estas estrategias dentro de otras estrategias de oscilación.

8. Reencadenamiento de trayectorias.

Proporciona una integración muy útil de las estrategias de intensificación y diversificación.

Se basa en explorar trayectorias que conectan soluciones élite para generar nuevas soluciones, empezando desde una de estas soluciones, etiquetada como

solución inicial y generando una trayectoria en el espacio de entornos que conduce hacia otras soluciones llamadas soluciones guía.

Esta estrategia no solamente introduce atributos de alta calidad, sino también subordina todas las consideraciones de elegir movimientos que introduzcan los atributos de las soluciones guía con el fin de crear una nueva composición de atributos en la solución actual.

En cada paso la composición se determina eligiendo el mejor movimiento, mediante los criterios usuales de elección del conjunto restringido de movimientos que incorporan un máximo número de los atributos de las soluciones guía.

Una vez identificada una colección de una o más soluciones élite para guiar la trayectoria a partir de una solución dada, los atributos de estas soluciones guía reciben pesos como incitadores para ser seleccionadas.

Los atributos que se presentan con mayor frecuencia en las soluciones guía reciben mayores pesos.

En una colección establecida de soluciones élite, el papel de las soluciones guía pueden alternarse, es decir pueden generarse simultáneamente un conjunto de soluciones actuales desarrollando trayectorias diferentes y permitiendo que una solución inicial sea reemplazada como una solución guía para otras.

Debido a que sus papeles pueden ser intercambiables las soluciones iniciales y las guías reciben el nombre de soluciones referenciales.

C. Operadores de los AGs.

Algunos sistemas logran adaptarse a su medio gracias al aprendizaje obtenido por medio de modelos de entrenamiento. Sin embargo otros lo hacen mediante la evolución, ocasionando que las generaciones posteriores se adecuen mejor a su entorno que sus progenitores.

En la evolución biológica los descendientes son generados, con algunas modificaciones, a imagen de sus predecesores por medio de la selección natural. Y los descendientes que sobreviven (los mejor adaptados) generarán a su vez otros descendientes.

Como se mencionó en el capítulo anterior los AGs son procedimientos que se basan en la teoría de la evolución[6] y la genética. Para comprender de qué manera se simulan los procesos evolutivos, es necesario analizar su analogía con los AGs.

1. Conceptos biológicos y genéticos.

El Genoma es la parte fundamental de los organismos biológicos ya que contienen a todos los cromosomas y por medio de éstos se heredan las características de los seres vivos. Los cromosomas están formados por una molécula de *Ácido desoxiribonucleico* (ADN), mostrado en la figura 4. El ADN de un organismo puede contener desde un gen hasta miles de genes, como es el caso de la composición del ser humano.

[6] Propuesta por Carlos Darwin.

Figura 4. Estructura del ADN.

El ADN esta compuesto por cuatro compuestos que son *Adenina* (A), *Citosina* (C), *Guanina* (G) y *Timina* (T) y la secuencia de éstos son los que determinan la estructura de cualquier organismo.

Como se mencionó en el párrafo anterior, los genes son una porción del ADN, los cuales generalmente se encargan de producir las proteínas. A su vez los genes contienen valores denominados *alelos*.

Existen células las cuales están constituidas por dos juegos de cromosomas, las cuales se denominan *diploides;* si el alelo de ambos juegos es diferente se etiquetan como *heterocigos* de lo contrario se llaman *homocigos.*

Las células que contienen un solo cromosoma o varios cromosomas que tengan la misma secuencia de genes es designada *haploide.*

Existen unas células exclusivas denotadas *gametos,* las cuales tienen como misión transmitir la información genética de los padres en el momento de la reproducción. Se conoce como *reproducción* a la creación de un nuevo individuo ya sea por reproducción asexual o sexual.

Se conoce como *individuo* a un miembro de la población. La *población* es un conjunto de individuos, con características similares, que interactúan con la finalidad de realizar ciertas actividades como la reproducción.

Un individuo posee dos tipos de características, unas son externas y se denota como *fenotipo*; las otras no son visibles a simple vista y se conocen como *genotipo*. Podemos decir que el fenotipo representa a las características externas de un individuo mientras que el genotipo es la información genética de éste.

Todo organismo está rodeado por un *ambiente* ya sea abiótico o biótico. Sin importar cual sea el tipo de ambiente, los organismos ocupan un nicho ecológico, el cual influye sobre su aptitud dentro de todo el ambiente. Un *nicho ecológico* es una estrategia de supervivencia de las especies por ejemplo la pesca o la caza. Una *especie* es un conjunto de organismos con características similares, que les permiten reproducirse entre sí para engendrar descendientes similares a ellos. La *aptitud* de un individuo es la capacidad de adaptación al ambiente actual, también representa la posibilidad de vivir para poder reproducirse.

Al proceso de elegir a los individuos que se reproducirán se denomina *selección*. La aptitud de un individuo nos ayuda para saber cuales son los mejores candidatos para reproducirse.

Existen dos tipos de selecciones: la blanda y la dura. La primera, utiliza un proceso probabilístico para mantener a individuos con bajas aptitudes como padres. La segunda únicamente mantiene a los individuos con mejores aptitudes para crear las próximas generaciones.

Durante la reproducción se puede presentar un error de copiado del ADN y se le conoce como *mutación*.

Cuando se presenta una transferencia de genes de un individuo de una subpoblación a otra se presenta la migración. Una *subpoblación* es aquella en donde los individuos únicamente pueden reproducirse con los individuos de esa misma subpoblación.

En algunas ocasiones las subpoblaciones permanecen aisladas de la población principal originando en sus individuos cambios genéticos que darán origen a una nueva especie, a este proceso se le conoce como *especiación.*

2. Conceptos biológicos utilizados en los AGs.

Los AGs, al igual que los organismos biológicos realizan las siguientes procesos: *selección, reproducción* y *mutación*, estos procesos necesitan de los mismos elementos como lo hacen los seres vivos.

En los AGs un *cromosoma* es representado por un arreglo de una cadena de genes, generalmente esta cadena es de ceros y unos (figura 5 y 6). Su función es la representación de las variables de decisión.

1	0	0	1	1	0	0	1

Figura 5. Representación de un cromosoma.

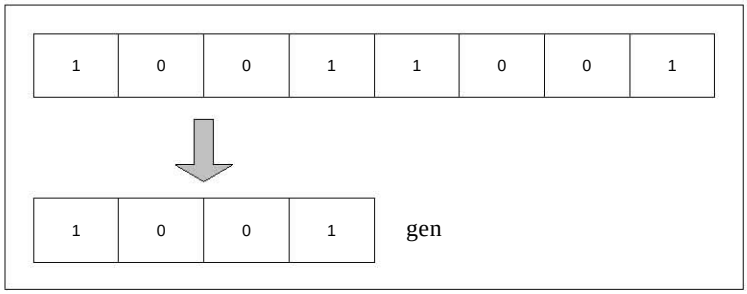


Figura 6. Representación de un gen.

Los *genes* generalmente representan a un solo parámetro y son un subelemento del cromosoma, por ejemplo el gen de arriba representa al parámetro nueve. Estos genes contienen valores denominados alelos, por ejemplo si nuestro cromosoma es una cadena binaria el alelo puede valer 0 o 1 (figura 7).

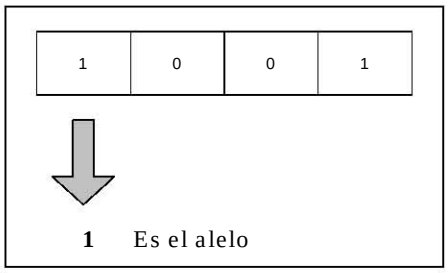


Figura 7.

Un *individuo* en los AGs es un cromosoma que representa a una solución de toda la población de posibles soluciones de un problema, también puede almacenar información relacionada con su aptitud.

La *aptitud* es el valor asignado al individuo para poder visualizar que tan bueno es para resolver el problema en cuestión. Por ejemplo si $f(x) = (x-4)^2$, esta función es la de aptitud, entonces $f(1101_2) = 81$.

Después de aplicar la función de aptitud a cada solución posible obtenemos una hipersuperficie denominada *paisaje de aptitud*.

Los individuos contienen una codificación y decodificación del problema denominadas *genotipo* y *fenotipo* respectivamente (figura 8).

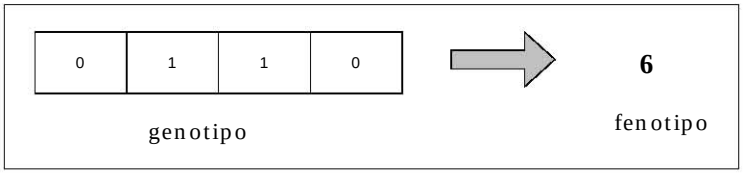


Figura 8. Ejemplo de un genotipo y fenotipo

En la implementación de los AGs se conoce como *generación* a una iteración en donde se crea una nueva población por medio de la reproducción.

Al igual que en un sistema biológico podemos dividir a la población en pequeños grupos denominados *subpoblaciones.* Aquí es posible que los individuos de diferentes subpoblaciones se reproduzcan y se le denomina *población panmítica.* Las subpoblaciones pueden definirse mediante restricciones.

Una característica especial de las subpoblaciones es que permiten la *migración* de genes de un individuo de una subpoblación a otra.

Sin embargo, sí el esquema creado sólo permite que los individuos se reproduzcan con individuos de su misma subpoblación se nombra *especiación.*

Cuando hablamos de *esquema* nos referimos a una guía de los alelos de genes de un cromosoma, en esta idea se encuentran los fundamentos matemáticos de los AGs llamado teorema de los esquemas que se analizará mas adelante. En algunos se puede incluir un estado de *no importa*, el cual se indica mediante el símbolo # o *, por ejemplo si tenemos una cadena 0*11**1, el símbolo * puede tomar el valor de 0 o 1, una instancia de este esquema es 0111011.

Al igual que la técnica BT los AGs contienen las estrategias de *explotación* y *exploración*. La primera de estas es cuando se utiliza la información de los puntos visitados con anterioridad para decidir cuales son los espacios de búsqueda más convenientes a inspeccionar a continuación. La segunda de estas tácticas es el proceso de visitar regiones que no han sido investigadas con anterioridad. La explotación ayuda a encontrar óptimos locales mientras que la exploración evita quedar atrapado en estos.

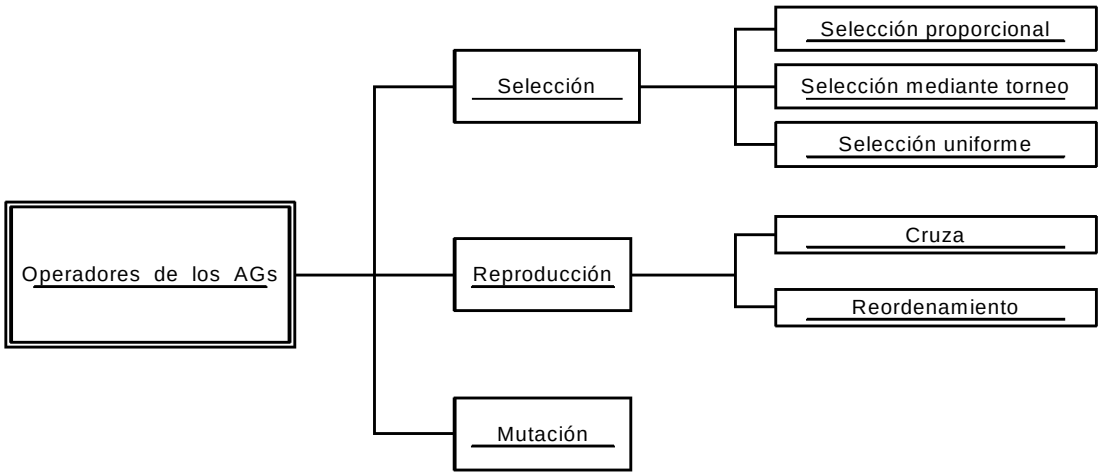
Además de las estrategias anteriores una muy utilizada por los AGs es el *elitismo*, la cual se utiliza para garantizar que los cromosomas más aptos se pasen a la siguiente generación sin ninguna modificación. De igual manera garantiza que la aptitud nunca disminuya de una generación a otra, un inconveniente del elitismo es que nos lleva a una convergencia más rápida.

En algunos casos después de que los genes han evolucionado y se introducen en cualquier cromosoma se obtiene una probabilidad muy alta de incrementar la aptitud de dicho cromosoma; a este grupo de genes se le conoce como bloque constructor. No obstante Goldberg estudió este proceso y dedujo que cuando se combinan buenos bloques constructores se reduce la aptitud en vez de incrementarla y a esta manifestación se le conoce como decepción.

En algunos casos la función objetivo tiene varios óptimos con el mismo valor, en estos casos se considera a cada óptimo como un *nicho ecológico* y se crea una subpoblación a en cada óptimo. Con esto se previene que el AG converja únicamente en un óptimo.

3. Diagramas de los operadores de los Algoritmos Genéticos

En los siguientes diagramas se muestran los operadores utilizados por los AGs y las propuestas de varios investigadores para implementar cada uno de ellos.



Aparte de los tres operadores mencionados existen otras características que son de gran importancia para los AGs como: la representación, el valor de los parámetros, las restricciones, y la teoría de los esquemas, la cual es la base teórica.

Para implementar con éxito un AG es de suma importancia tener en cuenta los siguientes componentes:

1. La elección adecuada de la codificación del dominio en términos de los cromosomas.

2. La manera en que se crea la población inicial.

3. La creación de la función de aptitud.

4. La incorporación de los operadores genéticos, que hacen posible la creación de nuevos individuos.

5. La elección de los parámetros utilizados por el AG.

4. Algoritmo de los AGs.

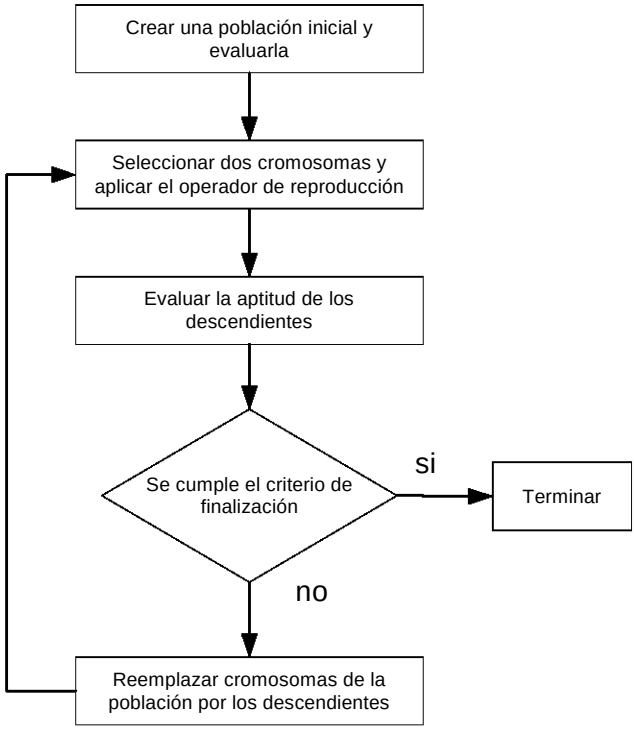
El proceso de un AG puede representarse de la siguiente manera:

```
PROCEDURE (AG)
begin
    t=0
    inicializar P(t)
    evaluar P(t)
    while (la condición de finalización no se cumpla) do
    begin
        t=t+1
        seleccionar P(t)  desde  P(t-1)
        alterar  P(t) //aplicar los operadores genéticos para producir una
        nueva  población.
        Evaluar P(t)
    end
end
```

El algoritmo anterior fue tomado del libro *How to solve it: modern heuristics*[7]. En el algoritmo P(t) representa a la población P creada en la generación t.

[7] Michalewicz Z. y Fogel D. *How to solve it: modern heuristics, pp. 151.*

En forma de diagrama lo que un AG hace es lo siguiente:



5. Selección.

Al igual que en la naturaleza el AG debe de seleccionar a los individuos que se reproducirán. Cuando se lleva a cabo la selección por primera vez es importante que exista una población, en los AGs existen dos alternativas para generar la población inicial. La primera de estas es crearla aleatoriamente y la otra es obteniendo soluciones aproximadas con otro método, heurística ó metaheurística. La manera más usual de concebir la población inicial es por medio de la primera de las opciones mencionadas.

El objetivo de la selección es elegir a los individuos mejor adaptados para que den origen a una descendencia con buenos valores de la función objetivo.

Existen diferentes técnicas para realizar esta selección, las más importantes son: selección proporcional, selección mediante torneo y selección de estado uniforme.

a. Selección proporcional.

En ésta se eligen a los individuos de acuerdo a su aptitud con respecto a la población. Ejemplos de ésta es la siguiente:

Ruleta: para implementar este tipo de selección se calcula el promedio de las aptitudes de todos los individuos y se denota como $\overline{f}$. El siguientes paso es calcular la suma del valor esperado de cada individuo denotada por T, el valor esperado de cada individuo está dado por la aptitud entre el promedio de las aptitudes. Después se genera un número aleatorio r entre 0.0 y T, enseguida se suman los valores esperados hasta que esta suma sea mayor o igual a r. Y se selecciona a este individuo, pues fue el que hizo que la suma fuera igual o mayor a r.

b. Selección mediante torneo.

La idea básica de esta selección es comparar las aptitudes de *k* individuos (generalmente *k*=2) y escoger al más apto. Es necesario que los individuos se revuelvan *k* veces para obtener *n* padres donde *n* es el tamaño de la población. Por ejemplo si tenemos una población de cuatro individuos y *k*=2 tendremos que revolver dos veces a los individuos de la población y comparar de dos en dos para obtener de la primera vez que se revuelvan dos padres y de la segunda otros dos.

c. Selección uniforme.

Esta técnica es usada en los AGs no generacionales (las poblaciones que se van generando en cada generación son muy similares a las anteriores). La selección uniforme permite el reemplazo de pocos individuos en cada generación, comúnmente estos individuos son los menos aptos. Los individuos son reemplazados por los hijos obtenidos de la reproducción de los individuos más aptos. Este tipo de selección fue propuesta por el Dr. Whitley en 1989.

