GENETIC ALGORITHMS + DATA STRUCTURES= EVOLUTION PROGRAMS (1999)

Zbigniew Michalewicz

RESUMEN
CAPITULOS 1-6

1. ¿ Qué son los algoritmos genéticos?

1.1 Introducción

Para algunos problemas de optimización duros podemos utilizar algoritmos probabilísticos, estos algoritmos no garantizan que se obtenga el máximo global, pero si una buena solución.

Los algoritmos genéticos (GA) son más robustos que los métodos de búsqueda dirigida ya existentes. Otra propiedad destacable es que los GAs mantienen una población de potenciales soluciones. Además realiza el intercambio de información entre las distintas direcciones. El resto de métodos únicamente procesan un único punto en el espacio de búsqueda.

La población de soluciones va evolucionando, en cada generación las soluciones buenas se reproducen mientras que las relativamente malas soluciones son eliminadas.

Un GA debe tener para un determinado problema los componentes siguientes:

1) Una representación genética para las potenciales soluciones del problema.
2) Un mecanismo para crear una población inicial de potenciales soluciones.
3) Una función de evaluación, que de un criterio para indicar que solución es la mejor para nuestro problema.
4) Operadores genéticos que alteren la composición de la población de soluciones.

2. ¿Como trabajan los GA?

2.1 Problema inicial

Supongamos en primer lugar, sin perdida de generalidad que vamos a trabajar únicamente con problemas de maximización. Si el problema fuese minimizar una función f, el problema sería equivalente a maximizar $g=-f$.

$$min\big(f(x)\big) = max\big(g(x)\big) = max\{-f(x)\}$$

Además suponemos que la función de coste f toma valores positivos en su dominio.

Supongamos que se quiere maximizar una función de k variables:

$$f(x_1,x_2,.....,x_k): \mathrm{R}^k \rightarrow \mathrm{R}$$

Supongamos además que cada variable x_i puede tomar valores dentro de un dominio $D_i=[a_i,b_i] \subseteq R$. $i=1,...,k$.

Deseamos optimizar la función f con una precisión de 6 cifras exactas.

Es evidente que para conseguir tal precisión cada dominio D_i, debe ser dividido en $(b_i-a_i)\cdot 10^6$ rangos de igual tamaño.

Sea m_i el entero más pequeño para el cual se verifica que:

$$(b_i\text{-}a_i)\cdot 10^6 \leq 2^{mi}\text{-}1$$

Por lo tanto cada variable x_i puede ser codificada, como un string binario de longitud m_i.

Entonces dado un string binario el equivalente real sería dado por:

$$x_i = a_i + decimal(1001...001_2)\cdot\frac{b_i - a_i}{2^{mi} - 1}$$

Donde decimal() representa el valor decimal del string binario.

Así cada cromosoma, que es una potencial solución, está representada por un string binario de longitud $m=m_1+m_2+...+m_k$. Los primeros $m_1$bits dan un valor para x_1 dentro del rango D_1, y así sucesivamente.

La inicialización de la población se realiza de manera aleatoria, se configura el tamaño de la población a generar *pop_size***.**

El resto del algoritmo es inmediato:

- En cada generación t, se evalúa cada cromosoma de la población.
- Se selecciona una nueva población, basándose en el valor de su función de coste.
- Se alteran los cromosomas de la nueva población mediante mutación y operadores de cruce.
- Después de un determinado número de generaciones, cuando no se observa ninguna mejora adicional, el mejor cromosoma representará una solución optima que puede ser que se trate del óptimo global.

2.1.1 Proceso de selección de una nueva población.

Para el proceso de selección de una nueva población ,se utiliza el algoritmo conocido como "*tiro de una ruleta*" con sectores circulares de tamaño proporcional al valor de ajuste de cada cromosoma. Construimos la ruleta de la siguiente manera:

1) Calculamos el valor de ajuste $f(\mathbf{v_i})$ para cada cromosoma $\mathbf{v_i}$,(i=1,...,*pop_size*)
2) Calculamos la suma de los valores de ajuste , de todos los cromosomas de la población:

$$F = \sum_{i=1}^{pop\_size} f(\vec{v}_i)$$

3) Calculamos la probabilidad de selección p_i para cada cromosoma $\mathbf{v_i}$ (i=1,...,*pop_size*).

$$p_i = \frac{f(\vec{v}i)}{F}$$

4) Calculamos la probabilidad acumulada q_i para cada cromosoma $\mathbf{v_i}$ (i=1,...,*pop_size*).

$$q_i = \sum_{j=1}^{i} p_j$$

El proceso de selección está basado en girar la ruleta *pop_sizes* veces, cada vez seleccionamos un único cromosoma para la siguiente población de la siguiente manera:

- Generamos un número real aleatorio *r* dentro del rango [0. ,1.]
- Si $r<q_1$ entonces seleccionamos el primer cromosoma v_1, sino seleccionamos el cromosoma i-esimo v_i ($2 \leq i \leq pop\_size$) tal que $q_{i-1} < r \leq q_i$

Obviamente los mejores cromosomas, serán seleccionados más veces, mientras que los peores tenderán a desaparecer.

2.1.2 Aplicación de los operadores genéticos.

Operador de recombinación
Sea la probabilidad de cruce p_c. Esta probabilidad nos dará el número de cromosomas que se utilizarán en la operación de cruce $p_c\ pop\_size$.

Para cada cromosoma en la nueva población:

- Generamos un número aleatorio real *r* en el rango [0.,1.].
- Si $r<p_c$ se selecciona el cromosoma para el cruce.

Ahora debemos emparejar los cromosomas seleccionados aleatoriamente. Para cada pareja de cromosomas emparejados, seleccionamos un número entero aleatoriamente *pos* dentro del rango [1,*m*-1] donde m es el número de bits totales del cromosoma. El número *pos* indica la posición del punto de cruce.

