HANDBOOK OF GENETIC ALGORITHMS

Lawrence Davids

(1991)

RESUMEN
CAPITULOS 1-5

1. ¿ QUÉ ES UN ALGORITMO GENÉTICO ?

1.1 La Inspiración inicial: La evolución natural.

La evolución tiene lugar en los cromosomas, dispositivos orgánicos para codificar la estructura de los seres vivos. Algunas características de la evolución en los cromosomas son:

- La evolución natural actúa sobre los cromosomas más que sobre los seres vivos a los que codifican.
- La selección natural es el enlace entre cromosomas y la representación de sus estructuras decodificadas. Los procesos de selección natural producen que los cromosomas que codifican estructuras adecuadas se reproduzcan más a menudo que las que no las poseen.
- La evolución sucede en el momento de la reproducción. La mutación quizás cause que los cromosomas de los hijos sean diferentes a los de los padres. Los procesos de recombinación quizás creen cromosomas en los hijos que sean bastante diferentes a los de los padres, combinando material genético de los dos padres.
- La evolución biológica no tiene memoria.

En la década de los 70 John Holland intrigado por las características de la evolución natural pensaba que si se implementaban de forma adecuada dichas características dentro de un algoritmo matemático, podría ser utilizado para resolver problemas que eran difíciles de abordar con otro tipo de algoritmos. Esta nueva familia de algoritmos basados en las características de la evolución natural recibió el nombre de Algoritmo Genético (GA)

1.2 Una visión general de un algoritmo genético

Veamos la forma en que Holland empaquetó las características de la evolución natural en su algoritmo genético.

Los mecanismos que enlazan el algoritmo genético con el problema que se desea resolver son:

1. Una forma de codificar las soluciones al problema en forma de cromosomas.
2. Una función de evaluación o de coste que devuelve la valía de un cromosoma en el contexto del problema. El valor que se obtiene al evaluar dicho cromosoma en la función de coste se *fitness*.

Las técnicas para codificar las soluciones pueden variar de problema a problema y de un algoritmo genético a otro. Una codificación muy utilizada es considerar que los cromosomas son cadenas de bits. Se ha de tener en cuenta una técnica de codificación no trabaja igual de bien para todos los problemas.

Una función de evaluación toma como entrada un cromosoma y devuelve un número o una lista de números que son una medida de la adecuación de dicho cromosoma para resolver el problema. La función de evaluación juega el mismo papel que juega el

entorno en la evolución natural. Nos da un número que es utilizado en los procesos de reproducción.

En la el siguiente cuadro aparecen los principales pasos de un algoritmo genético:

Algoritmo Genético

1. Crear aleatoriamente una población inicial de cromosomas.
2. Evaluar cada cromosoma de la población inicial en la función de coste para obtener su *fitness*.
3. Crear nuevos cromosomas aplicándoles el operador de recombinación y el operador de mutación sobre los cromosomas actuales.
4. Eliminar miembros de la población (aquellos que presenten un menor *fitness*) para hacer sitio a los nuevos cromosomas.
5. Evaluar los nuevos cromosomas en la función de coste para obtener su *fitness* e introducirlos dentro de la población.
6. Si se han conseguido los objetivos se termina y se devuelve el mejor cromosoma, sino hay que volver al paso 3

Si todo va bien este proceso de evolución simulada nos devolverá poblaciones hijos de cromosomas que irán siendo mejores que las generación de inicio

1.3 La Anatomía de un algoritmo genético.

Se va a considerar el algoritmo genético GA 1-1, que consta de tres módulos:

- *Modulo de evaluación*: contiene una función de evaluación.
- *Modulo de población*: contiene una población de cromosomas y técnicas para crear y manipular esta población.
- *Modulo de reproducción*: contiene técnicas para crear nuevos cromosomas durante la reproducción.

Cuando el GA 1-1 se ejecuta, los tres módulos trabajan conjuntamente para producir la evolución de la población de cromosomas iniciales. Primero, la técnica de inicialización genera una población de 100 cromosomas.. Esta población es la primera generación de cromosomas. El paso siguiente es evaluar cada individuo de la población en la función de evaluación, lo que nos da el *fitness* de cada cromosoma, es decir, una medida de la capacidad de cada cromosoma para poder reproducirse.

Ahora GA 1-1 comienza una serie de ciclos donde se reemplaza la generación actual de cromosomas por otra nueva. En cada ciclo hay 50 eventos de reproducción, para producir 100 cromosomas nuevos. En el proceso de reproducción se selecciona dos cromosomas padres utilizando una técnica de selección de padres. El modulo de reproducción aplica el operador de recombinación en un punto y mutación. Para generar dos nuevos cromosomas denominados cromosomas hijos. Este proceso de generación de poblaciones continua hasta que se han generado 4000 individuos en la generación nº 40.