6. Reproducción.

Para que un individuo pueda heredar sus características a otros individuos es necesario que exista la reproducción. En los AGs existen dos operadores de reproducción: cruza y reordenamiento.

a. Cruza.

En los AGs después de haber seleccionado a los mejores individuos estos pasan a la siguiente generación o se cruzan para que sus genes se mezclen y se generen dos cromosomas hijos. Existen diferentes tipos de cruzamiento los tradicionales son: cruza de un punto, cruza de dos puntos y cruza uniforme[8]. Desafortunadamente cuando trabajamos con problemas de optimización combinatoria estos operadores no son útiles, ya que se generarían hijos no válidos, por esta circunstancia se han creado cruzamientos especiales como los siguientes:

Order Crossover (OX): supongamos que tenemos los siguientes padres.

P1=1-5-6-8-4-2-3-7 y P2=8-5-6-2-3-1-7-4

[8] Las definiciones están ubicadas en el glosario de este trabajo.

Seleccionamos al azar una subcadena de P1 8-4-2 y generamos un hijo colocando esa subcadena en la misma posición y dejando en blanco las otras posiciones, la cadena sería la siguiente:

H1=x-x-x-8-4-2-x-x

Después borramos en P2 la subcadena tomada de P1

P2'=x-5-6-x-3-1-7-x

y sustituimos los valores de izquierda a derecha de P2' en H1, obteniendo:

H1=5-6-3-8-4-2-1-7

Para obtener el otro hijo se hace lo mismo solo que ahora la subcadena será de P2.

Partially Mapped Crossover (PMX): generamos al azar dos puntos de cruza de los siguientes padres:

P1=1-5-4-7|8-3|2-6 y P2=5-8-6-2|4-7|3-1

intercambiamos los dos segmentos generados en los hijos produciendo los siguientes hijos:

H1=x-x-x-x|4-7|x-x y H2=x-x-x-x|8-3|x-x

los otros valores los obtenemos haciendo un mapeo entre los padres. Primero copiamos los valores de P1 que no están en H1 y luego copiamos los valores de P2 que no están en H2.

H1=1-5-x-x|4-7|2-6 y H2=5-x-6-2|8-3|x-1

y efectuamos el mapeo entre ambos padres para encontrar los valores que faltan.

H1=1-5-8-3-4-7-2-6 y H2=5-4-6-2-8-3-7-1

b. Reordenamiento.

Este operador cambia el orden de los genes con el propósito de reunir a los genes que se hallen interrelacionados para permitir la obtención de bloques constructores. Una manera de intercambiar los genes es por medio de la *inversión* la cual elige dos puntos al azar en el cromosoma y genera un corte e invierte la sucesión de los genes que se encuentran entre estos puntos.

7. Mutación.

La mutación es un operador genético que no se presenta frecuentemente, la forma en determinar si se efectúa una mutación o no es manejado por medio de una probabilidad, algunos investigadores sugieren que sea de 0.001 ó 0.01, estas probabilidades son usadas cuando el cromosoma es binario. Otros investigadores han propuesto que la probabilidad este dada por $p = 1/L$ donde L es la longitud de la cadena del cromosoma.

Al igual que en la cruza existen diversas formas de aplicar la mutación, en nuestro caso donde las soluciones están dadas por permutaciones, las más usuales son las siguientes:

Mutación por inversión: seleccionamos dos posiciones al azar e intercambiamos sus valores, supongamos que elegimos la posición cuatro y seis de la siguiente cadena:

H=5-4-2-6-3-1-7

por lo tanto las posiciones seleccionadas son las que contienen los valores seis y uno, después de intercambiarlos la cadena obtenida es la siguiente:

H'=5-4-2-1-3-6-7

Mutación por inserción: se elige dos posiciones al azar, una que nos indicará el valor que tenemos que quitar y la otra que nos dirá donde debemos insertar ese valor. Por ejemplo seleccionamos las posiciones tres y siete del siguiente cromosoma:

H=6-4-3-7-1-5-2

entonces el valor que tenemos que mover es tres y lo insertaremos en la posición siete, por lo tanto se obtiene:

H'=6-4-7-1-5-3-2.

8. Codificación del dominio.

Al mencionar el término de codificación del dominio nos referimos a la representación de la población del AG.

Como se mencionó en párrafos anteriores uno de los componentes básicos del AG es la codificación del dominio o representación. Y esto influye en la obtención de soluciones óptimas.

En los AGs tradicionales propuestos por Holland[9] los cromosomas eran representados por cadenas binarias (ceros y unos), pero debido a que muchos

[9] Holland J. *Adaptation in Natural and Artificial Systems*.

problemas reales no pueden codificarse sin que se obtengan soluciones no válidas, se han creado otro tipo de representaciones.

Sin embargo la representación binaria ha demostrado favorecer a la producción de soluciones óptimas siempre y cuando el problema en cuestión se pueda codificar mediante ceros y unos. Esto se produce debido a que la codificación binaria produce un número mayor de esquemas y esto a su vez conlleva a la fabricación de bloques constructores.

En seguida se muestran algunas de las representaciones que han sido agregadas a los AG.

Codificación Gray: esta codificación fue creada debido a que en la representación binaria dos números consecutivos difieren en dos o más bits. Por ejemplo $7_2 = 0111 \quad y \quad 8_2 = 1000$ son diferentes en sus cuatro bits. La característica de la codificación gray es que dos números consecutivos únicamente difieren en un bit. Para convertir un número binario a gray sólo aplicamos XOR a los bits consecutivos de derecha a izquierda y conservamos el último bit de la izquierda.

Aunque la codificación gray mejoró el funcionamiento de los AG con codificación binaria, se presenta un problema cuando se desea representar a un número muy grande o un número decimal, pues sería necesario la utilización de una cadena extremadamente larga, la cual nos conduciría a un trabajo computacional muy grande debido a la decodificación, también la búsqueda sería más compleja.

Por este motivo se introdujo el concepto de la codificación real y entera. A pesar de que la idea original de los AGs sólo permite cromosomas binarios existen

otras técnicas de la computación evolutiva como las estrategias evolutivas y la programación genética que usan representaciones diferentes a las binarias. Estas codificaciones han mostrado ser útiles en la simbolización de problemas reales y por tal motivo se han creado operadores de cruza y mutación apropiados a este tipo de representaciones.

Codificación real: En ésta representación cada alelo tiene el valor de un número real, por ejemplo:

P=1.24-32.5-5.01-10.75-15.89-25.03

podemos observar que el cromosoma contiene seis alelos con un valor real cada uno.

Codificación entera: Aquí cada alelo contiene un valor entero, en algunas ocasiones, dependiendo del tipo de problema, el cromosoma representa a un número real por ejemplo:

P=1-6-8-5-4

representa al número 16.854; es necesario mantener fija la posición del punto decimal para poder aplicar los operadores genéticos. La mutación constará de cambiar un número por otro aleatorio entre cero y nueve o de disminuir o restar su valor, por ejemplo sumarle o restarle un valor entre dos y cinco.

9. Representación del dominio.

Cuando se habla de representación del dominio nos referimos a cómo vamos a interpretar los resultados obtenidos, se ha mencionado que en los AGs las soluciones están representadas por cromosomas (vectores) en donde cada uno de

estos es una posible solución. El problema es: si damos un tipo de representación como sabremos qué nos indican los alelos de este cromosoma.

Para el problema del agente viajero existen tres representaciones comunes las cuales se explicarán a continuación.

Representación ordinal: En esta se utilizan dos vectores de tamaño *n* (número de ciudades); A los cuales denominaremos *L* y *V*. El vector *L* contendrá los nombres de las ciudades y el *V* la representación ordinal de una ruta.

Para ejemplificar esta representación consideremos L=1-2-3-4-5-6-7-8 y V=2-1-1-2-4-1-2-1. El vector *V* nos indica la siguiente ruta S=2-1-3-5-8-4-7-6. La pregunta es ¿Cómo sabemos que *V* nos indica la ruta mencionada?, la respuesta es la siguiente: el primer valor de *V* es 2 lo cual nos indica la posición de la ciudad en *L* de ahí que el primer elemento de la ruta sea 2, como esa ciudad ya fue incluida la eliminamos de *L* y recorremos las ciudades restantes.

Este es un proceso iterativo, por lo cual leemos el siguiente elemento de *V* que es 1, la ciudad que ocupa la posición 1 en *L* es 1, el paso a seguir es eliminarlo y recorrer las ciudades restantes; ahora nuestra ruta es 2-1.

Para obtener toda nuestra ruta tendremos que leer los siguientes elementos de *V* y actualizar nuestro vector *L* con las eliminaciones de las ciudades correspondientes y recorriendo las restantes.

Representación Adyacente: Esta representación utiliza un vector que se nombró *V* de tamaño *n* (número de ciudades), este vector nos proporciona la representación adyacente de la ruta. Se establece V=5-6-7-2-4-3-1 que simboliza a la ruta S=2-6-3-7-1-5-4. Para utilizar esta representación es necesario que sepamos en

que ciudad se tiene que empezar el recorrido, en este caso es 2 de esta manera la primera ciudad de la ruta es 2, la siguiente ciudad es la que se encuentra en la posición 2 de *V*, la cual es la 6, la próxima es la que está en la posición 6 de *V*, es decir la número 3.

Para completar las otras ciudades seguimos leyendo las que ocupan la posición de la última ciudad colocada en la ruta.

Representación Path: Esta representación al igual que la anterior se auxilia de un vector *V* de tamaño *n* (número de ciudades). Este tipo de representación es la más natural, debido a que los elementos de V nos muestran directamente cual es la ruta por ejemplo:

Sea V=2-5-8-4-7-6-1-3 el cual representa a la ruta S=2-5-8-4-7-6-1-3.

En las representaciones anteriores no es necesario indicar que la ruta termina con la ciudad que empezó pues sabemos que el problema del agente viajero tiene que regresar a esta ciudad. Para asegurar esto nos auxiliamos del manejo de las restricciones, el cual se explicará más adelante.

10. Valor de los parámetros.

Existen diferentes investigaciones dedicadas a tratar de encontrar que valores son los que se les deben asignar a los parámetros de los AGs para que obtengamos soluciones óptimas, desafortunadamente no se ha encontrado una forma estándar de seleccionar los valores de los parámetros del AG.

Los principales parámetros que deseamos conocer son tres:

- El tamaño de la población
- La probabilidad de cruza
- La probabilidad de mutación.

El investigador Grefenstette sugiere los siguientes valores: una población de cincuenta individuos, utilizar cruza de un punto con un probabilidad de 0.95, una probabilidad de mutación de 0.01 y utilizar la estrategia de elitismo.

Grefenstette adquirió estos valores usando un AG cuya función era optimizar los parámetros de otro AG. Dentro de este estudio observó que si la mutación es mayor a 0.05 o no se presenta, tiende a disminuir la aptitud de los individuos. Si nuestra población contiene de veinte a cuarenta individuos, la aptitud de esto se mejorará si la probabilidad de cruza es alta y la de mutación baja o viceversa.

Goldberg realizó un estudio basado en el valor esperado de número de esquemas para obtener el tamaño de la población cuando los cromosomas son binarios. La función que obtuvo fue la siguiente:

$$tamaño\_población = 1.65^{2^{(0.21L)}}$$

donde L es la longitud de la cadena, de esta forma si tenemos una cadena con una longitud de cuarenta el tamaño de la población será de 557 individuos.

Goldberg concluyó que entre más grande sea el valor de la población mayor será el desempeño del AG.

De Jong hizo un estudio en donde creó dos grupos para analizar el comportamiento de los parámetros. Uno de ellos es llamado *desempeño de línea* y el otro *desempeño fuera de línea.* El primero de éstos concentra la búsqueda en las regiones más promisorias y el segundo permite la exploración en regiones no prometedoras del espacio de búsqueda siempre y cuando se alcance una mejor aptitud.

Finalmente la conclusión es que, los parámetros óptimos para el funcionamiento de un AG son difíciles de determinar, por esta circunstancia recientes estudios afirman que es mejor la auto-adaptación.

La auto-adaptación sugerida por Davis nos obliga a asignar una aptitud a cada operador genético. La aptitud esta basada en el número de individuos con buenas aptitudes que han creado. Cada vez que un operador crea a un individuo con una aptitud alta recibe un estímulo. La desventaja de implantar la auto-adaptación es el costo computacional que implica.

11. Manipulación de las restricciones.

Una dificultad que se presenta en los AG son las restricciones ya que normalmente los problemas que se tienen que resolver son de la siguiente estructura:

$$\begin{aligned} &\textit{Minimizar} \quad f(X) \\ &\textit{sujeto} \quad a \\ &g_i(x) \le 0 \quad i = 1, 2, \cdots, n \end{aligned}$$

Los AGs resuelven problemas de optimización sin restricciones, esto nos exige plantear un elemento que nos ayude a introducir la información sobre las

restricciones en la función de aptitud. Existen diferentes formas que hacen posible la inclusión. Las más usadas son las siguientes:

Funciones de penalización: Estas funciones extienden el dominio de la función objetivo con el siguiente procedimiento:

$$Aptitud_i(x) = f_i(x) \pm Q_i$$
$$donde \quad Q_i = c(\sum_{i=1}^{n} g_i(x)^2)$$

c representa un elemento de penalización dado por el usuario.

Para penalizar a un individuo, se puede hacer si no es factible o mediante la suma de infactibilidad que tenga.

Dentro de las funciones de penalización encontramos diferentes técnicas para su asignación, por ejemplo:

Pena de muerte: su procedimiento es asignar una aptitud cero a un individuo si este no es factible. El inconveniente es que si la población inicial no tuviera ningún individuo factible, la búsqueda se detendría.

12. Teorema de los esquemas.

Como se señaló en los apartados anteriores la base matemática de los AGs se encuentra en el teorema de los esquemas. Este modelo matemático fue propuesto por Holland en el cual usa la selección por ruleta y la cruza de un punto. Para explicar en que consiste es necesario mencionar que todo esquema tiene un orden denotado por O(H) y una longitud de definición representada por δ(H).

O(H) está dada por el número de posiciones que no sean "no importa", mientras que δ(H) es obtenida por las distancias mas alejadas de los valores que no son "no importa". Tomando como ejemplo la cadena siguiente:

*10***11

obtenemos que O(H)=4, debido a que únicamente cuatro posiciones son "no importa" y δ(H)=6 ya que la primera posición que no es "no importa" es la segunda y la última es la octava por lo tanto 8-2=6.

Podemos deducir que el espacio de búsqueda está representado por un esquema de orden cero, sin importar la longitud.

El número de esquemas que se genera en una cadena está dado por $(c+1)^L$, donde c es la cardinalidad del alfabeto usado y L es la longitud de la cadena. Por ejemplo el número de esquemas que tienen la siguiente cadena P=1-0-1-1-0-1-1 es $(2+1)^7 = 2187$.

Otra característica de los esquemas es que representan a c^r cadenas, donde r es el número de símbolos no importa en la cadena. De igual manera cualquier cadena de longitud L es obtenida de 2^L esquemas diferentes.

De las propiedades anteriores concluimos que si en una población todas las cadenas son iguales, entonces existirán 2^L esquemas distintos, pero si todas las cadenas son diferentes, entonces habrá $n2^L$, donde n es el tamaño de la población (es decir el número de cadenas existentes).

Es por el motivo anterior que un AG al evaluar la aptitud de las cadenas pertenecientes a la población, también estima, de manera implícita, las aptitudes de un número mayor de esquemas.

La aptitud promedio de un esquema se denota $\overline{f}(H)$ y se define como la aptitud promedio de todos los representantes de ese esquema. A las funciones de aptitud de los representantes del esquema denominémoslos $f_1, f_2,\ldots, f_k$. Llamemos $m(H,t)$ al número de representantes del esquema H en la generación t. Entonces:

$$\overline{f}(H) = \frac{f_1 + f_2 +,\ldots,+ f_k}{m(H,t)}$$

Sea $\overline{f}$ la aptitud promedio de la población en la generación t; para obtenerla basta con determinar la aptitud promedio de todos los individuos de la población. Especificando $f_1, f_2,\ldots, f_n$ a las aptitudes de los individuos de la población obtenemos que:

$$\overline{f} = \frac{f_1 + f_2 +,\ldots,+ f_n}{n}$$

Con los conceptos anteriores podemos calcular el valor esperado de individuos en la generación $t+1$ de la siguiente forma:

$$m(H,t+1) = n\, m(H,t)\frac{\overline{f}(H)}{f_1 + f_2 +,\ldots,+ f_n} = m(H,t)\frac{\overline{f}(H)}{\overline{f}}$$

La expresión anterior considera el proceso de selección sin analizar los efectos del operador de cruza ni el de mutación; sin embargo es un hecho que al aplicar estos operadores algunas de las cadenas generadas no pertenecerán al

esquema progenitor, mientras que otras que no pertenecían al esquema con la aplicación de los operadores serán parte de dicho esquema y por último se puede dar el caso en que otras cadenas pasen sin ninguna modificación a la siguiente generación.

Para analizar el efecto del operador de cruza nombremos a P_c la probabilidad de cruza, después elegimos a un representante del esquema *H* para ser padre. Si los hijos generados son representantes del esquema *H*, entonces el esquema sobrevive.

Definamos a $S_c(H)$ como la probabilidad de supervivencia del esquema *H* al efectuarse el operador de cruza, entonces:

$$S_c(H) \geq 1 - P_c\left(\frac{\delta(H)}{L-1}\right)$$

Donde $\left(\frac{\delta(H)}{L-1}\right)$ nos indica la probabilidad de que los hijos no pertenezcan al esquema.

El efecto del operador de mutación es intercambiar los valores de los alelos fijos. Consideremos P_m la probabilidad de mutación, para que un esquema sobreviva ningún alelo debe cambiar de valor (recordemos que O(H) representa al número de alelos que pueden intercambiarse). La probabilidad de que ningún alelo cambie de valor está dada por $1-P_m$, Ahora supongamos que al cambiar el valor de un alelo se genera un evento independiente, obtenemos que la probabilidad de supervivencia al efectuarse el operador de mutación es:

$$S_m(H) = (1 - P_m)^{O(H)}$$

Si adoptamos los efectos de selección, cruza y mutación en una sola expresión obtenemos:

$$m(H, t + \textit{selección} + \textit{cruza} + \textit{mutación}) \geq m(H,t)\frac{\overline{f}(H)}{\overline{f}}\left[1 - P_c\left(\frac{\delta(H)}{L-1}\right)\right](1-P_m)^{O(H)}$$

Esta expresión es conocida como el *Teorema de los Esquemas,* y se encarga de describir el crecimiento de un esquema de una generación a otra.