Sean dos cromosomas:

$$(b_1,b_2,.....,b_{pos},b_{pos+1},....b_m)$$
$$(c_1,c_2,.....,c_{pos},c_{pos+1},....,c_m)$$

Se generarían los cromosomas hijo siguientes:

$$(b_1,b_2,.....,b_{pos},c_{pos+1},....c_m)$$
$$(c_1,c_2,.....,c_{pos},b_{pos+1},....,b_m)$$

Operador de mutación :
Se caracteriza por el parámetro p_m probabilidad de mutación. Que nos da el número esperado de mutaciones $p_m \cdot pop\_size$. Cada bit en los cromosomas de toda la población tienen la misma probabilidad de mutar, cambiar de 0 a 1 o viceversa.

Se procede de la siguiente manera:

- Generamos un número aleatorio real *r* en el rango [0.,1.].
- Si $r<p_m$ mutamos el bit.

2.2 Ejemplo.

Se configuró un GA con los siguientes valores:

pop_size=20
p_c=0.25
p_m=0.01

Supóngase que se pretende maximizar la siguiente función:

$$f(x_1, x_2) = 21.5 + x_1 sin(4\pi x_1) + x_2 sin(20\pi x_2)$$
$$-3.0 \le x_1 \le 12.1$$
$$4.1 \le x_2 \le 5.8$$

Supóngase que requiere una precisión de cuatro cifras decimales para cada variable x.

El dominio de la variable x_1 tiene una longitud de 15.1. La precisión requerida implica que el rango [-3.0,12.1] debería ser divido en 15.1*10000 rangos de igual tamaño. Esto significa que se requieren 18 bits, para x_1 en la primera parte del cromosoma:

$$2^{17} < 151000 \le 2^{18}$$

Para el caso de x_2 la longitud del rango de que se dispone es 1.7, debería ser dividido en 17000 partes iguales, lo cual requiere 15 bits, y constituirá la segunda parte del cromosoma.

$$2^{14} < 17000 \le 2^{15}$$

La longitud total del cromosoma es m=18+15=33 bits, los 18 primeros codifican a x_1 y los 15 restantes a x_2.

$$v_i = (b_1, ..., b_{18}, \mathbf{b_{19}, ..., b_{33}}).$$

Por ejemplo el cromosoma:

(010001001011010000 | 1111100100010)

Los primero 18 bits representan el valor de x_1 en binario. Para pasarlo a decimal se aplicaría la siguiente fórmula:

$$x_1 = -3.0 + decimal(010001001011010000) \cdot \frac{12.1 - (3.0)}{2^{18} - 1} = -3.0 + 70352 \cdot \frac{15.1}{262143} = 1.052426$$

Los otros 15 bits corresponden a x_2=5.755330.

Luego el cromosoma:

(010001001011010000 | 1111100100010)=(x_1,x_2)=(1.052426, 5.755330)

Que tiene un valor de ajuste: $f(x_1,x_2)$=20.252640.

Para optimizar la función f utilizamos un algoritmo genético, creamos una población de *pop_size*=20. Los 33 bits de cada cromosoma son inicializados aleatoriamente.

En la siguiente *Tabla 1* aparecen, el valor de ajuste de los 20 cromosomas de la primera generación *t*=1.

Cromosoma	**f(v_i)**	$p_i = \frac{f(\vec{v}i)}{F}$	$q_i = \sum_{j=1}^{i} p_j$
v_1	26.01	0.067099	0.067099
v_2	7.58	0.019547	0.086647
v_3	19.52	0.050355	0.137001
v_4	17.40	0.044889	0.181890
v_5	25.34	0.065350	0.247240
v_6	18.10	0.046677	0.293917
v_7	16.02	0.041315	0.335232
v_8	17.95	0.046315	0.381546
v_9	16.12	0.041590	0.423137
v_{10}	21.27	0.054873	0.478009
v_{11}	23.41	0.060372	0.538381
v_{12}	15.01	0.038712	0.577093
v_{13}	27.31	0.070444	0.647537
v_{14}	19.87	0.051257	0.698794
v_{15}	30.06	0.077519	0.776314
v_{16}	23.86	0.061549	0.837863
v_{17}	13.69	0.035320	0.873182
v_{18}	15.41	0.039750	0.912932
v_{19}	20.09	0.051823	0.964756
v_{20}	13.66	0.035244	1.000000

Tabla 1

A la vista de la *Tabla 1* se puede afirmar que el cromosoma v_{15} es el que presenta el mayor valor de ajuste, mientras que v_2 presenta el menor valor de ajuste. El ajuste total es F=387.77.

Ahora estamos listos para girar la ruleta, se debe girar 20 veces, cada vez se selecciona un único cromosoma para la siguiente población. Supongamos que la secuencia aleatoria de 20 números reales en el rango [0,1] es la que se muestra en la *Tabla 2*.

Fijémonos en que el primer valor de r=0.513870 es mayor que q_{10} y menor que q_{11}, esto significa que el cromosoma v_{11} es seleccionado para la nueva población. De igual manera el segundo r=0.1757 es mayor que q_3 y menor que q_4, lo que significa que el cromosoma v_4 es seleccionado para la siguiente generación, etc.

La probabilidad de recombinación es p_c=0.25, es decir el 25% de la población será seleccionada para realizar recombinación es decir 5 de 20. Para cada cromosoma en la nueva población ***v'***, generamos un número aleatorio real en el rango r=[0,1]. Si r<0.25 seleccionamos el cromosoma para el cruce. Supongamos que se ha generado la secuencia de números que se muestra en la *Tabla 3*.

r	Cromosoma	Nueva

	seleccionado de la generación t=1	población t=2
0.5138	v_{11}	v_1'
0.1757	v_4	v_2'
0.3086	v_7	v_3'
0.5345	v_{11}	v_4'
0.9476	v_{19}	v_5 '
0.1717	v_4	v_6'
0.7022	v_{15}	v_7'
0.2264	v_5	v_8'
0.4947	v_{11}	v_9'
0.4247	v_3	v_{10}'
0.7038	v_{15}	v_{11}'
0.3896	v_9	v_{12}'
0.2772	v_6	v_{13}'
0.3680	v_8	v_{14}'
0.9834	v_{20}	v_{15}'
0.0053	v_1	v_{16}'
0.7656	v_{10}	v_{17}'
0.6464	v_{13}	v_{18}'
0.7671	v_{15}	v_{19}'
0.7802	v_{16}	v_{20}'