1.3.1 El modulo de evaluación

En GA 1-1 la función de evaluación es binaria y se denomina *f6*. La función *f6* hace tres cosas:

1) Decodifica una cadena de 44 bits en dos números reales.
2) Introduce los dos números reales dentro de una función matemática.
3) Retorna el valor de la función para esos dos números. Este número es la evaluación del cromosoma.

Cada cromosoma estará compuesto de una lista de números binarios.

El proceso de decodificación y la función de evaluación pueden ser cambiados, sin que se tenga que producir ningún cambio en los otros dos módulos, siempre que el número de bits en los cromosomas permanezca constante.

1.3.2 El módulo de población

La población inicial son cromosomas formados por cadenas de bits generados aleatoriamente.

La técnica de eliminación consiste en eliminar todos los cromosomas de la población anterior una vez que se ha generado una nueva población de cromosomas.

La técnica de selección de padres para generar cromosomas hijos es la conocida como *roulette wheel*.

Se utilizar el valor o *fitness* que se obtiene al evaluar el cromosoma en la función de evaluación para indicar la capacidad de reproducción del cromosoma.

El tamaño de la población se fija en 100 cromosomas. Si se generan 40 generaciones el número de individuos o cromosomas que se habrán generado será 4000.

1.3.3 El modulo de reproducción

Este modulo utiliza un operador que consta de recombinación en un solo punto de cruce y mutación. La probabilidad de recombinación se ha tomado como 0.65 y la probabilidad de mutación se ha tomado como 0.008.

1.4 Selección de padres, mutación y recombinación.

1.4.1 Selección de padres: Roulette Wheel

El propósito de la selección de padres es dar más oportunidades de reproducirse a aquellos individuos de una población que son los que mejor se ajustan tras haber sido evaluados por la función de evaluación. Una técnica muy útil para la selección de padres es la conocida como *roulette wheel*.

Algoritmo de selección de padres "*roulette wheel*"

1) Se suman los fitnes de todos los miembros de la población, llamamos a dicha suma *fitness* total.
2) Se genera un número aleatorio *n* entre 0 y el *fitness* total.
3) Se devuelve el primer miembro de la población cuyo *fitness* añadido al *fitness* del miembro anterior sea mayor o igual que *n*.

Un ejemplo de la aplicación de este algoritmo es el siguiente:

Cromosoma	1	2	3	4	5	6	7	8	9	10
Fitness	8	2	17	7	2	12	11	7	3	7
Fitness total corrido	8	10	27	34	36	48	59	66	69	76

Tabla 1

En la *Tabla 1* se muestra una población de 10 cromosomas con un *fitness* total igual a 76. La primera fila contiene el índice de cada cromosoma, la segunda fila contiene el *fitness* de cada cromosoma y la tercera el *fitness* total acumulado.

Numero aleatorio	23	49	76	13	1	27	57
Cromosoma elegido	3	7	10	3	1	3	7

Tabla 2

En la *Tabla 2* se muestra siete números generados aleatoriamente, junto con el índice del cromosoma que ha sido seleccionado por el algoritmo roulette wheel. En cada caso el cromosoma elegido es el primero para el cual el *fitness* total acumulado es mayor o igual que el número aleatorio.

El efecto del algoritmo *roulette wheel*, es devolver un padre seleccionado aleatoriamente, si bien la oportunidad de selección de cada padre es directamente proporcional a su *fitness*. En promedio este algoritmo, cuando se lleven ejecutadas varias generaciones, eliminará a aquellos individuos con un menor *fitness* y contribuirá a la expansión de material genético entre los miembros de la población.

Unicamente añadir que cuando se utiliza este algoritmo de selección de padres el valor de *fitness* de los cromosomas deben ser números positivos.

1.4.2 Cruce en un solo punto y mutación.

El operador que implementa la recombinación y la mutación primero hace una copia de los dos padres, se acaban de generar dos cromosomas hijo. Después se aplican dos funciones para modificarlos.

1.4.2.1 Mutación de bits

Consiste en seleccionar dentro de un cromosoma varios bit, cada bit tiene una determinada probabilidad de mutación normalmente se trata de un número muy pequeño. En el GA 1-1 la probabilidad de mutación se asigna como 0.008

En la *Tabla 3* se muestran varios ejemplos de mutación de bits:

Cromosoma antiguo				**Números aleatorios**				**Nuevo bit**	**Cromosoma nuevo**			
1	0	1	0	.801	.102	.266	.373	-	1	0	1	0
1	1	0	0	.120	.096	**.005**	.840	0	1	1	0	0
0	0	1	0	.760	.473	.894	**.001**	1	0	0	1	**1**

Tabla 3: Ejemplo de mutación de bits.