Sin embargo una de las limitaciones de éste teorema es que no siempre es exacto, sólo ha sido planteado para cadenas binarias y no puede predecir a largo plazo como se comporta el AG. Por estos motivos la base matemática de los AGs sigue siendo un campo de investigación, a pesar de que en la práctica han demostrado ser eficientes para encontrar soluciones óptimas o cercanas a ésta.

Son los problemas sin resolver, no los resueltos, los que mantienen activa la mente.

ERWIN GUIDO KOLBENHEYER

III. FORMULACIÓN DEL MODELO.

A. Descripción del problema.

La empresa X tiene que distribuir su producto a cada una de las sucursales que tiene localizadas en el interior del país. La matriz de esta empresa, es decir la sucursal que se encarga de la producción del producto, se encuentra localizada en la ciudad de México.

La empresa cuenta con dos zonas de distribución la zona norte y la zona sur. La zona para la que se encontrará la ruta mínima es la de la zona sur.

Para minimizar los costos de transporte es necesario que se optimice el tiempo de recorrido lo cual implica encontrar la ruta óptima. Para poder encontrar esta ruta es necesario tomar en cuenta las siguientes restricciones:

- Se debe de empezar el recorrido en la ciudad de México.
- Se tiene que surtir cada sucursal solamente una vez, esto implica que únicamente se pasará por cada sucursal una vez.
- Se tiene que regresar a la ciudad de México.
- Sólo se puede viajar por las carreteras establecidas.
- La distancia recorrida es simétrica, por ejemplo si la distancia de la Ciudad de México a Tlaxcala es de 118 kilómetros, entonces la distancia de Tlaxcala a la Ciudad de México es la misma.

Sin incluir a la matriz, la empresa cuenta con quince sucursales ubicadas en los siguientes lugares: Izucar de Matamoros y San Martín Texmelucan en Puebla y también su capital Puebla; Tlaxcala; Matías Romero, Huajuapan y Salina Cruz en Oaxaca y su capital Oaxaca; Coatzacoalcos, Tuxpan, Poza Rica, Acayucan, La Tinaja, Naranjos y Santiago Tuxtla en Veracruz y la capital del mismo estado, Jalapa.

Para facilitar el manejo de las sucursales se numeraron de la siguiente manera:

Número de ciudad	***Nombre de la ciudad***
1	Ciudad de México
2	San Martín Texmelucan
3	Izucar de Matamoros
4	Puebla
5	Tlaxcala

Número de ciudad	*Nombre de la ciudad*
6	Matías Romero
7	Huajuapan
8	Salina Cruz
9	Oaxaca
10	Coatzacoalcos
11	Poza Rica
12	Acayucan
13	Santiago Tuxtla
14	Jalapa
15	La Tinaja
16	Tulancingo

Para representar las distancias entre cada una de las sucursales, elaboramos la matriz de costo, en donde los números representan a cada una de las sucursales tomando como referencia la lista anterior. La distancia está dada en kilómetros. Cuando no se puede ir de una sucursal a otra directamente el valor es cero, al igual que si la sucursal destino es la misma que la de inicio.

Para saber cual es la distancia de una sucursal a otra interceptamos el renglón y la columna de dichas ciudades. Por ejemplo si queremos saber la distancia de San Martín Texmelucan a Puebla, interceptamos el renglón 2 (Texmelucan) con la columna 4 (Puebla) y observamos que la distancia es de 28 kilómetros.

Matriz de costo

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1	**0**	95	165	123	118	1126	682	1069	470	1486	210	1273	596	402	356	120
2	95	**0**	166	28	23	496	287	855	595	915	498	912	715	1116	420	510
3	165	166	**0**	67	97	1223	154	969	786	1442	566	898	960	728	597	847
4	123	28	67	**0**	33	533	588	734	340	816	481	693	579	318	251	730
5	118	23	97	33	**0**	803	739	1020	791	961	258	815	680	319	466	210
6	1126	496	1223	533	803	**0**	1098	241	910	536	1374	147	695	879	292	1455
7	682	287	154	588	739	1098	**0**	720	194	813	1513	759	1012	1321	244	1413
8	1069	855	969	734	1020	241	720	**0**	267	797	1640	639	980	1380	699	763
9	470	595	786	340	791	910	194	267	**0**	679	1562	386	810	961	293	1031
10	1486	915	1442	816	961	536	813	797	679	**0**	884	71	183	641	477	1080
11	210	498	566	481	258	1374	1513	1640	1562	884	**0**	879	645	304	851	150
12	1273	912	898	693	815	147	759	639	386	71	879	**0**	140	263	200	1704
13	596	715	960	579	680	695	1012	980	810	183	645	140	**0**	123	503	699
14	402	1116	728	318	319	879	1321	1380	961	641	304	263	123	**0**	63	818
15	356	420	597	251	466	292	244	699	293	477	851	200	503	63	**0**	456
16	120	510	847	730	210	1455	1413	763	1031	1080	150	1704	699	818	456	**0**

La red que representa a las carreteras que interconectan las sucursales se muestra en la figura 1, en donde el costo de los arcos representa la distancia en kilómetros entre una sucursal y otra.

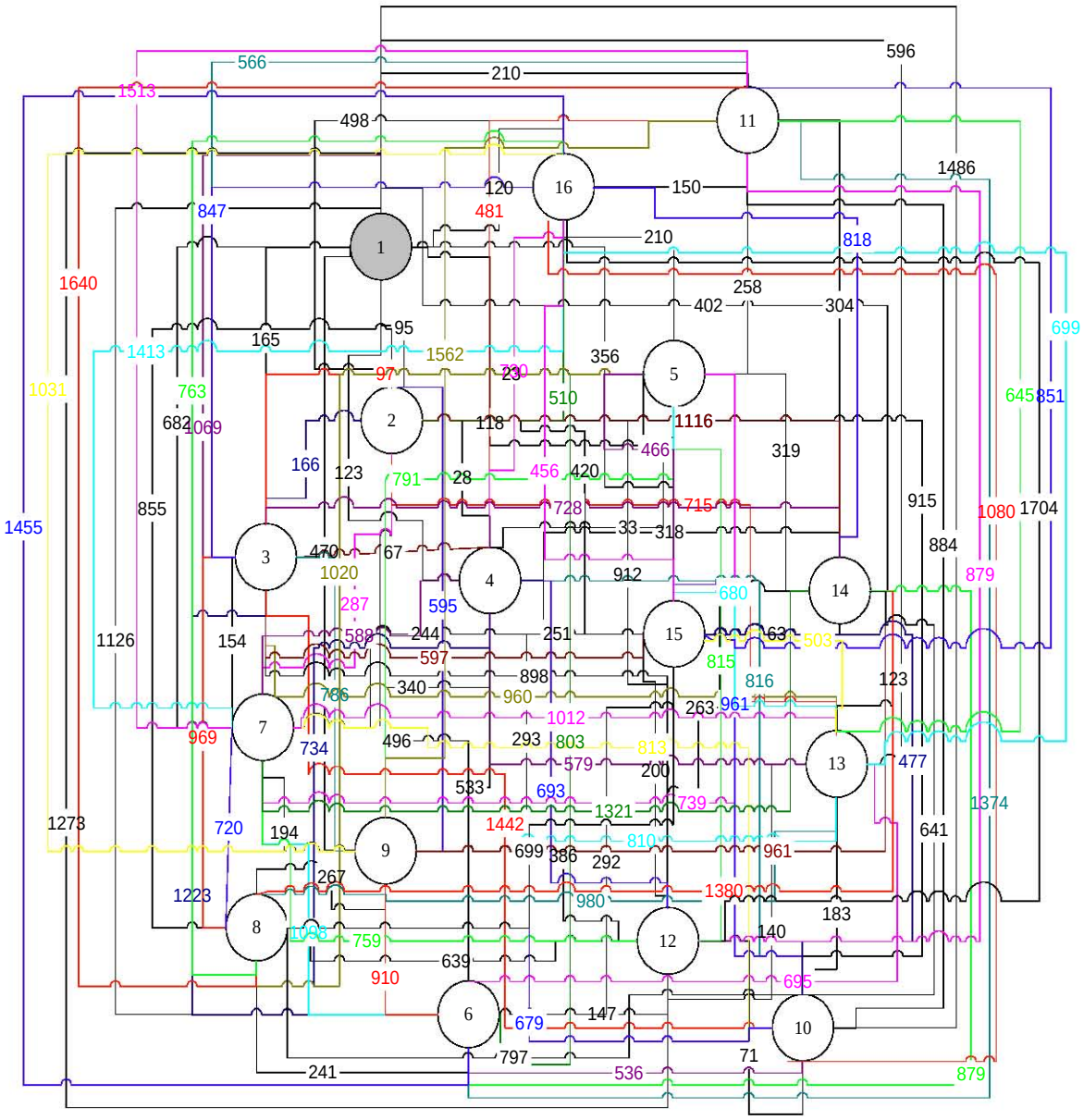

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
596
566
210
1513
498
1486
120
150
847
481
210
818
1640
258
402
304
699
95
165
1413
1562
356
1031
763
97
23
645
851
682
1069
510
1116
118
319
166
123
791
28
456
420
466
855
728
715
915
1080
1704
1455
33
318
884
470
67
912
879
1020
680
287
595
1126
154
588
244
251
503
597
815
898
816
123
340
960
263
961
969
1012
734
496
293
803
813
579
200
477
533
693
1321
739
1374
1278
720
194
1442
641
699
386
292
961
267
1380
1223
980
183
1058
759
140
639
910
695
679
147
797
71
241
536
879

Figura 1

La formulación de este problema conociendo que n = 16 es:

$$Min\ z = \sum_{i=1}^{16}\sum_{j=1}^{16} c_{ij}x_{ij}$$

sujeto a

$$\sum_{i=1}^{16} x_{ij} = 1, \quad j = 1,2,..,16.$$

$$\sum_{j=1}^{16} x_{ij} = 1, \quad i = 1,2,..,16.$$

$$x_{ij} = 0\ \acute{o}\ 1$$

En donde:

c_{ij} es la distancia entre la ciudad i y j.

x_{ij}=1 significa que el agente viajero va desde la ciudad i a la ciudad j.

x_{ij}=0 significa que el agente viajero no va de la ciudad i a la ciudad j.

Desarrollando las sumatorias anteriores y sustituyendo los valores de las distancias mostradas en la matriz de costo del problema descrito anteriormente, obtenemos la siguiente función:

$$
\begin{aligned}
Min\ z = {} & 95x_{1,2} + 165x_{1,3} + 123x_{1,4} + 118x_{1,5} + 1126x_{1,6} + 682x_{1,7} + 1069x_{1,8} + 470x_{1,9} + \\
& 1486x_{1,10} + 210x_{1,11} + 1273x_{1,12} + 596x_{1,13} + 402x_{1,14} + 356x_{1,15} + 120x_{1,16} + \\
& 95x_{2,1} + 166x_{2,3} + 28x_{2,4} + 23x_{2,5} + 496x_{2,6} + 287x_{2,7} + 855x_{2,8} + 595x_{2,9} + 915x_{2,10} + \\
& 498x_{2,11} + 912x_{2,12} + 715x_{2,13} + 1116x_{2,14} + 420x_{2,15} + 510x_{2,16} + 165x_{3,1} + 166x_{3,2} + 67x_{3,4} + \\
& 97x_{3,5} + 1223x_{3,6} + 154x_{3,7} + 969x_{3,8} + 786x_{3,9} + 1442x_{3,10} + 566x_{3,11} + 898x_{3,12} + \\
& 960x_{3,13} + 728x_{3,14} + 597x_{3,15} + 847x_{3,16} + 123x_{4,1} + 28x_{4,2} + 67x_{4,3} + \\
& 33x_{4,5} + 533x_{4,6} + 588x_{4,7} + 734x_{4,8} + 340x_{4,9} + 816x_{4,10} + 481x_{4,11} + 693x_{4,12} + \\
& 579x_{4,13} + 318x_{4,14} + 251x_{4,15} + 730x_{4,16} + 118x_{5,1} + 23x_{5,2} + 33x_{5,4} + \\
& 803x_{5,6} + 739x_{5,7} + 1020x_{5,8} + 791x_{5,9} + 961x_{5,10} + \\
& 258x_{5,11} + 815x_{5,12} + 680x_{5,13} + 319x_{5,14} + 466x_{5,15} + 210x_{5,16} + 1126x_{6,1} + \\
& 496x_{6,2} + 1223x_{6,3} + 533x_{6,4} + 803x_{6,5} + 1098x_{6,7} + 241x_{6,8} + \\
& 910x_{6,9} + 536x_{6,10} + 1374x_{6,11} + 147x_{6,12} + 695x_{6,13} + 879x_{6,14} + \\
& 292x_{6,15} + 1455x_{6,16} + 682x_{7,1} + 287x_{7,2} + 154x_{7,3} + 588x_{7,4} + 732x_{7,5} + 1098x_{7,6} + \\
& 720x_{7,8} + 194x_{7,9} + 813x_{7,10} + 1513x_{7,11} + 759x_{7,12} + 1012x_{7,13} + 1321x_{7,14} + \\
& 244x_{7,15} + 1413x_{7,16} + 1069x_{8,1} + 855x_{8,2} + 969x_{8,3} + 734x_{8,4} + \\
& 1020x_{8,5} + 241x_{8,6} + 720x_{8,7} + 267x_{8,9} + 797x_{8,10} + 1640x_{8,11} + \\
& 639x_{8,12} + 980x_{8,13} + 1280x_{8,14} + 699x_{8,15} + 763x_{8,16} + \\
& 470x_{9,1} + 595x_{9,2} + 786x_{9,3} + 340x_{9,4} + 791x_{9,5} + \\
& 910x_{9,6} + 194x_{9,7} + 267x_{9,8} + 679x_{9,10} + 1562x_{9,11} + 386x_{9,12} + \\
& 810x_{9,13} + 961x_{9,14} + 293x_{9,15} + 1031x_{9,16} + 1486x_{10,1} + 915x_{10,2} + \\
& 1442x_{10,3} + 816x_{10,4} + 961x_{10,5} + 536x_{10,6} + 813x_{10,7} + 797x_{10,8} + \\
& 679x_{10,9} + 884x_{10,11} + 71x_{10,12} + 183x_{10,13} + 641x_{10,14} + 477x_{10,15} + 1080x_{10,16} + \\
& 210x_{11,1} + 498x_{11,2} + 566x_{11,3} + 481x_{11,4} + 258x_{11,5} + 1374x_{11,6} + 1513x_{11,7} + \\
& 1640x_{11,8} + 1562x_{11,9} + 884x_{11,10} + 879x_{11,12} + 645x_{11,13} + 304x_{11,14} + 851x_{11,15} + \\
& 150x_{11,16} + 1273x_{12,1} + 912x_{12,2} + 898x_{12,3} + 693x_{12,4} + 815x_{12,5} + 147x_{12,6} \\
& 759x_{12,7} + 639x_{12,8} + 386x_{12,9} + 71x_{12,10} + 879x_{12,11} + 140x_{12,13} + \\
& 263x_{12,14} + 200x_{12,15} + 1704x_{12,16} + 596x_{13,1} + 715x_{13,2} + 960x_{13,3} + 579x_{13,4} + \\
& 680x_{13,5} + 695x_{13,6} + 1012x_{13,7} + 980x_{13,8} + 810x_{13,9} + 183x_{13,10} + 645x_{13,11} + 140x_{13,12} + \\
& 123x_{13,14} + 503x_{13,15} + 699x_{13,16} + 402x_{14,1} + 1116x_{14,2} + 728x_{14,3} + \\
& 318x_{14,4} + 319x_{14,5} + 879x_{14,6} + 1321x_{14,7} + 1280x_{14,8} + 961x_{14,9} + \\
& 641x_{14,10} + 304x_{14,11} + 263x_{14,12} + 123x_{14,13} + 63x_{14,15} + 818x_{14,16} + 356x_{15,1} + \\
& 420x_{15,2} + 597x_{15,3} + 251x_{15,4} + 466x_{15,5} + 292x_{15,6} + 244x_{15,7} + 699x_{15,8} + 293x_{15,9} + \\
& 477x_{15,10} + 851x_{15,11} + 200x_{15,12} + 503x_{15,13} + 63x_{15,14} + 456x_{15,16} + 120x_{16,1} + 510x_{16,2} + \\
& 847x_{16,3} + 818x_{16,4} + 210x_{16,5} + 1455x_{16,6} + 1413x_{16,7} + 763x_{16,8} + 1031x_{16,9} + \\
& 1080x_{16,10} + 150x_{16,11} + 1704x_{16,12} + 699x_{16,13} + 818x_{16,14} + 456x_{16,15}
\end{aligned}
$$

