UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE CIENCIAS EXACTAS

Algoritmos genéticos y su aplicación en optimización de redes

Tesis de Maestría

Autor: Ing. José Luis Hernández
Director: MSc. Raúl Gallard
Co Director: Lic Javier Díaz

A mi familia
Los verdaderos dueños de mi tiempo
A mis padres, que me lo regalaron

Agradecimientos

Umberto Eco escribió que es de mal gusto agradecer en una tesis al director. Es una opinión que no debería pasar por alto teniendo en cuenta quien la emite. No declararé aquí, entonces, mi profundo reconocimiento a mis directores. No diré que sin su guía todo hubiera sido más difícil, ni que lo escrito en estas páginas tiene mucho de ellos, no por conformarlos después de cada revisión sino como documentación de lo aprendido. No voy a expresar públicamente mi gratitud para con el Profesor Raúl Gallard y el Profesor Javier Díaz. Ellos ya lo saben.

A la Universidad Nacional de Río Cuarto, como siempre y desde siempre. Gracias a su apoyo desde mi Facultad de Ingeniería y la Escuela de Posgrado puedo escribir estas líneas.

A mis compañeros: muchas veces por el apoyo, y muchas otras por soportar mi *seteo* en modo parco.

Índice

Síntesis

El presente trabajo realiza un análisis de los algoritmos genéticos que se diseñan para resolver problemas de optimización, que involucran no sólo funciones objetivo continuas y derivables sino aplicados a funciones con puntos de discontinuidad o de no derivabilidad. Asimismo se aplican a problemas de secuenciación en donde el espacio de soluciones está determinado por un conjunto de secuencias una de las cuales es la óptima, presente en muchos problemas de optimización en redes. En este tipo de problemas están presente las permutaciones y su representación intrínseca ha constituido un reto para los algoritmos genéticos.

Se presenta un análisis de diferentes representaciones de los cromosomas que pueden ser utilizados en la resolución de los distintos problemas y del funcionamiento de los AG en los distintos casos, representaciones y parámetros que los gobiernan.

Se seleccionan algunos ejemplos de aplicaciones de algoritmos genéticos en redes en los cuales se distinguen diferentes tipos de problemas y de aportes de los AG en cada ejemplo. Se hace hincapié en los operadores genéticos seleccionando para cada caso los más apropiados. Posteriormente se encara el diseño e implementación de un AG. utilizando el problema del viajante para un testeo preliminar de los AG y finalmente se aplica este diseño en uno de los ejemplos seleccionados. Conjuntamente se implementan algunas técnicas clásicas para contrastar los resultados.

Finalmente se realiza una interpretación de los resultados justificando la exploración de estas técnicas como una alternativa válida en problemas de optimización de redes de datos, analizando las ventajas y desventajas de estos métodos frente a técnicas clásicas.

Introducción

Los algoritmos genéticos (AG) se incluyen dentro de las técnicas de Computación Evolutiva. La filosofía que subyace es la de intentar simular los mecanismos de la evolución natural para solucionar los problemas de optimización que resultan de la adaptación de los seres vivos al medio ambiente. La simulación tiene que ver por establecer analogías entre los AG y los mecanismos que provee la naturaleza buscando adaptar soluciones potenciales (cromosoma o individuos) a un medio en función de su *fitness* o ajuste (dado por el mérito del individuo a la población).

El funcionamiento de los AG se basa en la modificación iterativa de una población de individuos candidatos a solución mediante el uso de operadores genéticos. La teoría en la cual se apoyan los AG está explicitada en Goldberg [1] y Michalewicz [2].

Los problemas de optimización de funciones reales, sin la exigencia necesaria de continuidad ni derivabilidad pueden ser resueltos por los AG utilizando, entre otras, una representación de las soluciones potenciales en forma de cadenas de bits en cada uno de los cromosomas que conforman la población. Los operadores genéticos clásicos funcionan correctamente dado que fueron creados a tales efectos.

Los problemas de secuenciación, presentan una dificultad adicional dado que su representación no se ajusta, a priori, a los AG. Sin embargo un adecuado rediseño de la representación y de dichos operadores hacen que puedan solucionarse problemas como el viajante de comercio, el problema de la mochila, el coloreo de grafos , problemas de ruteo en redes, etc.

El objetivo principal de este trabajo es investigar el comportamiento de los AG en problemas de optimización de redes y presentarlo como una alternativa a los algoritmos ya existentes, para lo cual se presenta un estudio de los mismos y aplicaciones a la resolución de problemas de secuenciación con soluciones conocidas u obtenidas por otros métodos con el fin de comparar los resultados.

Sin embargo, a pesar de los resultados obtenidos en distintas publicaciones acerca del tema, no existe una consolidación del tema y, para alcanzarla, resta profundizar en

ciertos aspectos importantes.

Objetivo general

La aplicación de los AG a problemas de optimización en redes.

Objetivos específicos

Reportar el comportamiento de los AG diseñados para resolver problemas de optimización en redes.

Presentar un análisis de las representaciones utilizadas en AG

Presentar un análisis de los operadores utilizados por los AG.

Trabajos relacionados

Algunos de los resultados presentados en este trabajo o relacionados con el tema de tesis han sido expuestos por el autor en distintas reuniones científicas citadas en las referencias.

Estructura del trabajo

La tesis se divide en seis capítulos y un anexo organizados de la siguiente manera:

El capítulo I trata el tema de los algoritmos genéticos en general. Sus definiciones principales y su estructura funcional a nivel global. Se analiza la forma de representación de cromosomas y se aborda una descripción general de los componentes de un AG.

El capítulo II describe con más en detalle un algoritmo genético desde el punto de vista funcional mostrando como pueden ser utilizados los operadores de cruzamiento y mutación, considerando representación de cromosomas a través de cadenas de bits. Se considera también el elitismo y se analizan distintos tipos de cruzamiento para cromosomas representados por cadenas binarias.

El capítulo III realiza una presentación de aplicaciones de los AG en problemas de optimización en redes, en las cuales se muestran los distintos operadores y estrategias utilizadas en cada caso, como así también la función de los AG como optimizadores y como predictores. Se analiza el problema del balance dinámico de carga en un sistema distribuido, la asignación de tareas en un sistema paralelo, la obtención de una ruta óptima en una red. Este último problema se tomara para la prueba del AG cuyo diseño y análisis son la razón de ser de los capítulos siguientes.

El capitulo IV cubre el problema del TSP (*Traveling Salesman Problem*) dado que configura un buen problema de testeo de los AG para la resolución de problemas de secuenciación en general. Se realiza un estudio de los distintos operadores genéticos que pueden utilizarse, teniendo en cuenta que la representación de los cromosomas no será de cadenas binarias sino de vectores enteros. Así se analizan distintos tipos de representación de cromosomas y de operadores. Finalmente se ofrecen los resultados de la experimentación con dichos algoritmos para resolver problemas de TSP con distintos parámetros.

El capítulo V cubre el análisis y resolución del problema de la ruta de mínimo costo en

una red. Se desarrolla una técnica clásica y un AG y se analizan los resultados obtenidos en cada caso. Se incorpora en este capítulo el cruzamiento por arcos como una mejora a los operadores de cruzamiento analizados en el capitulo IV.

Finalmente las conclusiones a las que se arriban en la presente tesis, los trabajos futuros y recomendaciones basados en ella y sus limitaciones están plasmadas en el capítulo VI.

CAPÍTULO I

1 ALGORITMOS GENÉTICOS

1.1 Introducción

Los algoritmos genéticos (AG) son algoritmos de búsqueda basados en los mecanismos de la selección natural y la genética natural (Goldberg [1]). Los seres vivos buscan la mejor adaptación al medio en el cual deben desempeñarse mediante algún tipo de "medición" de dicha adaptabilidad, transmitiendo a su descendencia las características que aportan a que el individuo sea más apto y eliminando por selección natural aquellas características que son adversas al desempeño del individuo en su contexto. Así los individuos más aptos sobreviven mientras que los de peor desempeño se extinguirán al cabo de algunas generaciones.

Los AG simulan esta situación, sólo que millones de años se ven reflejados en milisegundos, y con un modelo mucho más reducido y simplificado que el utilizado por la naturaleza. Un individuo es una propuesta de solución a un problema determinado, y su aptitud se mide con relación a que tan buena solución al problema proporciona el individuo. Una población es un conjunto de individuos que se aparearán para generar descendencia que tendrá características similares a la de sus padres. Luego, padres e hijos competirán entre sí sobreviviendo los mejores y pereciendo los de menor adaptación al medio. Al cabo de varias generaciones la población estará compuesta con mejores individuos de los cuales podrá obtenerse la mejor solución al problema.

1.2 Definiciones

La terminología utilizada en AG deriva, en cierta medida de la correspondiente a la genética natural. Así se redefinirán conceptos como individuo, cromosoma, gen, genotipo, fenotipo etc. que van a devenir en las estructuras de datos que se utilizaran en la implementación de los AG.

Una **población** es un conjunto de **individuos** que generará descendencia y competirán

entre sí posteriormente. Un **individuo** es un conjunto de **genes** que lo caracteriza. Un **gen** puede tomar valores determinados pertenecientes a un conjunto de datos posibles. Dichos valores son llamados **alelos.**

Desde el punto de vista de estructura de datos podemos representar una población como un arreglo de individuos (cromosomas) cada uno de los cuales estará representado por un valor real o una string de bits. En función de la representación elegida, distintas funciones proveerán las conversiones que sean necesarias.

1.3 Estructura de un AG simple

Como cualquier estructura de un algoritmo evolutivo, la estructura básica de un AG comienza con una generación aleatoria de una población inicial con las potenciales soluciones. En una iteración cualquiera *t* el vector de cromosomas $P(t)=\{x_1^t, ..., x_n^t\}$ es evaluado de acuerdo a una función de ajuste o *fitness* para cada x_i^t. Luego se forma una nueva población, $P(t+1)$ con los individuos que obtuvieron mejor ajuste. La nueva población es sometida a alteraciones mediante operadores como el **cruzamiento** y la **mutación** para formar nuevas soluciones que volverán a competir entre ellas mediante la aplicación de la función de ajuste, hasta que, después de la aplicación de un criterio de detención el algoritmo finalice.

```
*     AG simple.
      t=0;
      iniciar P(t);
      hacer_mientras t<tmax
            t=t+1;
            selección (P(t)) desde P(t-1);
            cruza(P(t));
            mutación(P(t));
      fin_de_hacer
      fin
```

Fig. 1.1. Estructura de un AG simple con criterio de detención por número de iteraciones

Un AG para un problema particular, debe tener los siguientes componentes [Michalewicz]:

- Una representación de las soluciones potenciales
- Una forma de crear la población inicial
- Una función de evaluación (para medir que tan buena es cada solución)

- Operadores genéticos que alteren la composición de los hijos.
- Parámetros generales que usa el AG (tamaño de la población, etc.).

1.4 Representación de soluciones potenciales

Hay una gran dependencia entre la forma de representación y el problema a resolver. Si, por ejemplo, el AG se utiliza para optimizar una función simple, de una sola variable en un dominio fijado por un intervalo, se puede usar una representación en un vector binario que represente los valores reales que puede tomar la variable independiente. Es este el caso de determinar el mínimo (o máximo) de una función *f(x)* donde x ε [a, b]. La población de soluciones potenciales estará formada por un conjunto de vectores binarios donde cada uno de ellos almacenará una solución x_i. Para determinar los valores binarios que tomará cada x_i se divide el dominio [a, b] en la cantidad de intervalos que requiera la precisión deseada a fin de determinar la cantidad de bits que se utilizará. Posteriormente se asigna a x=a el valor binario 000... 000_B y a x=b el valor 111... 111_B. Valores intermedios se obtienen por interpolación lineal:

$$x_R = a + x_B . \frac{(b-a)}{max_b}$$

donde:

x_R es el valor real de la solución potencial.

x_B es el valor en binario representado en k bits.

max_B es el máximo número representable con k bits.

a es el mínimo valor real que puede tomar x.

b es el máximo valor real que puede tomar x.

Si el problema a resolver tiene que ver con encontrar un trayecto o ruta que minimice costos (por ejemplo el TSP), la población estará formada por caminos alternativos o soluciones potenciales: cada individuo de la población representará una ruta que una nodos de una red de *k* nodos posibles. Una población de este tipo se ve representada en la figura 1.2.

1 2 5 6 9 7 4 3 8
1 5 6 8 3 7 4 9 2
.....
.....
.....
.....
1 6 2 7 3 8 4 9 5
1 3 5 7 9 2 4 6 8
1 2 3 4 5 6 7 8 9

Fig.1.2. Una población de rutas

Como puede apreciarse, el problema de la representación es muy importante. Se han presentado dos formas de representar cromosomas. Sin embargo existen otras; en cada caso las mismas son más aptas para resolver un determinado tipo de problema. Pueden citarse las siguientes representaciones de cromosomas:

- Cadena de bits
- Vectores de números enteros
- Vectores de números reales
- Matrices
- Estructuras de datos

1.5 Creación de una población inicial

De manera similar al problema de la representación, la creación de una población inicial tiene que ver con el tipo de problema que se quiere resolver. En los dos ejemplos dados en los puntos anteriores, puede verse claramente que una generación de soluciones potenciales en el caso de representación por vectores binarios, puede realizarse la generación de una población inicial por medio de una generación aleatoria adecuada. En el primer caso la única restricción tiene que ver con los límites impuestos al espacio factible por lo que basta la generación aleatoria dentro del intervalo válido de la variable independiente. Si la función fuera multivariable, cada una de ellas tendrá acotados sus limites superior e inferior.

Un párrafo aparte merece la resolución de problemas de optimización que incluyen

restricciones que no tienen que ver con los límites de los intervalos entre los cuales se les permite moverse a las variables independientes, sino que delimitan el espacio de soluciones factibles dentro del dominio acotado por los límites de validez de las variables. Puede verse en el ejemplo mostrado en la figura 1.3 que dichos límites son diferentes. Mientras las variables **x** e **y** podrían moverse dentro del área rectangular de limitadas por los valores x1 y x2 para x y los valores y1 e y2 para la variable y, el espacio de soluciones factibles es el área marcada en la figura que cumple la restricción

$$ax+by=K$$

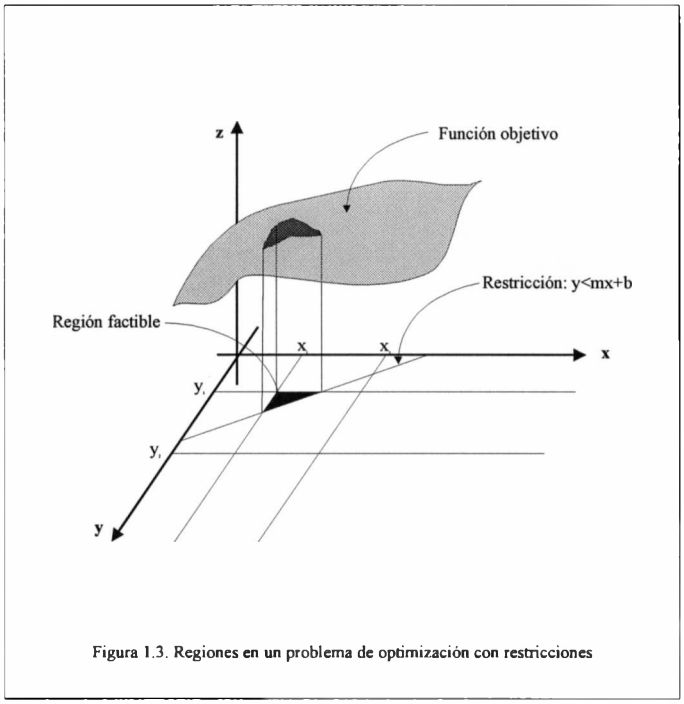


Figura 1.3. Regiones en un problema de optimización con restricciones

En este caso se suele aplicar penalizaciones a la solución [1] [2] que puede ir desde la pena de muerte (desechar la solución de plano) hasta la aplicación de funciones más complejas que,acorde con el grado de violación de la restricción, disminuyen el fitness del individuo. De esta manera se preservan algunos individuos que, si bien no son factibles, pueden poseer material genético que ayude en las futuras generaciones.

En el caso de TSP o en cualquier otro donde lo que representa cada individuo de una población es una ruta, aparecen restricciones relacionadas con la validez de la ruta. El algoritmo de inicialización deberá garantizar en este caso la generación de rutas válidas.

1.6 Función de evaluación

Una vez representadas la variables y generada la población inicial, es necesario ponderar cuan buenos son los candidatos (cada uno de los individuos de la población). La bondad de los mismos se refiere a cuan buena es la solución que proponen y si se intenta minimizar una función, costo en este caso, los individuos de menor costo serán los más apropiados. El valor de la función objetivo a minimizar, el costo de cada ruta considerada, en general el ajuste o *fitness* da la medida de la bondad de la solución.

La evaluación de cada individuo en función de su adaptabilidad al medio le otroga un mérito o competencia que, obien le ayudará a sobrevivir en las próximas generaciones o bien significará su muerte como resultado de la competencia con sus congéneres.

1.7 Ciclo de evolución

Puede observarse en el pseudocódigo la presencia de un lazo con un criterio de detención que es el encargado de poner fin a la evolución de la población. Cada ciclo representa una nueva generación. Este ciclo está controlado por el criterio de detención adoptado. En la bibliografía se citan como válidos los siguientes:

- Detener después de N iteraciones
- Detener cuando el *fitness* medio de la población cae dentro de una banda predefinida.
- Detener cuando la diversidad poblacional es baja
- Otros

En general, los criterios de detención pueden clasificarse en dos grupos:

- Condiciones independientes del proceso de evolución
- Condiciones atadas a dicho proceso

Dentro del ciclo de evolución pueden observarse tres acciones concretas:

Seleccionar

Cruzar

Mutar

La selección implica determinar cuales cromosomas serán considerados para procrear y

así formar la siguiente generación. Es importante aquí el criterio con que esa solución es hecha. Una forma de selección consiste en asignar a cada cromosoma una probabilidad de ser seleccionado, proporcional a su función de ajuste, aunque este no es el único criterio utilizado.

Seleccionar es escoger individuos para entrar en una etapa de modificación de la población que viene de la mano de los llamados operadores genéticos (como se verá más adelante).

1.8 Operadores genéticos que alteren la composición de los individuos

Los individuos de una población *i* deben generar descendencia transmitiendo sus características genéticas a sus hijos. Para ello deberá contarse con operadores que permitan dicha transformación y que necesariamente estarán influidos por el tipo de representación y de problema a resolver. Dos operadores utilizados en algoritmos genéticos son los operadores de cruzamiento y mutación, que si bien se verán en detalle más adelante, en forma concisa puede considerarse que el cruzamiento permite generar hijos a partir de padres seleccionados y la mutación consiste en realizar una modificación, normalmente pequeña, en el cromosoma de un individuo.