En la primera columna se muestran cuatro cromosomas con cuatro genes cada uno. Se generan números aleatorios entre 0 y 1 para cada uno de los bits de los cuatro cromosomas, son los que aparecen en la segunda columna.

En el caso del primer cromosoma (1,0,1,0) el test de probabilidad no es pasado por ninguno de sus genes, no sale ningún número por debajo de 0.008, por lo que no hay que generar ningún bits aleatoriamente y el nuevo cromosoma es el mismo (1,0,1,0).

En el segundo cromosoma (1,1,0,0) el tercer bit pasa el test de probabilidad ya que 0.005<0.008, luego se ha de generar otro bit aleatoriamente y resulta 0, que es el mismo valor que ya tenía ese bit, por lo que el nuevo cromosoma será (1,1,0,0).

En el tercer cromosoma (0,0,1,0) el cuarto bit pasa el test de probabilidad ya que 0.001<0.008, se genera un bit aleatoriamente para sustituirle y se obtiene 1,luego el nuevo cromosoma sería (0,0,1,1).

Otra variante muy conocida es este proceso de mutación es si un bit de un cromosoma pasa el test de probabilidad entonces es cambiado su valor, es decir, si era 1 pasa a 0 y si era 0 pasa a 1. Obsérvese que si la mutación se utiliza de esta manera la probabilidad de mutación es dos veces mayor que de la forma anterior.

1.4.2.2 Cruce en un punto

En la naturaleza el cruce ocurre cuando dos padres intercambian parte de sus correspondientes cromosomas. En un algoritmo genético, el cruce recombina el material genético de dos cromosomas padres para formar dos cromosomas hijos.

En un operador de cruce en un solo punto, el cruce ocurre cuando a partir de un determinado bit de los cromosomas padres se intercambian los genes para crear dos cromosomas hijos.

Ejemplo: Se tienen dos cromosomas padres con 6 genes o bit cada uno de ello el punto de cruce se toma en el 4º bit seleccionado contando desde la izquierda, luego se van a intercambiar los bit 5º y 6º de cada cromosoma padre para formar dos cromosomas hijo.

Padre 1 : 1 1 1 1 | 1 1 → Hijo 1 : 1 1 1 1 0 0

Padre 2 : 0 0 0 0 | 0 0 → Hijo 2 : 0 0 0 0 1 1

Ejemplo: Se tienen dos cromosomas padres con 6 genes o bit cada uno de ello el punto de cruce se toma en el 3º bit seleccionado contando desde la izquierda, luego se van a intercambiar los bit 5º y 6º de cada cromosoma padre para formar dos cromosomas hijo.

Padre 1 : 1 0 1 | 1 0 1 → Hijo 1 : 1 0 1 1 0 0

Padre 2 : 0 0 1 | 1 0 0 Hijo 2 : 0 0 1 1 0 1

Características del operador de cruce en un punto:

- Puede producir hijos que son radicalmente diferentes de sus padres. Es lo que sucede en el primer ejemplo.
- No se introducirán diferencias si el valor de los bit a intercambiar es el mismo. Es lo que sucede en las posiciones 2,3,4 y 5 que tienen el mismo valor en los cromosomas padre y en los cromosomas hijo. Un caso extremo ocurre cuando ambos padres son idénticos, en ese caso el operador de cruce no ha producido ninguna diversidad.

El operador de cruce es el elemento básico que caracteriza a un algoritmo genético. De hecho para muchos expertos si se elimina el operador de cruce el algoritmo resultante ya no es un algoritmo genético.

1.5 Schemata.

El operador de cruce actúa para combinar *bloques de construcción* . En el cuadro inferior se muestra un único cromosoma de longitud 5 y varios de los bloques de construcción contenidos en dicho cromosoma. Cada bloque de construcción se presenta como una lista constituida de tres caracteres (1,0 y #). Un 1 o un 0 en cualquier posición dentro del bloque de construcción significa, que el valor del cromosoma debe tener ese mismo valor en esa posición para poder contener dicho bloque de construcción. Una # en cualquier posición en el bloque de construcción significa que el valor del cromosoma en dicha posición es irrelevante para determinar si el cromosoma contiene el bloque de construcción.