sujeto a

$$x_{2,1}+x_{3,1}+x_{4,1}+x_{5,1}+x_{6,1}+x_{7,1}+x_{8,1}+x_{9,1}+x_{10,1}+x_{11,1}+x_{12,1}+x_{13,1}+x_{14,1}+x_{15,1}+x_{16,1}=1$$
$$x_{1,2}+x_{3,2}+x_{4,2}+x_{5,2}+x_{6,2}+x_{7,2}+x_{8,2}+x_{9,2}+x_{10,2}+x_{11,2}+x_{12,2}+x_{13,2}+x_{14,2}+x_{15,2}+x_{16,2}=1$$
$$x_{1,3}+x_{2,3}+x_{4,3}+x_{5,3}+x_{6,3}+x_{7,3}+x_{8,3}+x_{9,3}+x_{10,3}+x_{11,3}+x_{12,3}+x_{13,3}+x_{14,3}+x_{15,3}+x_{16,3}=1$$
$$x_{1,4}+x_{2,4}+x_{3,4}+x_{5,4}+x_{6,4}+x_{7,4}+x_{8,4}+x_{9,4}+x_{10,4}+x_{11,4}+x_{12,4}+x_{13,4}+x_{14,4}+x_{15,4}+x_{16,4}=1$$
$$x_{1,5}+x_{2,5}+x_{3,5}+x_{4,5}+x_{6,5}+x_{7,5}+x_{8,5}+x_{9,5}+x_{10,5}+x_{11,5}+x_{12,5}+x_{13,5}+x_{14,5}+x_{15,5}+x_{16,5}=1$$
$$x_{1,6}+x_{2,6}+x_{3,6}+x_{4,6}+x_{5,6}+x_{7,6}+x_{8,6}+x_{9,6}+x_{10,6}+x_{11,6}+x_{12,6}+x_{13,6}+x_{14,6}+x_{15,6}+x_{16,6}=1$$
$$x_{1,7}+x_{2,7}+x_{3,7}+x_{4,7}+x_{5,7}+x_{6,7}+x_{8,7}+x_{9,7}+x_{10,7}+x_{11,7}+x_{12,7}+x_{13,7}+x_{14,7}+x_{15,7}+x_{16,7}=1$$
$$x_{1,8}+x_{2,8}+x_{3,8}+x_{4,8}+x_{5,8}+x_{6,8}+x_{7,8}+x_{9,8}+x_{10,8}+x_{11,8}+x_{12,8}+x_{13,8}+x_{14,8}+x_{15,8}+x_{16,8}=1$$
$$x_{1,9}+x_{2,9}+x_{3,9}+x_{4,9}+x_{5,9}+x_{6,9}+x_{7,9}+x_{8,9}+x_{10,9}+x_{11,9}+x_{12,9}+x_{13,9}+x_{14,9}+x_{15,9}+x_{16,9}=1$$
$$x_{1,10}+x_{2,10}+x_{3,10}+x_{4,10}+x_{5,10}+x_{6,10}+x_{7,10}+x_{8,10}+x_{9,10}+x_{11,10}+x_{12,10}+x_{13,10}+x_{14,10}+x_{15,10}+x_{16,10}=1$$
$$x_{1,11}+x_{2,11}+x_{3,11}+x_{4,11}+x_{5,11}+x_{6,11}+x_{7,11}+x_{8,11}+x_{9,11}+x_{10,11}+x_{12,11}+x_{13,11}+x_{14,11}+x_{15,11}+x_{16,11}=1$$
$$x_{1,12}+x_{2,12}+x_{3,12}+x_{4,12}+x_{5,12}+x_{6,12}+x_{7,12}+x_{8,12}+x_{9,12}+x_{10,12}+x_{11,12}+x_{13,12}+x_{14,12}+x_{15,12}+x_{16,12}=1$$
$$x_{1,13}+x_{2,13}+x_{3,13}+x_{4,13}+x_{5,13}+x_{6,13}+x_{7,13}+x_{8,13}+x_{9,13}+x_{10,13}+x_{11,13}+x_{12,13}+x_{14,13}+x_{15,13}+x_{16,13}=1$$
$$x_{1,14}+x_{2,14}+x_{3,14}+x_{4,14}+x_{5,14}+x_{6,14}+x_{7,14}+x_{8,14}+x_{9,14}+x_{10,14}+x_{11,14}+x_{12,14}+x_{13,14}+x_{15,14}+x_{16,14}=1$$
$$x_{1,15}+x_{2,15}+x_{3,15}+x_{4,15}+x_{5,15}+x_{6,15}+x_{7,15}+x_{8,15}+x_{9,15}+x_{10,15}+x_{11,15}+x_{12,15}+x_{13,15}+x_{14,15}+x_{16,15}=1$$
$$x_{1,16}+x_{2,16}+x_{3,16}+x_{4,16}+x_{5,16}+x_{6,16}+x_{7,16}+x_{8,16}+x_{9,16}+x_{10,16}+x_{11,16}+x_{12,16}+x_{13,16}+x_{14,16}+x_{15,16}=1$$

$$x_{1,2}+x_{1,3}+x_{1,4}+x_{1,5}+x_{1,6}+x_{1,7}+x_{1,8}+x_{1,9}+x_{1,10}+x_{1,11}+x_{1,12}+x_{1,13}+x_{1,14}+x_{1,15}+x_{1,16}=1$$
$$x_{2,1}+x_{2,3}+x_{2,4}+x_{2,5}+x_{2,6}+x_{2,7}+x_{2,8}+x_{2,9}+x_{2,10}+x_{2,11}+x_{2,12}+x_{2,13}+x_{2,14}+x_{2,15}+x_{2,16}=1$$
$$x_{3,1}+x_{3,2}+x_{3,4}+x_{3,5}+x_{3,6}+x_{3,7}+x_{3,8}+x_{3,9}+x_{3,10}+x_{3,11}+x_{3,12}+x_{3,13}+x_{3,14}+x_{3,15}+x_{3,16}=1$$
$$x_{4,1}+x_{4,2}+x_{4,3}+x_{4,5}+x_{4,6}+x_{4,7}+x_{4,8}+x_{4,9}+x_{4,10}+x_{4,11}+x_{4,12}+x_{4,13}+x_{4,14}+x_{4,15}+x_{4,16}=1$$
$$x_{5,1}+x_{5,2}+x_{5,3}+x_{5,4}+x_{5,6}+x_{5,7}+x_{5,8}+x_{5,9}+x_{5,10}+x_{5,11}+x_{5,12}+x_{5,13}+x_{5,14}+x_{5,15}+x_{5,16}=1$$
$$x_{6,1}+x_{6,2}+x_{6,3}+x_{6,4}+x_{6,5}+x_{6,7}+x_{6,8}+x_{6,9}+x_{6,10}+x_{6,11}+x_{6,12}+x_{6,13}+x_{6,14}+x_{6,15}+x_{6,16}=1$$
$$x_{7,1}+x_{7,2}+x_{7,3}+x_{7,4}+x_{7,5}+x_{7,6}+x_{7,8}+x_{7,9}+x_{7,10}+x_{7,11}+x_{7,12}+x_{7,13}+x_{7,14}+x_{7,15}+x_{7,16}=1$$
$$x_{8,1}+x_{8,2}+x_{8,3}+x_{8,4}+x_{8,5}+x_{8,6}+x_{8,7}+x_{8,9}+x_{8,10}+x_{8,11}+x_{8,12}+x_{8,13}+x_{8,14}+x_{8,15}+x_{8,16}=1$$
$$x_{9,1}+x_{9,2}+x_{9,3}+x_{9,4}+x_{9,5}+x_{9,6}+x_{9,7}+x_{9,8}+x_{9,10}+x_{9,11}+x_{9,12}+x_{9,13}+x_{9,14}+x_{9,15}+x_{9,16}=1$$
$$x_{10,1}+x_{10,2}+x_{10,3}+x_{10,4}+x_{10,5}+x_{10,6}+x_{10,7}+x_{10,8}+x_{10,9}+x_{10,11}+x_{10,12}+x_{10,13}+x_{10,15}+x_{10,16}=1$$
$$x_{11,1}+x_{11,2}+x_{11,3}+x_{11,4}+x_{11,5}+x_{11,6}+x_{11,7}+x_{11,8}+x_{11,9}+x_{11,10}+x_{11,12}+x_{11,13}+x_{11,14}+x_{11,15}+x_{11,16}=1$$
$$x_{12,1}+x_{12,2}+x_{12,3}+x_{12,4}+x_{12,5}+x_{12,6}+x_{12,7}+x_{12,8}+x_{12,9}+x_{12,10}+x_{12,11}+x_{12,13}+x_{12,14}+x_{12,15}+x_{12,16}=1$$
$$x_{13,1}+x_{13,2}+x_{13,3}+x_{13,4}+x_{13,5}+x_{13,6}+x_{13,7}+x_{13,8}+x_{13,9}+x_{13,10}+x_{13,11}+x_{13,12}+x_{13,14}+x_{13,15}+x_{13,16}=1$$
$$x_{14,1}+x_{14,2}+x_{14,3}+x_{14,4}+x_{14,5}+x_{14,6}+x_{14,7}+x_{14,8}+x_{14,9}+x_{14,10}+x_{14,11}+x_{14,12}+x_{14,13}+x_{14,15}+x_{14,16}=1$$
$$x_{15,1}+x_{15,2}+x_{15,3}+x_{15,4}+x_{15,5}+x_{15,6}+x_{15,7}+x_{15,8}+x_{15,9}+x_{15,10}+x_{15,11}+x_{15,12}+x_{15,13}+x_{15,14}+x_{15,16}=1$$
$$x_{16,1}+x_{16,2}+x_{16,3}+x_{16,4}+x_{16,5}+x_{16,6}+x_{16,7}+x_{16,8}+x_{16,9}+x_{16,10}+x_{16,11}+x_{16,12}+x_{16,13}+x_{16,14}+x_{16,15}=1$$
$$x_{ij}=0\ \acute{o}\ 1$$

Como se mencionó en el capítulo anterior el espacio de búsqueda del problema del agente viajero está dado por $S = \frac{(n-1)!}{2}$, entonces para el problema anterior el espacio de búsqueda es $\frac{(15)!}{2} \approx 653837184000$, esto nos indica el número de posibles soluciones para el problema en cuestión.

Calcular todas estas posibilidades aún para una máquina resultaría costoso ya que el tiempo necesario para la ejecución es costoso en tiempo. Es por este motivo que el uso de los métodos tradicionales resulta insuficiente. Para este tipo de problemas es conveniente usar las técnicas heurísticas y metaheurísticas, pues sin explorar todo este espacio de búsqueda éstas logran encontrar soluciones que se acercan a la óptima o en el mejor de los casos encuentran la solución óptima global.

El uso de la BT y los AGs para resolver este tipo de problemas ha demostrado ser eficiente. En el siguiente apartado se resolverá el problema de la empresa con cada una de las metaheurísticas mencionadas.

B. Planteamiento del problema utilizando Búsqueda Tabú.

En el capítulo II se describieron los elementos necesarios para implementar la metaheurística llamada BT. Una de las características principales de este método es que se necesita una solución inicial, para el problema de reparto mencionado anteriormente ordenaremos las ciudades de menor a mayor de acuerdo a las distancias que se tienen que recorrer de una a otra.

Este ordenamiento se auxiliará de las restricciones planteadas en el apartado *A* de este capítulo, con la finalidad de generar rutas iniciales que sean factibles.

Las siguientes rutas se obtuvieron con software que genera las rutas de manera pseudoaleatoria:

1. 1-14-13-10-15-9-7-3-8-6-4-5-16-11-12-2-1
2. 1-16-7-2-3-4-13-12-10-9-8-6-15-11-14-5-1
3. 1-3-7-9-12-10-13-14-11-16-5-2-4-15-6-8-1
4. 1-4-3-5-10-7-8-6-14-13-12-9-15-2-11-16-1
5. 1-16-15-12-13-6-7-3-4-2-9-8-10-5-11-14-1
6. 1-11-10-12-8-9-2-3-15-14-13-7-6-4-5-16-1
7. 1-5-4-11-10-16-15-12-13-14-3-7-9-6-8-2-1
8. 1-11-10-7-8-6-14-13-4-5-16-15-9-12-2-3-1
9. 1-13-12-6-7-3-4-2-15-14-8-9-16-11-10-5-1
10. 1-5-4-11-10-16-15-12-13-14-3-7-6-9-8-2-1

El costo para cada una de las rutas anteriores obtenido por las distancias mostradas en la matriz de costo del problema es el siguiente:

1. 5848
2. 5914
3. 3844
4. 6031
5. 6542

6. 6621
7. 6692
8. 7070
9. 7504
10. 7622

Otro aspecto que sea consideró vital para el éxito de la BT en investigaciones realizadas es el tamaño de la lista tabú, para este problema utilizaremos un tamaño de 20. Este tamaño se obtuvo de un promedio de los 3 tamaños que son considerados los más adecuados, el primer valor se obtuvo de los artículos publicados en donde se menciona que la lista debe tener un tamaño de 7, el segundo lo obtuvimos de la siguiente igualdad $t = \sqrt{n}$ y dado que n es 16, $t = 4$, el tercero fue encontrado por una aplicación realizada por Knox[1] en donde $t = 3n$, de tal manera que $t = 48$.

La manera de implementar esta lista tabú es por medio de un arreglo circular FIFO, la pregunta es: ¿cuánto tiempo permanecerá un elemento en esta lista antes de ser eliminado? El tiempo se medirá en iteraciones, de esta manera un elemento permanecerá tabú $0.0003(n^4)$ iteraciones. Sustituyendo el valor de *n* en la expresión anterior obtenemos:

$$0.0003(n^4) = 0.0003(16^4) \approx 19$$

[1] Michalewicz Zbigniew y Fogel David. *How to solve it: modern heuristics*, pp. 133.

La relación anterior fue obtenida por Knox[2] en la misma investigación en la que obtuvo la relación para encontrar el tamaño de la lista tabú.

Para decidir que elementos serán tabú (es decir permanecerán en la lista tabú) lo haremos mediante la función objetivo. Nombremos i^* a una solución para que ésta no sea tabú, la función objetivo del problema debe ser menor o igual a la solución obtenida anteriormente nombrada i de tal manera que se cumpla la siguiente desigualdad:

$$f(i^*) \leq f(i)$$

La BT realiza una serie de movimientos para moverse de un punto a otro en el espacio de búsqueda. Estos movimientos tienen un valor que hacen posible saber si la exploración de soluciones se acerca a la óptima o no.

Para obtener el valor de un movimiento obtendremos la diferencia de la función objetivo antes de efectuar un movimiento y después de efectuarse el movimiento. Suponiendo que x es un movimiento y x^* el siguiente movimiento entonces:

$$valor\_movimiento = f(x) - f(x^*)$$

La BT utiliza una memoria y la lista tabú para evitar que se visiten soluciones que ya fueron revisadas. Para este problema se implementará la memoria a corto y largo plazo, debido a que hay un gran número de posibles soluciones.

Debido a que el problema está planteado como un problema binario, la primera de éstas memorias utilizará las restricciones basadas en recencia, así podremos almacenar en que momento una variable $x_{i,j}$ ha cambiado su valor de 0 a

[2] Michalewicz Zbigniew y Fogel David. *How to solve it: modern heuristics*, pp. 134.

1 y conseguiremos evitar volver a realizar ese cambio. La manera en que se implementará es: generando un contador que iniciará en 0 y se incrementará en 1 cada que ocurra un movimiento, también se creará una matriz que se inicializará con 0, y si la variable $x_{i,j}$ cambia de valor actualizaremos la posición i,j de la matriz con el valor que tenga el contador, esto se hace con la finalidad de saber en que iteración ocurrió el cambio y determinar en que iteración podremos volver a cambiar el valor de la variable.

Para que la variable $x_{i,j}$ pueda volver a cambiar su valor se tiene que cumplir la siguiente desigualdad:

$$matriz(i, j) + tenencia\_tabú < contador$$

El valor de la *tenencia_tabú* es fijo para todas las variables, en este trabajo es de 16 debido a que nuestro problema tiene dieciséis ciudades, en algunas otras aplicaciones este valor puede ser un múltiplo de la raíz cuadrada del total de atributos que se tengan.

La memoria a corto plazo cuenta con los elementos: Recencia, Niveles de aspiración, Restricciones y Lista de candidatos; nuestro problema se implementará por tiempo independiente, esto nos garantizará que si encontramos una solución con la mejor obtenida hasta el momento clasificada como tabú, podamos utilizarla y dirigir la búsqueda hacia otra región.

También manejaremos una lista de candidatos élite la cual almacenará las 15 mejores soluciones encontradas, la lista se actualizará si surge una solución que sea mejor que cualquiera de las almacenadas en ésta.

La implementación de la memoria a largo plazo se basará en frecuencia por medio de la cual se guardará el historial de los movimientos de la variable $x_{i,j}$ para saber cuantas veces a cambiado su valor de 0 a 1 y viceversa.

Implícitamente los conceptos de intensificación y diversificación son usados, ya que si el proceso no logra encontrar una solución que mejore la obtenida hasta el momento, se procede a utilizar la memoria a corto plazo con el criterio establecido en los niveles de aspiración y entonces se estará realizando la diversificación del espacio de búsqueda. La intensificación se desarrolla cuando se examina la lista de candidatos élite debido a que la memoria a corto plazo no es suficiente para encontrar soluciones mejores.

También implementaremos la estrategia de oscilación cambiando los valores de 0 a 1 en las variables $x_{i,j}$, se creará una matriz que almacene 1 si esa variable está en la solución actual y 0 en caso contrario. En esta matriz se generarán los cambios de los valores de las variables $x_{i,j}$ y se actualizará en cada iteración, dependiendo de las variables que se tengan en la solución actual.

Para implementar el reencadenamiento de trayectorias utilizaremos la lista de candidatos élite y la solución actual, obtendremos al azar una subcadena tanto de la solución actual como una de las almacenadas en la lista de candidatos y las intercambiaremos para generar una nueva solución actual.

C. Planteamiento del problema utilizando AGs.

En el capítulo anterior se puntualizaron los componentes que se tienen que considerar para implementar un AG. Empezaremos por el tipo de codificación que tendrá el problema.

Se utilizará una representación entera en donde cada alelo contendrá el valor de una ciudad y un cromosoma nos indicará una posible ruta, por ejemplo las ciudades se numeraron de uno a dieciséis y sabemos que nuestra ciudad origen y destino esta etiquetada con el número 1.

De tal manera un cromosoma tendrá la siguiente forma:

1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	1

Cada comodín (*) representa el valor de una de las etiquetas de las ciudades, es decir puede tomar el valor de entre 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 sin repetir algún número.

Se decidió esta codificación porque es fácil interpretar los resultados que se obtengan después de ejecutar el programa del AG. Además de que la aplicación de los operadores, de esta metaheurística, no genera soluciones invalidas como sería en el caso de la representación binaria.

La representación que se usará para identificar la ruta es la denominada en ingles Path (La representación del camino). Un ejemplo de esta representación es la siguiente ruta:

1-3-7-9-8-6-12-10-13-14-15-4-2-5-11-16-1

La cual será representada por el siguiente vector (denominado cromosoma)

1	3	7	9	8	6	12	10	13	14	15	4	2	5	11	16	1

Dado que la ciudad inicial como la final son conocidas sólo nos concentraremos en encontrar la ruta de las 15 ciudades restantes considerando que la ruta obtenida empezará y terminará en la ciudad número 1. Esto se hizo con el

propósito de ahorrar recursos computacionales, puesto que el funcionamiento del AG es el mismo para 10 o para 30 ciudades.