Tabla 2

r	Nueva población t=2	Cromosoma seleccionado para cruce ¿r<0.25 ?
0.8229	v_1'	
0.9172	v_2'	SI
0.0315	v_3'	
0.5818	v_4'	
0.1519	v_5 '	SI
0.5197	v_6'	
0.8699	v_7'	
0.3892	v_8'	
0.6254	v_9'	
0.4011	v_{10}'	
0.1665	v_{11}'	SI
0.2002	v_{12}'	SI
0.3146	v_{13}'	
0.6067	v_{14}'	
0.6745	v_{15}'	
0.556	v_{16}'	
0.3469	v_{17}'	
0.7854	v_{18}'	
0.7584	v_{19}'	
0.8269	v_{20}'	

Tabla 3

Se ha tenido suerte pues el número de cromosomas seleccionados es par, si hubiese salido impar tendríamos que o bien añadir un cromosoma más o bien eliminar uno.
Ahora hay que emparejar los cromosomas aleatoriamente por ejemplo: (v'_2 con v'_{11}) y (v'_{13} con v'_{18}).

Se genera un entero aleatorio *pos* en el rango [1,32] que indicará la posición del punto de cruce. Resultó que *pos*=9 por lo que se tiene que:

$$v'_2 = (100011000 \mid 10110100111000001110010)$$
$$v'_{11} = (111011101 \mid 10111000010001111011110)$$

que dan los hijos siguientes:

$$v''_2 = (100011000 \mid 10111000010001111011110)$$
$$v''_{11} = (111011101 \mid 10110100111000001110010)$$

Por otra parte resultó *pos*=20 y se generaron v''_{13} y v''_{18}

La probabilidad de mutación es p_m=0.01 luego el 1% de los bits sufrirá mutación. Luego como 33x20=660bits es el número de bits de toda la población, solo 6.6 de estos 660 sufrirán mutación. Debemos generar un número aleatorio r entre [0.1] para cada uno de los bits. Si se cumple que r<0.01 mutaremos el bit.

Se encontró que tras generar 660 números reales aleatoriamente 5 bit tuvieron un valor menor que 0.01 (ver *Tabla 4*).

r	Cromosoma	Bit seleccionado dentro del cromosoma por cumplir r<0.01
0.000213	v'_4	13
0.0009945	v''_{11}	19
0.008809	v''_{13}	22
0.005425	v''_{13}	33
0.002836	v'_{19}	8

Tabla 4

Esto significa que de los cuatro cromosomas afectados por el operador de mutación uno de ello v''_{13} tiene dos bits cambiados.

Ya se tendría lista la población correspondiente a *t*=2. Ahora habría que realizar su evaluación.

Con la generación t=2 se encuentra que el ajuste total *F* es mucho mayor *F*=447.049688 que el de la generación *t*=1 que era de *F*=387.776822.

El mejor cromosoma es ahora el v_{11}=33.3518 antes era el v_{15}=30.060205.

En t=1000 generaciones el mejor valor es de 35.477938, sin embargo el mejor cromosoma en t=396 tenía un valor de 38.8275. esto es debido al error estocástico de muestreo. Es por ello que se deben ir almacenando el mejor individuo de cada generación.

3. ¿Binario o Flotante ?

Algunos de los problemas que presentan los GA son:

- Convergencia prematura a un mínimo local.
- Incapacidad para realizar un ajuste fino.
- Incapacidad para operar en la presencia de restricciones no triviales.

La representación binaria tradicionalmente utilizada en los GA tiene algunos inconvenientes cuando es utilizada para problemas multidimensionales y de alta precisión numérica.

Si utilizamos representación en coma flotante, esta más cercana al espacio del problema ,lo cual permite la implementación de operadores más transparentes y dinámicos.

En la implementación en coma flotante cada cromosoma fue codificado como un vector de números en coma flotante, de la misma longitud que el vector solución. Cada componente o elemento del vector fue forzado a estar dentro del rango de valores deseado, y los operadores fueron diseñados para que se cumpliera esta restricción.

La precisión de un aproximación de este tipo depende de la máquina que se vaya a utilizar, pero en general es mucho mejor que la representación binaria. Además la representación en coma flotante permite representar grandes rangos y es mucho más fácil para el diseño de herramientas que traten con restricciones no triviales.

4. GAs: Temas seleccionados

La teoría de GA suministra alguna explicación de por que, dado un determinado problema, podemos obtener la convergencia hacia el óptimo deseado. Desafortunadamente las aplicaciones prácticas no siguen siempre a la teoría, por las siguientes razones:

- La codificación de los problemas a menudo desplaza al GA a operar en un espacio diferente al del problema.
- Existe un limite al hipotético número ilimitado de iteraciones.
- Existe un límite al hipotetico número ilimitado de miembros de la población.

Uno de los problemas que presentan los GA, al igual que otros tipos de algoritmos de optimización es el de la convergencia prematura a un mínimo local. La información de valor desarrollada en parte de la población se pierde.

Para combatir la convergencia prematura se puede recurrir algunas estrategias como:

- Utilización de cruce uniforme.
- Detectar cromosomas duplicados.

4.1 Mecanismos de muestreo.

Por mecanismos de muestreo, se entiende la forma de seleccionar cromosomas para generar las próximas generaciones.

Existen dos temas muy importantes en los procesos de evolución, por un lado está la diversidad de la población y por otro se encuentra la presión de selección. Estos factores se encuentran fuertemente relacionados de tal manera que un aumento en la presión de selección produce una disminución en la diversidad de la población.

Un aumento de la presión de selección conducirá a una prematura convergencia de la búsqueda del GA, y una presión de selección débil conducirá a una búsqueda poco efectiva.

Según *DeJong* en 1975. Consideró varias variaciones de los mecanismos de selección.

1) La primera variación es llamada el *modelo elitista* y consiste en conservar siempre el mejor cromosoma.
2) La segunda variación, llamada *modelo del valor esperado*, disminuye el error estocástico de la rutina de selección. Esto se consigue mediante la introducción de un contador para cada cromosoma **v** , que se configura inicialmente con el valor $\frac{f(\vec{v})}{\bar{f}}$ y se disminuye en 0.5 o 1 cuando el cromosoma es seleccionado para reproducción con cruce o mutación, respectivamente. Cuando el contador del cromosoma cae por debajo de 0, el cromosoma ya no está disponible para ser seleccionado.
3) La tercera variación es una combinación de las otras dos.