Schemata en cadena de bits		
Cromosoma (01101) contiene 32 shemata incluyendo:		
0##0#	#1###	011#1
#####	01101	0#1#1

Holland denominó a cada bloque de construcción un *schemata*, y estableció que cuando se ejecuta el GA este manipula *schematas*.

2. MEJORAS EN LA REALIZACIÓN.

2.1 GA 1-1 en acción.

Cuando se corre un algoritmo genético se suelen observar las siguientes fases:

1) En la fase inicial la población es heterogénea, consiste de cromosomas que son muy distintos entre si. En este momento el algoritmo genético se comporta como un algoritmo que genera cromosomas aleatorios y luego los evalúa.
2) A partir de la segunda generación el cruce comienza a jugar un importante papel en la recombinación de los mejores cromosomas.
3) Cuando ya se han generado 3000 individuos, es decir, 30 generaciones, la población ha convergido hacia una única solución.
4) Desde este punto hasta que finalice el algoritmo se va estar buscando en un entorno de la región de la solución, siendo la mutación el principal elemento que introduce diversidad. Cuando una población consiste primordialmente de individuos muy similares se dice que ha convergido.
5) Cuando la población ha convergido sobre un cromosoma que requeriría mutación de muchos bits para causar cualquier mejora, se puede decir que la carrera del algoritmo esta completa.

Una vez que se está en una región cercana a la solución las mejoras que se producen ya no son muy apreciables. Mucho mejor que dejar el algoritmo iterar un número mayor de veces generando más individuos es más aconsejable volver a correr el algoritmo usando una semilla aleatoria diferente y coger el mejor resultado.

En nuestro caso el algoritmo se dejo correr 20 veces. El comportamiento de un algoritmo genético se examina tomando el valor de la función de evaluación para el mejor cromosoma de la población en determinados momentos de la carrera del algoritmo.

De forma gráfica esto se puede ver de la siguiente manera, como los algoritmos genéticos son estocásticos su comportamiento varia de una carrera a otra por eso es mucho más corriente tomar el promedio del mejor cromosoma de cada generación para las distintas carreras (p.ej. 20 veces) que se efectuén con el algoritmo.

La forma de estas curvas es típica , sube rápidamente al comienzo de la carrera, sube más lentamente cuando se encuentra cerca de un mínimo local y se mantiene más o menos plana en su parte final. Por lo tanto como ocurre con la mayoría de los algoritmos de optimización, la mayor parte de las mejoras se producen en el comienzo de la carrera y pequeñas mejoras o ninguna al final de la carrera.

2.2 Mejorando el GA 1-1

2.2.1 Normalización lineal

El algoritmo busca la mejor solución a la función *f6* mediante búsqueda aleatoria, el comportamiento de GA 1-1 se degrada si se realizan pequeñas modificaciones en la función *f6*.

Sea la función *f6* la siguiente :

$$f6 = 0.5 - \frac{\left(sin\left(\sqrt{x^2 + y^2}\right)^2\right) - 0.5}{\left(1.0 + 0.001\left(x^2 + y^2\right)\right)^2}$$

Y se la función *f6m*:

$$f6m = 999.5 - \frac{\left(sin\left(\sqrt{x^2 + y^2}\right)^2\right) - 0.5}{\left(1.0 + 0.001\left(x^2 + y^2\right)\right)^2}$$

Es obvio ver que la única diferencia entre *f6* y *f6m* es que se le ha subido el valor de la constante inicial. De hecho la curva son las mismas pero elevada en 999.

Cuando GA 1-1 se ejecuta utilizando la función *f6* como función de evaluación de los cromosomas sucede que con la población generada inicialmente:

- La evaluación del mejor cromosoma de la población es en promedio 0.979.
- La evaluación del caso peor es 0.066.
- La evaluación promedio entre el mejor y el peor es 0.514.

En consecuencia existe una fuerte presión de selección a favor del mejor cromosoma y en contra del peor.

Por otra parte, cuando GA 1-1 se ejecuta utilizando la función *f6m* como función de evaluación de los cromosomas sucede que con la población generada inicialmente:

- La evaluación del mejor cromosoma de la población es en promedio 999.979.
- La evaluación del caso peor es 999.066.
- La evaluación promedio entre el mejor y el peor es 999.514.

El mejor y el peor cromosoma producirán un número de descendientes similar en la siguiente generación. Y por lo tanto el efecto de la selección natural sobre el mejor cromosoma prácticamente no existirá.