El tamaño de la población será de acuerdo a la relación encontrada por Goldberg $tamaño\_población = 1.65^{2^{(0.21L)}}$, donde L es igual a 15. De esta manera:

$$tamaño\_población = 1.65^{2^{(0.21(15))}} = 85.21349173$$

Redondeando el tamaño de la población será de 85 individuos, nuestra población inicial se generará de manera aleatoria.

La selección será una mezcla de la proporcional por ruleta y la uniforme, además de implementar la estrategia elitista. El algoritmo a seguir será el siguiente:

1. Se evaluarán los individuos de acuerdo a su función de aptitud.
2. El más apto se pasará intacto a la siguiente generación.
3. Por medio de la selección proporcional por ruleta se elegirán a los individuos que reemplazarán a los menos aptos para pasar a la siguiente generación, al igual que a los individuos que se cruzarán.

Se escogió este mezcla de selección ya que ambos tipos de selecciones al igual que la estrategia elitista nos dan la oportunidad de conservar a los individuos con una mejor aptitud y a la vez nos permiten deshacernos de los individuos que tienen la peor aptitud; el hecho de cruzar a los individuos nos permite crear diversidad en nuestra nueva población.

De acuerdo a diferentes investigaciones realizadas en especial la presentada en el COMCEV'03 (Congreso de Computación Evolutiva) por Everardo Gutiérrez

López y Carlos A. Brizuela Rodríguez[3] y los estudios realizados por Grefenstette los parámetros para las operadores de cruza y mutación son equivalentes al promedio de los resultados de las investigaciones mencionadas.

La cruza se realizará con un probabilidad de 0.8, si la probabilidad de cruza es menor entonces se remplazarán los individuos menos aptos con los individuos obtenidos por medio de la selección, en caso de que la probabilidad de cruza sea igual o mayor a 0.8 se cruzarán los individuos derivados de la selección.

La mutación tendrá una probabilidad de 0.292

El número máximo de generaciones es de 200 ya que el número de ciudades no es mayor a 30

Habiendo definido las probabilidades de cruza y mutación especificaremos que tipo de operadores se usarán. Para la cruza se ha concluido que con la codificación y representación seleccionadas, el operador *Order Crossover* es el más conveniente por los resultados logrados en experimentos anteriores.

El operador de mutación que se aplicará es por *inversión*.

Las restricciones se manejarán por medio de la matriz de costo. Para saber si se puede ir de una ciudad a otra bastará con saber si la intersección de estas ciudades es diferente de cero. Las restricciones que nos aseguran que de una ciudad *x*, solo visitaremos una sola ciudad y viceversa lo verificaremos sumando el número de valores que sean diferentes de cero en el renglón y columna en cuestión de la matriz de costo; si el resultado de esta suma es mayor a 1 nos indicará que esa ruta no es permitida.

[3] Memorias del primer Congreso de Computación Evolutiva, pp. 81.

Dado que en nuestro problema el objetivo es minimizar una función objetivo, sabemos que este problema lo podemos convertir en un problema de maximización si multiplicamos por –1, sin embargo si hacemos esto todos nuestros costos serían negativos. De esta manera nuestra función de adaptación no puede depender de la función objetivo, por lo que se procederá a evaluar a los individuos obtenidos en una generación por medio de la función objetivo y a ordenarlos de mayor a menor dependiendo de la evaluación, finalmente consideraremos que 100 es la mejor aptitud obtenida y 0 la peor, sacaremos la relación con los individuos y se les asignará su aptitud.

Los números aleatorios se generarán utilizando el paquete java.util.Random el cual toma un número de una secuencia grande de números para generar pseudoaleatorios con un valor entre 0.0 y 1. La semilla para la generación de estos es la hora actual del sistema. Los pseudoaleatorios se generan con un probabilidad normal con una media de 0.0 y una desviación estándar de 1.0

Para obtener pseudoaleatorios enteros utilizaremos el mismo paquete de java, con una multiplicación adicional para obtener enteros entre 0 y 16.

Los mejores libros son aquellos que quienes los leen creen que también ellos pudieron haberlos escrito.

PASCAL

IV. ANÁLISIS DE LAS METAHEURÍSTICAS.

A. Resultados obtenidos con Búsqueda Tabú.

Los parámetros utilizados para la ejecución del algoritmo de Búsqueda Tabú, descritos en el capitulo III, se presentan a continuación:

- Total de iteraciones igual a 100.
- Valor de tenure igual 19.
- Tamaño de la lista tabú igual a 20.
- Siempre se empleara el criterio de aspiración.

- Tenencia igual a 16.
- Las rutas iniciales se describen en el capitulo anterior en el apartado B.

El algoritmo de búsqueda tabú se proceso 10 veces y para cada una de estas ejecuciones se utilizó una ruta inicial diferente.

La *primera ejecución* generó en un tiempo de 26659 milisegundos equivalente a 0.44433166667 minutos, la siguiente ruta:

1-3-7-15-14-12-6-8-9-10-13-11-16-5-4-2-1

con un costo de 3567 kilómetros, la ruta inicial fue : 1-14-13-10-15-9-7-3-8-6-4-5-16-11-12-2-1 con un costo de 5848 kilómetros esto implica una reducción de 2281 kilómetros.

En la *segunda ejecución* se obtuvo la siguiente ruta:

1-2-4-3-7-15-9-8-6-12-10-13-14-11-16-5-1

con un costo de 2695 kilómetros, el tiempo de proceso fue de 30474 milisegundos equivalentes a 0.5079 minutos, la ruta inicial fue: 1-16-7-2-3-4-13-12-10-9-8-6-15-11-14-5-1 con un costo de 5914 kilómetros esto implicar una reducción de 3219 kilómetros.

La *tercera ejecución* se proceso en un tiempo de 27750 milisegundos equivalente a 0.4625 minutos, la ruta obtenida es:

1-3-7-9-12-10-13-14-11-16-5-2-4-8-6-15-1

con un costo de 3614, la ruta utilizada como inicial fue: 1-3-7-9-12-10-13-14-11-16-5-2-4-15-6-8-1 con un costo de 3844, la diferencia muestra una reducción de 230 kilómetros.

En la *cuarta ejecución* se produjo la siguiente ruta:

1-3-7-9-12-10-13-14-11-16-5-2-4-8-6-15-1

con un costo de 3371 kilómetros, se proceso en un tiempo de 23744 milisegundos equivalente a 0.3957333 minutos. La ruta inicial fue: 1-4-3-5-10-7-8-6-14-13-12-9-15-2-11-16-1 con un costo de 6031 kilómetros esto indica que hubo una reducción de 2260 kilómetros.

De la *quinta ejecución* se obtuvo la siguiente ruta:

1-3-4-5-14-13-10-12-15-7-9-8-6-2-11-16-1

con un costo de 3371 kilómetros, la ruta se proceso en 19989 milisegundos equivalente a 0.33315 minutos. 1-16-15-12-13-6-7-3-4-2-9-8-10-5-11-14-1 fue la ruta inicial con un costo de 6542, lo que proporciona un disminución de 3171 kilómetros.

En la *sexta ejecución* la ruta obtenida es:

1-11-2-9-8-6-12-10-13-14-15-7-3-4-5-16-1

con un costo de 3226 kilómetros, el tiempo en el que se generó fue de 27500 milisegundos equivalente a 0.4583333 minutos. La ruta inicial fue 1-11-10-12-8-9-2-3-15-14-13-7-6-4-5-16-1 con un costo de 6621, de tal forma que el costo se redujo en 3395 kilómetros.

La *séptima ejecución* generó la siguiente ruta:

1-16-11-5-14-13-10-12-6-8-9-15-7-3-4-2-1

con un costo de 2760 kilómetros, el tiempo de ejecución fue de 20880 milisegundos equivalente a 0.348 minutos, la ruta inicial utilizada fue 1-5-4-11-10-16-15-12-13-14-3-7-9-6-8-2-1 con un costo 6692 kilómetros 3932, esto indica que hubo una reducción de 2260 kilómetros.

La *octava ejecución* provee como resultado la siguiente ruta:

1-2-7-15-14-13-10-12-6-8-9-4-3-5-11-16-1

con un costo 2753 de kilómetros en un tiempo de 23213 milisegundos equivalente a 0.3868833 minutos, la ruta inicial fue 1-11-10-7-8-6-14-13-4-5-16-15-9-12-2-3-1 con un costo de 7070 kilómetros, la diferencia muestra una reducción de 4317 kilómetros.

La *novena ejecución* produce la ruta siguiente:

1-13-12-15-4-2-3-7-9-8-6-10-14-11-16-5-1

en un tiempo de 30574 milisegundos equivalente a 0.509566667 minutos, con un costo de 4196 kilómetros. La ruta inicial fue 1-13-12-6-7-3-4-2-15-14-8-9-16-11-10-5-1 con un costo 7504 kilómetros, lo que implica una reducción de 3308 kilómetros.

En la *décima ejecución* se produjo la ruta siguiente:

1-16-11-5-14-13-10-12-6-8-9-15-7-3-4-2-1

en un tiempo de 18296 milisegundos equivalente a 0.3049333 minutos, con un costo de 2760 kilómetros; la ruta utilizada como inicial fue 1-5-4-11-10-16-15-12-13-14-3-7-6-9-8-2-1 la cual tiene un costo de 7622 kilómetros, la diferencia muestra una reducción de 4862 kilómetros.

Los resultados obtenidos con la implementación de la metaheurística BT se sintetizan en la tabla 1*.

Resultados obtenidos con BT					
Ejecución	Ruta Inicial	Costo Inicial (km.)	Ruta	Costo (km.)	Tiempo (min.)
1	1-14-13-10-15-9-7-3-8-6-4-5-16-11-12-2-1	5848	1-3-7-15-14-12-6-8-9-10-13-11-16-5-4-2-1	3567	0.44433166667
2	1-16-7-2-3-4-13-12-10-9-8-6-15-11-14-5-1	5914	1-2-4-3-7-15-9-8-6-12-10-13-14-11-16-5-1	2695	0.5079
3	1-3-7-9-12-10-13-14-11-16-5-2-4-15-6-8-1	3844	1-3-7-9-12-10-13-14-11-16-5-2-4-8-6-15-1	3614	0.4625
4	1-4-3-5-10-7-8-6-14-13-12-9-15-2-11-16-1	6031	1-3-4-5-14-13-10-12-15-7-9-8-6-2-11-16-1	3371	0.3957333
5	1-16-15-12-13-6-7-3-4-2-9-8-10-5-11-14-1	6542	1-3-4-5-14-13-10-12-15-7-9-8-6-2-11-16-1	3371	0.33315
6	1-11-10-12-8-9-2-3-15-14-13-7-6-4-5-16-1	6621	1-11-2-9-8-6-12-10-13-14-15-7-3-4-5-16-1	3226	0.4583333
7	1-5-4-11-10-16-15-12-13-14-3-7-9-6-8-2-1	6692	1-16-11-5-14-13-10-12-6-8-9-15-7-3-4-2-1	2760	0.348
8	1-11-10-7-8-6-14-13-4-5-16-15-9-12-2-3-1	7070	1-2-7-15-14-13-10-12-6-8-9-4-3-5-11-16-1	2753	0.3868833
9	1-13-12-6-7-3-4-2-15-14-8-9-16-11-10-5-1	7504	1-13-12-15-4-2-3-7-9-8-6-10-14-11-16-5-1	4196	0.509566667
10	1-5-4-11-10-16-15-12-13-14-3-7-6-9-8-2-1	7622	1-16-11-5-14-13-10-12-6-8-9-15-7-3-4-2-1	2760	0.3049333

Tabla 1

* Nota: Las pantallas que muestran las ejecuciones de la BT se presentan en el apéndice 1.

B. Resultados obtenidos con Algoritmos Genéticos.

Los parámetros para la ejecución del AG de acuerdo al diseño del modelo planteado en el capitulo III, son los siguientes:

- Una probabilidad de mutación igual a 0.292.
- Una probabilidad de cruza igual a 0.8.
- El máximo de generaciones igual a 200.
- El tamaño de la población igual a 85.

El algoritmo se ejecutó 10 veces en diferentes horas debido a que la semilla para la generación de números pseudoaleatorios es la hora del sistema.

La *primera ejecución* se llevó a cabo a la 1:12 p.m. la mejor ruta fue:

1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1

obtenida en un tiempo de 818147 milisegundos equivalente a 13.63578333 minutos, el costo de esta ruta es de 2477 kilómetros, se obtuvo en la generación número 12 y el algoritmo iteró 18 generaciones.

De la *segunda ejecución* la ruta que se obtuvo fue:

1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1

en un tiempo de 724989 milisegundos que equivale a 12.08315 minutos, la ruta se obtuvo en la generación 8 y el número de generaciones iteradas fue 15. La hora de ejecución fue 1:47 p.m.

La *tercera ejecución* se llevó a cabo a las 2:59 p.m. y se obtuvo la siguiente ruta:

1-3-4-2-5-16-11-14-15-12-10-13-6-8-9-7-1

con un costo de 3543 kilómetros, la ruta se calculó en un tiempo de 864834 milisegundos equivalente a 14.4139 minutos. El número de generaciones fue de 17, y el mejor individuo se encontró en la generación 8.

La *cuarta ejecución* se realizó a las 4:26 p.m. y generó la siguiente ruta:

1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1

el costo es de 2477 kilómetros. La ruta se calculó en un tiempo de 1172276 milisegundos equivalente a 19.53793333 minutos. El número de generaciones fue de 21 y el mejor individuo se encontró en la generación 18.

La *quinta ejecución* se efectuó a las 5:50 p.m. y generó la siguiente ruta:

1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1

con un costo de 2477 kilómetros, en un tiempo de 1206275 milisegundos equivalente a 20.10458333 minutos. El mejor individuo se encontró en la generación 11; el total de generaciones procesada fue de 18.

La *sexta ejecución* se realizó a las 5:13 p.m. y se obtuvo la siguiente ruta:

1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1

en un tiempo de 1205373 milisegundos equivalente a 20.08955 minutos con un costo de 2477 kilómetros. El número de generaciones fue de 19 y el mejor individuo se localizó en la generación 15.

La *séptima ejecución* se llevó a cabo a las 8:38 p.m. y generó la siguiente ruta:

1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1

con un costo de 2477 kilómetros. La ruta se computó en un tiempo de 1355138 milisegundos equivalente a 22.58563333 minutos. El total de generaciones fue 21 y el mejor individuo se halló en la generación 14.

La *octava ejecución* se procesó a las 9:07 p.m. originando la siguiente ruta:

1-5-2-4-3-7-9-8-6-12-10-13-14-15-16-11-1

con un costo de 2495 kilómetros. La ruta se calculó en un tiempo de 1142923 milisegundos equivalente a 19.04871667 minutos. El mejor individuo se alcanzó en la generación 12. El total de generaciones fue de 18.

En la *novena ejecución* se generó la siguiente ruta:

1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1

se obtuvo en un tiempo de 839808 milisegundos equivalente a 13.9968 minutos con un costo de 2477 kilómetros. Las generaciones computadas sumaron 15. El mejor individuo se encontró en la generación 13. El inicio de ejecución fue a las 10:49 p.m.

Finalmente en la *décima ejecución* procesada a la 1:06 p.m. se creó la siguiente ruta:

1-5-2-4-3-7-9-8-6-12-10-13-14-15-16-11-1

con un costo de 2495 kilómetros en un tiempo de 892483 milisegundos equivalente a 14.87471667. El total de ejecuciones iteradas fue 15 generaciones y el mejor individuo se obtuvo en la generación 11.

Los resultados obtenidos con el AG se sintetizan en la tabla 2*.

Resultados obtenidos con el AG

Ejecución	Ruta	Costo (km.)	Tiempo (min.)	Generación de la mejor ruta	Total Generaciones
1	1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1	2477	13.63578333	12	18
2	1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1	2477	12.08315	8	15
3	1-3-4-2-5-16-11-14-15-12-10-13-6-8-9-7-1	3543	14.4139	8	17
4	1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1	2477	19.53793333	18	21
5	1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1	2477	20.10458333	11	18
6	1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1	2477	20.08955	15	19
7	1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1	2477	22.58563333	14	21
8	1-5-2-4-3-7-9-8-6-12-10-13-14-15-16-11-1	2495	19.04871667	12	18
9	1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1	2477	13.9968	13	15
10	1-5-2-4-3-7-9-8-6-12-10-13-14-15-16-11-1	2495	14.87471667	11	15

Tabla 2

* Nota: Las pantallas que muestran las ejecuciones del AG se presentan en el apéndice 2.

C. Comparación de las metaheurísticas.

Acorde a los resultados, de cada una de las metaheurísticas, obtenidos y mostrados en el apartado anterior los resultados a comparar son los siguientes:

El tiempo promedio de ejecución, en minutos, de la BT para la muestra de tamaño 10 es el siguiente:

$$\frac{\sum_{i=1}^{n} tiempo}{n} = \frac{4.151331534}{10} = 0.4151331534 \approx 0.4 \quad \min.$$

Donde $n =$ *tamaño de la muestra*

El costo promedio, en kilómetros, de ejecución de la BT para la muestra de tamaño 10 es el siguiente:

$$\frac{\sum_{i=1}^{n} \cos to}{n} = \frac{32313}{10} = 3231.3 \quad km.$$

Donde $n =$ *tamaño de la muestra*

El costo promedio de la ruta inicial, en kilómetros, es el siguiente:

$$\frac{\sum_{i=1}^{n} \cos to}{n} = \frac{63688}{10} = 6368.8 \quad km.$$

Donde $n =$ *tamaño de la muestra*

La mejor ruta, generada por la BT fue:

1-2-4-3-7-15-9-8-6-12-10-13-14-11-16-5-1

con un costo de 2695 kilómetros en un tiempo de 0.5079 minutos.