1.9 Parámetros generales que usa el AG (tamaño de la población, etc.).

Distintos parámetros y condiciones iniciales son necesarios para el ajuste de un algoritmo genético. La velocidad de convergencia se ve afectada entre otros por la cantidad de individuos en una población, la probabilidad de cruzamiento, la probabilidad de mutación, el elitismo, etc.

CAPÍTULO II

2 DESCRIPCIÓN DE UN ALGORITMO GENÉTICO

Se analizan aquí las características del algoritmo genético mostrado en la figura 1.1 Como se puede apreciar, el algoritmo hace evolucionar sucesivas generaciones de individuos hasta obtener la solución óptima. La creación de una nueva generación a partir de los operadores genéticos es un punto muy importante e involucra a los operadores de cruzamiento, mutación y selección.

El operador **selección** intenta simular la selección natural de los individuos vivos. Este operador selecciona los padres de la próxima generación (los que sobrevivirán) **en función de su *fitness***. De esta forma, los mejores individuos pasarán a la próxima generación sus características genéticas. La clave de la selección es que los más aptos tengan mayor probabilidad de recombinarse y contribuir con sus hijos a la próxima generación.

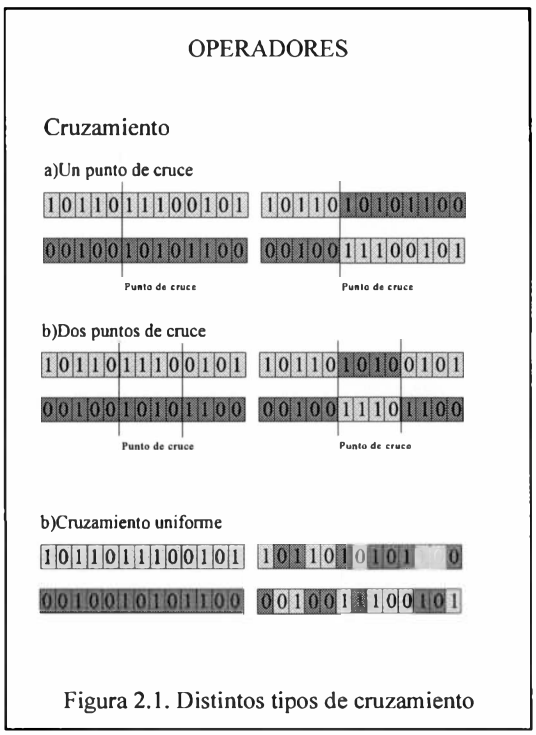


Figura 2.1. Distintos tipos de cruzamiento

Una vez seleccionados los padres, se aplica el operador de **cruzamiento** el cual recombina los cromosomas de los progenitores para generar hijos con características

similares. La figura 2.1 muestra algunas formas de cruzamiento que pueden aplicarse, utilizando un sistema de representación de los cromosomas en string de bits.

Formalmente el cruzamiento podría definirse como una función

$$C : S^q \rightarrow S^r \qquad C(x_1^t,...,x_q^t) \rightarrow C(x_1^{t'},...,x_r^{t'})$$

donde

q es la cantidad de padres

r es la cantidad de hijos

x_i^t es el cromosoma *i* de la población P^t antes del cruzamiento

x_i^t es el cromosoma *i* de la población P^t después del cruzamiento

Finalmente, al igual que lo que ocurre en la naturaleza, con una probabilidad muy baja se aplica el operador de **mutación**. Éste permite alterar la información genética de los hijos. Dicha alteración en general debe ser pequeña y tiende a evitar la convergencia prematura a mínimos locales. La figura 2.2 muestra el efecto de la mutación en un cromosoma.

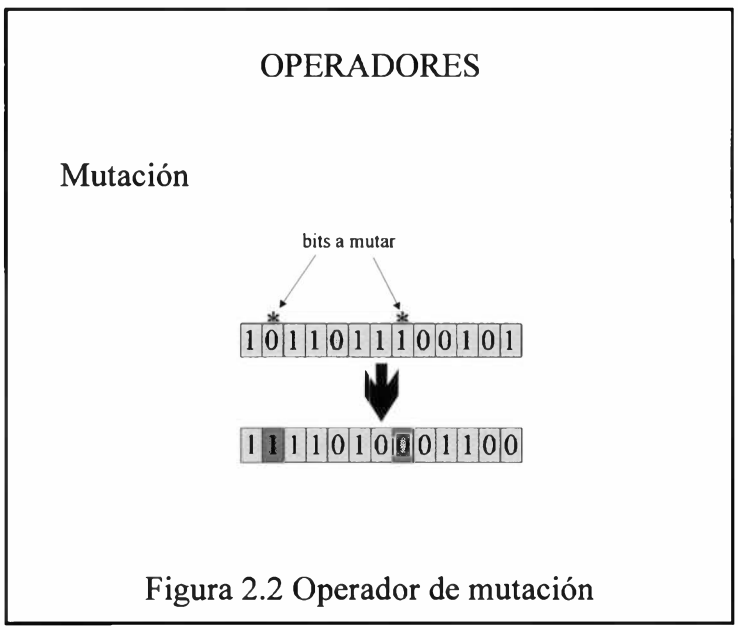


Figura 2.2 Operador de mutación

El punto crucial de un algoritmo genético, parece ser la generación de una nueva población. El algoritmo de la figura 2.3 resume el proceso.

```
*       Crear nueva población.
        hacer_mientras P(t) no este llena
                seleccionar padres p1 y p2 de P(t);
                generar hijos ←cruza(p1,p2,probcruza);
                producir mutación(hijos,probmuta);
                Agregar Hijos a P(t+1);
        fin_de_hacer
        fin
```

Fig. 2.3. Generación de una nueva población P_{t+1} desde una generación P_t

2.1 Selección de padres

Se mencionó que el criterio de selección de progenitores tiene que ver con la función de ajuste. Se puede lograr este objetivo a través de varios mecanismos. Se describe aquí el método de la ruleta, propuesto por Holland en sus trabajos más tempranos en la cual la probabilidad de ser padres es proporcional al *fitness* de cada individuo.

Una ruleta honesta, consta de ranuras en una de las cuales caerá la bolita. Eso asegura que todos los números tienen la misma probabilidad de ser elegidos. Pero si el ancho de cada ranura fuera diferente, los números que identifican a las ranuras más anchas tendrán mayor probabilidad de ser elegidos que los que pertenecen a ranuras estrechas. Entonces si lo que se busca es que la probabilidad de que un cromosoma sea elegido como padre sea proporcional a su *fitness*, el ancho de cada ranura será justamente proporcional a la medida de aptitud.

En esta particular ruleta el ancho de cada ranura es el aporte del *fitness* de un cromosoma *i*, (F_i) con relación a la suma de los *fitness* de todos los individuos de la población (FT). Si la población tiene N individuos entonces

$$FT = \sum_{i=1}^{N} F_i$$

y la probabilidad de que un cromosoma *i* sea seleccionado estará dada por

$$P_i = \frac{F_i}{FT}$$

Entonces la simulación del lanzamiento de la bolita se consigue generando un numero aleatorio entre 0 y FT y utilizando los valores de *fitness* acumulados se determina en

que ranura cae y por lo tanto cual individuo es seleccionado para procrear. El siguiente ejemplo podrá aclarar el procedimiento: sobre una población supuesta de 5 individuos se tabulan sus *fitness*, el aporte de ellos al total y los acumulados.

INDIVIDUO (CROMOSOMA)	FITNESS	APORTE AL TOTAL	ACUMULADOS
10001010101	100	0.186	0.186
00101000101	135	0.251	0.437
11010100010	40	0.074	0.511
00010101000	200	0.372	0.883
11011000111	62	0.115	1.000
TOTAL FITNESS	537	1.000	

Tabla 2.1. Selección de padres por el método de la ruleta

A continuación se genera un número aleatorio de 0 a 537 en valores absolutos o entre 0 y 1 en valores normalizados. Supóngase .516. El individuo seleccionado es el primero para el cual se cumple que el valor acumulado es mayor que .516. En el ejemplo es el cuarto individuo.

Otra forma de visualizar este efecto es dibujar las ranuras de ancho proporcional al *fitness* de cada individuo. Como se puede apreciar en la siguiente figura la probabilidad de que la bolita virtual caiga en la ranura más ancha es, obviamente mayor.

.186	.251	.074	.372	.115

El pseudocódigo asociado al método de la ruleta se muestra en la figura 2.4.

```
*       Función de Selección de Padres
        bola←Suma_de_fitness*random(0,1);
        j←0;
        repetir
                j←j+1;
                suma_parcial ← suma_parcial + Fitness(j);
        hasta (suma_parcial>=bola or j=N)
        retornar (j)
        fin
```

Figura 2.4. Algoritmo de selección de padres por el método de la ruleta

En realidad, como puede apreciarse, el método de la ruleta no es un método novedoso. Mas bien es una expresión diferente del método de Montecarlo.

2.2 Cruzamiento

Se analiza a continuación más detalladamente el operador de cruzamiento. Una vez seleccionados los padres la procreación se lleva a cabo a través de los operadores genéticos, de los cuales el cruzamiento es uno de ellos. En primer lugar se supone una representación de los cromosomas en strings de bits, para luego analizar la representación con valores enteros donde cada cromosoma representa una ruta posible.

El cruzamiento es simplemente un intercambio de información genética entre los padres de tal manera que los hijos que resultan **sean también cromosomas válidos**. Esto es, si los cromosomas son strings de bits que representan soluciones factibles en un espacio de soluciones, entonces los hijos también deberán serlo; si los cromosomas son secuencias de enteros que representan rutas posibles, ellos deberán generar hijos que representen rutas igualmente válidas.

Para representación en strings de bits, pueden utilizarse los cruzamientos propuestos por Holland [23] a saber:

El cruzamiento con un **único punto de cruce**, genera dos hijos a partir de los padres mediante el intercambio de las substrings determinadas por el punto de cruce aleatoriamente escogido entre los l-1 puntos posibles, siendo l la longitud del

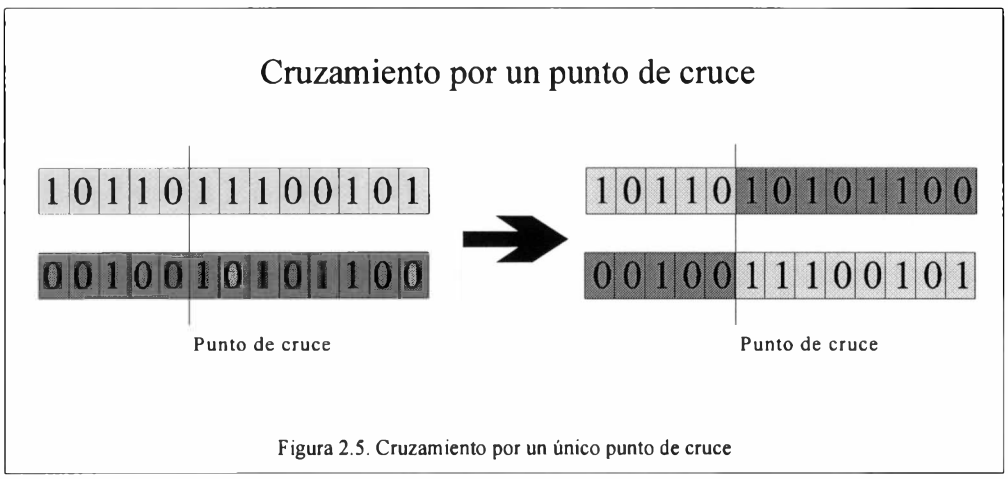


Figura 2.5. Cruzamiento por un único punto de cruce

cromosoma. En la figura 2.5 se ilustra este tipo de operador.

El cruzamiento **por dos puntos de cruce** es similar al anterior; en este caso se intercambian los bits de los padres que caen entre los dos puntos de cruce generados aleatoriamente. La figura 2.6 ilustra este tipo de cruzamiento.

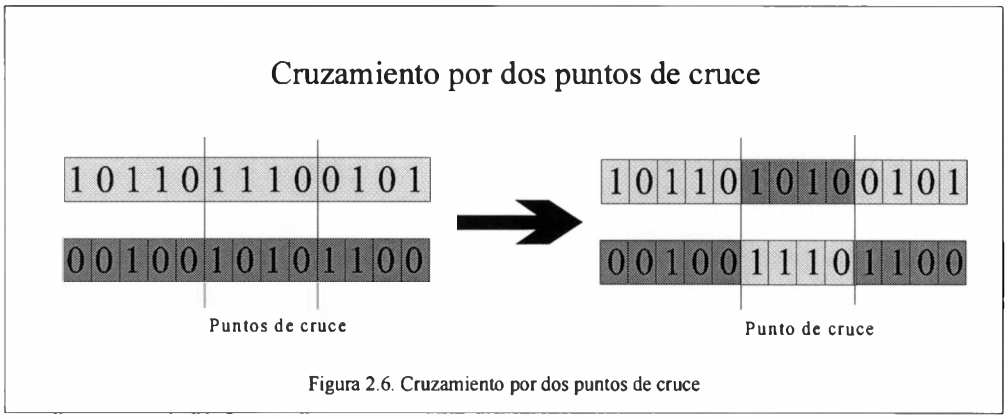


Figura 2.6. Cruzamiento por dos puntos de cruce

Finalmente el **cruzamiento uniforme** es una extensión del anterior aplicando intercambio de todos los bits homólogos entre los padres con una probabilidad del 50% una vez que se ha determinado que los dos padres se han de cruzar. Esto también puede lograrse mediante una expresión lógica, generando una máscara en forma aleatoria del mismo largo que los cromosomas y procediendo de la siguiente manera: si el bit *j* de la máscara es 1 en el hijo se copia el bit del padre en uno de los hijos y el de la madre en el otro hijo. En caso contrario la operación se invierte. Se ven los resultados del cruzamiento uniforme en la figura 2.7.

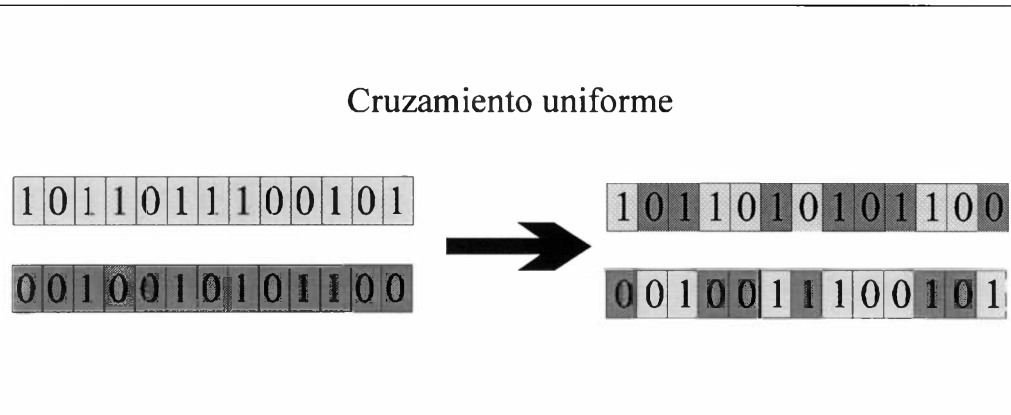


Figura 2.7. Cruzamiento uniforme

2.3 Mutación

El operador de mutación también dependerá, como ya se expresó, de la representación de los cromosomas. En el caso de representación por strings de bits, la mutación simplemente implica alterar bits de un cromosoma con cierta probabilidad. Esta operación implica decidir, en cada posición de bit y acorde con la probabilidad de mutación, si el bit debe mutar o no.

2.4 Elitismo

Se ha explicado que a partir de una generación dada se obtiene la siguiente por medio de los operadores descriptos. Sin embargo, los mejores individuos, si bien son los padres de los que forman la próxima generación, quedan excluidos. El elitismo tiende a evitar dicha situación. Especificando la cantidad de individuos de elite, se puede permitir el paso de algunos de ellos a la siguiente generación. De esta manera se asegura que la siguiente generación tendrá siempre un individuo de igual o mejor ajuste que en la anterior. El procedimiento es copiar, antes de comenzar a aplicar los operadores genéticos, los mejores k individuos de una población a la siguiente. k es un parámetro del AG.

2.5 Relación entre el *fitness* y la función objetivo

Nótese que el método de la ruleta considera anchos de ranura proporcionales al *fitness*. Sin embargo no se ha introducido, hasta ahora, ninguna restricción a los valores que puede tomar la función de ajuste. Es claro que si todos los valores son positivos la función de *fitness* coincidirá con la función objetivo. Pero si ésta tomara algunos valores negativos, no es posible asignar un ancho de ranura válido.

Por otra parte, como se asigna mayor probabilidad de ser padres a los individuos de mayor *fitness* el proceso tiende a convergir a la maximización.

Si lo que se desea es minimizar, o bien si la función objetivo toma valores menores que cero en alguna porción del dominio bajo estudio, será necesaria una transformación previa a la evaluación.

Sea *f(x)* la función objetivo a optimizar (maximizar o minimizar). Puede realizarse una transformación

$$g(x)=T(f(x))$$

que permita sortear los obstáculos planteados. Los casos que pueden distinguirse son:

a) Maximizar *f(x)* con $x \varepsilon D=[a,b]$

Si $x>0$ para todo $x \varepsilon$ D entonces T es la transformación identidad ($T(f)=f$), resultando

$$g(x)=f(x)$$

Si $x<0$ para algún $x \varepsilon$ D entonces T es una transformación tal que produce un cambio de coordenada mediante un desplazamiento del eje de abscisas en una constante *c* ($T(f)=f+c$), resultando

$$g(x)=f(x)+c$$

donde

$$c=|\min(f(x)| \text{ para todo } x \varepsilon D$$

Si min(f(x)) no es conocido, *c* deberá estimarse. Nótese que esta transformación sólo implica un cambio de coordenadas para forzar a la función de fitness a ser positiva en el intervalo considerado, lo cual puede apreciarse en la siguiente figura:

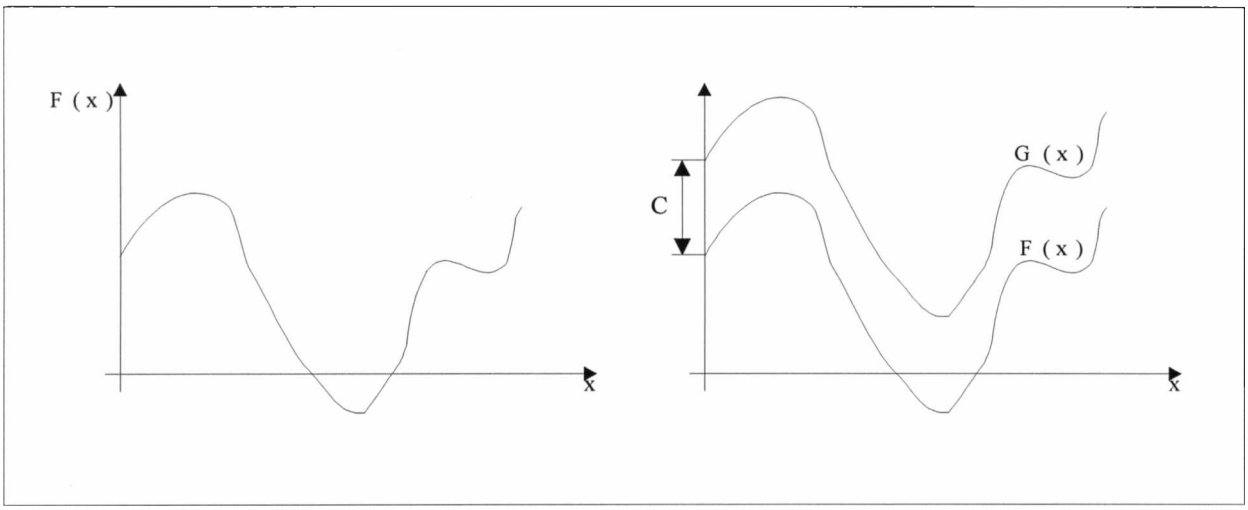


Figura 2.8

Esta transformación es lícita dado que el punto crítico sobre el dominio permanece invariable.

b)Minimizar $f(x)$ con $x \varepsilon D=[a,b]$

La minimización de $f(x)$ se obtiene mediante la maximización de $-f(x)$; por lo que, en principio, la transformación T producirá un cambio de signo ($T(f)=-f$). El resultado es

$$g(x)=-f(x)$$

Pero aún resta el evitar que existan valores negativos en el *fitness*. La solución a este problema viene de la mano de la aplicación del mismo criterio explicado en el punto a). Haciendo

$$g(x)=-f(x)+ C_{máx}$$

y eligiendo convenientemente $C_{máx}$ para que todos los valores de $g(x)$ sean positivos, el problema de minimización queda salvado.