Para mejorar a la función *f6m* y que esto no ocurra hay que recurrir a las que llamaremos *técnicas de ajuste.* Dos de estas técnicas son la de *windowing* y la de *Normalización lineal.*

La técnica de ajuste conocida como *Normalización Lineal* consiste en:

- Ordenar los cromosomas en orden decreciente de su valor *fitness* obtenido en la función de coste.
- Crear un ajuste que comience en una constante y decrezca linealmente. Los parámetros de esta técnica son el valor constante α y la razón de disminución β.

Ejemplo de técnicas de ajuste						
Evaluación original	200	9	8	8	4	1
El ajuste es la evaluación	200	9	8	8	4	1
Ajuste Lineal con β=1	100	99	98	97	96	95
Ajuste Lineal con β=20	100	80	60	40	20	1

Tabla 4

Se ha construido la *Tabla 4* para poner de manifiesto dos características típicas que se suelen presentar al correr un algoritmo genético:

1) La existencia de un cromosoma que es mucho mejor que su siguiente competidor. Este cromosoma es un *superindividuo* que probablemente eliminará al resto de competidores en una generación o dos cuando el ajuste sea la evaluación. Si esto sucede el superindividuo tendrá muy pocas oportunidades de combinarse con los otros miembros de la población y la población probablemente experimente una rápida convergencia.

2) La existencia de individuos cuyas evaluaciones están muy próximas. Cuando existen competidores muy próximos quizás se desee aumentar la presión de selección para que los mejores competidores se reproduzcan de manera no proporcional.

Si se toma como ajuste el valor obtenido en la evaluación, ambas características perdurarán, es decir, la existencia de un superindividuo y la existencia de competidores cercanos.
Si se hace un ajuste lineal, el superindividuo tardará más generaciones en dominar y se aumenta la competición entre competidores próximos. Hay que darse cuenta que cuanto más grande sea β mayor será la presión de selección en favor del valor de evaluación mayor.

Es por esto que tomaremos el ajuste lineal en lo que resta.

GA 1-1 con ajuste lineal en los valores evaluados → GA 2-1

Con GA 2-1 se consigue no tratar a los competidores cercanos como iguales que es lo que sucedía en GA 1-1.

2.2.2 Elitismo

Ya nos hemos dado cuenta de que el mejor miembro de una población quizás falle en producir descendientes en la próxima generación. La estrategia elitista resuelve esta posible causa de error copiando los mejores individuos de una generación en la siguiente generación. La estrategia elitista quizás incremente la velocidad de dominio de una población a través de un superindividuo pero en general parece mejorar el comportamiento del algoritmo genético.

GA 2-1 con reemplazamiento generacional por reproducción con elitismo →GA 2-2

El GA 2-1 posee reemplazamiento generacional por reproducción sin elitismo.

2.3 El Algoritmo genético tradicional

El GA 2-2 puede ser considerado como un *algoritmo genético* tradicional ,las características de dicho algoritmo se pueden resumir en:

1) Codificación binaria.
2) Reproducción generacional con elitismo.

3) Técnica de ajuste.
4) Operador de cruce en un único punto.
5) Operador de Mutación

2.4 Técnica de Reproducción Steady State.

Cuando GA 2-2 se reproduce, se reemplaza el conjunto completo de padres por sus hijos. Esta técnica de reproducción tiene algunos inconvenientes:

- Una es que incluso utilizando reproducción con elitismo, muchos de los mejores individuos puede que no se reproduzcan en absoluto y sus genes pueden perderse.
- Otra es que la mutación o el cruce pueden alterar los genes de los mejores cromosomas incluso algunos pueden ser destruidos.

Por supuesto ninguna de estos problemas es deseable.

Una solución a este problema es modificar la técnica de reproducción de tal forma que solamente se reemplacen uno o dos individuos al mismo tiempo mejor que no todos simultáneamente, es la técnica de reproducción conocida como *steady-state.*

Reproducción "steady-state"

1) Se crean *n* hijos por reproducción.
2) Se eliminan los *n* peores miembros de la población para hacer sitio para los hijos.
3) Se evalúan y se insertan los hijos dentro de la población.

La reproducción *steady-state* tiene un solo parámetro, el número de cromosomas a crear. Notar que el reemplazamiento generacional es un caso especial de la reproducción *steady-state* en el que el parámetro asume el tamaño de la población.

GA 2-1 sustituyendo la reproducción generacional por steady-sate →GA 2-3

2.5 Steady- State sin duplicados.

La reproducción *steady-state* es inferior al reemplazamiento generacional en el sentido de que crea muchos individuos que son idénticos.