El tiempo promedio, en minutos, de ejecución del AG para la muestra de tamaño 10 es el siguiente:

$$\frac{\sum_{i=1}^{n} tiempo}{n} = \frac{170.3707667}{10} = 17.03707667 \approx 17 \quad \min.$$

Donde $n = tamaño\ de\ la\ muestra$

El costo promedio, en kilómetros, de ejecución del AG para la muestra de tamaño 10 es el siguiente:

$$\frac{\sum_{i=1}^{n} costo}{n} = \frac{25872}{10} = 2587.2 \quad km.$$

Donde $n = tamaño\ de\ la\ muestra$

El número total de generaciones promedio en donde se obtuvo la mejor ruta de ejecución del AG para la muestra de tamaño 10 es el siguiente:

$$\frac{\sum_{i=1}^{n} No.\ Generaciones\ Mejor\ Ruta}{n} = \frac{122}{10} = 12.2 \approx 12$$

Donde $n = tamaño\ de\ la\ muestra$

El número total de generaciones promedio obtenidas en la ejecución del AG, para la muestra de tamaño 10 es el siguiente:

$$\frac{\sum_{i=1}^{n} No.\ Generaciones}{n} = \frac{177}{10} = 17.7 \approx 18$$

Donde n = tamaño de la muestra

La mejor ruta obtenida por el AG fue:

1-2-5-4-3-7-9-8-6-12-10-13-14-15-16-11-1

con un costo de 2477 kilómetros en un tiempo de 12.08315 minutos.

La tabla 3 muestra las comparaciones de los resultados obtenidos por cada una de las metaheurísticas.

Comparación de los resultados obtenidos por BT y AG

	BT	AG
Tiempo Promedio de Ejecución (min.)	0.4151331534 ≈ 0.4	17.03707667 ≈ 17
Costo promedio (km.)	3231.3	2587.2
Tiempo de Ejecución de la mejor ruta obtenida (min.)	0.5079	12.08315
Costo de la mejor ruta obtenida (km.)	2695	2477

Tabla 3

Al analizar los resultados de la tabla anterior, observamos que el AG encuentra mejores rutas que la BT, sin embargo la BT produce rutas factibles en un tiempo menor al del AG. Es necesario considerar aparte de los datos mostrados en la tabla 3, las características esenciales para que cada una de las metaheurísticas se ejecute. Para la BT se requiere de una ruta inicial, y a pesar de que se elaboro un software para generarlas esto implica un tiempo extra. El algoritmo genético no requiere de una solución previa. El tiempo de ejecución depende del número de ciudades, es decir, si las ciudades aumentan el tiempo aumenta en ambas metaheurísticas.

Conclusiones

Con base en los resultados obtenidos de las dos metaheurísticas se llegó a las siguientes conclusiones:

El algoritmo genético (AG) es el que obtiene una mejor ruta, sin embargo el tiempo de ejecución es en promedio 34 veces mayor que el de la búsqueda tabú (BT). Por esta razón la decisión de usar una metaheurística u otra depende de las condiciones de la empresa.

Si la empresa requiere una solución factible en un tiempo rápido, debido a que la demanda de reparto es alta, la alternativa para resolver el problema es la BT ya que aproximadamente en 0.4 min. encontrara una ruta con un costo promedio de 3231.3 km.

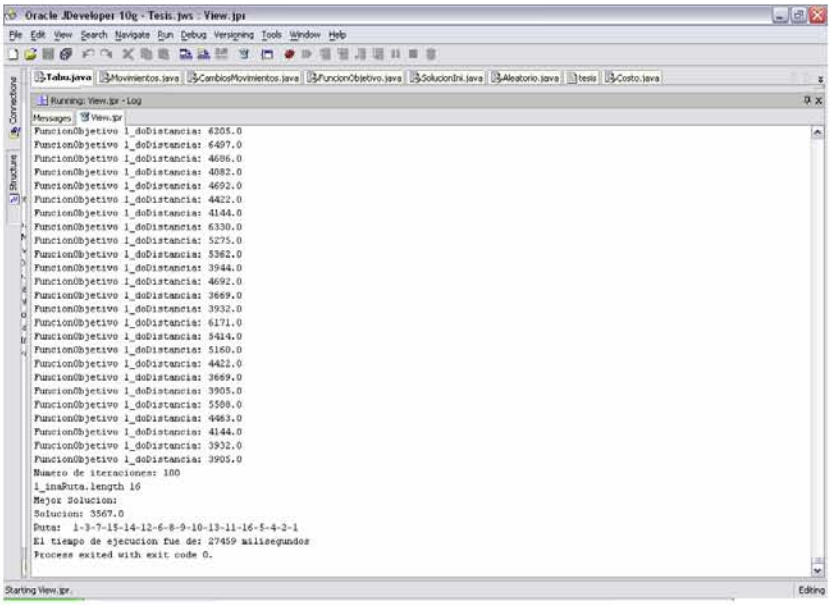
Sin embargo si la demanda de reparto no es tan alta y se cuenta con un tiempo de holgura, el AG es una alternativa eficaz ya que en un promedio de 17 min. generará una ruta con un costo promedio de 2587.2 km.

Debido a que el comportamiento de los problemas del tipo Agente Viajero para n ciudades se pueden representar de la misma forma que el problema planteado, los datos generados en tiempo y distancias con las dos metaheurísticas comparadas en este trabajo es igual para todos los problemas de este tipo, considerando que el tiempo de ejecución es proporcional al numero de ciudades, es decir si el numero de ciudades aumenta el tiempo de ejecución aumenta y si el numero de ciudades disminuye el tiempo de ejecución disminuye.

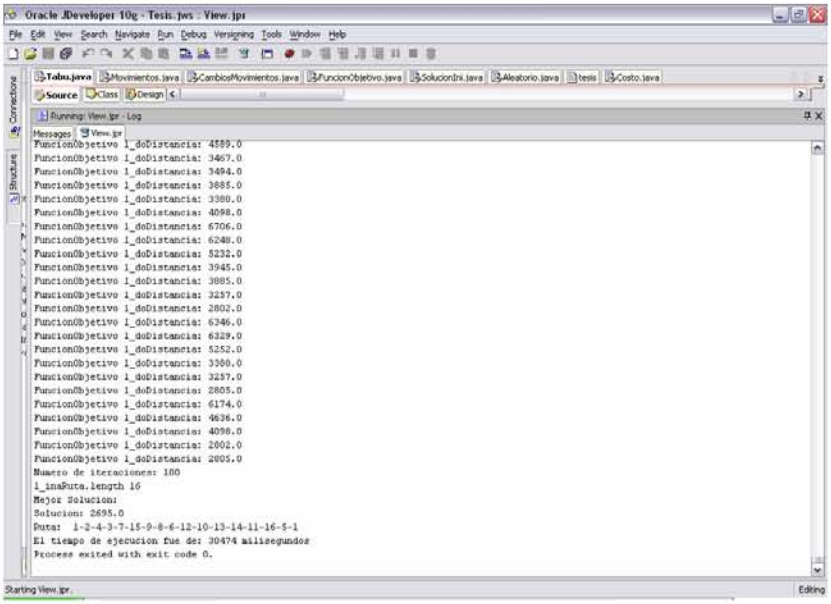
APÉNDICE 1.

Ejecuciones de la BT.