Podría pensarse en elegir un valor suficientemente grande para $C_{máx}$,pero entra a tallar otro problema que tiene que ver con la escala y los valores relativos de $g(x)$. Por ejemplo, si se quiere minimizar la función

$$f(x)=sen(x) \text{ para } x \varepsilon D=[0, \Pi]$$

y se toma $C_{máx}=2$

tal como se puede apreciar, la diferencia entre la función objetivo entre $x_1= \Pi /2$ y $x_2= \Pi$ es 1, Proporcionalmente la relación está dada por

$$g(x_1)/g(x_2) =1/2$$

Pero si $C_{máx}=100$y se calcula la proporción

$$g(x_1)/g(x_2) =(100-\text{sen}(x_1)) / (100-\text{sen}(x_2)) = 99/100 = 0.99$$

Se puede observar que para $C_{máx}=2$ el ancho de la ranura para $x_1= \Pi /2$ es el doble que para $x_2= \Pi$; pero son casi iguales si $C_{máx}=100$. Por lo tanto la selección otorgará idéntica probabilidad a dos individuos que en realidad deberían tener un ajuste muy diferente.

Una forma de zanjar el problema es elegir $C_{máx}$ después de una primera inspección de la función en el intervalo D, ajustando dicho valor en las sucesivas iteraciones del AG cuando se detectan valores menores que cero en la función objetivo $g(x)$.

2.6 Algoritmo genético aplicado a optimización de funciones reales

Se presenta a continuación la ejecución de un AG puesto a encontrar el máximo de una función simple a valores reales corrido "a mano" para ilustrar su funcionamiento y explicitar la forma en la cual el mismo evoluciona. Posteriormente se presentan resultados obtenidos con otras funciones más complejas.

Sea $f(x)=x^2$ con x ε D=[0,15] ^ x ε Z^+

Se plantea encontrar x^* tal que $f(x^*)=f_{máx}$

Está claro, en este sencillo ejemplo que el máximo de la función está en el extremo del intervalo. Es decir

$$x^*=15$$

y

$$f(x^*)=225$$

De acuerdo con lo expresado anteriormente, se siguen los pasos para la construcción del AG.

2.6.1 Codificación (representación)

Se elige una codificación en forma de string de bits de 4 bits de longitud, suficiente para representar todo el dominio. Se considera la representación binaria directa ($0000_b=0_d$,... $1111_b=15_d$).

2.6.2 Generación de la población inicial

Como criterio de diseño, se decide trabajar con una población de tamaño fijo de 4 individuos (recuérdese que *x* pertenece al conjunto de los números enteros). En forma aleatoria se generan dichos individuos como población inicial.

$$P^0=\{(0101)\ (1000)\ (1100)\ (0011)\}$$

2.6.3 Evaluación de la población inicial

El primer individuo (0101) corresponde al valor decimal 5. Su *fitness es* $5^2=25$ y el aporte respecto del *fitness* total es del 10.3% tal como se muestra en la tabla 2.2

Generación 0

Individuo	Represent.	Valor x	Fitness fi	Prob. Selecc $fi/\Sigma fi$	Prob. acumulada
x_1^0	0101	5	25	0.103	0.103
x_2^0	1000	8	64	0.264	0.367
x_3^0	1100	12	144	0.595	0.962
x_4^0	0011	3	9	0.038	1
			Σfi =242		
			f_{medio}=60.5		

Tabla 2.2

La ruleta queda conformada como se muestra en la figura 2.9

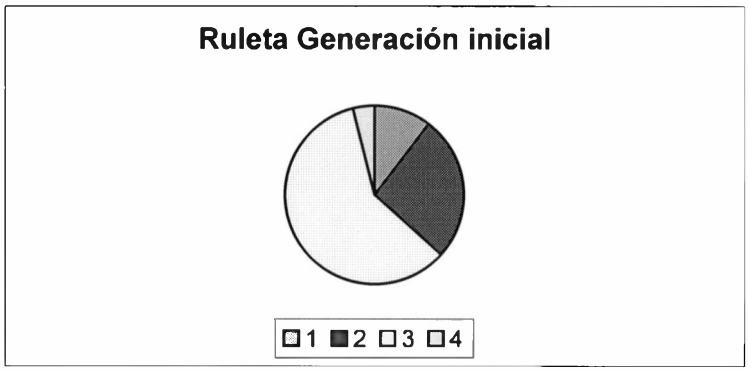


Figura 2.9

2.6.4 Ciclo de evolución

2.6.4.1 Selección

Se generan cuatro números en forma aleatoria, mostrados en la tabla 2.3. La segunda columna de la tabla indica que individuo fue seleccionado como padre para la próxima generación.

N°	Individuo Seleccionado
0.030	x_1^0
0.341	x_2^0
0.856	x_3^0
0.708	x_3^0

Tabla 2.3

A continuación, también en forma aleatoria se forman las parejas que procrearán, las cuales quedan conformadas como sigue:

$$\text{Pareja } 1: x_1^0 - x_3^0$$

$$\text{Pareja } 2: x_2^0 - x_3^0$$

2.6.4.2 Cruzamiento

De la primera pareja

$$x_1^0 = 01|01$$

$$x_3^0 = 11|00$$

y tomando un punto de corte generado en forma aleatoria luego del 2do bit los hijos generados resultan

$$x_1^1 = 0100$$

$$x_2^1 = 1101$$

mientras que de la segunda pareja

$$x_2^0 = 100|0$$

$$x_3^0 = 110|0$$

y tomando un punto de cruce luego del 3er bit los hijos que resultan del apareamiento son

$$x_3^1 = 1000$$

$$x_4^1 = 1100$$

2.6.4.3 Mutación

Admítase que x_3^1 muta su segundo bit para pasar a

$x_3^1 = 1010$

La nueva población P^1 que conformada como sigue

$$P^1=\{(0100)\ (1101)\ (1010)\ (1100)\}$$

2.6.5 Evaluación de la nueva población

Nuevamente se construye una tabla para visualizar la evaluación de la nueva población. La tabla 2.4, muestra ya una evolución: el valor del *fitness* promedio ha aumentado a $f^1_{med}=107.25$.

Generación 1

Individuo	Represent.	Valor x	Fitness fi	Prob. Selecc $fi/\Sigma fi$	Prob. acumulada
x_1^1	0100	4	16	0.037	0.037
x_2^1	1101	13	169	0.394	0.431
x_3^1	1010	10	100	0.233	0.664
x_4^1	1100	12	144	0.336	1
			Σfi =429		
			f_{medio}=107.25		

Tabla 2.4

Por otra parte, comparando la distribución de los individuos en las dos poblaciones P^0 y P^1 puede observarse una concentración, leve aún, hacia la derecha en la segunda población. El método comienza a converger. Tal distribución es mostrada en la figura 2.10.

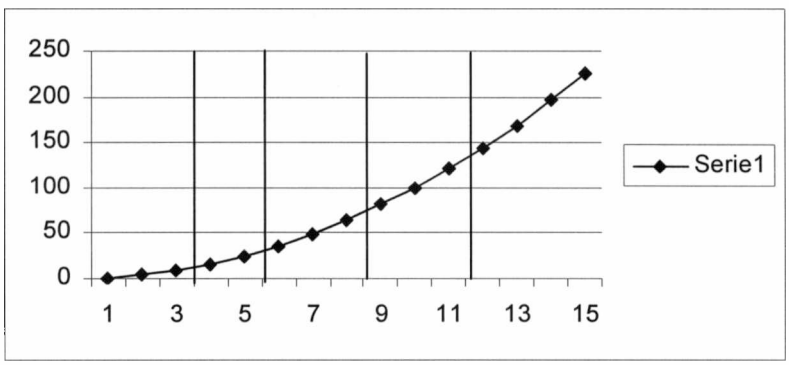


Figura 2.10 Distribución de individuos Generación 1

Se van obteniendo, repitiendo estos pasos, las siguientes generaciones, cuyos resultados se muestran en las tablas y figuras a continuación:

Generación 2

Rand=[0.749, 0.232, 0.351, 0.703]

x_3^1	101|0	1011	x_1^2
x_2^1	110|1	1100	x_2^2
x_2^1	11|01	1100	x_3^2
x_4^1	11|00	1101	x_4^2

Individuo	Represent.	Valor x	Fitness fi	Prob. Selecc $fi/\Sigma fi$	Prob. acumulada
x_1^2	0011	11	121	0.209	0.209
x_2^2	1100	12	144	0.249	0.458
x_3^2	1100	12	144	0.249	0.707
x_4^2	1101	13	169	0.292	1
			Σfi =578		
			f_{medio}=144.50		

Tabla 2.5

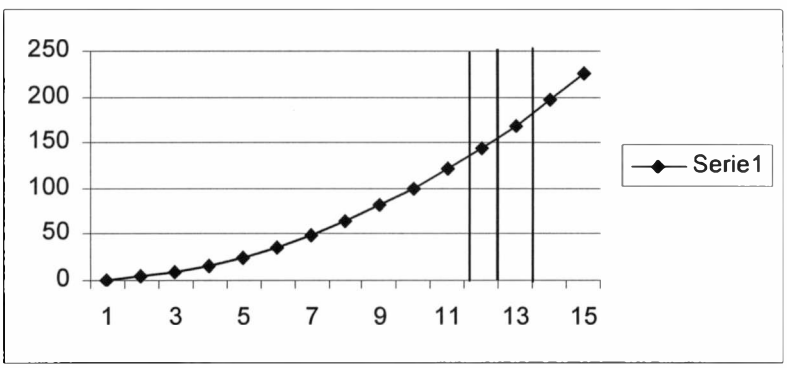


Figura 2.11 Distribución de individuos Generación 2

Generacion 3

x_1^2	10|01	1111	x_1^3
x_2^2	10|11	1001	x_2^3
x_3^2	1|100	1101	x_3^3

x_4^2	1|101	1100	x_4^3

Individuo	Represent.	Valor x	Fitness fi	Prob. Selecc $fi/\Sigma fi$	Prob. acumulada
x_1^3	1111	15	225	0.363	0.363
x_2^3	1001	9	81	0.131	0.494
x_3^3	1101	13	169	0.273	0.767
x_4^3	1100	12	144	0.233	1
			Σfi =619		
			f_{medio}=154.75		

Tabla 2.6

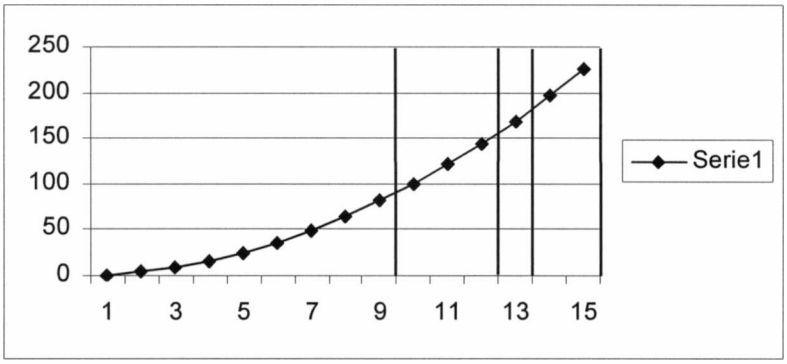


Figura 2.12. Distribución de individuos Generación 3

Finalmente, resulta interesante observar, si bien en este ejemplo son pocos los puntos establecidos, la convergencia del método a través de la gráfica de la figura 2.13 en la que se muestra el mejor individuo en cada población.

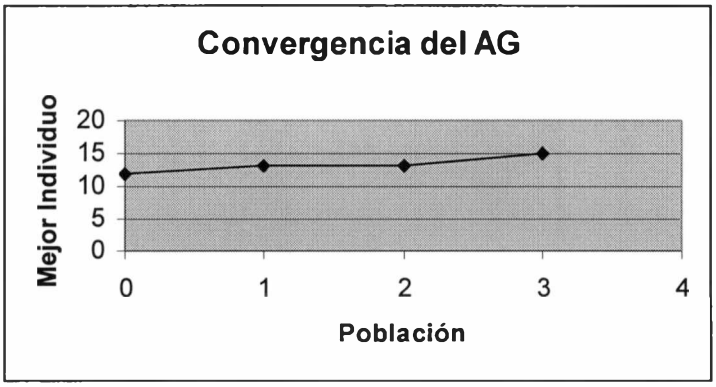


Figura 2.13. Mejor individuo de cada población

La evolución también puede reflejarse en los cambios de los *fitness* promedio de cada generación. La figura 2.14 ilustra dicha variación.

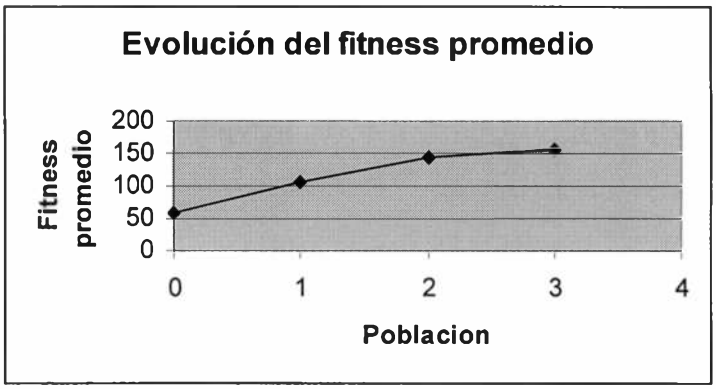


Figura 2.14. Evolución de f_{med}

2.7 Conclusión

Se ha explicado en esta sección el funcionamiento de un algoritmo genético simple y los ejemplos dados corresponden a la representación de cromosomas como strings de bits, para la cual los operadores de cruzamiento por puntos de cruce único o múltiple dan como resultado otra string de bits igualmente válida. Las restricciones impuestas por los límites mínimo y máximo que puede tomar cada variable se salvan con el sistema de representación adoptado, tal como se explicó al inicio y por lo tanto, la descendencia de estos cromosomas generará hijos dentro del rango de validez.

En el caso de que los cromosomas representen rutas alternativas como en el caso del TSP o de ruta óptima, los operadores no podrán ser utilizados tal cual se describieron sino que deben adaptarse para cumplir la propiedad de generar hijos válidos. En las secciones siguientes se describen los operadores para construir un algoritmo genético que permita resolver este tipo de problemas.

CAPÍTULO III

3 APLICACIONES DE LOS ALGORITMOS GENÉTICOS EN PROBLEMAS DE REDES.

3.1 Introducción

Las aplicaciones de los Algoritmos genéticos cubren un espectro cada vez más amplio: problemas de optimización en general, problemas de tipo combinatorio (*scheduling*, localizaciones, distribuciones, etc.) son ámbitos naturales donde existen los AG.

Más particularmente en redes las aplicaciones no son menores. Este capítulo expone algunos de los diferentes casos en los cuales se aplican los AG con buena actuación en lo referente a redes, para luego en el próximo capítulo encarar de lleno el diseño de un AG para solucionar un problema concreto de redes de datos.

3.2 Balance de Carga en un sistema distribuido

Un sistema de cómputo distribuido (*Distribution Computing System - DCS*) consta de procesadores autónomos e interconectados a través de una red de comunicaciones. La *performance* global del sistema mejora significativamente cuando la carga entre los procesadores está equitativamente distribuida. Esto es, que no haya procesadores con exceso de tarea en el mismo momento que otros estén ociosos.

La idea es, entonces, distribuir la carga teniendo en cuenta los objetivos de maximizar la utilización de los procesadores y minimizar el tiempo de respuesta medio.

En general, los algoritmos de balance de carga pueden ser clasificados en dos categorías a saber: estáticos y dinámicos. La diferencia principal entre ambos tipos de algoritmos es obvia después de haber mencionado los nombres de las categorias: un balance de carga estático implica una distribución de tareas antes de su ejecución y basada en información previa de dichas tareas. Por otra parte, en un esquema de balance dinámico, un procesador "cargado" envía el exceso de tarea a otro procesador que tenga menos

trabajo; y eso ocurre durante la ejecución.
Hay trabajos en los cuales se resuelve el DCS estático aplicando algoritmos genéticos (Mansour [10] o Kidwell [11]). El balance de carga estático es, desde el punto de vista matemático, un problema de optimización combinatorial y los AG pueden competir hasta con ventaja en este campo.

Los trabajos de Munetono [9] en referencia a balance de carga dinámico utilizando AG incluyen las capacidades de aprendizaje adaptivo para resolver el problema. En efecto, el balance de carga dinámico no es un problema de optimización simple: la decisión de migrar una tarea se toma en un procesador y está basada en información que puede no ser absolutamente segura dado que es afectada por los tiempos de latencia de las comunicaciones. La clave del balance de carga dinámico eficiente en un DCS el conocimiento de la distribución de la carga.

Esquivel [12] plantea un esquema similar pero incluye la utilización del promedio exponencial ponderado para determinar un subconjunto, dentro del conjunto de procesadores candidatos a ser requeridos, tratando de predecir la aceptación de la solicitud.