La reproducción *steady-state* sin duplicados es una técnica que descarta aquellos hijos que sean iguales algún miembro de la generación actual antes de insertarlos en la población. Utilizando esta técnica nos aseguramos que todos los miembros de la población son diferentes entre si.

El único inconveniente que presenta esta técnica es que en el módulo de población hay que realizar muchos test de igualdad entre los posibles hijos y los miembros ya

existentes para elegir aquellos que sean distintos, lo cual supone una pérdida de tiempo de computación.

Sin embargo en la experiencia real se comprueba que el mayor tiempo de cálculo de un algoritmo genético se gasta en evaluar cada cromosoma. Por lo que el tiempo que pueda utilizar en realizar comparaciones es despreciable frente al anterior.

GA 2-3 sin duplicados $\rightarrow$GA 2-4

Conviene tener en cuenta que los GA no son sistemas simples, sino que son no lineales y su comportamiento es difícil de predecir y de formalizar.

2.6 Valores de los parámetros mejorados.

El modificar el valor de alguno de los parámetros que caracteriza un GA puede influir enormemente en el comportamiento del algoritmo.

Si GA 2-4 se aumenta la razón de cruce en un punto de 0.6 a 0.8 y la razón de mutación de un bit de 0.008 a 0.04 $\Rightarrow$ GA 2-5

2.7 Robustez

Un algoritmo se dice que es *robusto* en la medida que se comporta adecuadamente para distintos tipos de problemas.

Se ha decir que cuanto mejor se ajuste un GA para resolver un problema especifico se estará perdiendo en robustez, es decir, en capacidad para resolver otro tipo de problemas. Ya que la robustez es inversamente proporcional a la mejora en el comportamiento para un determinado problema.

La meta de este texto no es construir GA robustos sino mostrar que los GA son los mejores algoritmos de optimización para cierto tipo de problemas.

Pienso que para obtener un algoritmo que se comporte bien en un dominio, es necesario especializar dicho algoritmo sobre ese dominio. Tal especialización tal vez requiera el uso de técnicas, parámetros y operadores adaptados especialmente para el tipo de problema que se pretende resolver. Pero tal especialización produce una disminución en la robustez

3. EVOLUCION ADICIONAL DEL ALGORITMO GENETICO

3.1 Operadores Independientes

El operador de cruce en un punto y de mutación de bits es en realidad es en realidad dos operadores unidos en uno. A partir de ahora se van ha considerar como dos operadores distintos: El operador de cruce en un punto y el operador de mutación de bits . De tal manera que durante un evento de reproducción se aplicará un operador u otro pero no los dos, así la mutación no se aplicará al resultado del cruce.

Desde ahora cuando se vaya a producir un evento de reproducción se procederá de la siguiente forma:

1) Se seleccionará un operador de una lista de operadores disponibles.
2) Se aplicará dicho operador a un número de padres adecuado.
3) Se devolverán los hijos que se generen al modulo de población.

Ahora cada operador tiene un nuevo parámetro, su *fitness*. Se manipularán los *fitness* de cada operador para que sumen entre todos los de la lista 100.

A la hora de seleccionar un operador u otro se aplicará un criterio de selección del tipo *roulette wheel*, de tal manera que la probabilidad de que un operador sea elegido dependerá de su *fitness*.

Modificando el GA 2-5 de la forma indicada en las líneas anteriores ⇒ GA 3-1

3.2 Otros Operadores de cruce.

Sean dos cromosomas con las siguientes estructuras:
Cromosoma 1 : **1**1**0**11001011101**1**
Cromosoma 2 : 0001**0**110**1**11100

Los bits señalados en negrita constituirían bits relevantes cuya información conviene que no fuese perdida. Tendríamos dos *schemata*. Pues bien con el operador de cruce en un solo daría igual donde se eligiese el punto de cruce siempre se perdería la primera *schemata*.

Una posible solución a este tipo de problemas surge con la utilización del operador de dos puntos de cruce estos dos puntos de cruce son seleccionados aleatoriamente. Veamos el siguiente ejemplo del operador de dos puntos de cruce :

Padre 1 : **1**1**0**1 100101 101**1** Hijo 1 : **1**1**0**1 **0**1101**1** 101**1**
Padre 2 : 0001 **0**1101**1** 1100 Hijo 2 : 0001 100101 1100

Se ha logrado introducir en un solo hijo las dos *schemata*. si embargo existen *schemata* que el operador de dos puntos de cruce no puede combinar. Se podría pensar en considerar un operador con varios puntos de cruce.