Otros métodos para muestrear una población están basados en la introducción de pesos artificiales. Los cromosomas son seleccionados proporcionalmente a su rango más que por su valor de ajuste. Estos métodos están basados en la creencia ampliamente extendida de que la causa de una convergencia prematura es la existencia de superindividuos que presentan un valor de ajuste mucho mayor que la media del valor de ajuste del resto de individuos de la población. Estos superindividuos presentarán un elevado número de hijos, debido al valor constante de pop_size, que impedirá la contribución de otros individuos, y se eliminará material genético que puede ser muy valioso para la búsqueda del máximo global. Produciendo la rápida convergencia del algoritmo.

4.1.1 Algoritmo Genetico modificado (AGm)

La estructura del AGm es la que se muestra a continuación :

```
t=0
inicializar P(t)
evaluar P(t)

while ( condición de terminación) do
t=t+1
seleccionar- padres de P(t-1)
seleccionar - difuntos de P(t-1)
formar P(t) : reproducir los padres
evaluar P(t)
end
```

La diferencia de AGm con el AG está en que en AGm no se realiza el paso de seleccionar P(t) de P(t-1), sino que se seleccionan r cromosomas (que no tienen por que ser distintos) para reproducirse y r cromosomas (distintos) para ser eliminados.

Estas selecciones se llevan a cabo utilizando el valor de ajuste relativo del cromosoma , un cromosoma con un valor de ajuste mayor que el valor de ajuste medio tiene una mayor probabilidad de ser elegido para reproducirse, cromosomas con un valor por debajo de la media tienen altas probabilidades de ser elegidos para ser eliminados.

Después de los pasos de seleccionar padres y seleccionar difuntos existen tres grupos, que no tienen por no encontrarse solapados, de cromosomas dentro de la población:

- *r* cromosomas, que no tienen por que ser distintos, para reproducirse, padres.
- *r* cromosomas distintos para ser eliminados.
- Los restantes cromosomas que se denominan *neutros.*

El número de cromosomas neutros en una generación, sera menor que *pop_size-2r* y mayor que *popsize-r.*

Después una nueva población P(t+1) es formada, consistente de *pop_size - r* cromosomas (todos los cromosomas que habían antes menos los que fueron seleccionados para ser eliminados). Y *r* hijos de los r padres.

Este algoritmo AGm tiene un paso potencialmente problemático que es como seleccionar los r cromosomas que serán eliminados. Obviamente la selección se quiere llevar a cabo, de tal manera que los cromosomas más fuertes tengan menos probabilidad de perecer. Se consigue esto cambiando la forma de seleccionar la nueva población P(t+1) a esta otra:

1) Seleccionar r padres de P(t). Cada cromosoma será marcado para ser aplicado a una determinada operación genética.
2) Seleccionar *pop_size* cromosomas distintos de P(t) y copiarlos a P(t+1).
3) Permitir que los r cromosomas padres produzcan exactamente r hijos.
4) Insertar los r nuevos hijos dentro de la población P(t+1).

La selecciones en los pasos 1 y 2 se realiza de acuerdo al valor de ajuste de los cromosomas.

Diferencias entre AGc y AGm

- En AGm tanto el padre como el hijo tienen una buena probabilidad de estar presentes en una nueva generación. Un individuo con un valor de ajuste por encima de la media tienen buenas probabilidades de ser elegido como padre en el paso 1 y al mismo tiempo de ser seleccionado en una nueva población de *pop_size -r* individuos. Si esto sucede así uno de sus hijos ocuparía alguna de las restantes r posiciones.

- Se aplican los operadores genéticos a todos los individuos. Esto produce un tratamiento uniforme de todos los operadores utilizados en el programa de evolución. Así si se utilizaran tres operadores (mutación, cruce e inversión) algunos de los padres se utilizarían bajo mutación, algunos otros con cruce y el resto con inversión

- En AGm se produce una mejor utilización de los recursos de almacenamiento disponibles. El nuevo algoritmo evita dejar duplicados de un mismo cromosoma en la nueva población. Lo cual podría llegar a producirse por otros medios pero es muy improbable. Por otro lado el AGc era muy vulnerable a la creación de duplicados.

- Los AGm para valores pequeños de r pertenece a la clase de AG de Estado Estacionario (SSGA). La diferencia entre los GAs y loa SSGAs está en que solo unos pocos individuos de la población son cambiados.

- En modGA pop_size-r cromosomas son colocados en una nueva población sin introducirles ningún cambio. En particular si r=1, solo un cromosoma es reemplazado en cada generación.

4.2 Características de la función

Existen una serie de medidas, que pueden ser tomadas y que pueden ser muy útiles, en relación a las características de la función a ser optimizada. Podemos dividir estos mecanismos en tres categorías, de acuerdo con *Golberg*:

1) *Escalado Lineal.* En este método el valor de ajuste actual del cromosoma se escala de la siguiente manera:

$$f_i' = a \cdot f_i + b$$

Los parámetros *a* y *b* son seleccionados para que el valor de ajuste promedio se mantenga y el mejor valor de ajuste se vea incrementado por un múltiplo deseado del valor de ajuste promedio. Este mecanismo, aunque bastante potente, puede introducir valores de ajuste negativo, con los que se debe dar un trato adecuado. Además los parámetros *a* y *b* son normalmente fijados por la vida de la población y no son dependientes del problema.

2) *Truncación sigma*.

$$f_i' = f_i + (\bar{f} - c \cdot \sigma)$$

c es elegido como un entero bajo en el rango [1,5]
σ es la desviación estándar del ajuste de la población. Los valores negativos de f' son puestos a 0.

3) *Escalado potencial*. En este método el nuevo valor de ajuste se calcula como:

$$f_i' = f_i^k$$

Con *k* cercano a la unidad. De algunos estudios se deduce que la elección de k debería ser dependiente del problema.

4.2.1 Condiciones de terminación del algoritmo.

Es importante destacar la importancia de la condición de terminación utilizada en nuestro algoritmo.