A los efectos de poder realizar comparaciones con otros métodos, se considera el algoritmo *sender-iniciate* en el cual cada procesador, en forma independiente, decide sobre la migración de una tarea. El algoritmo prevé la emisión de mensajes desde un procesador que se considera con carga pesada a otro. En este caso el procesador que ha decidido intentar migrar una tarea envía un mensaje de requerimiento (*request message*) a otro procesador, elegido al azar. Si el destino puede hacerse cargo de la tarea, esto es su carga actual es "liviana", contesta con un mensaje de aceptación (*Accept message*). De lo contrario, se niega mediante un mensaje de rechazo (*Reject message).* Si el procesador emisor recibe un mensaje de aceptación se produce la migración de la tarea. Si recibe un rechazo, elige otro procesador y vuelve a hacer el requerimiento. Si todos los requerimiento enviados son rechazados, no hay migración de tarea y el procesador deberá afrontar su procesamiento.

Como se podrá apreciar en los próximos párrafos, este algoritmo funciona bien cuando la carga del sistema a través del DCS es relativamente liviana. Esto es así por cuanto el

algoritmo encuentra más fácilmente una respuesta de aceptación a su *requet message.* Pero cuando el sistema, globalmente, está sometido a carga más pesada, es más dificil encontrar procesadores que acepten la migración de la tarea. Esto se traduce en sucesivos *reject* a los pedidos de requerimiento para, quizá, finalmente determinar que la tarea no sufrirá migración. Esto es por supuesto costoso en tiempo.

La solución con algoritmos genéticos mantiene el esquema de mensajes, pero se determina una combinación de *request message* enviados antes de la migración, a través de un procedimiento de aprendizaje llevado a cabo dentro de cada procesador. El objetivo de este aprendizaje es reducir el tiempo de decisión del procesador destino al cual enviar mensajes de requerimiento y mejorar la tasa de aceptación de los requerimientos en medio de un sistema que va cambiando en forma dinámica de acuerdo con la distribución de carga e influido por ella.

El aprendizaje se hace vía operadores genéticos (selección, cruzamiento y mutación) sobre una población de cadenas de bits en la cual si el bit *i* está en 1, significa que se le enviará un *request* al procesador *i.*

Por ejemplo, en un esquema de siete procesadores (P_0, P_1, ... P_7) y, considerando el cuarto (P_3) , un individuo de la población del cuarto procesador podría ser:

$$\mathbf{P}=(0\ 1\ 0\ 1\ *\ 0\ 1\ 0)$$

Que significa que se enviará, en caso de decidir migración, los requerimientos a los procesadores P_1, P_3 y P_6 . en ese orden. La migración se realizará contra aquel procesador que la haya aceptado.

¿Porqué **P** es una buena opción que reduce tiempos y optimiza? ¿Qué hace que esta combinación sea buena? ¿Cómo medir su bondad? En otras palabras, ¿cuál es su fitness?

La función de *fitness* tiene que ver con el éxito o fracaso de los requerimientos enviados a los otros procesadores. En general la función de *fitness* viene expresada como un promedio de valores de pesos individuales de cada individuo o *payoff.* El *payoff* de un

individuo es inversamente proporcional al numero de *request* enviados, salvo en el caso en que todas las solicitudes de migración fueran rechazadas. En este caso la string tendrá un *payoff* nulo. En otras palabras, el *payoff* de una string se define como

$$payoff = \begin{cases} 0 & si\ todas\ las\ solicitudes\ fueron\ rechazadas \\ g(m) & caso\ contrario \end{cases}$$

Donde g es una función monótonamente decreciente que asegure valores positivos para elementos positivos del dominio y m es el número total de solicitudes a enviar según la string que se considera (es la cantidad de bits en 1). Un ejemplo de g utilizado es

$$g(x) = \frac{1}{\sqrt{x}}$$

Nótese que, con este criterio, las strings que tienen previsto menor cantidad de mensajes de solicitud tendrán un valor de *payoff* más alto, pero si todas las solicitudes encuentran respuestas negativas, el *payoff* de la string se vuelve nulo.

En este caso, los algoritmos genéticos son usados como predictores de la respuesta del procesador requerido. Su *fitness* tiene que ver con la bondad de las predicciones anteriores y, por lo tanto, el valor predicho tiene que ver con la historia del sistema.

En los trabajos de Esquivel [12] y otros, se proponen estrategias para mejorar la eficiencia computacional: toma no más del 50% de los procesadores candidatos a ser requeridos y la predicción de la respuesta del procesador viene dado por el promedio exponencial ponderado. Es decir, sobre una base similar a la planteada en [9] el fitness de cada individuo es considerado mayor cuanto más exitosos fueron los requerimientos. En ese caso se tomó el *fitness* de cada cromosoma como el resultado de la relación Nro de éxitos/total de intentos.

Si un cromosoma seleccionado fuera

P=(1,0,-,0,1,1,1,1,1,0,0,1,1,0,0,0,1,0,1)

Habría un total de 9 procesadores candidatos. Son 9 resquerimientos. Entonces si 3 procesadores contestan afirmativamente el *fitness* de P es *f=3/9=0.333;* Seis respuestas positivas arrojarían un *fitness* de 6/9=.667 y 1 es el valor que corresponde al éxito total.

Después de cada requerimiento, se computa el *fitness* de cada cromosoma y se genera una nueva población a partir de la anterior.

Otro aporte de este trabajo es el realizar los requerimientos sólo a un subconjunto de procesadores elegidos entre los candidatos marcados por el cromosoma seleccionado. La ventaja principal es la de reducir el tráfico. Los experimentos mostraron una mejor eficiencia al tomar los mejores N/2 individuos candidatos y enviarles el mensaje de requerimiento (donde N es el número total de candidatos en el cromosoma).

Par decidir quienes son los mejores, se aprovecha el conocimiento que cada nodo tiene. Se trata de predecir cuales procesadores aceptarán el requerimiento. Se tomo el promedio exponencial ponderado, dado por

$$V_{n+1}=\alpha P_n + (1- \alpha) V_n$$

V_n representa la historia antigua mientras que P_n es la última muestra. α es un factor que permite ponderar ambos términos de la expresión. Si el valor de α tiende a 0 es la historia la que tiene mayor importancia; si α tiende a 1 es el valor de la última muestra la que adquiere mayor importancia.

3.2.1 Operadores genéticos

3.2.1.1 Selección

Se utiliza selección proporcional, es decir que una string es seleccionada con una probabilidad proporcional a su valor de *fitness.* Esto es, la probabilidad que j-ésima string sea seleccionada es

$$p_j = \frac{f_j}{\sum_{k=0}^{n-1} f_k}$$

3.2.1.2 Cruzamiento

El cruzamiento utilizado es el uniforme, descripto en el capítulo anterior. Hay una restricción que considerar y es el hecho que una string debe tener un *payoff* único asociado; por lo tanto no se permite la duplicación de strings en la población. Las mejores N strings diferentes de una poblacion *t* pasan a la siguiente población *t+1*.

3.2.1.3 Mutación.

Se utiliza la mutación estándar de una cadena de bits a valores binarios tal cual fuera descripto en el capítulo II.

3.2.1.4 Resultados

Se toman los resultados obtenidos y publicados en [Munetomo]. Las condiciones en las cuales fueron hechos los experimentos involucran una red de 16 procesadores independientes conectados con una topología en bus a 10 Mbps, cada paquete es de 1024 bytes. Como parámetros del AG se tomaron los siguientes: una población de 10 individuos, la función de fitness igual al promedio de las últimas 5 instancias del valor de *payoff*, una probabilidad de cruzamiento de 0.05 y una de mutación de 0.1. El valor de la función de *payoff* fue tomada como

$$g(x) = 1/\sqrt{x}$$

Los resultados comparativos pueden verse en las figuras 3.1 y 3.2. En el primer caso se supone que las tareas nacen uniformemente en todos los procesadores con una misma tasa de arribo λ para todos los procesadores. En el segundo caso, sólo 4 procesadores generan tareas a una tasa promedio de 4λ. No nacen tareas en los otros procesadores. En ambas figuras se muestra el tiempo medio de respuesta vs. La tasa promedio de arribo. Las tres condiciones para las cuales se muestran los resultados son: sin balance, balance de carga por *sender-initiated* y balance de carga por AG.

Este es un buen ejemplo de mejora de la eficiencia por medio de la utilización de algoritmos genéticos. Nótese que en este caso, el AG es utilizado como predictor. La historia pasada del sistema se utiliza para realizar dicha predicción. Los éxitos y fracasos proveen la realimentación con la cual el AG realiza sus ajustes.

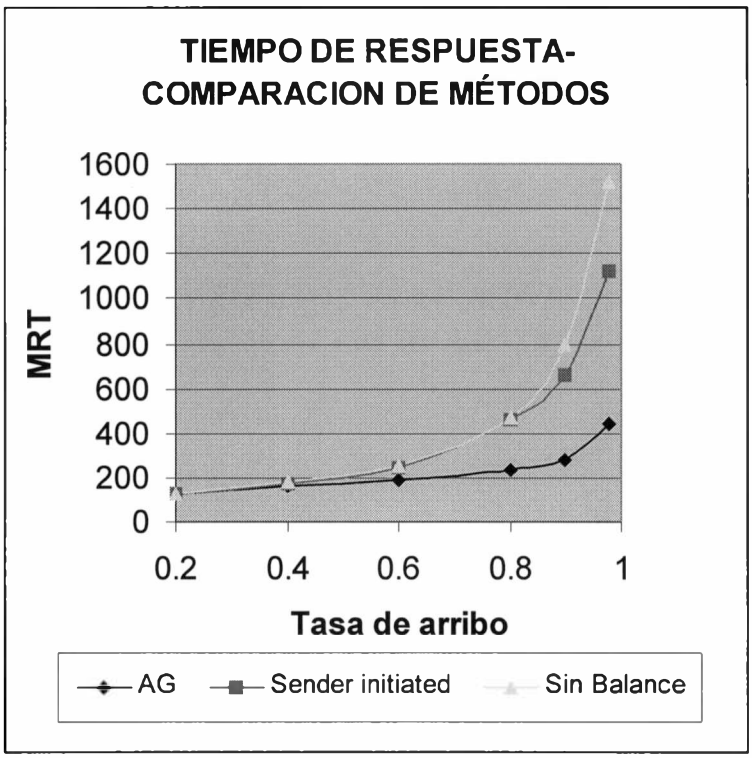


Figura 3.1. Tiempo medio de respuesta v.s. tasa de arribo para un sistema de distribución de carga uniformemente espaciada.

Por otra parte, los resultados obtenidos por Esquivel [12], reportan mejores performances en varios escenarios, no sólo en lo referente al tiempo de respuesta del sistema sino en cantidad de requerimientos realizados. Además se obtuvieron mejores resultados cuando se aplicó el promedio exponencial ponderado y se seleccionó la mejor mitad de los candidatos a enviar requerimientos, que cuando se les envió mensajes a todos los marcados por la información del cromosoma seleccionado.

Como conclusión, los tiempos de respuesta del sistema disminuyen al aplicar métodos de balance de carga. Se obtienen mejoras significativas de eficiencia cuando se aplican las técnicas evolutivas a los efectos de predecir las respuestas de los procesadores que pueden hacerse cargo de la tarea de uno que sufre una sobrecarga eventual.

Finalmente la selección de un subconjunto de candidatos colabora a la reducción de mensajes que se envían y la consecuente disminución de tráfico, proporcionando una mejora adicional.

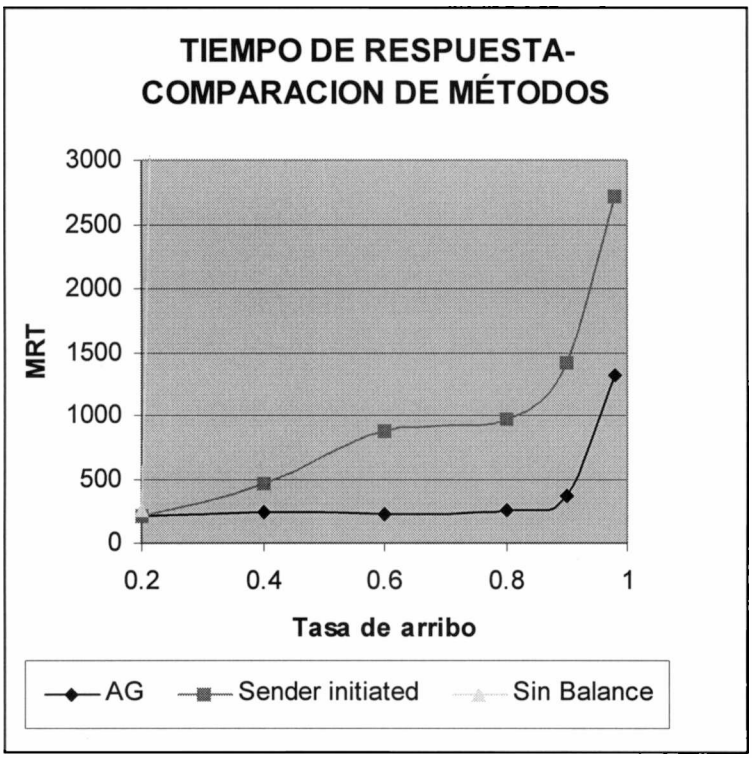


Figura 3.2. Tiempo medio de respuesta v.s. tasa de arribo para un sistema de distribución de carga no uniformemente espaciada.

3.3 Localización de sensores en redes en procesos industriales.

La información que brinda el monitoreo de una planta química deberá permitir una estimación completa de las variables que se requiere conocer a los fines de realizar las tareas de supervisión, control, planeamiento, optimización, etc.

En general el esquema de monitoreo involucra un conjunto de sensores, colocados en determinados puntos de la planta que transmiten al centro de control, la información obtenida en el proceso de medición.

Pero las propias variables del proceso presentan, en general, dependencias de unas con respecto a las otras. No es necesario medir absolutamente todas las variables dado que algunas pueden ser determinadas a partir de otras. Esto permite tener distintas disposiciones de los instrumentos de medición, una de las cuales será la de mínimo costo de instrumentación.

Por otra parte, se presentan restricciones en lo que hace a la calidad de la medición. Se requieren, generalmente, distintos valores de exactitud (que, por supuesto vienen de la mano de instrumentos más costosos) lo que complica un poco más el problema. Se trata de encontrar, no solo la ubicación de mínimo costo para que las variables especificadas sean conocidas (medidas o calculadas) sino que deben respetarse las exactitudes requeridas en cada caso.

Este problema, tratado en Carnero y otros [25], es de tipo combinatorio e involucra el uso de variables continuas y discretas. La clasificación de las variables utilizadas se plantea en términos de factorizaciones ortogonales y no es tratado aquí. Sin embargo se menciona por que el proceso de clasificación de variables, entre otros resultados, determina si las todas las variables especificadas pueden ser conocidas (sea por medición o por cálculo). En definitiva y desde el punto de vista del AG, esta técnica es utilizada para restringir el espacio de búsqueda.

El algoritmo genético utilizado en este caso [25] es el mismo que se plantea en los capítulos anteriores y que se diseña en los que siguen.

El AG mantiene una población de individuos donde cada uno de ellos es una propuesta de localización de instrumentos. La función de *fitness* es muy simple: viene dada por la suma de los costos de cada instrumento que se ubica. El AG tratará de encontrar la de menor costo. La clasificación de variables se usa para determinar si se cumple la restricción impuesta de que todas las variables especificadas sean determinables. Si eso no ocurre, la solución propuesta es penalizada, modificando de esta manera el valor de *fitness*.

La representación elegida en este caso fue de vectores de números enteros, donde el valor de cada componente número representa el tipo de instrumento y su índice el lugar de la localización. Un valor nulo indica que no se propone colocar instrumento en el lugar indicado. Así, el vector P, dado por

$$P=(1, 0, 0, 3, 0, 2, 0, 0)$$

propone, entre 8 sensores ubicables, colocar un instrumento de tipo 1 en la primera ubicación, uno de tipo 3 en la cuarta y uno de tipo 2 en la sexta. Las otras variables no serán medidas. La clasificación de las variables será la encargada de determinar la factibilidad de esta solución.

El costo asociado a P es la suma de los costos de los instrumentos utilizados en esta propuesta de solución

$$C_p=C_1+C_3+C_2$$

Finalmente entre todas las soluciones factibles el AG determinará la de menor costo.

3.3.1 Operadores genéticos

3.3.1.1 Selección

Al igual que en el caso presentado anteriormente se utilizó selección proporcional,. Esto es, la probabilidad que j-ésima string sea seleccionada es

$$p_j = \frac{f_j}{\sum_{k=0}^{n-1} f_k}$$

3.3.1.2 Cruzamiento

Se utiliza cruzamiento uniforme, sólo que se intercambian valores enteros en vez de bits. Hecha esta salvedad, el cruzamiento que se realiza es el estándar.

3.3.1.3 Mutación.

Se utiliza mutación estándar con la única salvedad que se mutan valores enteros y dichos valores deben pertenecer a un conjunto finito (el de tipos de instrumentos disponibles).

3.3.1.4 Resultados

En el trabajo citado se compararon los resultados con valores obtenidos por otras técnicas observando una buena convergencia al valor óptimo del AG. La estructura del cromosoma utilizada fue la de vectores de números enteros y, para el caso prueba de cuatro corrientes de flujo y midiendo caudales con tres tipos de instrumentos distintos, se utilizó una población de 10 individuos, probabilidad de cruzamiento del 65%, prob de mutación del 0.5% y un individuo de elite. La figura 3.3 muestra una curva de convergencia del algoritmo puesto a resolver el problema.

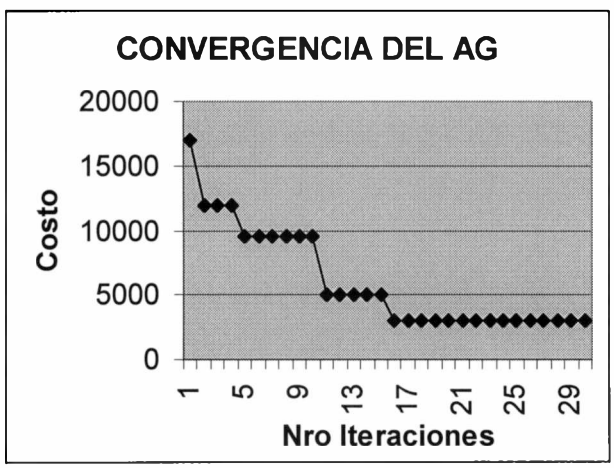


Figura 3.3. Convergencia de un AG en localización óptima de instrumentos.

En este caso, a diferencia del anterior se utilizó el AG como optimizador de la función de costo y con el agregado de la imposición de restricciones. En este trabajo en

particular, la penalización por violación de la restricción fue la pena de muerte para el individuo. Trabajos futuros considerarán otras funciones de penalización para pode aprovechar algunos individuos que, si bien proponen soluciones no factibles, pueden ser intermediarios gracias a los cuales encontrar un buen individuo dentro del espacio de soluciones factibles. La pena de muerte tiene la desventaja cortar de raíz dicha posibilidad.