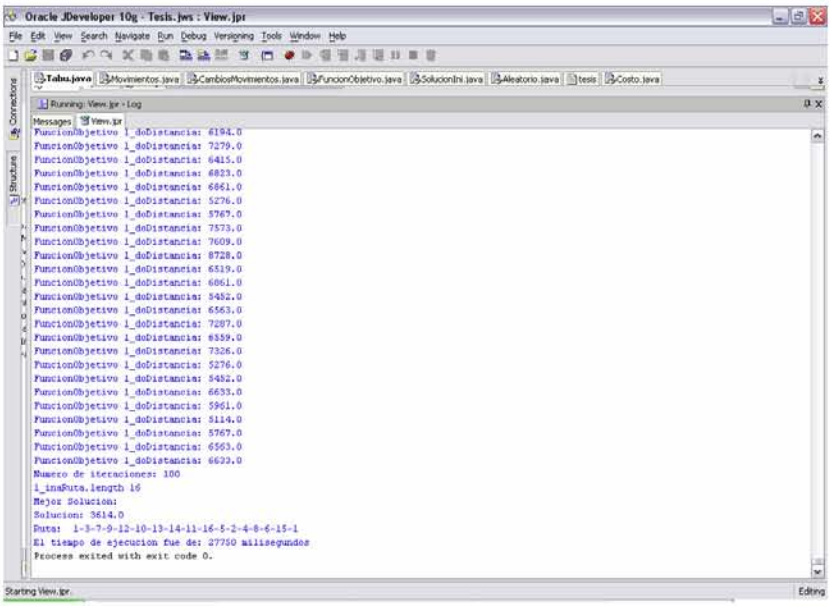
- Primera ejecución



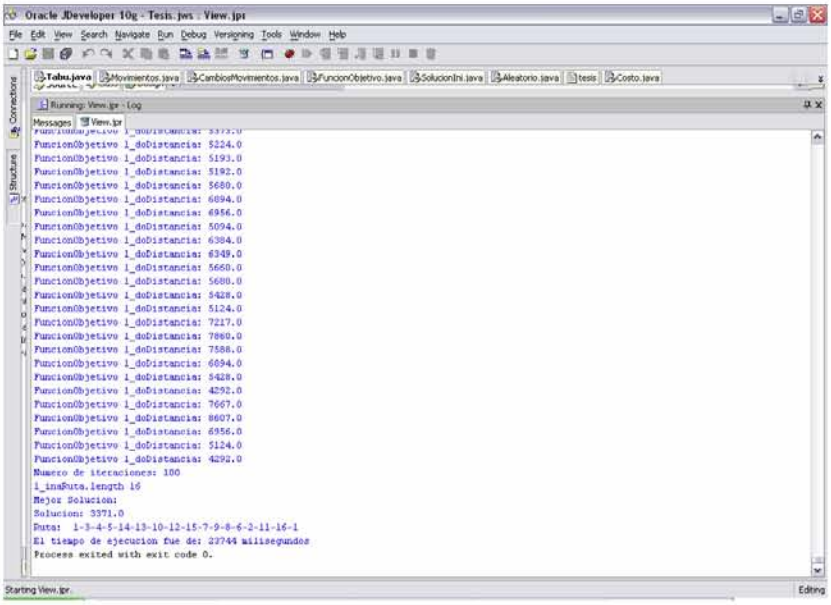
- Segunda ejecución



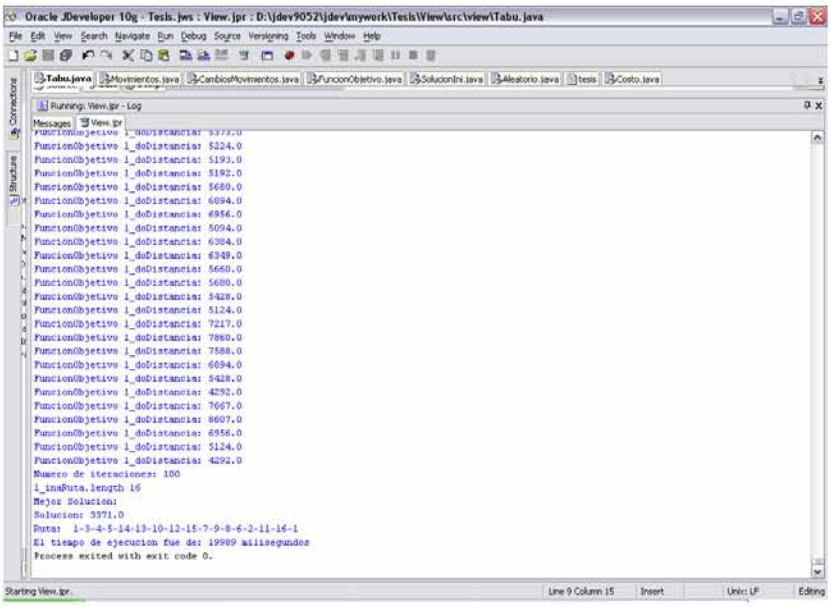
- Tercera ejecución



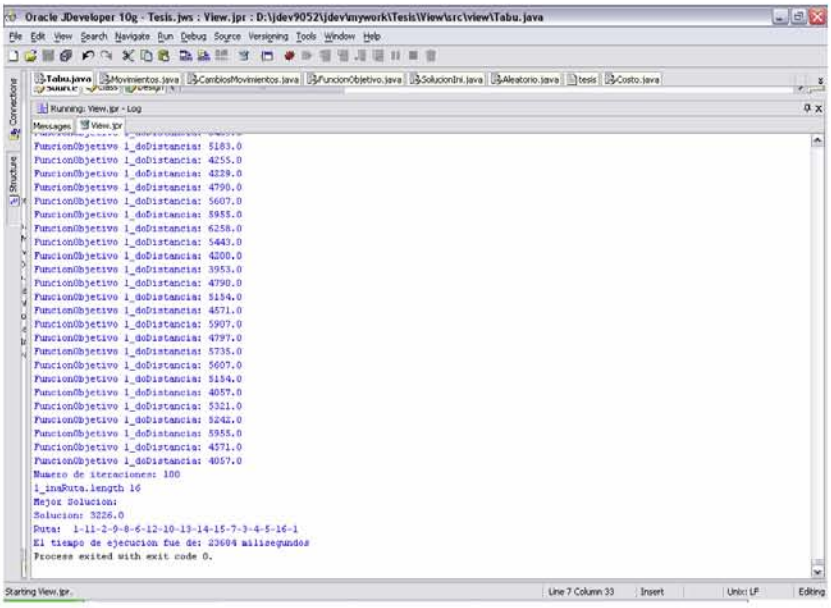
- Cuarta ejecución



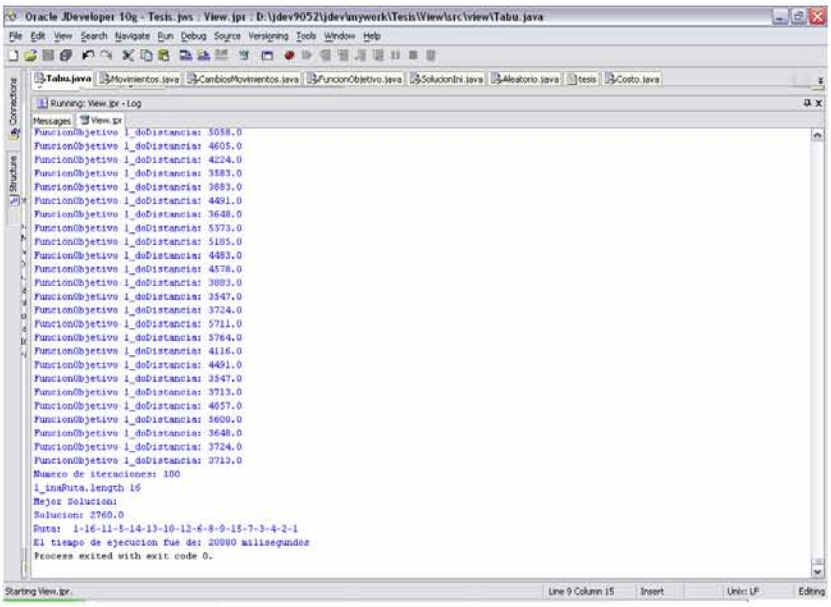
- Quinta ejecución



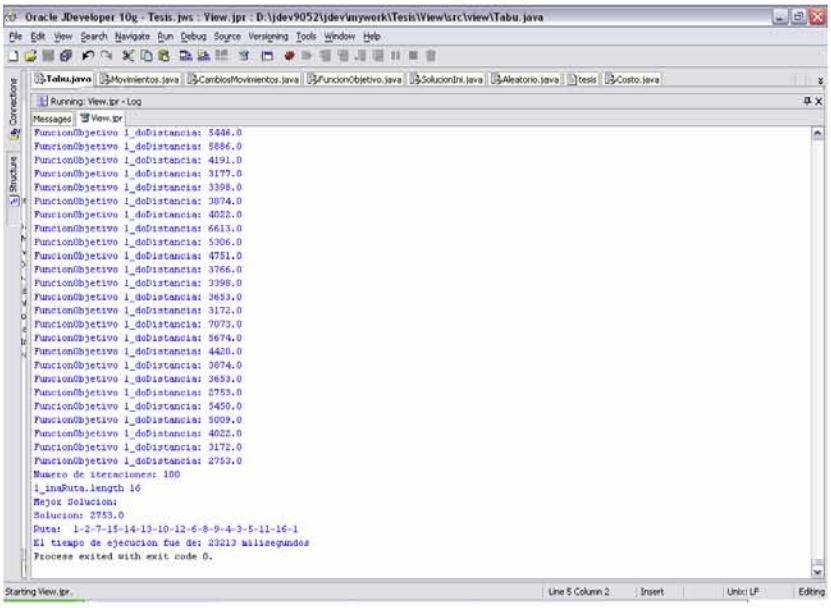
- Sexta ejecución



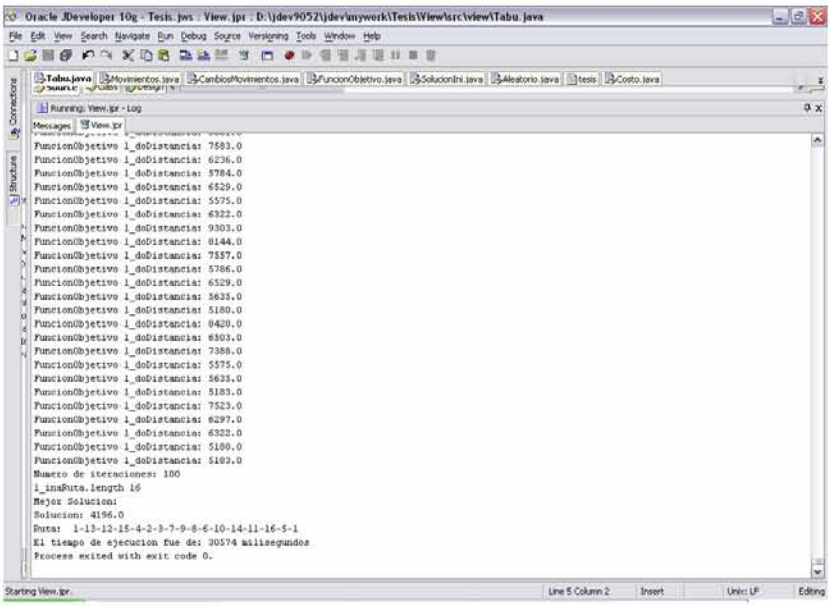
- Séptima ejecución



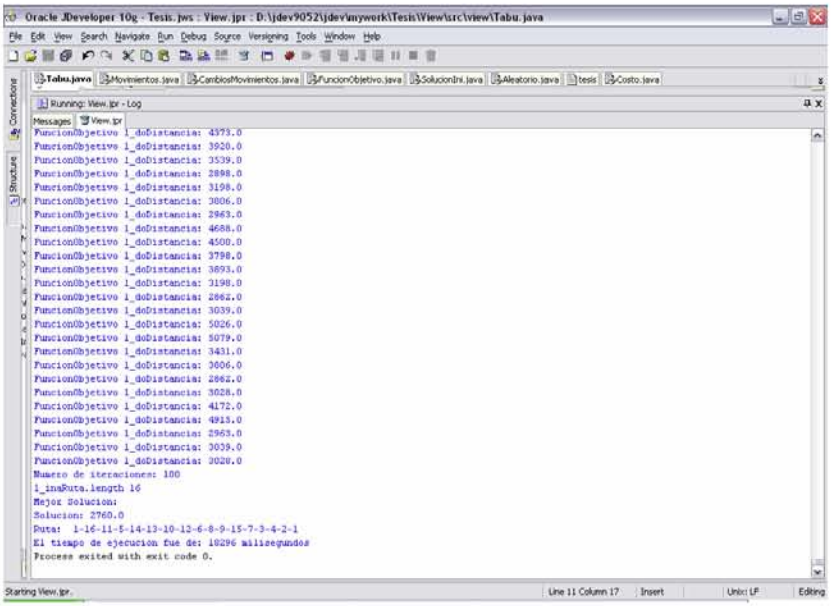
- Octava ejecución



- Novena ejecución



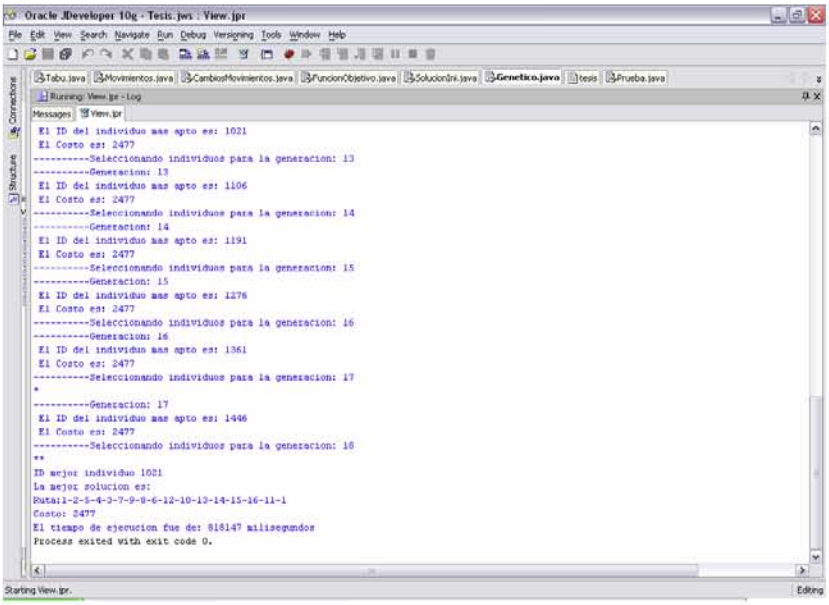
- Décima ejecución



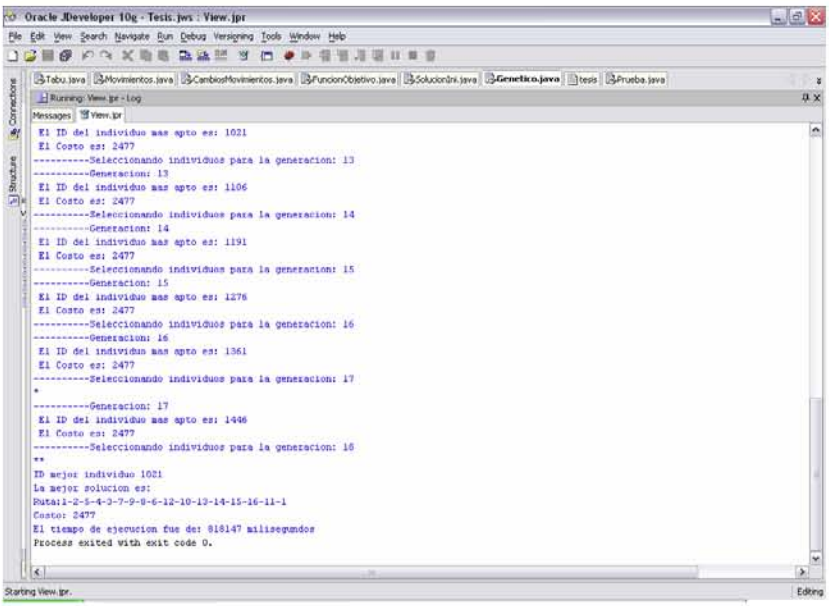
APÉNDICE 2.

Ejecuciones del AG.

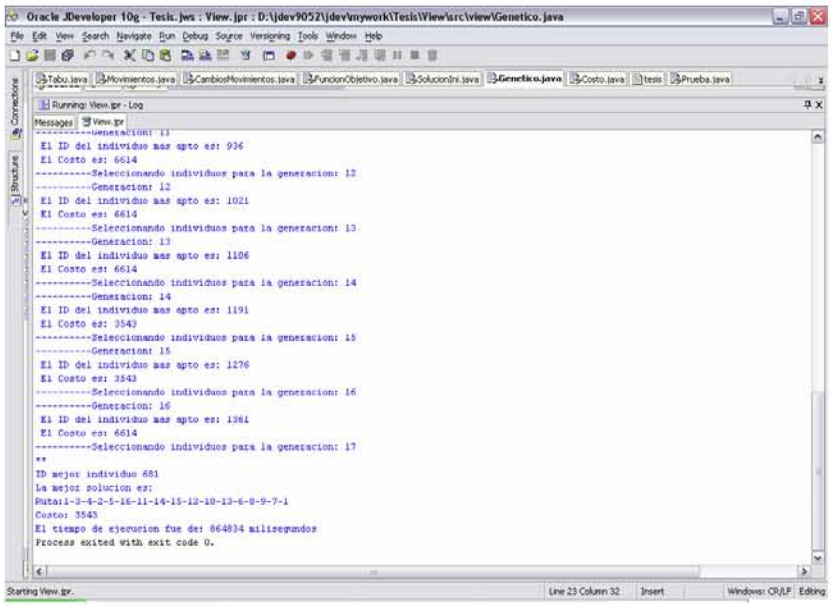
- Primera ejecución



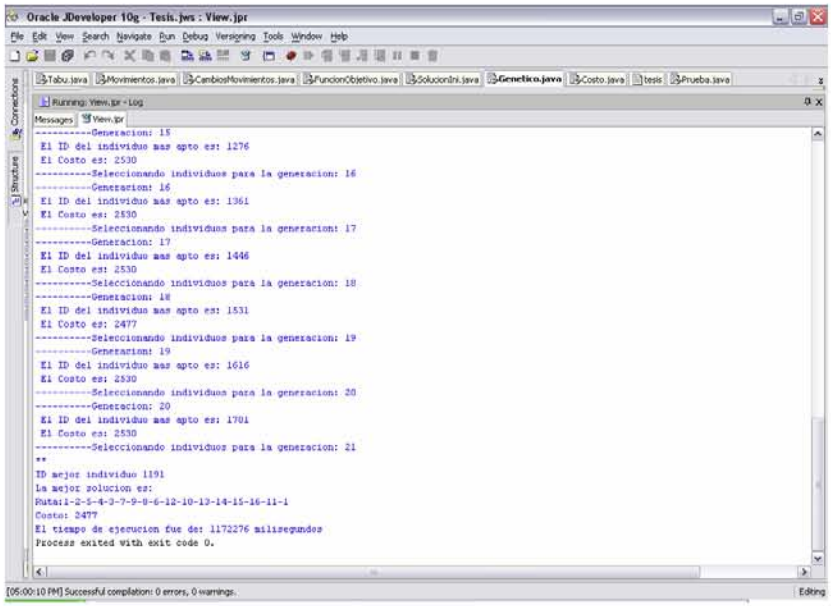
- Segunda ejecución



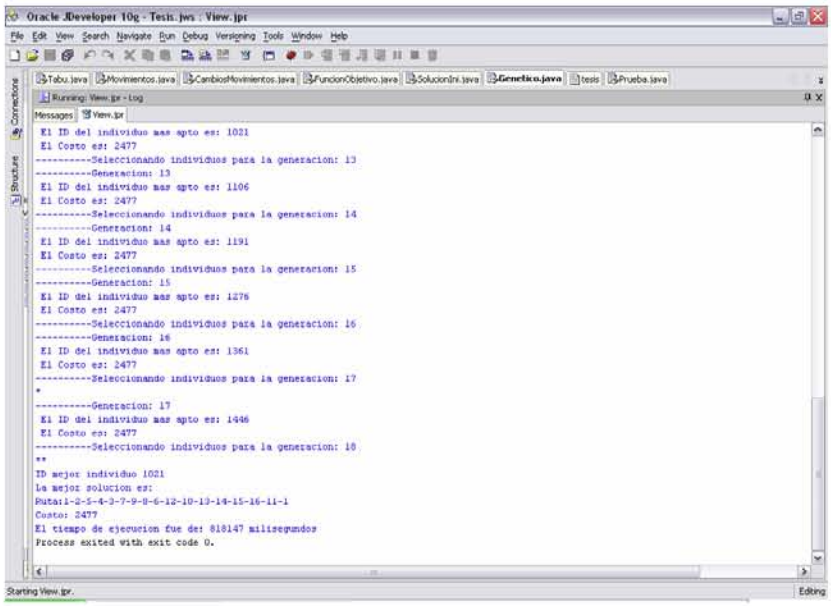
- Tercera ejecución



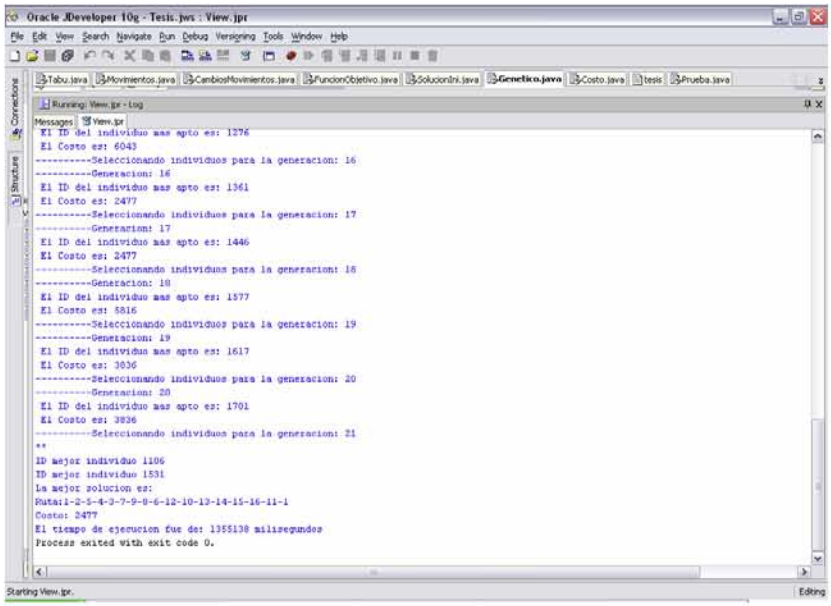
- Cuarta ejecución



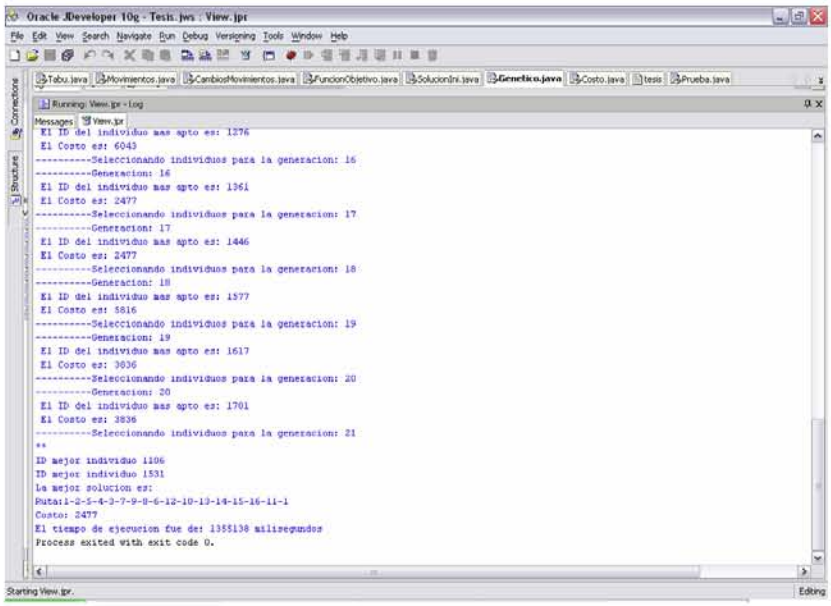
- Quinta ejecución



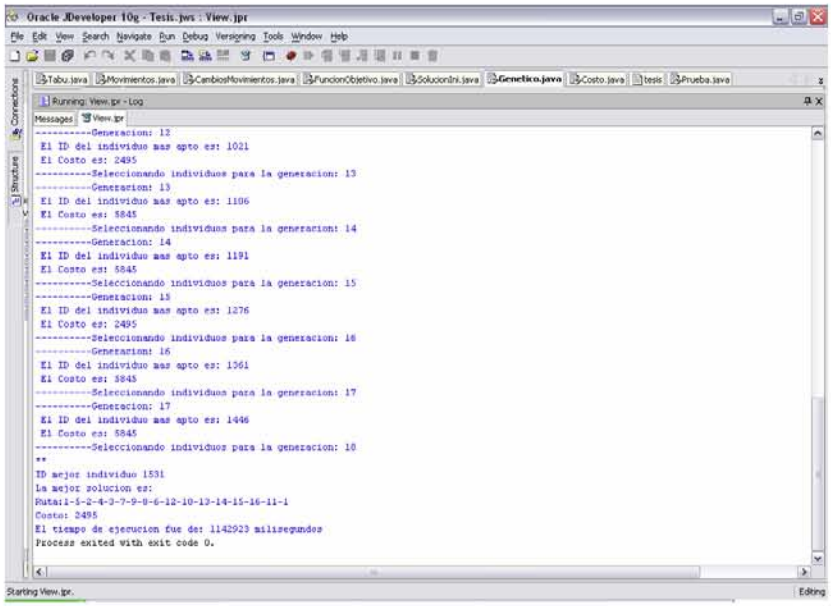
- Sexta ejecución



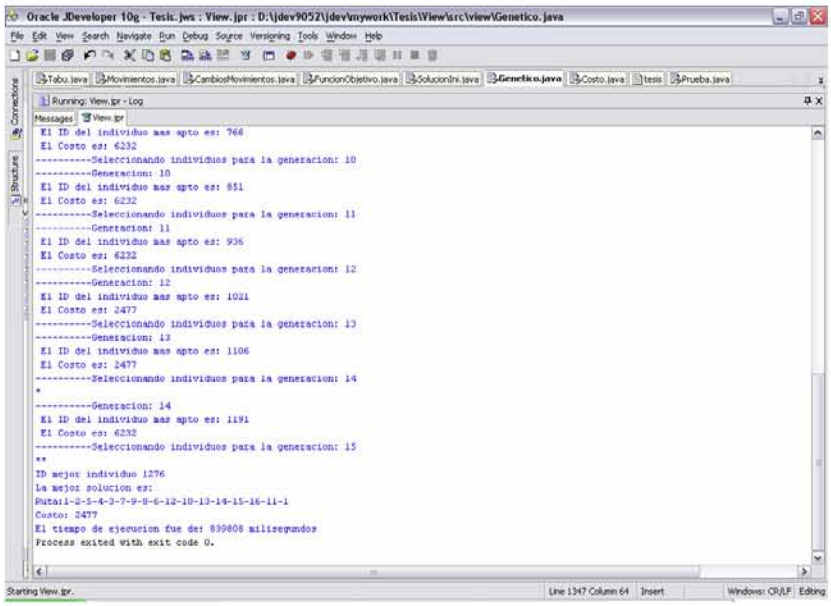
- Séptima ejecución



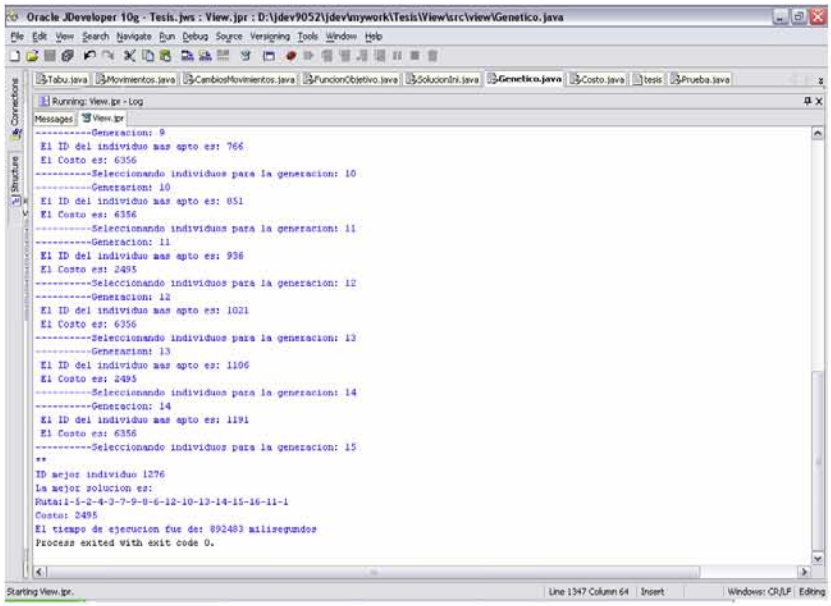
- Octava ejecución



- Novena ejecución



- Décima ejecución



ANEXO 1.

Características del Hardware y Software.

Para ejecutar el Programa se utilizó una PC con las siguientes características:

- Procesador Pentium IV.
- Velocidad 1.3 GHz.
- Memoria RAM de 512 MB.
- Disco duro de 80 GB.

El Sistema Operativo de la PC es:

- Microsoft Windows XP Profesional, version 2002.