<u>*Operador de cruce uniforme*</u> descrito por Syswerda (1989) , este operador trabaja de la siguiente forma :
1. Se escogen dos padres :

Padre 1 : 1 0 0 1 0 1 1
Padre 2 : **0 1 0 1 1 0 1**

2. Se genera un cromosoma aleatorio con el significado de que un 1 se selecciona el bit de esa posición del padre1 si es un 0 se selecciona el bit de esa misma posición del **padre 2**.
   Elección 1 1 0 1 0 0 1
3. Se decide aleatoriamente que padre contribuye para cada posición al primer hijo, el que no resulte elegido contribuirá con su bit de esa posición al segundo hijo
   Hijo 1 : 1 0 **0** 1 **1 0** 1
   Hijo 2 : **0 1** 0 **1** 0 1 **1**
   En los hijos el bit señalado en negrita es la contribución del **padre 2**

El cruce uniforme difiere del cruce en un punto y en dos por las siguientes características más relevantes:

- La posición de un bit donde se pueda encontrar información relevante del cromosoma es indiferente para el operador de cruce uniforme. Por lo que un peligro potencial de este operador es que puede dar un trato violento a todo lo que es bueno en el cromosoma.
- Los operadores de cruce en un punto y en dos son más locales que el operador de cruce uniforme y preservaran probablemente las características que estén codificadas más compactamente, es decir, en bit consecutivos.

GA 3-1 con operador de cruce uniforme ⇒ GA 3-2

3.3 Interpolando el valor de los parámetros.

Se denominan *parámetros fijos* de un algoritmo genético, a aquellos que no modifican su valor durante todo el transcurso de la carrera del algoritmo, es decir, se fijan al principio y no se modifican. Sin embargo se puede mejorar el comportamiento del GA si se modifica el valor de los parámetros durante el transcurso de la carrera del algoritmo.

3.3.1 Interpolando el fitness de los operadores

Se va a colocar la técnica de interpolación de pesos de los operadores en la lista de técnicas de parametrización, dentro del modulo de reproducción. Esta técnica tiene como parámetros los valores inicial y final del *fitness* que se desea que tengan cada operador. Con esta técnica se conseguirá el variar el *fitness* de cada operador gradualmente a lo largo de una carrera del algoritmo, desde un valor inicial hasta un valor final.

¿Por qué sería una buena idea alterar el *fitness* de varios operadores durante el transcurso de una carrera?. Es esta una pregunta interesante. Existen fuertes interacciones entre los pesos relativos de los operadores en un algoritmo genético, el grado de convergencia de esa población y la cercanía de los mejores individuos a una optima solución local. En cualquier punto de la carrera del algoritmo existe una razón de los *fitness* de los operadores que conducirá hasta la solución optima , sin embargo esta razón variará durante el transcurso de la carrera.

No se puede prescindir de ninguno de los dos operadores ni el de cruce ni el de mutación pues cada uno de ellos tiene un papel importante en las distintas etapas del algoritmo genético. Al principio el operador de cruce es el más importante pero al final cuando se esta cerca de la zona de convergencia es más útil el operador de mutación ya que va a introducir diversidad El ir interpolando el valor de *fitness* de un operador durante el transcurso del algoritmo, siempre dará mejores resultados que dejar estos valores de *fitness* fijos. Además en el peor de los casos nunca se empeora interpolando el valor que se obtuviera dejándolos fijos.

GA 3-2 interpolando el valor de *fitness* del operador de cruce uniforme de 70 a 50 y el valor del *fitness* del operador de mutación de 30 a 50 $\Rightarrow$ GA 3-3

4. ALGORITMOS GENETICOS HIBRIDOS

Si en un algoritmo genético se introduce lo mejor de otros algoritmos de otro tipo para resolver un cierto problema, se generará un algoritmo genético hibrido que en general será mejor que sus competidores.

4.1 ¿ Por qué hibridar?

Una meta central para los investigadores de algoritmos genéticos ha sido obtener algoritmos que fuesen robustos. Mediante la utilización de codificación binaria y un único operador esta era la meta que se pretendía alcanzar.

Aunque GA con representación binaria y cruce en un sólo punto y mutación son algoritmos robustos, casi nunca suelen ser el mejor algoritmo para un determinado problema.

4.2 Como adaptar un algoritmo genético.

Supóngase que una determinada persona, un usuario, está intentando resolver un problema en una determinada área de conocimiento utilizando un algoritmo que vamos a denominar *algoritmo corriente.* El usuario estará familiarizado con dicho algoritmo corriente y con la forma como trabaja.