3.4 Diseño de redes mesh

El diseño de redes de comunicaciones enmalladas es un problema complejo y multirestringido y de tipo combinatorio. También aquí los AG pueden jugar un buen papel.

El problema que se presenta es unir un cierto número de ciudades con una red respetando determinados requerimientos de tráfico, capacidades de enlaces, ruteos y conexionado. El resultado es una topología que respeta las restricciones impuestas y minimiza los costos de conexión.

King-Tim Ko y otros [4], resuelven el problema de conectar diez ciudades chinas con una red de conmutación de paquetes. La red tiene un tráfico especificado entre ciudades, representado por una matriz de tráfico donde el elemento T_{ij} representa el tráfico especificado entre la ciudad *i* y la ciudad *j*. La representación de la topología se realiza a través de la matriz de adyacencia: matriz booleana cuyos elementos $\mathbf{a_{ij}}$ son tales que almacenan un 1 si hay un enlace entre los nodos *i* y *j* y un 0 en el caso contrario. Asumiendo que el enlace es bidireccional (*i* puede enviar mensajes a *j* y viceversa) la matriz de adyacencia es simétrica. La matriz de adyacencia configura en si misma un cromosoma que representa a un individuo. Sin embargo y a los efectos de realizar una simplificación que permite la utilización de los operadores genéticos estándares, se realiza una vectorización de la matriz. Cada componente *k* del vector se calcula como

$$\mathbf{v}_k = \mathbf{a}_{ij}$$

donde,

$$k=jX-[j(j+1)/2]+i-j-1, \text{ para } j>i \text{ y}$$

$$\mathbf{a}_{ij}=0 \text{ si } i=j=0$$

El objetivo es obtener la mejor topología del sistema que respete las especificaciones mencionadas. La representación de un cromosoma que se utiliza es una string de bits de una longitud n(n-1)/2 elementos donde n es el número de ciudades consideradas.

3.4.1 Operadores genéticos

En este caso se adoptó selección proporcional y operadores estándar para cruzamiento y mutación pero con algoritmos de reparación de individuos. Esto es, cuando por efecto de cruzamiento o mutación se genera un hijo que no cumple con la restricción de biconexión impuesta, se ejecuta un algoritmo de reparación que corrige esa situación. Es diferente de la estrategia utilizada en el caso anterior donde la violación de las restricciones se penalizaba con la muerte del individuo.

3.5 Conclusiones

Se han presentado diferentes casos en los que los algoritmos genéticos cumplen un buen papel, en problemas de optimización que involucran redes. Cada caso explora alguna característica diferente del AG. Así el algoritmo funciona intentando predecir la respuesta de un procesador al cual se le requiere migrar una tarea, encuentra una localización óptima de instrumentos de medición o un esquema de conexión de mínimo costo en una red de conmutación de paquetes.

El algoritmo básico es el mismo en todos los ejemplos. Una diferencia está en la forma de representar el cromosoma, lo que trae como consecuencia en algunos casos, la realización de algunas modificaciones en los operadores de cruzamiento y mutación. Otra diferencia en los ejemplos estriba en el tratamiento de las restricciones. Mientras que en el caso de localización de sensores, la violación es penalizada con la eliminación del individuo, en el ejemplo de diseño de redes *mesh* la solución no factible es convertida antes de proceder a su evaluación.

En los capítulos que siguen, se analiza el diseño de un AG que pueda resolver alguno de estos problemas. Se tomará como referencia el caso de la obtención del camino de mínimo costo entre dos nodos de una red y en torno a dicho problema se irá implementando el AG.

Una base para la solución de este tipo de problemas es el problema del viajante. El próximo capítulo toma el TSP y desarrolla un AG completo a su alrededor. La resolución de este problema permite el análisis de distintos tipos de operadores y estrategias que serán usadas en la resolución de otros problemas, como el de la obtención de la ruta óptima que será el tema del capítulo V.

CAPITULO IV

4 ALGORITMOS GENÉTICOS Y EL PROBLEMA DEL VIAJANTE (TSP)

4.1 Introducción

En las secciones anteriores se explicó el funcionamiento de los algoritmos genéticos para resolver problemas de optimización en distintas aplicaciones. Los algoritmos enunciados suponen la representación binaria de los cromosomas. En el TSP, como en la mayoría de los problemas de secuenciación dicha representación ofrece más desventajas que beneficios lo que conlleva a elegir otros tipos de representaciones y por lo tanto tampoco los operadores genéticos podrán ser aplicados sin modificaciones. Esta sección presentará el problema del viajante y los operadores genéticos asociados a la resolución del problema como una base para la resolución del problema de la ruta óptima.

4.2 El problema del viajante

El problema del viajante (Traveling Salesman Problem) o TSP conceptualmente es simple: Un viajante debe visitar todas las ciudades de su zona, recorriendo todas las ciudades exactamente una vez y retornar al punto inicial. Cada paso de una ciudad a otra tiene un costo determinado y el objetivo es planificar un viaje que cumpla con los requerimientos del problema y tenga el mínimo costo.

El espacio de búsqueda es un conjunto de permutaciones de n ciudades, por lo tanto su tamaño es $n!$. Cualquier recorrido completo conduce a una solución del problema (aunque no sea la óptima). La solución óptima es el recorrido para el cual el costo es mínimo. Dicha solución puede ser expresada como

$$(i_1, i_2, ...,i_n)$$

donde $(i_1, i_2, ...,i_n)$ es una permutación sobre el conjunto $\{1,2,3, ...,n\}$. La representación binaria de estas soluciones no aportará ventajas. Más aún, son muchas las desventajas que acarrea, puesto que los cruzamientos o mutaciones pueden producir recorridos ilegales, ciudades duplicadas u omitidas, que exigirán la aplicación de algoritmos especiales de reparación.

Se han propuesto, en los últimos años varios tipos de representaciones relación al TSP de las cuales a su vez se analizan las siguientes: **por adyacencia, ordinal** y **por caminos** [2].

4.3 Representación por adyacencia.

El recorrido posible se representa como una lista de n ciudades donde la ciudad j es listada en la posición i si y solo si el recorrido conduce de la ciudad i a la j. Por ejemplo el vector

(2 4 8 3 9 7 1 5 6)

representa el siguiente recorrido

1-2-4-3-8-5-9-6-7

La desventaja que tiene esta representación es que no soporta el cruzamiento clásico y es necesario aplicar algoritmos de reparación para evitar recorridos ilegales. La ventaja es que permite un análisis de esquemas de cromosomas.

4.4 Representación ordinal

El recorrido se representa mediante una lista de n ciudades donde el i-ésimo elemento de la lista es un número entre 1 y n-i+1. Además una lista base servirá de punto de referencia para las listas ordinales. Por ejemplo, suponiendo que la lista base esta dada por:

B=(1 2 3 4 5 6 7 8 9)

un recorrido como el siguiente:

1-2-4-3-8-5-9-6-7

se representa por una lista de referencias respecto de la lista base. Resultando:

l=(1 1 2 1 4 1 3 1 1)

donde el vector l debe ser interpretado como sigue:

El primer número de la lista l es 1. Entonces se toma el primer elemento de la lista B como la primera ciudad del recorrido. Además se elimina el elemento de B que queda ahora con ocho elementos de los cuales el primero es el 2. El recorrido parcial es

1

El segundo número de la lista l es también un 1. Se toma el primer elemento de la lista

B (que ahora es un 2) como segunda ciudad, y se lo elimina de la lista B. El recorrido parcial es

1-2

El tercer elemento de *l* es 2. Se toma el segundo elemento de B, que es 4. 4 es la tercera ciudad y se lo elimina de la lista B. El recorrido parcial ahora es

1-2-4

Así siguiendo, el recorrido final es

1-2-4-3-8-5-9-6-7

A pesar de lo rebuscado de esta representación, su principal ventaja es que soporta el cruzamiento clásico. Sin embargo los resultados experimentales que se han obtenido juntando la representación ordinal y cruzamiento clásico no han sido muy buenos para la resolución del TSP.

4.5 Representación por caminos

Esta es la representación de recorridos, si se quiere, más natural: un recorrido es representado simplemente por las ciudades que recorre en el orden que lo hace. Es decir el recorrido

5-1-7-6-4-2-3-9-8

es representado por la lista

(5,1,7,6,4,2,3,9,8)

Esta representación ofrece como ventaja su simplicidad inherente aunque no es posible aplicar el cruzamiento clásico. En general las representaciones de los cromosomas en TSP requerirán de adaptaciones en los operadores genéticos.

4.6 Operadores genéticos para TSP

4.6.1 Cruzamiento

La simplicidad de la representación por caminos es una ventaja en contraposición de la desventaja de no soportar cruzamiento clásico, por cuanto pueden generarse recorridos no válidos. Surge la necesidad por lo tanto de definir algoritmos de cruzamiento para resolver esta cuestión. Goldberg y Lingle [1] proponen el cruzamiento **PMX** (Partially-mapped Xover), Davis el **OX** (order Xover) y Oliver ha propuesto el cruzamiento cíclico **CX** (Cyclic Xover).

4.6.2 Operador de transformación parcial (PMX)

Se discute a continuación el cruzamiento PMX elegido para la implementación del AG aplicado al TSP. Debe recordarse que el objetivo del cruzamiento es generar hijos válidos, a partir de los cromosomas padres. El algoritmo trabaja eligiendo una subsecuencia del recorrido a través de dos puntos de corte, intercambia la información entre los padres e intenta mantener el orden y la posición de tantas ciudades como sea posible asegurando que el recorrido obtenido sea válido. Por ejemplo, si los dos padres seleccionados son

p1=(1 2 3 | 4 5 6 7 | 8 9)

p2=(4 5 2 | 1 8 7 6 | 9 3)

donde también se han indicado los dos puntos de cruce (representados por '|'), la descendencia de estos padres se logra de la siguiente manera: primero se intercambian los genes entre los dos puntos de cruce. Comienzan a formarse dos hijos h1 y h2 de los que por el momento sólo se conocen los genes entre los puntos de cruce

h1=(x x x | 1 8 7 6 | x x)

h2=(x x x | 4 5 6 7 | x x)

Este intercambio define una tabla de correspondencias: 1↔4, 8↔5, 7↔6 y 6↔7. Esta tabla se utilizará para resolver conflictos.

Pero antes de utilizar la tabla se pueden llenar las ciudades que no llevan a conflicto; esto es que no haya ciudades repetidas en cada hijo. No hay problema para 2, 3 y 9 de p1 ni para 2, 9 y 3 de p2. Hasta ahora los hijos tomarían la siguiente forma:

h1=(x 2 3 |1 8 7 6 | x 9)

h2=(x x x | 4 5 6 7| 9 3)

Se puede observar que el primer elemento de p1 no puede transferirse a h1 por cuanto la ciudad 1 aparecería repetida; aquí es donde la tabla de correspondencias entra a tallar: la ciudad 1 se corresponde con la 4, por lo tanto el primer elemento de h1 será 4. Con un análisis similar el octavo elemento de h1 no puede ser 8; la tabla de correspondencia indica que h1 tendrá la ciudad 5 en el octavo lugar. Trabajando en forma similar con h2, la descendencia de p1 y p2 es

h1=(4 2 3 |1 8 7 6 | 5 9)

h2=(1 8 2 | 4 5 6 7| 9 3)

La ventaja que ofrece PMX es que permite explotar simultáneamente en valor y orden

siempre que se cuente con un buen plan de reproducción.

4.6.3 Operador de cruzamiento por orden (OX)

Este operador fue propuesto por Davis y construye la descendencia preservando el orden relativo de las ciudades de los padres. Siguiendo el ejemplo anterior, si los padres son

p1=(1 2 3 | 4 5 6 7 | 8 9)

p2=(4 5 2 | 1 8 7 6 | 9 3)

la descendencia se produce de la siguiente manera: primero se copian a los hijos las ciudades entre los puntos de cruce generados aleatoriamente, obteniendo

h1=(x x x | 4 5 6 7 | x x)

h2=(x x x | 1 8 7 6 | x x)

Posteriormente se realiza la secuencia de ciudades en uno de los padres comenzando por el segundo punto de cruce y recomenzando por el principio del vector cuando se alcanza el fin del mismo. Esto para el segundo padre arroja la siguiente secuencia:

9-3-4-5-2-1-8-7-6

Ahora se eliminan de la secuencia las ciudades que existen ya en el otro hijo, parcialmente armado (en este ejemplo h1 tiene ya las ciudades 4, 5, 6 y 7). La secuencia modificada que se obtiene es

9-3-2-1-8

Finalmente esta secuencia de h2 se coloca en h1 comenzando en el segundo punto de corte y siguiendo cíclicamente por el principio de h1 hasta obtener el primer hijo definitivo:

h1=(2 1 8 | 4 5 6 7 | 9 3)

Trabajando análogamente se obtendrá para el segundo hijo

h2=(3 4 5 | 1 8 7 6 | 9 2)

OX permite explotar una propiedad de la representación por caminos que es que importa el orden en el cual se recorren las ciudades y no sus posiciones. En otras palabras los recorridos

R1=(9 3 4 5 2 1 8 7 6)

R2=(4 5 2 1 8 7 6 9 3)

tienen el mismo costo.

4.6.4 Operador de cruzamiento cíclico (CX)

Fue propuesto por Oliver y su característica principal es que trabaja a nivel de posiciones de ciudades y no a nivel de subsecuencias: cada hijo tendrá la ciudad *i* en la misma posición que uno de sus padres. Esto permite garantizar que los cromosomas hijos hereden de sus padres todas las posiciones de las ciudades. El método, aplicado al un ejemplo es el siguiente: tomando p1 y p2 como cromosomas padres

p1=(1 2 3 4 5 6 7 8 9)

p2=(4 5 2 8 9 7 6 1 3)

se toma la primera ciudad de p1 (la ciudad 1) y se la ubica en h1:

h1=(1 x x x x x x x x)

para elegir la siguiente ciudad se toma la que está en la misma posición que 1 pero en el segundo padre, en este caso es 4. La ciudad 4 se coloca en h1 **en la posición** que tiene la ciudad 4 en p1

h1=(1 x x 4 x x x x x)

En p2 la ciudad en la misma posición que 4 es la ciudad 8. Se ubica, por lo tanto el 8 en la misma posición que le corresponde en p1

h1=(1 x x 4 x x x 8 x)

En p2 la ciudad en la misma posición que 8 es la ciudad 1. Como 1 ya está en h1, el proceso se detiene completando las ciudades restantes con las que se encuentran en las mismas posiciones en p2:

h1=(1 5 2 4 9 7 6 8 3)

Análogamente se obtiene h2 como

h2=(4 2 3 8 5 6 7 1 9)

Obsérvese que h1 hereda las posiciones de las ciudades 1, 4 y 8 de p1 y las de 2, 3, 5, 6, 7, 9 de p2mientras que h2 hereda justamente lo inverso: las posiciones de las ciudades 1, 4 y 8 son heredadas de p2 mientras que las posiciones de las ciudades 2, 3, 5, 6, 7, 9 de p1. Lo que preserva este operador son las posiciones absolutas de los elementos de la secuencia de los padres.

4.6.5 Operadores de Mutación

El operador de mutación tampoco puede ser el clásico en el TSP. La alteración de una ciudad producirá un recorrido ilegal por cuanto no puede haber ciudades repetidas y todas las ciudades deben estar en el recorrido. Por lo tanto la mutación debería producir un cambio aleatorio de orden entre pares de ciudades. Una forma de mutación que se

propone es generar dos números enteros aleatorios *c1* y *c2* entre 1 y el largo del cromosoma e intercambiar las ciudades apuntadas por aquellos. De esta forma un cromosoma

h2=(1 8 2 4 5 6 7 9 3)
^ ^

con los números aleatorios n1=2 y n2=6, producirán el intercambio de los elementos 8 y 6 en h2 resultando

h2'=(1 6 2 4 5 8 7 9 3)
^ ^

Otro operador de mutación, podría resultar del intercambio de los genes acotados por *c1* y *c2*. En el mismo ejemplo el cromosoma

h2=(1 8 2 4 5 6 7 9 3)
^ ^

se transforma, mediante los intercambios 6-8 y 2-5 en

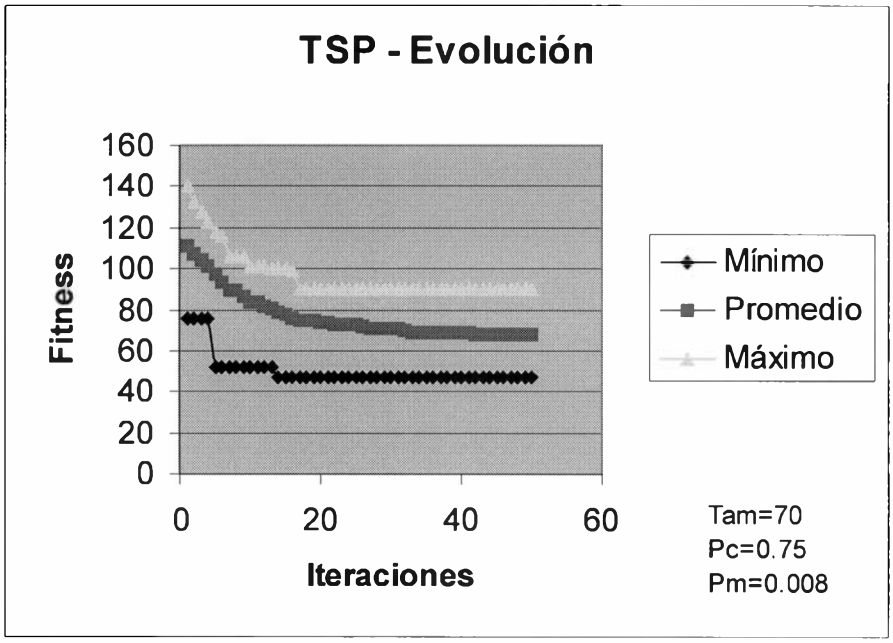
h2'=(1 6 5 4 2 8 7 9 3)
^ ^

4.6.6 Experimentación y análisis de resultados

El conjunto de experimentos fueron realizados alterando los parámetros que gobiernan al AG para poder realizar la comparación de resultados. En los casos de número de ciudades relativamente pequeño, los resultados pueden compararse con los obtenidos por otros métodos clásicos, aún por búsqueda exhaustiva. Dado que los operadores son probabilísticos las curvas se construyeron considerando un conjunto de ejecuciones del algoritmo y relevando los valores de fitness máximo, promedio y mínimo opara cada generación. La figura 4.1 muestra los resultados de uno de los experimentos considerando una probabilidad de cruzamiento de 0.75 y una probabilidad de mutación de 0.008 como operadores genéticos que afectan a una población constante de 70 individuos, para un TSP de 10 ciudades .

Se puede determinar que el parámetro de probabilidad de cruzamiento, debe ser adaptado para lograr una solución de compromiso. En efecto, si la probabilidad de cruzamiento tiende a 1 se aumenta la proporción de individuos a cruzarse, pero como

contrapartida se incrementa el costo computacional. Si, por otra parte, la probabilidad de cruzamiento tendiera a cero, no se generarían nuevos individuos por obra de este operador; la población tiende a estancarse sin el aporte de nuevos individuos que favorezcan la evolución.