La condición de terminación más simple es aquella que va comprobando el número actual de generación *t,* la búsqueda se termina si el número total de generaciones excede un valor constante predefinido., es decir si $t \leq T$.

En muchas versiones de programas de evolución, no todos los individuos necesitan ser reevaluados, algunos de ellos pasan a la siguiente generación sin sufrir ninguna alteración. En tales casos podría ser útil contar el número de evaluaciones de cada cromosoma y terminar la búsqueda cuando el número de evaluaciones exceda algún valor constante predefinido.

Sin embargo, las condiciones de terminación anteriores suponen el conocimiento del usuario de las características de la función, lo cual influye en la longitud de la búsqueda. Parece que sería mucho mejor si el algoritmo terminase la búsqueda cuando la probabilidad de una mejora significativa fuese muy pequeña.

Existen dos categorías para las condiciones de terminación:

- Basadas en la estructura del cromosoma, trabajan observando el número de genes del cromosoma que han convergido. Un gen se considera que ha convergido si algún porcentaje predeterminado de la población tiene el mismo valor para ese gen. Si el número de gen convergidos supera algún valor de porcentaje total la búsqueda se da por concluida.

- Basadas en el significado de un cromosoma., miden el progreso realizado por el algoritmo en un determinado número de generaciones. Si el progreso conseguido es menor que algún valor épsilon (que es un parámetro del método) la búsqueda se termina.

4.3 Algoritmos genéticos (Constractive mapping)

Una posible aproximación a la explicación de las propiedades de convergencia de los algoritmos genéticos quizás estén basadas en el teorema del punto fijo de Banach. Este teorema suministra una explicación intuitiva a la convergencia de los algoritmos genéticos (sin elitismo), la única condición requerida es que debería haber una mejora en las subsiguientes poblaciones (que no tienen por que ser una mejora del mejor individuo de la población).

4.4 Algoritmos genéticos con variación de la población.

La elección del tamaño de la población *pop_size* es muy importante, y puede ser un factor crítico en muchas aplicaciones.

Si el tamaño de la población es muy pequeño el algoritmo genético puede converger muy pronto. Si es muy grande quizás este malgastando recursos computacionales, el tiempo de espera para producir una mejora podría ser elevado.

Tanto la diversidad de la población como la presión de selección están influenciados por el tamaño de la población.

En esta sección se discutirá un algoritmo genético con tamaño de la población variable. (GAVaPS). Este algoritmo introduce el concepto de "edad" de un cromosoma, que es equivalente al número de generaciones que un determinado cromosoma permanece vivo.

La edad de un cromosoma reemplaza al concepto de selección, y va ha influir al tamaño de la población en cada etapa del proceso de evolución.

Parece razonable suponer que según el estado del proceso de evolución, habrá un operador genético que tendrá más importancia en la mejora de la población. Del mismo modo según el estado de la población habrá un tamaño de la población que será el más adecuado u óptimo.

El esquema del GAVaPS es:

```
t=0
inicializar P(t)
evaluar P(t)

while ( condición de terminación) do
t=t+1
Incrementar la edad de cada individuo en 1
Recombinar P(t)
Evaluar P(t)
Eliminar de P(t) todos los individuos cuyo
Edad sea mayor que su tiempo de vida.
end
```

El algoritmo GAVaPS procesa en el instante t a la población de cromosomas P(t). Durante el paso de recombinación, se genera una población nueva auxiliar (es una generación de hijos). El tamaño de esta población auxiliar es proporcional al tamaño de la población original.

$$AuxPopSize(t) = Pop\_size(t) \cdot \rho$$

Donde ρ es *la razón de reproducción*. Cada cromosoma de la población original puede ser elegido para reproducirse (es decir, situar los hijos en la población auxiliar) con igual probabilidad, independientemente de su valor de ajuste. Los hijos se crean tras aplicar los operadores de cruce y mutación a los cromosomas seleccionados. Como no hay ningún paso de selección se tiene que trabajar con los conceptos de *edad* y *tiempo de vida* de un cromosoma.

El parámetro *tiempo de vida*, es asignado una sola vez a cada cromosoma (bien en la población inicial o bien a los hijos nuevos de la población auxiliar). Este tiempo de vida se mantiene constante desde el nacimiento del cromosoma hasta su muerte.

La muerte de un determinado cromosoma ocurre cuando su edad excede a su tiempo de vida. El tiempo de vida de un cromosoma determina el numero de generaciones durante las cuales ese cromosoma es mantenido dentro de la población. Después de que su tiempo de vida es superado el cromosoma muere.

El tamaño de la próxima generación P(t+1) vendrá dado por:

$$Pop\_size(t+1) = Pop\_size(t) + AuxPopSize(t) - D(t)$$

Donde D(t) es el número de cromosomas que mueren en la generación P(t).

Las estrategias de calculo del tiempo de vida deberían conseguir dos objetivos:

1) Reforzar a los individuos que presenten un valor de ajuste por encima de la media.
2) Ajustar el tamaño de la población en cada estado del proceso de evolución.

Puesto que la probabilidad de que un cromosoma de la población original sea elegido para aportar hijos a la población auxiliar es la misma para todos los cromosomas, el número esperado de hijos que aportará cada cromosoma será proporcional a su tiempo de vida. Así los individuos con un valor de ajuste deberían tener garantizado un tiempo de vida superior al resto de individuos.

Para el cálculo del tiempo de vida se van a utilizar los siguientes valores, obtenidos del estado actual t del proceso de búsqueda:

AvgFit	valor de ajuste promedio
MaxFit	valor de ajuste máximo.
MinFit	valor de ajuste mínimo
AbsFitMax	valor de ajuste máximo encontrado hasta esta generación
AbsFitMin	valor de ajuste mínimo encontrado hasta esta generación

Por otra parte el cálculo del tiempo de vida debería ser computacionalmente sencillo, con el fin de ahorrar tiempo de cálculo.