El Lenguaje que se utilizó para implementar los algoritmos de la BT y del AG es:

- Java™ 2 SDK, Edición Estándar, versión 1.4.2 de Sun Microsystems.

El editor que se utilizó para escribir el código de los algoritmos es:

- Oracle® JDeveloper 10g, Versión 9.0.5.2

El almacenamiento de los datos obtenidos al ejecutar el algoritmo genético se guardó en una BD, el manejador que se utilizó fue el siguiente:

- Microsoft SQL Server 8.00.194

ANEXO 2.

Características de la programación de los algoritmos.

- El tiempo de ejecución de los algoritmos está dado en milisegundos.
- El paradigma de programación es orientado a objetos.
- Se utilizó el Framework OpenTS_10-exp10 para la programación del algoritmo de Búsqueda Tabú.

Glosario

1. Abiótico: lugar o medio en donde la vida no es posible.

2. Aleatorio: sometido al azar, a las leyes de la probabilidad.

3. Alfabeto: un conjunto finito de símbolos.

4. Algoritmo genético messy: variante del AG en donde la longitud de los cromosomas es variable.

5. Algoritmo híbrido: combinación de dos o más metaheurísticas para resolver diferentes tipos de problemas.

6. Arreglo circular: estructura de datos llamada cola circular, en la cual el primero elemento en entrar es el primero en salir.

7. Binario: sistema de numeración basado en el dos. Cuenta únicamente con los números 0 y 1 con los cuales se expresan los demás números.

8. Biótico: lugar que cuenta con las condiciones necesarias para que la vida exista.

9. Búsqueda exhaustiva: método que numera todas las soluciones posibles de un problema para resolverlo.

10. Búsqueda local: es un método de optimización para un espacio de búsqueda discreto.

11. Cardinalidad: número de miembros o elementos que tiene un conjunto.

12. **Circuito hamiltoriano:** es un circuito que pasa por cada nodo de un grafo exactamente una vez excepto el nodo inicial y el nodo final.

13. **Circuito:** es un paseo cerrado en donde ningún nodo se repite.

14. **Convergencia:** cuando un procedimiento llega a su fin.

15. **Convexo:** un conjunto es convexo si para dos puntos cualesquiera, que pertenezcan al conjunto, el segmento rectilíneo que los une también se encuentra dentro del conjunto.

16. **Corte de plano:** es un método que se utiliza para resolver problemas de programación entera. Los principios de este método se lo debemos a Gomory.

17. **Cruza de dos puntos:** fue la primera técnica de cruza de más de un punto propuesta por DeJong, en ésta se escogen dos posiciones en cada padre para efectuar la cruza.

18. **Cruza de un punto:** es una técnica de cruza propuesta por Holland en la cual se escoge una posición en los cromosomas padres para efectuar la cruza y generar a los descendientes.

19. **Cruza uniforme:** técnica de cruza propuesta por Ackley y Syswerda, en donde la cruza es de *n* puntos y el número de puntos de cruza no es fijo.

20. **Cualitativos:** que expresa o determina atributos o propiedades de un objeto o ser.

21. **Determinístico:** fenómenos explicados por el principio de causalidad.

22. **Diploide:** célula que contiene dos copias de cada cromosoma.

23. **Dominio:** en una función de un conjunto de pares ordenados (x, y), los valores admisibles para x es llamado dominio.

24. **Estadístico:** muestra aleatoria de cualquier función de variables aleatorias.

25. **Estancamiento Local:** es denominado también óptimo local.

26. **Estocástico:** aquello que es determinístico.

27. **Framework:** en la programación orientada a objetos es un conjunto de clases que proporcionan el esqueleto para resolver cierto tipo de problemas.

28. **Fenotipo:** rasgos específicos de un individuo, los cuales son observables.

29. **Genotipo:** información genética contenida en el genoma de un organismo.

30. **Grafo dirigido:** es un conjunto de V (vértices) y una relación de E (arcos) asociada a él, donde $E \subseteq VxV$

31. **Grafo no dirigido:** Un grafo es no dirigido si y solo si E es simétrico. Cuando se escribe (v_j, v_i) denota que el arco va del nodo v_j al nodo v_i por lo tanto se tiene que $(v_j, v_i) \in E \Rightarrow (v_i, v_j) \in E, \forall v_i, v_j \in E$.

32. **Grafo:** es la dupla (V, E), donde V es el conjunto de datos llamados nodos y E el conjunto de datos denominados arcos. Cada arco se encuentra identificado por un par (v_j, v_i).

33. **Inteligencia artificial:** estudio de cómo lograr que las computadoras realicen tareas que por el momento los humanos hacen mejor.

34. **Investigación de operaciones:** es la aplicación de la metodología científica a través de modelos matemáticos, primero para representar al problema real

que se quiere resolver en un sistema y segundo para resolverlo. El objetivo es la minimización de costos y la maximización de utilidades.

35. **Mapeo:** asociación de puntos en el espacio de búsqueda por medio de una función.

36. **Matriz asimétrica:** es aquella en la que los elementos $a_{ij} \neq a_{ji}$ para toda i, j.

37. **Matriz simétrica:** es aquella en la que los elementos $a_{ij} = a_{ji}$ para toda i, j.

38. **Memoria:** facultad intelectual por la cual se retiene y recuerda lo pasado.

39. **Método de ramificación y búsqueda:** método para resolver problemas de programación entera, el cual redondea y acota las variables enteras resultantes de la solución de los problemas lineales. Opera de una manera secuencial heurística que permite eliminar las soluciones que están alejadas de la óptima.

40. **Método del gradiente:** se basa en el gradiente de una función, el cual nos indica hacia dónde se dirige el máximo de dicha función o hacia dónde se encuentra el mínimo de esa función.

41. **Métodos híbridos:** son aquellos que combinan algoritmos evolutivos con métodos deterministicos para resolver problemas de optimización.

42. **Molécula:** partícula más pequeña de una sustancia que tienen todas las propiedades de dicha sustancia.

43. **Paseo cerrado: es** una secuencia finita de nodos y arcos que empieza y termina con nodos, en donde ningún arco se repite. El nodo inicial es el mismo que el nodo final.

44. **Permutación:** un ensayo sin reemplazo en donde importa el orden de los elementos.

45. **Poliedro:** sólido terminado con superficies planas.

46. **Problemas de calendarización:** son conocidos como problemas secuenciales ya que para poder realizar una tarea es necesario haber realizado otra previamente; el objetivo en este tipo de problemas es minimizar los costos.

47. **Programación dinámica:** procedimiento secuencial por medio del cual se resuelven ciertos problemas de optimización.

48. **Programación lineal:** procedimiento que resuelve problemas matemáticos lineales.

49. **Reproducción asexual:** proceso a través del cual se genera un nuevo organismo a partir de un padre.

50. **Reproducción sexual:** proceso a través del cual se crea un nuevo organismo a partir de dos padres. En este proceso dos gametos se fusionan para crear un organismo con diploide.

51. **Risco de Hamming:** mapeo inadecuado del espacio de búsqueda con el espacio de representación.

52. **Vecindad:** es la región del espacio de búsqueda cercana a un punto en particular en este espacio

53. **XOR:** representa al or exclusivo y se representa con el símbolo $\oplus$, tiene dos entradas y su valor es 0 si las entradas son iguales (es decir que sean ambas

0 ó 1) y verdadero si las entradas son diferentes (que una entrada sea 1 y la otra 0 o viceversa).

54. **Zona factible:** región del espacio de búsqueda que se encuentra delimitada por las restricciones que incluye el modelo del problema.

55. **Azar:** sin rumbo ni orden, casualidad, sin determinación previa.

FUENTES DOCUMENTALES

FUENTES BIBLIOGRÁFICAS

1. Aguirre Hernández, Rosalía. *Grados de libertad efectivos en algoritmos genéticos.* Tesis de maestría (Maestría en Investigación de operaciones), UNAM, Facultad de Ingeniería. México 1999.

2. Alexander S. Fraser and D. Burnell. *Computer Models in Genetics.* Editorial McGraw Hill, New York 1970.

3. Bäck Thomas. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Editorial Oxford University, New York 1996.

4. Bello Cabanas Myriam. *Introducción a la técnica de los algoritmos genéticos.* Tesis de licenciatura (Ingeniero en Computación), UNAM, Escuela Nacional de Estudios Profesionales Aragón. México 1996.

5. Cobos Silva Sergio. *La técnica de la búsqueda tabú y sus aplicaciones.* Tesis de doctorado (Doctorado en Investigación de Operaciones), UNAM, Facultad de Ingeniería. México 1994.

6. Coello Coello Carlos. *Introducción a la computación evolutiva.* Editorial CINVESTAV-IPN, México 2001.

7. Díaz Adenso, Glover Fred, Ghaziri Asan, González J., Laguna Manuel, Moscato Pablo y Tseng Fan. *Optimización heurística y redes neuronales.* Editorial Paraninfo, España 1996.

8. Duc Truong Pham y Dervis Karaboga. *Intelligent optimization techniques: genetic algorithms, tabu search, simulated annealing and neural network.* Editorial Springer, New York 2000.

9. Esponda Darlington Carlos Fernando. *Periodo de vida y extinción en algoritmos genéticos*. Tesis de licenciatura (Ingeniero en Computación), Instituto Tecnológico Autónomo de México, Escuela de ingeniería, México 1995.

10. Falkenauer Emanuel. *Genetic algorithms and grouping problem.* Editorial John Wiley and Sons, England 1998.

11. Freeman James y Skapura David. *Redes neuronales algoritmos, aplicaciones y técnicas de programación.* Editorial Addison-Wesley, Estados Unidos de América 1993.

12. Galaviz Casas José de Jesús. *Algoritmos genéticos autoadaptables*. Tesis de maestría (Maestría en Ciencias de la Computación), UNAM, Colegio de Ciencias y Humanidades, Unidad Académica de los Ciclos Profesional y de Posgrado. México 1997.

13. Garduño Rivera Alan. *Simulación de procesos evolutivos mediante algoritmos genéticos.* Tesis de licenciatura (Licenciado en Matemáticas Aplicadas y Computación), UNAM, Escuela Nacional de Estudios Profesionales Acatlán. México 1998.

14. Glover Fred y Laguna Manuel. *Tabu search, modern heuristic techniques for combinatorial problems*. Editorial Oxford, New York 1993.

15. Glover Fred y Laguna Manuel. *Tabu search.* Editorial *Kluwer* Academic Publishers, Boston 1997.

16. Goldberg David. *Genetic Algorithms in search, optimization and machine learning.* Editorial Addison-Wesley, United States of America 1989.

17. Gutiérrez García Juan Jesús. *Métodos de optimización con restricciones usando algoritmos genéticos y funciones de penalización: análisis comparativo.* Tesis de maestría (Maestría en Ciencias de la Computación), UNAM, Colegio de Ciencias y Humanidades, Unidad Académica de los Ciclos Profesional y de Posgrado. México 2001.

18. Gutin G. y Punnen A. *The traveling salesman problem and its variations.* Editorial Kluwer Academic Publishers, Netherlands 2002.

19. Holland John H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

20. Hopcroft John E. y Ullman Jeffrey D. *Introducción a la teoría de autómatas, lenguajes formales y computación.* Editorial Compañía Editorial Continental, México 1993.

21. Iglesias León Juan Raymundo. *Optimización basada en algoritmos genéticos: un enfoque orientado a objetos.* Tesis de licenciatura (Licenciado en Informática), UNAM, Facultad de Contaduría y Administración. México 1994.

22. John R. Koza. *Genetic programming. On the programming of computers by means of natural selection.* Editorial The MIT Press, Cambridge, Massachusetts 1992.

23. Kuri Morales Ángel y Galaviz Casas José. *Algoritmos genéticos.* Editorial IPN, UNAM y Fondo de Cultura Económica, México 2002.

24. Lance Chambers. *The practical handbook of genetic algorithms: applications*. Editorial Chapman and Hall/CRC, United States of America 2001.

25. Lawler E., Lenstra J., Rinnooy A. y Shmoys D. *The traveling salesman problem.* Editorial John Wiley and Sons, Great Britain 1990.

26. Lawrence Davis. *Handbook of genetic algorithms.* Editorial Van Nostrands Reinhold, New York 1991.

27. Matt Ridley. *Genoma,* La autobiografía de una especie en 23 capítulos. Editorial Taurus Pensamiento, España 2000.

28. Memorias del primer Congreso de Computación Evolutiva (COMCEV'03), CIMAT, Guanajuato 2003.

29. Memorias del Simposium Internacional de Computación. *La computación: investigación, desarrollo y aplicaciones.* CINVESTAV-IPN, México 1998.

30. Michalewicz Zbigniew y Fogel David. *How to solve it: modern heuristics.* Editorial Springer, Germany 2002.

31. Michalewicz, Zbigniew. *Genetic Algorithms + Data Structures = Evolutionary Programs.* Editorial Springer, United States of America 1996.

32. Mora Vargas Jaime. *Algoritmos genéticos inteligentes y su aplicación a la investigación de operaciones.* Tesis de doctorado (Doctorado en Ingeniería), UNAM, Facultad de Ingeniería. México 2000.

33. Nils J. Nilsson. *Inteligencia artificial una nueva síntesis.* Editorial McGraw Hill, España 2001.

34. Prawda Juan. M*étodos y modelos de investigación de operaciones vol. 1 modelos deterministicos*. Editorial Limusa, México 2002.

35. Rayward-Smith, Osman, Reeves y Smith. *Modern heuristic search methods.* Editorial John Wiley and Sons, England 1996.

36. Reeves C. R. *Modern heuristic techniques for combinatorial problems*. Editorial John Wiley and Sons, United States of America 1993.

37. Schrijver Alexander. *Theory of linear and integer programming.* Editorial John Wiley and sons, England 1986.

38. Stuart Russell y Peter Norvig. *Inteligencia artificial un enfoque moderno.* Editorial Prenctice Hall Hispanoamericana, México 1996.

39. Vose Michael D. *The simple genetic algorithm: foundations and theory (complex adaptative systems).* Editorial MIT Press, Cambridge 1999.

40. W. D. Cannon. *The wisdom of the body*. Editorial Norton and Company, New York 1932.

41. Walpole R. y Myers R. *Probabilidad y estadística para ingenieros.* Editorial Interamericana, México 1987.

42. Beyer Ruiz María Emilia. *Gen o no gen, el dilema del conocimiento genético.* Editorial Lectorum, México 2002.

FUENTES HEMEROGRÁFICAS

1. Coello Coello Carlos, Hernández Arturo, y P. Buckles Bill. *Estrategias evolutivas: la versión alemana del algoritmo genético, parte I.* Soluciones avanzadas, Octubre 1998.

2. Coello Coello Carlos, Hernández Arturo, y P. Buckles Bill. *Estrategias evolutivas: la versión alemana del algoritmo genético, parte II.* Soluciones avanzadas, Diciembre 1998.

3. D. Beasley, D. R. Bull y R. R. Martin. *An overview of genetic algorithms: part 1, fundamentals.* University of Wales College of Cardiff, 1993.

4. D. Beasley, D. R. Bull y R. R. Martin. *An overview of genetic algorithms: part 21, research topics.* University of Wales College of Cardiff, 1993.

5. Glover Fred. *Tabu search: part I*, ORSA Journal of Computing, Vol. 1, No. 3, pp. 190-206, 1989.

6. Glover Fred. *Tabu search: part II,* ORSA Journal of Computing, Vol. 2, No. 1, pp. 4-32, 1990.

7. Mathison Turing Alan. *Mind.* Computing machinery and intelligence 59:94-1001, 1950.

8. Palacios Roji García Agustín y Palacios Roji García Joaquín. *México Atlas Turístico de Carreteras 2003.* Guía Roji S. A de C. V 2003.

REFERENCIAS DE INTERNET

1. Darell Whitley y Soraya B. Rana (1997). *Representation, Search and genetic algorithms*. Obtenido de la red mundial el 3 de junio del 2003: http://www.cs.colostate.edu/~genitor/Pubs.html

2. Darell Whitley, Timothy Starkweather y Daniel Shaner (1991). *The traveling salesman and sequence scheduling: quality solutions using genetic edge recombination*. Obtenido de la red mundial el 20 de septiembre del 2003: http://www.cs.colostate.edu/~genitor/Pubs.html

3. Darell Whitley, (1993, 10 de Noviembre). *A genetic algorithm tutorial*. Obtenido de la red mundial el 25 de febrero del 2003: http://www.cs.colostate.edu/~genitor/Pubs.html

4. Darrell Whitley, (2002). G*enetic algorithms and Evolutionary computing*. Obtenido de la red mundial el 3 de junio del 2003: http://www.cs.colostate.edu/~genitor/Pubs.html

5. García F. y Laguna M. *Optimización: conceptos fundamentales y tendencias actuales.* Obtenido de la red mundial el 21 de mayo del 2003: http://www.terra.es/personal/faustopedro.gar/files/A5.pdf

6. Glover F. y Hanafi S. *Tabu search and finite convergence.* Obtenido de la red mundial el 15 de octubre del 2003: http://spot.colorado.edu/~glover/Recentpapers.html

7. John Dzubera y Darell Whitley (1994). *Advance correlation analysis of operators for the traveling salesman problem*. Obtenido de la red mundial el 12 de agosto del 2003: http://www.cs.colostate.edu/~genitor/Pubs.html

8. Keith Mathias y Darell Whitley (1992). G*enetic operators, the fitness landscape and the traveling salesman problem*. Obtenido de la red mundial el 20 de marzo del 2003: http://www.cs.colostate.edu/~genitor/Pubs.html

9. Laguna M. y González J. (2000). *Multi -start and strategic oscillation methods- principles to exploit adaptative memory.* Obtenido de la red mundial el 15 de octubre del 2003: http://bus.colorado.edu/faculty/glover/multistart.pdf

10. Marti Rafael. *Tabu search.* Obtenido de la red mundial el 15 de octubre del 2003: http://www.uv.es/~rmarti/tabu.html

11. Parada V., Solar M. y Nasra V. *El problema de localización de concentradores. Resolución mediante búsqueda tabú.* Obtenido de la red mundial el 15 de octubre del 2003: http://www.usach.cl/Papers/Congreso/45Congreso.pdf

12. Ponce J., Solis G. Y Ulfe L. *Investigación de operaciones.* Obtenido de la red mundial el 7 de abril del 2003: http://www.lista-ioper.rcp.net.pe/OptCombinatoria.pdf