Cuando se hable con el usuario se le dirá que el plan que proponemos es hibridar el algoritmo genético con el algoritmo genético, el algoritmo híbrido que se obtenga actuará mejor o igual que el corriente., para ello se seguirá los tres principios siguientes:

Principios de Hibridación

1) Utilizar la codificación que utilice el algoritmo corriente.
2) Hibridar donde sea posible. Introduciendo las mejores características del algoritmo corriente.
3) Adaptar los operadores genéticos, creando operadores de cruce y de mutación para la codificación del algoritmo corriente

4.2.1 Utilizar la codificación del algoritmo corriente

Utilizar la codificación del algoritmo corriente tiene dos ventajas:

1) Se garantiza que el dominio de búsqueda comprendido en la codificación se mantendrá.
2) Se garantiza que el algoritmo genético híbrido operará con la misma clase de estructuras con las que trabajaba el algoritmo corriente.

4.2.2 Hibridar donde sea posible

Si el algoritmo corriente es rápido, se pueden utilizar las soluciones que produzca como población inicial para el algoritmo genético. Con ello se asegura que el algoritmo híbrido no empeora las soluciones que ofrezca sino que más bien las mejorará.

Si el algoritmo corriente lleva a cabo sucesivas transformaciones sobre la codificación sería muy útil incorporar estas transformaciones dentro del conjunto de operadores.

4.2.3 Adaptar los operadores genéticos.

Una vez que se ha adaptado la codificación del algoritmo corriente, ya no se pueden utilizar los operadores de cruce y de mutación que se tenían ya que estos operaban con bits. Es necesario adaptar estos operadores de acuerdo con la codificación que se ha adoptado.

Una idea ha tener muy en cuenta es que la codificación que se haya adoptado debe poder soportar los operadores genéticos. En el caso en que no sea posible adaptar los operadores genéticos a esa codificación habrá que buscar otra codificación

A la hora de adaptar los operadores genéticos habrá que tener en cuenta el conocimiento que se posea del problema a resolver, y la técnica de codificación.

5. HIBRIDACION Y REPRESENTACION NUMERICA.

5.1 Algoritmo de generación aleatoria y prueba para f6.

Una manera de encontrar soluciones para *f6* es probando con distintos valores de (x , y) generados aleatoriamente y almacenando el valor que no resulte, para después revisar que par de valores da un mejor resultado. Se trataría de un algoritmo de generación aleatoria y prueba, que se puede utilizar en un amplio campo de problemas y que a veces funciona sorprendentemente bien.

5.2 Usando la codificación corriente.

Las cadenas de bits tienen varias ventajas sobre otras codificaciones, son simples de crear y de manipular. Además son teóricamente fáciles de tratar, con ellos es fácil probar teoremas sobre los algoritmos genéticos.

Al hibridar seguramente ya no se podrá utilizar la codificación binaria y habrá que recurrir a otro tipo de codificación.

Vamos a alterar nuestro algoritmo genético para que utilice números reales en sus cromosomas. Las modificaciones en el modulo de población son las siguientes:

- Reemplazar la técnica de codificación binaria de GA 3-3 con una técnica de lista de números reales.
- Reemplazar la función de evaluación binaria *f6* de GA 3-3 con una función *f6* que trate con números reales,(no requiere de conversión de binaria a real).
- Reemplazar la técnica de inicialización aleatoria binaria por una técnica de inicialización aleatoria real. Esta técnica generará números reales entre dos límites uno inferior y otro superior que serán parámetros de la técnica.

Un ejemplo de un cromosoma generado con la técnica descrita es : (x,y)=(1563404,11204).

Una pregunta interesante sería: ¿Una carrera de un algoritmo con 100 cromosomas generados aleatoriamente que dará 3900 descendientes a través mutación y cruce será mejor que 4000 individuos generados aleatoriamente? La respuesta es SI

5.3 Adaptando los operadores genéticos.

5.3.1 Operador de cruce con codificación en números reales.

Tanto el operador de cruce en un punto como el de cruce uniforme únicamente requieren una lista.

Ahora con una codificación en números reales es posible utilizar operadores de cruce que incorporen un trato numérico: *el operador promedio*. Este operador coge dos padres y genera un hijo que es el resultado de promediar a los dos padres.

5.3.2 Operador de mutación con codificación en números reales.

El operador de mutación de números reales reemplaza un determinado número real por otro si supera un test de probabilidad. Sin embargo como en el cromosoma *(x,y)* se tienen dos números reales se aplica únicamente a uno de los dos, así actuará mejor que utilizar generación aleatoria y prueba, puesto que la mitad del cromosoma retendrá la su valor antiguo que quizás sea útil mantener constante durante la búsqueda.