En algunas corridas se observó una convergencia a un subóptimo. Es decir, en la comparación de los resultados por AG y por técnicas clasicas existe un apartamiento del óptimo global. Sin embargo, el hecho de tener toda una población de cromosomas clasificadas por fitness da un *ranking* de soluciones que pueden tomarse si el criterio adoptado permite un apartamiento aceptable del óptimo y tomar la próxima solución como una alternativa.

Otra cuestión a tener en cuenta es la dicotomía diversidad genética versus convergencia de la población. Una mayor diversidad permite mayor número de individuos distintos y facilitar la convergencia; pero la misma se dificulta si dicha diversidad se lleva a un extremo puesto que la dispersión de los *fitness* de la población es muy alta. La *"voracidad"* de un operador de selección en términos de cuan rápidamente se uniformiza la población con copias de los mejores individuos es la llamada *presión selectiva.* Una alta presión selectiva trae como consecuencia una mayor pérdida de la diversidad poblacional lo cual lleva a una convergencia prematura a un subóptimo. Para evitar este problema existen dos caminos:

a) Incrementar el tamaño de la población y

b) aumentar la probabilidad de cruzamiento y/o de mutación de los operadores de cruzamiento y mutación.

Ambas soluciones deberán ser contrastadas con los incrementos de costo computacional que traen aparejadas y buscar una solución de compromiso que balancee costos y beneficios que dependerá de las exigencias del problema en cuanto a la calidad requerida a las soluciones. Un análisis de la teoría de la presión selectiva puede encontrarse en Goldberg y Deb [17].

CAPÍTULO V

5 OBTENCION DE LA RUTA ÓPTIMA EN UNA RED

5.1 Introducción

Uno de los problemas a resolver en análisis de redes es la obtención de una ruta óptima que transporte un determinado tráfico desde un nodo origen a un nodo destino. La función objetivo en este caso es el costo del transporte y será igual a la suma de los costos asociados a las ramas por donde pasa el tráfico. La ventaja de esta optimización es obvia: redunda en un beneficio económico para un cierto flujo de datos dado. Se presenta en este capítulo, en primer lugar, la terminología utilizada para el análisis de redes; posteriormente se analiza la implementación de un algoritmo clásico desarrollado por Dijkstra y propuesto en Bronson [8] y finalmente se comparan los resultados obtenidos a través de un algoritmo genético implementado ad-hoc.

5.2 Diferencias con TSP

Hay una relación entre el problema del viajante y el de la obtención de la ruta óptima. El TSP configura una base que apoya y sirve de prueba a un AG puesto a buscar soluciones a un problema de secuenciación. Sin embargo y, como podría esperarse, hay algunas consideraciones que deben hacerse en el problema de ruta óptima que le otorgan características propias.

En primer lugar, el camino que une dos puntos en una red no necesariamente debe pasar por todos los nodos de ella. Incluso no es así en la mayoría de los casos. La representación del individuo, entonces, será diferente de la del TSP: los cromosomas involucrados en este AG tendrán longitud variable, aún dentro de la misma población.

Por otra parte, el TSP busca minimizar un costo global para todas las ciudades que están involucradas. La ruta óptima, si se quiere, tiene objetivos menos sociales y, sin importar lo que ocurra con el resto de la red, su propósito primordial es minimizar el costo de transportar un paquete entre dos puntos de la red.

5.3 Terminología utilizada

Para analizar una red se la considera como un conjunto de puntos llamados **nodos** y un conjunto de arcos llamados **ramas** que conectan pares de nodos. Una rama es **orientada** si se permite el flujo sólo en una dirección. Dos **ramas son conexas** si tienen un nodo en común. Una **ruta** es una secuencia de ramas tales que en la alternación de ramas y nodos no se repite ningún nodo. Una **red es conexa** si para cada par de nodos existe al menos una ruta que los conecte.

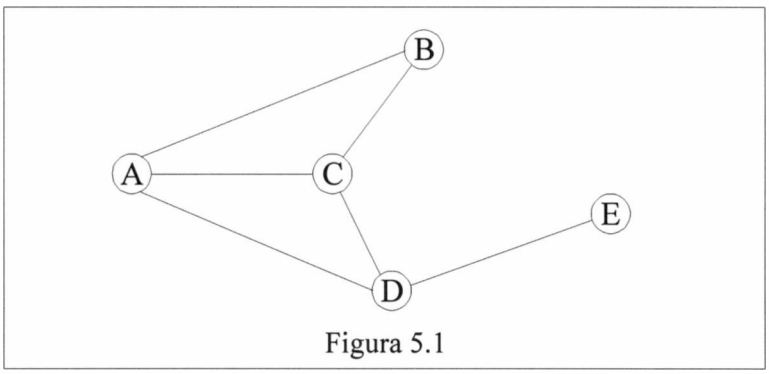


Figura 5.1

En la figura 5.1 las ramas AB y AC son conexas, mientras que las AB y CD no lo son. La red de la figura es una red conexa, puesto que dados dos nodos cualesquiera, existe un camino que los une a través de la red.

5.4 Utilización de una técnica clásica para la resolución del problema

El problema de la ruta óptima involucra a una red conexa con cada una de sus ramas con un costo no negativo asociado. A un nodo se le denomina *fuente* y a otro *destino.* El objetivo es encontrar una ruta que una fuente con destino de tal manera que la suma de los costos, asociados con las ramas en la ruta, sea mínima.

El algoritmo propuesto en Bronson [8] es, en realidad, una versión del algoritmo de Dijkstra para la obtención de la mínima ruta. También puede encontrarse una versión del mismo en la obra de Tanenbaum [3]. La clave del algoritmo propuesto por está en otorgar costos a los nodos. El costo del nodo se lo otorgan los vecinos. Se va eligiendo el menor costo que puede ser otorgado a un nodo partiendo desde la fuente hasta llegar al destino

Para ilustrar esto, considérese la red de la figura 5.2. Identificados el nodo 1 como fuente y el 8 como destino, se parte del fuente. Este nodo no tiene quien le otorgue

costo. Por tanto se le asigna costo 0. A partir de esta asignación, en toda la red sólo el nodo 1 esta disponible para asignar costos a sus vecinos (los nodos 2 y 3).

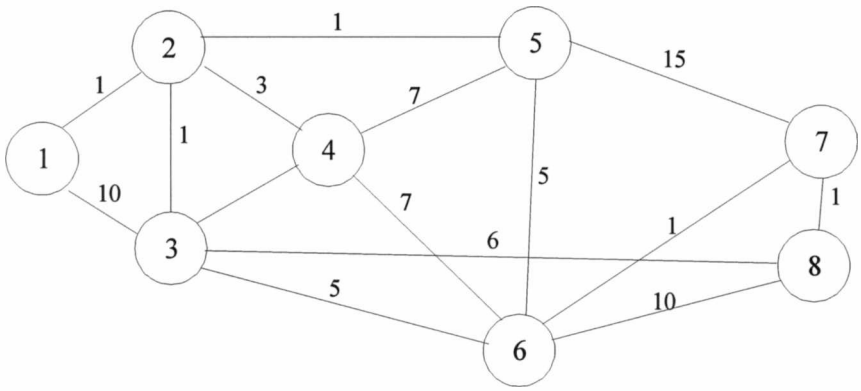


Figura 5.2. Red de prueba. El costo de transporte entre nodos está indicado en las ramas

Hasta ahora el nodo 2 solo puede recibir costo del 1, al igual que el 3. El costo que debería asignarse al nodo 2 es el costo del nodo 1 incrementado en el costo de la rama asociada (1-2). Ese valor es 0+1=1.

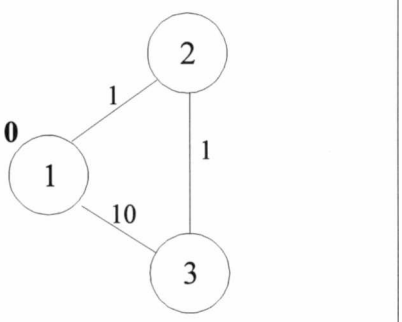


Figura 5.3

Procediendo en forma análoga el costo para el nodo 3 es de 10. Se asigna el **menor costo** al nodo correspondiente, quedando nodo1 con costo 0 y nodo 2 con costo 1.

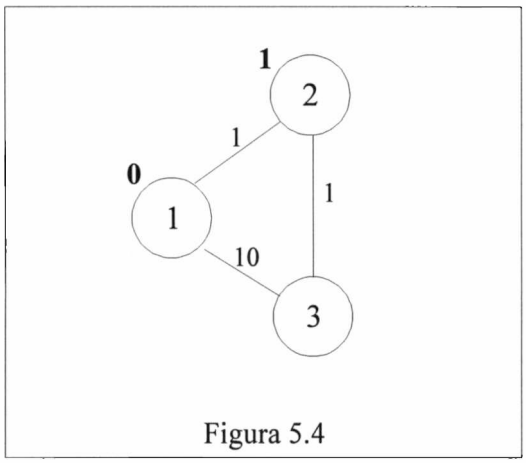


Figura 5.4

Posteriormente se consideran todos los nodos que pueden otorgar costos. En este caso los nodos 1 y 2. El nodo 1 otorga costo 10 a 3 y el nodo 2 otorga 2 al 5, 4 al 4 y 2 al 3. Hasta aquí hay un empate entre los nodos 5 y 3. En estos casos se resuelve arbitrariamente la cuestión. En el ejemplo se decidió por el nodo 5, que engrosará la lista de los nodos que otorguen costos.

Nuevamente se consideran todos los nodos que asignan costos y se selecciona el menor de todos ellos. A medida que se van cubriendo los nodos, se van marcando las ramas para no volver a utilizarlas. Y se lleva la cuenta de los costos asociados a cada rama.

Cuando el nodo al cual se le asigna costo es el destino, el algoritmo entra en su etapa final: el costo de la ruta óptima es el del costo del nodo destino y la ruta buscada se determina haciendo un camino hacia atrás desde el destino al origen.

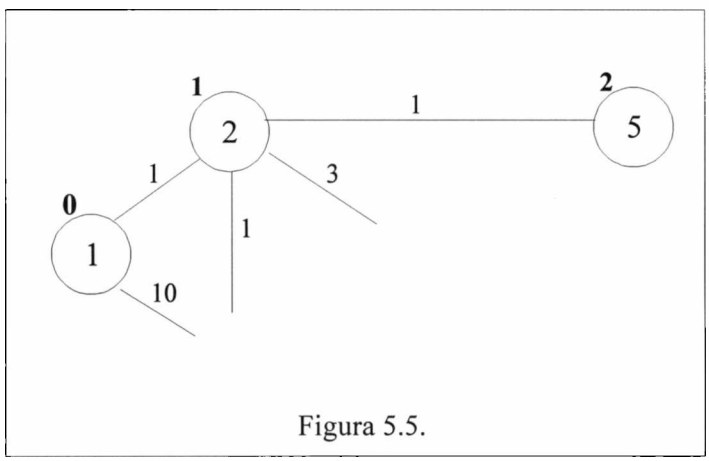


Figura 5.5.

El algoritmo completo para resolver el problema de la ruta óptima se encuentra en el apéndice.

5.5 Algoritmo genético para la determinación de la ruta óptima

La construcción del algoritmo genético para la resolución de un TSP sirve de base para la solución al problema de la ruta óptima. El problema es básicamente un problema de secuenciación. La mejor secuencia es la de menor costo. La diferencia entre el problema del TSP y el de la ruta óptima estriba, fundamentalmente, en que el largo del cromosoma debería ser variable. La ruta óptima no necesariamente será una que contenga todos los nodos de la red. De esta manera deberán considerarse cromosomas válidos a aquellos que no tengan todos sus genes con valor.

Para la realización de este trabajo se consideró un cromosoma donde es válido un nodo virtual numerado como 0.

Hay una consideración adicional que hacer para optimizar la resolución de este problema: en el caso del TSP se consideran cromosomas que representan rutas donde cada gen representa un nodo. El operador de cruzamiento trabaja sobre las posiciones relativas de los nodos en el cromosoma. En este caso, si bien el cromosoma esta formado también por genes que representan nodos, al momento de realizar el cruzamiento es más eficiente considerar los arcos que unen los nodos en una ruta, o pares de nodos. Esto último implica la reformulación del operador de cruzamiento para producir descendencia que considere mejores arcos en vez de mejores posiciones de nodos.

5.5.1 Representación del cromosoma

De lo expresado en el párrafo anterior surge la selección de la representación del cromosoma especificando la ruta que une origen con destino a través de una red conexa. De esta manera un cromosoma válido puede ser

$$P=(1,2,5,7,8)$$

Este cromosoma deberá incluir un relleno (*padding*) de ceros con el fin de mantener la representación matricial de la población. La longitud de todos los cromosomas está dada por el camino que involucra el máximo número de nodos posible. De esta forma P será representado como

P=(1,2,5,7,8,0,0,0)

5.5.2 Estructura del algoritmo genético

Determinada la representación de los cromosomas, se analizan a continuación la estructura funcional del algoritmo genético a aplicar, la cual es similar a la utilizada en la resolución de los otros problemas planteados con anterioridad. La filosofía del algoritmo se mantiene, aunque deben considerarse nuevos operadores, necesarios para cumplir la premisa de mantener una población de individuos válidos. Se proponen en este capítulo dos operadores para cruzamiento y mutación para resolver la cuestión planteada.

5.5.2.1 Selección de padres

El operador de selección utilizado emplea el método de selección proporcional ya explicado en el capítulo I. Los cromosomas de mejor ajuste tienen una probabilidad mayor de ser seleccionados como padres que, por intermedio de cruzamiento y mutación, engendrarán la próxima generación.

5.5.2.2 Cruzamiento por arcos

Tal como ya se mencionó, el cruzamiento por arcos permite que la descendencia de dos cromosomas padres contenga los arcos que están presentes en sus progenitores. Para evitar la repetición de nodos se construye una tabla o matriz de conexiones con los arcos de los progenitores. Por ejemplo, si los padres a cruzar fueran

P1=(1, 2 ,4 ,6 ,8)

P2=(1,3,4,5,6,7,8)

La tabla de conexiones de los padres sería la siguiente:

NODO	CONECTADO A
1	2-3
2	4
3	4
4	5-6
5	6
6	7-8
7	8
8	-

Se selecciona a continuación el nodo origen como primer nodo del hijo (todos los cromosomas comienzan en el nodo origen y terminan en el destino, tal como está propuesto el problema a resolver). En este caso el nodo que se selecciona es el 1. El hijo comienza a formarse

$$h=(1)$$

Ese nodo seleccionado se considera para inspeccionar los nodos conectados a él y se toma el criterio de seleccionar el nodo que conecte a su vez con el menor número de nodos. Esto disminuye la probabilidad de aislamiento de un nodo, lo cual exige una reparación del cromosoma agregando un arco que no está en ninguno de los padres. En caso de haber más de un nodo con el mínimo número de conexiones la selección se realiza al azar. En este caso el nodo que se elige es el 2 (tanto el nodo 2 como el 3, que son los conectados al 1, tienen solo una conexión con otros nodos).

Con el fin de evitar que el nodo seleccionado sea nuevamente elegido se elimina dicho nodo de la segunda columna de la tabla. Este nuevo nodo seleccionado es el próximo nodo del hijo el cual queda como

$$h=(1,2)$$

y la tabla modificada se transforma en

NODO	CONECTADO A
1	3
2	4
3	4
4	5-6
5	6
6	7-8
7	8
8	-

Si este fuera el nodo destino, el algoritmo concluye. En otro caso se prosigue con el análisis descripto hasta que se seleccione el nodo destino. En este caso debe proseguirse y dado que el nodo 2 solo conecta con el 4, el próximo es justamente el 4, con lo que

$$h=(1,2,4)$$

y la tabla queda

NODO	CONECTADO A
1	3
2	-
3	-
4	5-6
5	6
6	7-8
7	8
8	-

Teniendo en cuenta que el nodo 4 conecta con los nodos 5 y 6 y que a su vez 5 conecta con un solo nodo (el 6) y 6 conecta con dos (7 y 8), corresponde entonces seleccionar el nodo 5, obteniendo

$$h=(1,2,4,5)$$

y la tabla modificada

NODO	CONECTADO A
1	3
2	-
3	-
4	6
5	6
6	7-8
7	8
8	-

El nodo 5 conecta con el 6. El 6 es el próximo. Por lo tanto

$$h=(1,2,4,5,6)$$

y la tabla vuelve a actualizarse.

NODO	CONECTADO A
1	3
2	-
3	-
4	-
5	-
6	7-8
7	8
8	-

Finalmente el nodo 6 conecta con 7 y 8. 8 tiene 0 conexiones y 7 una. El próximo nodo es el 8, y dado que es el nodo destino, el algoritmo termina después de haber agregado dicho nodo. El hijo de p1 y p2 es, por fin,

$$h=(1,2,4,5,6,8)$$

El algoritmo de cruzamiento por arcos se incluye completo en el apéndice.

Nótese que, debido al criterio de elegir sólo el nodo con mínimas conexiones durante el proceso de construcción, el cruzamiento genera sólo un hijo. Sin embargo, sería posible generar un número variable de hijos en caso de considerar todas las interconexiones posibles. Este criterio distinto se aplicará en los futuros trabajos.

5.5.2.3 Mutación

El operador que se propone en este caso es aplicable a cualquier AG. En realidad se basa en un cambio del cromosoma, como cualquier operador de mutación, sólo que mucho más drástico, llamado macromutación. Simplemente se genera una ruta válida que reemplaza al individuo seleccionado para mutación. De esta manera se asegura el cumplimiento del requisito de validez del individuo sin necesidad de echar mano a la utilización de algoritmos de reparación, los cuales son muy costosos en tiempo. Este método tiene la ventaja de su muy fácil implementación dado que es simplemente un llamado a la rutina de generación de la población inicial con tamaño de población unitario. El resultado es un único individuo que se ubicará en la posición que le corresponde al que fue seleccionado para ser mutado.

5.5.3 Experimentación y análisis de resultados

El gráfico de la figuras 5.7a y 5.7b muestran el comportamiento del AG en la resolución del problema de la ruta óptima. En 5.7a se tomó una corrida cualquiera para observar la tendencia de convergencia. Se seleccionó una ejecución que muestre los distintos escalonamientos que se producen a medida que el algoritmo encuentra el mejor individuo en cada población.

Sin embargo, no todas las corridas dan como resultado una apariencia como la mostrada en la figura 5.7a. En algunos casos se observo una convergencia casi inmediata y en otros el valor optimo se obtuvo después de muchas iteraciones. También se observaron corridas en los que el algoritmo encontró un subóptimo. Para dar una idea más acabada de la convergencia del AG se realizaron 40 corridas del algoritmo, de las cuales se muestran en el gráfico de la figura los valores mínimo, promedio y máximo para cada iteración.