El tiempo de vida para el individuo i-esimo puede ser determinado(para problemas de maximización, funciones de coste tomando siempre valores positivos) por las siguientes ecuaciones:

1. Asignación proporcional:

$$lifetime[i] = min\left(MinLT + \eta \cdot \frac{fitness[i]}{AvgFit}, MaxLT \right)$$

$$\eta = \frac{1}{2}\left(MaxLT - MinLT\right)$$

Donde MinLT y MaxLT hacen referencia al tiempo de vida asignado mínimo y máximo respectivamente.

Esta asignación esta basada en la idea de selección mediante el tiro de una ruleta.El valor del tiempo de vida para el individuo [i] es proporcional a su valor de ajuste dentro

del rango [MinLT, MaxLT]. La principal desventaja de esta técnica de cálculo es que no tiene en cuenta la "bondad objetiva" de cada individuo., que puede ser calculada relacionando su valor de ajuste con el mejor valor de ajuste encontrada hasta esta generación t

2. *Asignación lineal*:

$$lifetime[i] = MinLT + 2\eta \cdot \frac{fitness[i] - AbsFitMin}{AbsFitMax - AbsFitMin}$$

$$\eta = \frac{1}{2}(MaxLT - MinLT)$$

En esta técnica el tiempo de vida es calculado en concordancia al valor de ajuste relacionado con el mejor valor actual. Sin embargo, si muchos individuos tienen su valor de ajuste igual o aproximadamente igual al del mejor individuo se producen tiempos de vida altos, que aumenta mucho el tamaño de la población.

3. *Asignación bilineal:*

$$lifetime[i] = \begin{cases} MinLT + \eta \cdot \dfrac{fitness[i] - MinFit}{AvgFit - MinFit} & si \quad AvgFit \geq fitness[i] \\ \dfrac{1}{2}(MinLT + MaxLT) + \eta \cdot \dfrac{fitness[i] - AvgFit}{MAxFit - AvgFit} & si \quad AvgFit < fitness[i] \end{cases}$$

$$\eta = \frac{1}{2}(MaxLT - MinLT)$$

Esta tercera técnica intenta hacer un compromiso entre las dos anteriores . Hace más pronunciada la diferencia entre los tiempos de vida de los individuos cercanos al mejor utilizando información del valor de ajuste medio, sin embargo también tiene en cuenta consideraciones de los valores de ajuste máximo y mínimo encontrados hasta esta etapa del proceso de evolución.

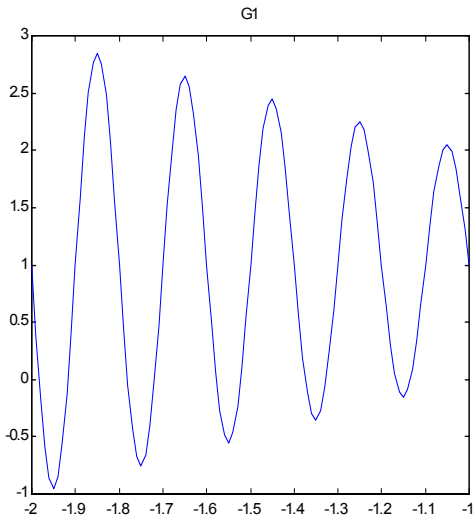
Algunas funciones donde se probó el GAVaPS fueron :

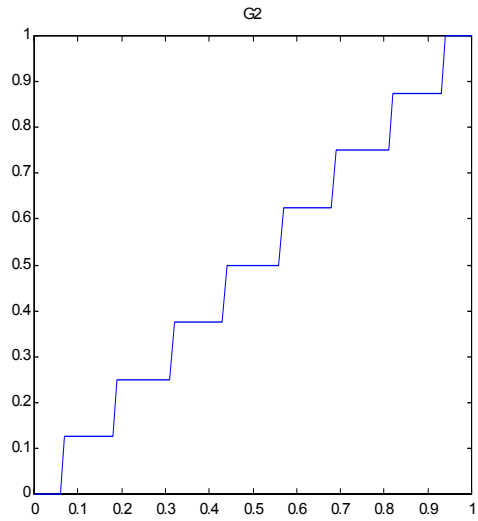
$$G1: \quad -x\sin(10\pi x) + 1 \quad -2.0 \leq x \leq 1.0$$

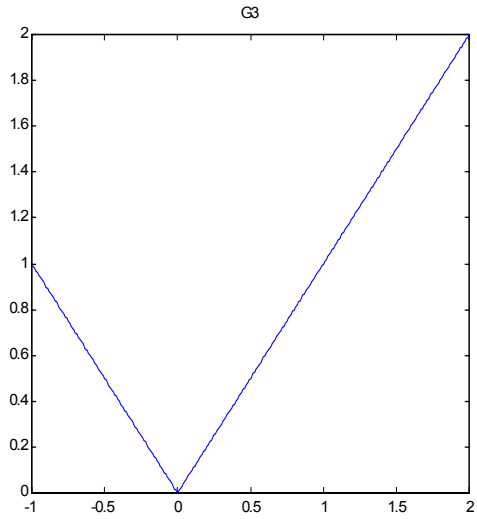
$$G2: \quad \frac{\text{int}\, eger(8x)}{x} \quad 0.0 \leq x \leq 1.0$$

$$G3: \quad x \cdot \text{sgn}(x) \quad -1.0 \leq x \leq 2.0$$

$$G4: \quad 0.5 + \frac{\sin^2\sqrt{x^2 + y^2} - 0.5}{\left(1 + 0.001(x^2 + y^2)\right)^2} \quad -100 \leq x, y \leq 100$$







El comportamiento del GAVaPS ha sido probado y comparado con el Algoritmo genético simple propuesto por Golberg (SGA). Para hacer esta prueba se ha utilizado representación binaria, y dos operadores genéticos: mutación y operador de cruce en un punto.

Tamaño inicial de la población (Para el SGA este valor permanece constante)	20
Tasa de reproducción ρ (No afecta al SGA)	0.4
Probabilidad de mutación p_m	0.015
Probabilidad de cruce p_c	0.65
Longitud del cromosoma m	20
Tiempo de vida máximo MaxLT (Permanece constante)	7
Tiempo de vida máximo MinLT (Permanece constante)	1
Número de carreras	20
conv	20

Para realizar la comparación de ambos algoritmos se eligieron dos parámetros:

- *evalnum(t)* es valor medio del número de evaluaciones de la función de coste realizadas a lo largo de todas las carreras para cada generación.
- *avgmax(t)* es el valor medio de los valores de ajuste máximos encontrados en todas las carreras para cada generación.

La condición de terminación de ambos algoritmos es la misma : Terminarán si no hay ninguna mejora significativa en el mejor valor de ajuste encontrado, durante conv=20 generaciones consecutivas.

En la *Tabla 5* se muestra pop_size(t) y el valor de ajuste medio de la población para una única carrera del GAVaPS, para el caso de G4 con técnica bilineal para el cálculo del tiempo de vida media.

La forma de pop_size(t) es muy interesante, Al principio cuando la variación del valor de ajuste es relativamente alto , el tamaño de la población aumenta. El GAVaPS está haciendo una búsqueda amplia del óptimo. Una vez que se encuentra en las proximidades de un posible óptimo, el algoritmo comienza a converger y se reduce el tamaño de la población.

El GAVaPS incorpora un mecanismo de auto ajuste, a través de la posibilidad de ir regulando el tamaño de la población en cada etapa del proceso de evolución.

Para el GAVaPS, la elección del valor de la tasa de reproducción ρ, influye fuertemente en el coste de la simulación evalnum(t). No obstante sin una pérdida de avgmax se puede reducir su valor. Así se encuentra que $\rho=0.4$ es una buena elección.

Como era de esperar para el SGA un tamaño de población pequeño implica un coste bajo y un pobre comportamiento. Incrementar el tamaño de la población supone en principio una mejora en el comportamiento, acompañada de un aumento en el coste. De hecho ocurre que hay un estado de saturación del comportamiento cuando todavía se está produciendo un crecimiento lineal del coste.

Para el caso del GAVaPS, el valor inicial de la población no tiene ninguna influencia ni en su comportamiento (muy bueno) ni en su coste (razonable y suficiente para un muy buen comportamiento).

Se ha de indicar que el SGA tiene la conducta óptima (alto comportamiento con bajo coste) cuando se corre con distintos tamaños de la población, para las cuatro funciones G1,G2,G3 y G4.

Tipo de algoritmo	Funciones							
	G1		G2		G3		G4	
	V	E	V	E	V	E	V	E
SGA	2.814	1467	0.875	345	1.996	1420	0.959	2186
GAVzPS(1) proporcional	2.831	1708	0.875	970	1.996	1682	0.969	2133
GAVzPS(2) lineal	2.841	3040	0.875	1450	1.996	2813	0.970	3739
GAVzPS(3) bilineal	2.813	1538	0.875	670	1.996	1555	0.972	2106

Tabla 5: Comparación entre las tres estrategias

En la *Tabla 5*, *V* hace referencia al mejor valor encontrado y *E* es el número de evaluaciones de la función de coste.

La estrategia lineal GAVzPS(2) se caracteriza por tener el mejor comportamiento y desgraciadamente el mayor coste. En otro lado se encuentra GAVzPS(3) bilineal, que es la más barata de todas en lo que a coste se refiere, pero su comportamiento no es tan bueno como en el caso lineal.. GAVzPS(1) proporcional suministra un comportamiento y un coste medio. Además GAVzPS para cualquier estrategia del tiempo de vida suministra un comportamiento mejor que en el caso del SGA., no obstante los resultados para el SGA mostrados en la tabla fueron obtenidos para el valor ideal de tamaño de población (que no tenía por que ser 20).

El conocimiento que se tiene acerca del valor de los parámetros característicos del GAVzPS, es fragmentario y se encuentra basado en la experiencia práctica.

4.5 Algoritmos genéticos y restricciones.

Existen tres formas de trabajar con AG y problemas de optimización que presenten restricciones.

4.5.1 Funciones de penalización

Una manera de trabajar con aquellos cromosomas que violan las restricciones del problema que se pretende resolver, es generar todo tipo de candidatos y luego penalizar utilizando una *función de penalización pen(**x**)* aquellos candidatos que no sean válidos. Existe una gran variedad de funciones de penalización, que pueden ser aplicadas de tipo constante, lineal cuadrático, exponencial, dependiendo del grado de violación que se haya producido así será la penalización que se imponga.

$$max(F(\vec{x})) = max(f(\vec{x}) - pen(\vec{x}))$$
$$pen(\vec{x}) = g(\vec{x}) + a$$

Otra posible penalización es eliminar a los individuos que no sean válidos. Esta técnica fue utilizada con éxito en estrategias de evolución para problemas de optimización. Esta técnica presenta como principal inconveniente, para algunos problemas utilizando operadores genéticos tradicionales la posibilidad de encontrar una solución válida es

muy pequeña, y se puede gastar mucho tiempo en generar y evaluar soluciones que no son válidas.

4.5.2 ***Algoritmos de reparación***.

Otra categoría de métodos para tratar restricciones está basada en la aplicación de algún algoritmo especial de reparación para corregir las soluciones generadas que no sean válidas por violar las restricciones.

Este clase de algoritmos requieren mucho gasto computacional y debe ajustarse para cada tipo de problema que se intente resolver.

Además puede ocurrir que el proceso para corregir la solución quizás sea más complicado de resolver que el problema original.

4.5.3 Decodificadores.

Este método se basa en la utilización de decodificadores que garanticen la generación de una solución válida o el uso de operadores específicos del problema que aseguren el generar una solución válida. Los decodificadores requieren mucho gasto computacional y no todas las restricciones pueden ser implementadas fácilmente de esta manera.

5. ¿Representación binaria o flotante ?

La representación binaria tiene algunos inconvenientes cuando es utilizada problemas de alta precisión numérica y multidimensionales. Por ejemplo, para un problema de 100 variables con dominios en el rango [-500,500], con una precisión de seis dígitos decimales, la longitud del cromosoma binario que se necesitaría es 3000. Lo que genera un espacio de búsqueda de 10^{1000}. Para estos problemas los algoritmos genéticos con representación binaria actúan mal.

Algunas de las ventajas de la representación binaria es la posibilidad de realizar un tratamiento teórico y conseguir operadores genéticos que actúen muy elegantemente.