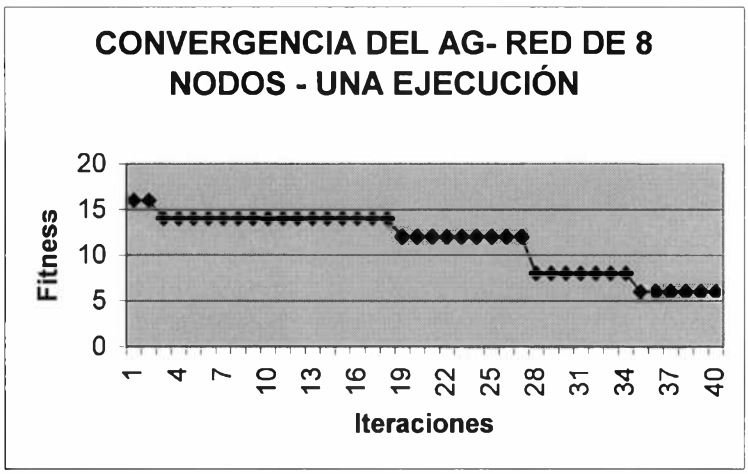


Figura 5.7[a]

El valor del óptimo real que, dado que el sistema es pequeño, puede obtenerse por búsqueda exhaustiva o bien por el método presentado en el punto 4.3, es 6. El tamaño de la población fue de 10 individuos, con probabilidad de cruzamiento de 0.75 y mutación variable con probabilidad de mutación inicial de 0.0015 pasando a 0.05 y un individuo de elite.

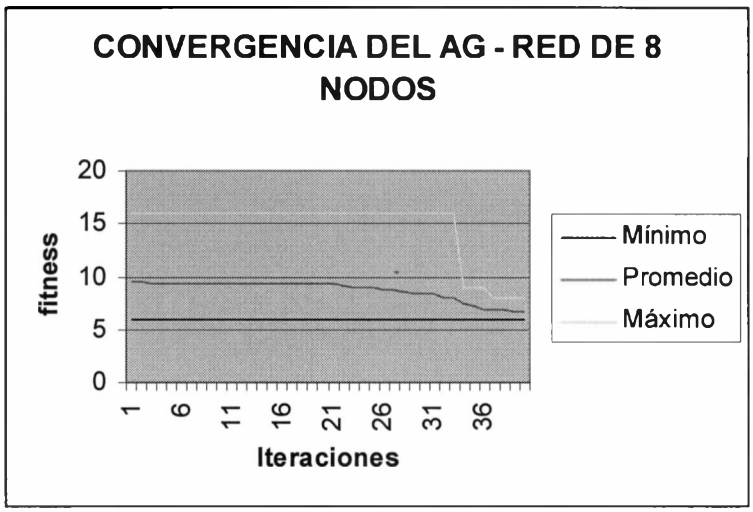


Figura 5.7[b]

Se observó una mejor convergencia del método aplicando criterio de mutación variable. Luego de un determinado número de iteraciones la población puede comenzar a perder diversidad genética. Si se está en el caso de un óptimo global esta es una relativamente buena señal, pero si el óptimo es local, el incremento de la probabilidad de mutación favorece a la diversidad genética. Puesto que el algoritmo trabaja con dos individuos de elite, el aumento de diversidad no afecta al óptimo ya encontrado. De esta manera se

evita la convergencia prematura al óptimo local como puede apreciarse en la figura.

El siguiente experimento se realizó con 12 nodos y con la red de la figura 5.8. Puede verse a simple vista que el camino mínimo tiene costo 9 dado que este ejemplo es una ampliación de la red del ejemplo anterior donde solo se conectan 4 nodos adicionales que conectan con los nodos 6, 7 y 8 de la red anterior. El camino de mínimo costo es p^*=(1 2 3 4 6 7 8 9 11 12) con costo $C_{mín}$=9.

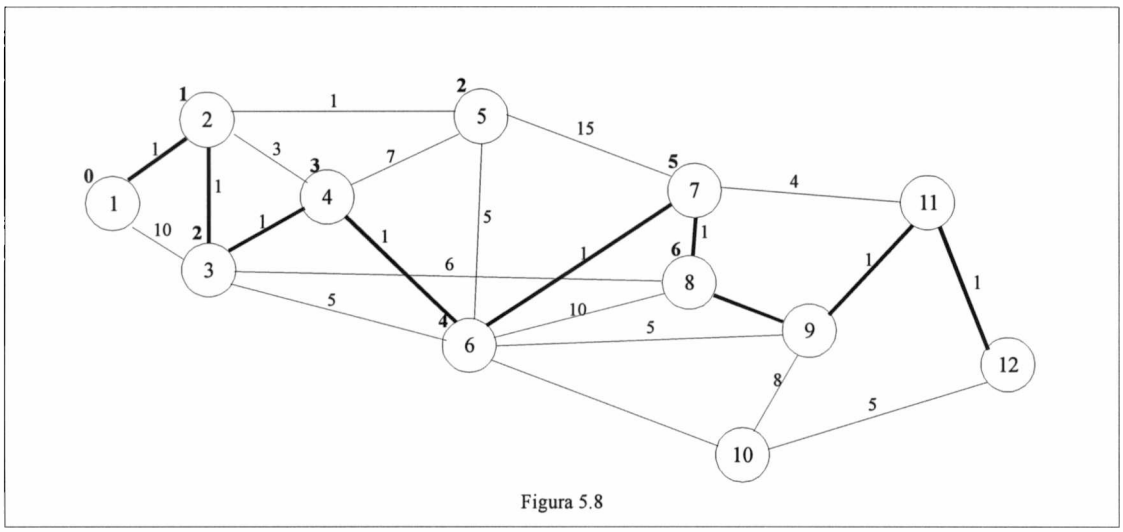


Figura 5.8

Nuevamente se grafican los valores mínimo, máximo y promedio de cada iteración en cada una de las 40 ejecuciones del algoritmo. En este experimento se tomo como tamaño de la población 10 individuos, una probabilidad de cruzamiento de 0.75, mutación variable y un individuo de elite. Los resultados pueden apreciarse en la figura 5.9.

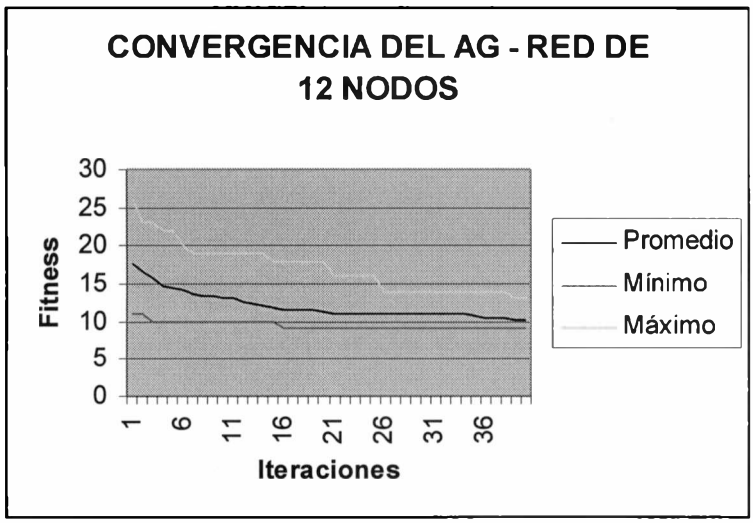


Figura 5.9

5.5.4 Comparación con técnicas clásicas

Se realizaron también comparaciones con el método clásico descrito anteriormente. Para ello se diseñó una red en módulos y se realizaron corridas con un número de módulos creciente, incrementando la complejidad de la red. En los casos de redes pequeñas, el algoritmo clásico ostenta mejores tiempos con la ventaja de obtener el resultado exacto.

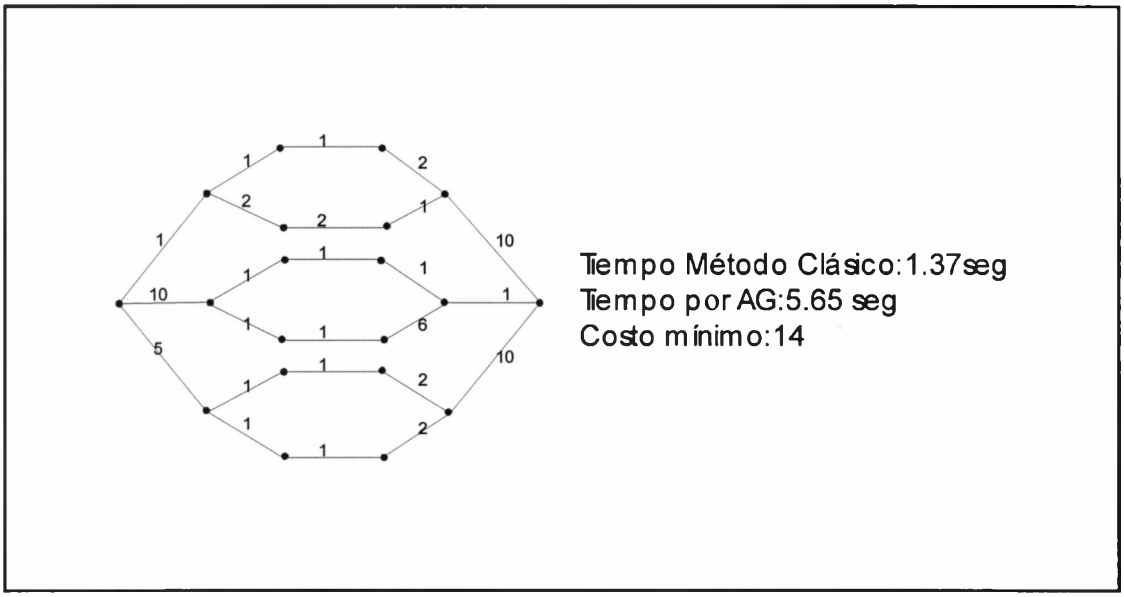


Figura 5.10. modulo de prueba para comparación de los algoritmos

Como puede apreciarse en la figura 5.10, los tiempos requeridos por los algoritmos clásico y genético no benefician en lo más mínimo a este último: ambos alcanzaron el mismo valor de mínimo costo pero el AG tarda más de cuatro veces más.

Sin embargo, el algoritmo genético comienza a imponerse cuando el tamaño y complejidad de la red se incrementa. Los tiempos comparativos pueden apreciarse en la figura 5.11 en la cual se contrastan las dos técnicas.

En dicha figura se grafica en abscisas el grado de complejidad de la red y en ordenadas los tiempos. La complejidad de la red se fue incrementando en módulos de una red prefijada que se anexaron a la correspondiente al caso anterior. Puede observarse un buen comportamiento del AG para redes más complejas y un punto de cruce por debajo del cual no es conveniente la aplicación del AG.

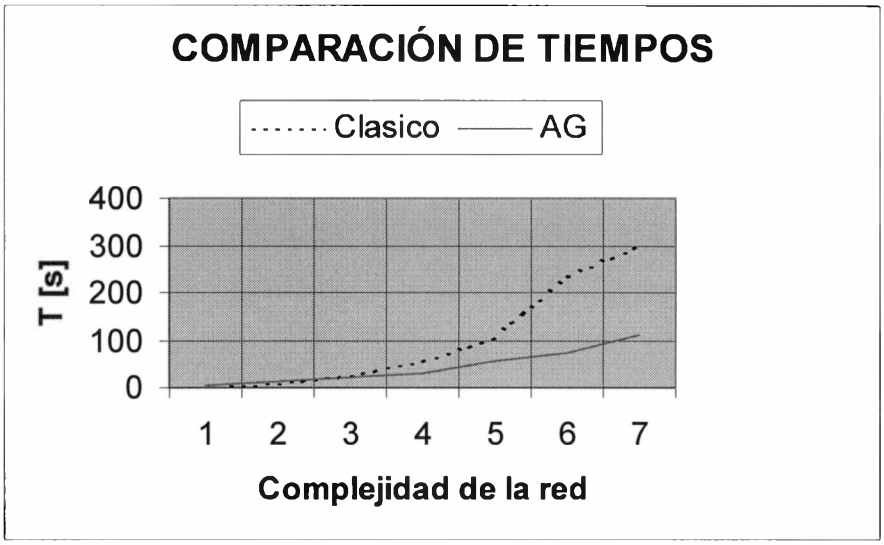


Figura 5.11

En cuanto a la calidad de los resultados obtenidos, se observaron variaciones con respecto al óptimo obtenido, por el AG; es decir, en algunos casos el algoritmo convergió a un subóptimo. Sin embargo dicho apartamiento del óptimo cayó dentro de valores que pueden considerarse admisibles. La figura 5.12 muestra comparativamente los valores óptimos alcanzados por los dos métodos.

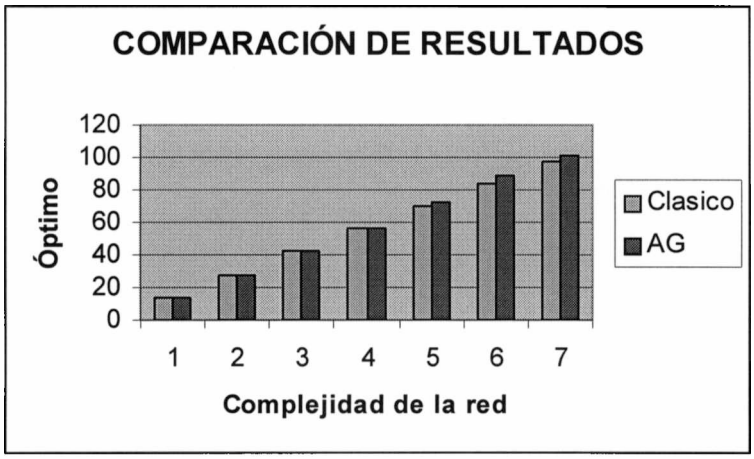


Figura 5.12

Por otra parte el AG brindó un conjunto de soluciones dentro del espacio factible permitiendo la construcción de una tabla de *ranking* con otras posibles soluciones al problema aunque hubiera un apartamiento del óptimo global.

5.5.5 Asignación de costos

Los AG planteados en este capítulo se basan, para evaluar el *fitness* de un cromosoma en la matriz de costos. Como se construye dicha matriz es un problema ajeno al AG. Sin embargo debe establecerse un criterio para la determinación del costo entre cada para de nodos.

En algunos casos el costo es tomado como inversamente proporcional a la velocidad de transferencia o proporcional al tiempo de transferencia de N bytes o bien en función de las tarifas de conexión.

Otros trabajos presentan una estructura más bien discreta y asignan los costos en función de las características de las líneas de transmisión. King-Tim Ko [4] propone la utilización de costos por distancia, parametrizados en velocidad de transferencia entre los nodos: 6 Mbps cuestan 1 unidad por kilómetro, 45 Mbps cuestan 4 unidades por Km, 150 cuestan 9 u/Km, etc.

También puede realizarse una métrica más compleja que considere otros elementos a tener en cuenta en una estructura de costos que incluso puede dar lugar a un análisis más dinámico. En efecto, si dentro de la métrica e incorpora el tráfico esperado distribuido en el intervalo de tiempo más conveniente, se puede prever caminos alternativos que consideren la variación de tráfico e incluso, llevar cuenta de las curvas de demandas que puedan proveer de la suficiente información para utilizar predictores que actúen directamente corrigiendo la matriz de costos en forma dinámica.

5.6 Conclusiones

El problema de la ruta óptima es un buen ejemplo para analizar el comportamiento de un AG sin triunfalismos. En comparación con otras técnicas de optimización, gana y pierde dependiendo de la complejidad del problema. Lo importante es que siempre se obtienen conjuntos soluciones factibles y, cuando el sistema es más complejo, se obtienen soluciones más rápidamente.

CAPÍTULO VI

6 CONCLUSIONES

En función de lo expresado en el desarrollo de esta tesis pueden arribarse a las siguientes conclusiones:

Se presenta una revisión general del estado del arte en algoritmos genéticos con énfasis en la aplicación de los mismos a los problemas de optimización en redes como dentro del marco de los problemas de secuenciación exponiendo las estructuras de representación de los cromosomas más utilizadas.

Se introducen algunos criterios de selección de los operadores de cruzamiento y/o mutación a utilizar dependiendo del problema a resolver y de la representación del cromosoma.

Se analiza el comportamiento de los AG en ejemplos concretos y se comparan con los resultados arrojados por técnicas clásicas. Como resultado de dicha comparación puede concluirse que el estudio de las técnicas de computación evolutiva presenta caminos alternativos a la resolución de problemas de optimización. En el caso particular del problema de encontrar la ruta óptima en una red tiene la ventaja adicional de seleccionar no solo el mejor cromosoma sino establecer además un *ranking* de soluciones que permitiría, en algunos casos,. tomar la siguiente solución cuando la primera no es factible.

Los algoritmos genéticos tienen la ventaja de no incorporar ninguna restricción a la función objetivo. La misma puede no ser continua ni derivable, sino tener cualquier característica.

Una desventaja de los AG estriba en que el óptimo global no siempre puede alcanzarse con cualquier precisión: cuando un problema particular esta asociado con un método que encuentra soluciones con un error menor al admisible en tiempos aceptables, es

conveniente la aplicación de dicho método. Cuando las buenas soluciones demandan tiempos demasiado largos o cuando los requerimientos de la función objetivo no pueden ser cumplidos, justamente por las características de dicha función, los algoritmos genéticos entran a tallar proporcionando no sólo una solución al problema sino un conjunto de ellas.

Se analiza en profundidad el problema de la obtención de la ruta de mínimo costo en redes. No obstante otros problemas de optimización pueden encontrar solución por medio de los algoritmos genéticos como en los ejemplos que analiza el capítulo III. Problemas de planeamiento, topologías, localizaciones, etc. pueden ser afrontados con esta técnica. La clave para la obtención de los mejores resultados tiene que ver con la selección apropiada del cromosoma y los parámetros que gobiernan el AG.

Queda planteado el tema una vez más. No está aún dicha la última palabra en algoritmos genéticos. No está aún finalizada su formalización y quedan propuestos, con el objeto de dar continuidad al presente trabajo los siguientes problemas y/o tareas de investigación:

a) Profundizar más en las tareas de clasificación de las representaciones utilizadas en los problemas abordados en particular y en problemas de secuenciación en general. Cuanto más se conozca acerca de ellas más se facilita la identificación de la relación representación-problema.
b) Continuar el desarrollo de los AG considerando otros operadores y aplicarlos a otros problemas de optimización.
c) Avanzar en el estudio de técnicas combinadas que permiten explotar el conocimiento acerca del problema que se resuelve, permitiendo mejorar en función de dichos conocimientos los operadores involucrados. De esta forma se puede trabajar con operadores de cruzamiento y mutación inteligentes que permitan disminuir los costos de computación.
d) Experimentar en el trabajo de estos algoritmos en procesamiento paralelo. Los AG tienen la ventaja de poder distribuir el procesamiento que mantiene a la población con suma facilidad. Los costos en tiempo podrían reducirse drásticamente mediante el procesamiento paralelo.
e) Profundizar en las estrategias y mecanismos de selección de individuos dado que

dicho operador aparece como uno de los motores principales para facilitar la explotación del espacio de búsqueda y balancear la presión selectiva y la diversidad genética armoniosamente hasta llegar a la obtención de las soluciones.

No se pretende en este trabajo reemplazar técnicas ya existentes y probadas de optimización sino visualizar estas técnicas emergentes donde lo probabilístico juega junto a lo determinístico y explorar críticamente otros caminos... aunque éstos sean *azarosos*.

Referencias y bibliografía

1. David Goldberg, Genetic Algorithms in Search, Optimization & Machine Learning. Addison Wesley. (1989).
2. Zbigniew Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs. Springer. 1995
3. A. Tanenbaum. Computer Networks. Segunda edición. 1991. Prentice Hall
4. King-Tim Ko,Kit-Sang Tang, Cheung-Yau Chan Kim-Fung Man, Sam Kwong. "Using Genetic Algorithms to design mesh Networks".Computer.V30 N°8. agosto 1997.
5. Kamlesh Mathur, Daniel Solow. Investigación de Operaciones. Prentice Hall. 1996.
6. Floudas Christodoulos, Nonlinear and Mixer-Integer Optimization, *Oxford University Press*, New York (1995).
7. Douglas Comer. Internetworking with TCP/IP. Vol 1. Prentice Hall. 1995.
8. Richard Bronson, Investigación de Operaciones, Mc. Graw Hill, 1993.
9. Munetono M, Takai Y y Sato Y., *"A Genetic Approach to Dinamic Load Balancing in a Distributed Computing System"*. 1° IEEE CEC, Junio 1994, Vol I. Pp 418-421.
10. Mansour, N, Fox, G., *"An hybrid genetic algorithm for task allocation in minicomputers"*. Proceedings of the fourth International Conference on Genetic Algorithms, pp 466-473, 1991
11. Kidwell, M. *"Using genetic algorithms to scheduledistributed tasks on a bus-based system"*. Proceedings of the fifth International Conference on Genetic Algorithms, pp 368-374, 1993
12. Esquivel, S., Leguizamón G., Gallard H. *"A hybrid strategy for load balancing in distributed system enviroments"*, Proceedings of then fourth IEEE International Conference on Evolutionary Computation ICEC 97),pp 127-132, 1997.
13. Díaz V, Herrera N, Leguizamón G., Gallard R. *"Algoritmos Genéticos aplicados a un problema de optimización combinatorial"*. Anales congreso Infocom'95 pp 321-331. Buenos Aires, 1995

14.Michalewicz, Z. Esquivel S., Gallard R., Michalewicz M, Tao, G. *"The Spirit of Evolutionary Algorithms"*. De próxima publicación.

15.Narshing Deo, Graph Theory with Applications to Engineering and Computer Science. Prentice-Hall, 1974.

16.Sen S., Shankar, N, Deb, K. *"Sensor Network design of linear processes using genetic algorithms"*. Computer Chemistry Engng. Vol 22. N° 3 pp 385-390, 1998.

17.Goldberg D., Deb, K. "A comparative Analysis of selectioon schemes used in Genetic Algorithms". Foundations of Genetic Algorithms, ed Rawlins (San Mateo CA:Morgan Kaufmann) pp 69-93.

18.K.S. Tang, K.F. Man and S. Kwong. "Genetic Algorithms and their applications". IEEE signal processing magazine. Noviembre 1996.

19.Baker, J. E. "Reducing Bias and Ineficiency in the selection algorithm". Proc of the 2nd International conference on Genetics Algorithms (Cambridge MA, 198) ed Grefenstette (Hillsdale, NJ Erlbaum) pp 14-21.

20.De Jong K."Genetics Algorithms are nor function optimizers". Foundations of Genetics Algorithms 2, ed L D Whitley. San Mateo CA: Morgan Kaufmann. pp 5-17.

21.De la Maza, M., Tidor, B."An analysis of selection procedure with particular attention paid to proportionaland boltzman selection"Proc of the 5th International Conference on Genetic Algorithms (urbna Champaign, IL, July 1993) ed S. Forrest. San Mateo CA. Morgan Kaufmann. pp 124-131.

22.Starkweather T, Mc Daniel S, Mathias K Whitley D and Whitley C – A comparison of genetic sequencing operators. Proc of the 4th Int. Conf. On Genetic Algorithms (San Diego, CA, 1991) ed R. K. Belew and L. B. Booker (San Mateo CA: Morgan Kaufmann) pp 69-76

23.Holland J. H. *Adaptation in Natural and Artificial System*. University of Michigan Press, Ann. Arbor, 1975.

24. Hernández, J.- Brignone, S.- Ruetsch, L.- Moitre, D.- Aromataris, L. *"Planificación de la Operación de Sistemas de Energía vía Optimización Evolutiva"*. **Anales del 7mo. Congreso Latinoamericano de Control Automático**. Vol. I, pp. 266-271. Buenos Aires, Argentina, 1996.

25. "DESPACHO ECONÓMICO DE UN SISTEMA DE SUMINISTRO DE ENERGÌA ELÈCTRICA: UN ENFOQUE COMBINADO. **VI Congreso Nacional de Ingenierìa Mecànica, Elèctrica y Ramas Afines. Bs As - III Congreso Nacional de Mantenimiento.** Bs As.

26. Moitre, D.- Brignone, S.- Ruetsch, L.- Maldonado, F.- Hernández, J.- Aromataris, L. *"Técnicas Emergentes en la Optimización de Sistemas de Suministro de Energía Eléctrica"*. **Ponencias del 2do. Congreso Latinoamericano de Distribución de Energía Eléctrica**. Viña del Mar, Chile, 1996.

27. Moitre, D.- Hernández, J.- Maldonado, F.- Ruetsch, L.- Aromataris, L. "Optimización del Despacho Económico de un Sistema de Suministro de Energía Eléctrica combinando técnicas deterministicas y estocásticas". **Resúmenes de las Sesiones Técnicas del Congreso: Transiciones Energéticas en México, Centro y Sudamérica** . Pag. 23-24. México, 1996.

28. Moitre, D.- Hernández, J.- Aromataris, L.- Rodriguez, G. *¨Programación de la Operación Optima Económica de Sistemas de Potencia sujeta a restricciones energéticas: una aplicación de Algoritmos Genéticos¨*. **RVP'97 IEEE - Sección México. Capítulo de Potencia.** Acapulco, México. 1997.

29. Riubrugent, J.-Moitre, D.- Hernández, J.- Aromataris, L. *¨Control de Sistemas de Potencia: aplicación de Algoritmos Genéticos a la Programación de la Operación¨*. **Memorias del VIII Simposio de Ingeniería Eléctrica .** Trabajo Técnico EEE 34. Santa Clara, Cuba. 1997.

30. Riubrugent, J.-Moitre, D.- Hernández, J.- Aromataris, L. ¨Control de Sistemas de Potencia: aplicación de Algoritmos Genéticos a la Programación de la Operación¨. **CIMAF'97 .** La Habana, Cuba. 1997.

31. Hernández, J.- Moitre, D.- Aromataris, L.- Rodriguez, G. *"Economic Optimal Dispatch of Electric Power Systems under Restrictions over the Environment: Application of Genetic Algorithms"*. **BIEL'97**. Buenos Aires, Argentina, 1997.

32. Moitre, D.- Hernández, J.- Aromataris, L.- Rodriguez, G. *¨Aplicación de Algoritmos Genéticos a la Coordinación Hidrotérmica de Corto Plazo¨*. **XII Congreso Chileno de Ingeniería Eléctrica.** Temuco, Chile. 1997.

33. Carnero M, Hernandez J, Sanchez M, Bandoni A. Diseño óptimo de

instrumentación en procesos.**VII Congreso Latinoamericano de transferencia de calor y materia "LATCYM 98".**

APÉNDICE

Evolución de las poblaciones

Evolución de las poblaciones TSP de 10 ciudades, con tamaño de la población de 10 individuos, probabilidad de cruzamiento 0.65, probabilidad de mutación de 0.001, 2 individuos de elite. Se muestran las poblaciones de las poblaciones 0, 20, 50 y 200.

POBLACION INICIAL										
INDIVIDUOS										FITNESS
1	5	8	9	4	10	7	2	3	6	257
1	5	9	7	8	10	2	3	4	6	204
1	4	6	10	2	7	3	5	8	9	278
1	3	10	6	2	7	8	5	4	9	293
1	8	10	6	3	5	7	9	4	2	239
1	7	5	9	6	8	3	4	2	10	205
1	8	3	9	10	2	4	7	6	5	159
1	6	3	9	2	7	10	5	8	4	250
1	10	8	4	9	6	5	7	3	2	291
1	5	4	6	7	9	10	3	8	2	165

POBLACION 5										
INDIVIDUOS										FITNESS
1	6	8	9	10	2	4	7	3	5	153
1	5	3	9	10	2	4	7	6	8	127
1	3	8	9	4	10	2	6	7	5	265
1	3	8	9	7	10	4	2	6	5	216
1	5	3	9	4	10	7	2	8	6	260
1	5	3	9	10	2	7	4	8	6	257
1	8	3	9	10	2	4	7	6	5	159
1	6	8	9	10	2	4	7	3	5	153
1	5	3	9	10	2	4	7	6	8	127
1	6	8	9	10	2	4	7	3	5	153

POBLACION 20										
INDIVIDUOS										FITNESS
1	5	6	9	2	10	4	7	8	3	116
1	5	6	9	2	10	4	7	8	3	116
1	5	6	9	2	10	4	7	8	3	116
1	5	6	9	2	10	4	7	8	3	116
1	5	6	9	2	10	4	7	8	3	116
1	5	6	9	2	10	4	7	8	3	116
1	5	3	9	2	10	4	7	6	8	127
1	5	6	9	2	10	4	7	8	3	116
1	5	3	9	10	2	4	7	8	6	183
1	5	6	9	2	10	4	7	8	3	116

POBLACION 200										
INDIVIDUOS										FITNESS
1	5	6	9	2	10	4	7	8	3	116
1	5	6	9	2	10	4	7	8	3	116
1	5	6	9	2	10	4	7	8	3	116
1	5	6	9	2	10	4	7	8	3	116
1	5	6	9	2	10	4	7	8	3	116
1	5	6	9	2	10	4	7	8	3	116
1	5	6	9	2	10	4	7	8	3	116
1	5	6	9	2	10	4	7	8	3	116
1	5	6	9	2	10	4	7	8	3	116
1	5	6	9	2	10	4	7	8	3	116

Se puede apreciar que al cabo de 20 iteraciones casi se uniformizó la población. Sin embargo se está en presencia de un óptimo local. Las próximas poblaciones no arrojan valores diferentes de fitness poblacional. Se manifiesta una clara y alta presión selectiva. Este experimento fue llevado a cabo con un tamaño de población relativamente pequeño. Como se explica en el desarrollo de la tesis un aumento del tamaño de la población o una estrategia diferente de los operadores de reproducción pueden disminuir la presión selectiva.

Las tablas siguientes muestran la evolución con los mismos parámetros salvo que el tamaño de la población es de 20 individuos. En la iteración número 20, puede verse una mayor diversidad genética que en el experimento anterior. Aún el fitness poblacional no se ha estacionado. En la iteración 50 se observa el estancamiento de la evolución. Por otra parte se observa un aumento en la precisión de la solución alcanzada.

POBLACIÓN INICIAL										
INDIVIDUOS										FIT
1	5	9	7	8	6	4	2	3	10	199
1	10	6	7	3	9	4	8	5	2	191
1	3	7	5	6	9	4	10	8	2	321
1	5	6	8	9	3	2	4	10	7	197
1	8	2	6	5	3	9	4	10	7	213
1	10	5	2	7	9	6	4	8	3	204
1	10	7	9	3	8	6	2	5	4	242
1	5	9	7	6	8	3	2	4	10	202
1	7	2	9	5	10	8	4	3	6	317
1	2	4	10	5	9	6	7	8	3	161
1	8	5	9	6	10	2	4	7	3	196
1	3	9	10	6	4	7	5	2	8	221
1	9	2	7	8	5	10	3	6	4	278
1	7	3	10	4	5	2	9	6	8	207
1	9	2	5	6	10	3	8	7	4	307
1	6	2	4	9	3	10	5	8	7	207
1	9	2	3	5	4	10	7	8	6	307
1	5	8	7	4	3	6	10	2	9	273
1	4	5	3	10	2	9	6	7	8	192
1	2	3	4	8	5	7	9	6	10	155

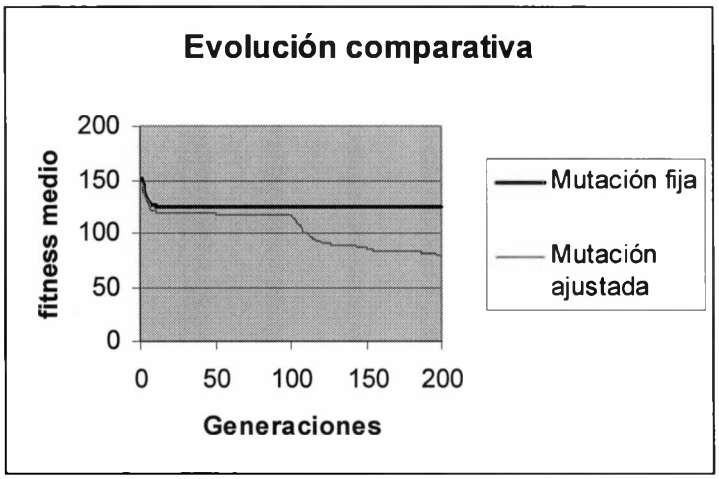
POBLACIÓN 20										
INDIVIDUOS										FIT
1	10	9	2	4	5	6	7	8	3	87
1	10	9	2	4	5	6	7	8	3	87
1	10	9	2	8	4	7	6	5	3	185
1	8	10	6	5	9	2	3	4	7	162
1	10	9	2	4	5	6	7	8	3	87
1	10	9	2	4	5	6	7	8	3	87
1	10	9	6	4	5	2	7	8	3	149
1	10	9	2	4	5	6	7	8	3	87
1	2	5	10	4	9	6	7	8	3	118
1	10	9	2	4	5	6	7	8	3	87
1	10	9	2	4	5	6	7	8	3	87
1	2	8	10	4	5	6	7	9	3	141
1	10	9	2	4	5	6	7	8	3	87
1	10	9	2	4	5	6	7	8	3	87
1	10	9	2	4	5	6	7	8	3	87
1	10	9	2	4	5	6	7	8	3	87
1	10	9	2	4	5	6	7	8	3	87
1	10	9	2	4	5	6	7	8	3	87
1	10	8	2	4	5	6	7	9	3	167
1	2	9	10	8	5	6	3	4	7	137

POBLACIÓN 50										
INDIVIDUOS										FIT
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69

POBLACIÓN 200										
INDIVIDUOS										FIT
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69
1	2	9	10	4	5	6	7	8	3	69

Variación de la probabilidad de mutación

Para la realización de este experimento se utilizaron los mismos parámetros de los experimentos anteriores. La probabilidad de mutación, sin embargo es incrementada drásticamente cuando la evolución se ha estancado. El siguiente gráfico representa una corrida con 10 ciudades, con tamaño de la población de 20 individuos, prob de cruzamiento 0.65, probabilidadd de mutación inicial de 0.001, y 2 individuos de elite.



En el caso de la curva de mutación fija puede observarse que el fitness promedio de la población se estabiliza en una generación determinada permaneciendo prácticamente sin cambios hasta la última.

En el caso de la mutación ajustada, se produjo un aumento de la probabilidad de mutación a 0.5 sólo en la generación 100. Puede observarse en la figura el efecto producido en una curva que hasta la centésima iteración pareciera haber convergido. La alta presión selectiva ha sido combatida aquí con una estrategia de cambio en los operadores de reproducción. Concretamente el aumento de la probabilidad de mutación en sólo una generación ha producido un recupero de la diversidad genética que, a su vez, ha conducido a una mejor solución.

Algoritmo de cruzamiento por arcos

```
function [hijo]=cruzarco(padre,madre,costo)
%Cruzarco.m
%Cruzamiento de cromosomas por arcos
% Llamada:
%     [hijo]=cruzarco(padre,madre,costo)
% Salida: una matriz con dos hijos con camino valido
%
n=size(costo,1);                                %Orden de la matriz de costos
conex=armacon(padre,madre,costo);               %Arma la matriz de conexiones
nodo_seleccionado=0;
ultimo=madre(largo_m);                          %nodo destino de la red
clear cant_cnx;                                 %cantidad de nodos conectado a uno
clear hijo

i=1;                                            %puntero al hijo
k=1;                                            %puntero a las conexiones de los padres

while (nodo_seleccionado~=ultimo)
  [nada,nodos_dek]=find(conex(k,:)~=infinito);  %Nodos conectados al nodo k
  if nodos_dek==ultimo                          %si el unico nodo conectado es el
    i=i+1;                                      %ultimo se asigna este al hijo
    hijo(1,i)=ultimo;                           %y se sale del lazo
    break
  end

  menor=Inf;
  for j=1:size(nodos_dek,2)                     %Para todos los nodos conectados a k
        nodo=nodos_dek(j);                      %1 nodo conectado a k (cuantos conecta?)
        [donde,aux]=find(conex(nodo,:)~=infinito);%vector con las colum de los elementos <> Inf
        cant_cnx=size(aux,2);                   %Cantidad de columnas en la fila j <>Inf
        if cant_cnx<menor & cant_cnx~=0         %busca el nodo de menor cantidad de ramas
                menor=cant_cnx;                 %conectados a el y que no este aislado
                nodo_seleccionado=nodos_dek(j);
        end
  end
                                                %El nodo seleccionado ya se incorpora al hijo
  i=i+1;                                        %puntero al hijo
  hijo(1,i)=nodo_seleccionado;                  %Carga del nodo seleccionado al hijo
  k=nodo_seleccionado;                          %Salir si el nodo seleccionado es el ultimo.
end

for k=i+1:size(padre,2)                         %Relleno con ceros el resto del cromosoma
  hijo(1,k)=0;
end
```

Cálculo del camino más corto en un grafo por el algoritmo de Dijkstra

Input: Un grafo D=(V,A) con costos c_{uv}>=0 sobre sus arcos y un nodo *s* ε V
Output: Las mínimas distancias entre *s* y todos los *v* ε V en un array ρ.

Set W:= {s};ρ[s]:=0;
For all y ε V-{s} do ρ[y]=c_{sy}
While W<>V do
    Find min{ρ[y]: y ~ε **W**}, say ρ[x];
    Set **W**:=**W union** {x};
    For all y ε **V-W do**
        ρ[y]:=min{ρ[y], ρ[x] + c_{xy}
    end
end