5.1 Comparación entre la representación binaria y en coma flotante.

La representación en coma flotante FP permite representar dominios bastante amplios, mientras que la representación binaria B debe sacrificar precisión para poder ampliar la anchura del dominio de búsqueda, para una longitud de cromosoma constante. En la FP es más fácil trabajar con restricciones.

Vamos a comparar ambas representaciones en relación a los operadores genéticos implementados:

- *Mutación aleatoria y cruce*: Los resultados son ligeramente mejores para la representación B. Por otra parte la representación FP es más estable con una desviación estándar menor. La mutación FP se comporta más aleatoriamente que la mutación B, ya que cambiar un solo bit no significa cambiar drásticamente el valor del cromosoma.

- *Mutación no uniforme*: El operador de mutación no uniforme FP presenta un mejor comportamiento en promedio, además es más estable que el operador B.
- *Cruce en varios puntos*. La representación FP tiene una superioridad aplastante.
- *Tiempo de cálculo*. El algoritmo en representación FP es mucho más rápido que el algoritmo en representación B.

Como conclusión se puede decir que la representación en coma flotante, es más rápida, más consistente de carrera a carrera y suministra una mayor precisión (sobre todo en rangos de dominio anchos). Además la representación FP permite un mayor acercamiento al espacio del problema y es más fácil diseñar otros operadores que incorporen un conocimiento especifico del problema.

6. Tratando restricciones.

Se entenderá por las abreviaturas NPL , problema no lineal . Consistirá en optimizar una función $f(x)$, con $x=(x_1,.....,x_q)$, sometido a las m siguientes restricciones:

$$g_j(x)=0 \text{ con } j=1,....,p. \text{ IGUALDADES}$$
$$h_j(x)\leq 0 \text{ con } j=p+1,.....,m. \text{ INIGUALDADES}$$

6.1 El sistema GENECOP.

En un problema de optimización con restricciones la forma geométrica del conjunto de soluciones en R^q es muy importante. Se va trabajar con conjuntos convexos.

Se desea optimizar $f(x_1, ...,x_q)$ perteneciente a R, donde $(x_1, ...,x_q)$ pertenece al conjunto D que es un subconjunto de R^q. Además se supone que *D* es convexo.

El dominio de D está definido por los siguientes rangos:

$$l_1\leq x_1\leq r_1$$
$$l_2\leq x_2\leq r_2$$
$$.$$
$$.$$
$$l_q\leq x_q\leq r_q$$

Y por el conjunto de restricciones C.

Propiedad 1:
Sea D un conjunto convexo. Se cumple que $y\in$ {left(k),rigth(k)} si $(x_1,x_2,...,x_{k-1},y,x_{k+1},...,x_q) \in D$ donde todos los x_i con i=1,...,k-1,k+1,...,q permanecen constantes.

La propiedad 1, es una propiedad básica utilizada por todos los operadores de mutación, cuando se considera representación en coma flotante.

Si el conjunto C de restricciones está vacío, en ese caso se cumple que:

left(k)=l(k) u rigth(k)=r(k) con k=1,....,q.

Ejemplo:

$-3 \le x_1 \le 3 \Rightarrow l(1)=-3\ r(1)=3$

$0 \le x_2 \le 8 \Rightarrow l(2)=0 \quad r(2)=8$

$x_1^2 \le x_2 \le x_1+4$ sería una restricción de C.

Sea el punto $(2,5) \in D$

Si $x_2=5$, para x_1 se cumple por la restricción de C que $1 \le x_1 \le \sqrt{5}$, luego left(1)=1 ,rigth(1)= $\sqrt{5}$.

Si $x_1=1$, para x_2 se cumple por la restricción C que $4 \le x_2 \le 6$

Propiedad 2:
Sean $\mathbf{x_1}$ y $\mathbf{x_2}$ del espacio de soluciones D. Entonces $a\mathbf{x_1}+(1-a)\mathbf{x_2}$ con $a \in [0,1]$ también es un vector solución perteneciente a D.

Esta propiedad es importante para los operadores de cruce.

Considérese una clase particular de problemas de optimización que son definidos sobre un conjunto convexo.

Optimizar $f(x_1,x_2,....,xq)$ sujeto a las restricciones siguientes:

1) *Restricciones del dominio*:

$$\vec{l} \le \vec{x} \le \vec{u}$$

$$\begin{bmatrix} l_1 \\ . \\ . \\ l_q \end{bmatrix} \le \begin{bmatrix} x_1 \\ . \\ . \\ x_q \end{bmatrix} \le \begin{bmatrix} u_1 \\ . \\ . \\ u_q \end{bmatrix}$$

2) *p restricciones de igualdad*:

$$A\vec{x} \le b$$

$$\begin{bmatrix} a_{11} & a_{1q} \\ & \\ a_{p1} & a_{pq} \end{bmatrix}_{pxq} \cdot \begin{bmatrix} x_1 \\ . \\ . \\ x_q \end{bmatrix}_{qx1} \le \begin{bmatrix} b_1 \\ . \\ . \\ b_p \end{bmatrix}_{px1}$$

$[a_{ij}]$ i=1,...p j=1,....q

3) *m restricciones de desigualdad*:

$$C\vec{x} \le d$$

$$\begin{bmatrix} c_{11} & & c_{1q} \\ & & \\ c_{m1} & & c_{mq} \end{bmatrix}_{mxq} \cdot \begin{bmatrix} x_1 \\ . \\ . \\ x_q \end{bmatrix}_{qx1} \le \begin{bmatrix} d_1 \\ . \\ . \\ d_m \end{bmatrix}_{mx1}$$

$[c_{ij}]$ i=1,...m j=1,....q

Se denomina GENOCOP a la abreviatura de Genetic algorithm for Numerical Optimization for Constrained Problems)

GENOCOP se basa en dos ideas principales:

- Eliminar el conjunto de p igualdades presentes en las restricciones.
- Diseño cuidadoso de los operadores genéticos que garanticen que todos los cromosomas están dentro del espacio de soluciones restringido.

Supóngase que dentro del conjunto de p restricciones de igualdad hay p variables independientes que pueden ser expresadas en función de las restantes q-p. Entonces podemos reescribir las ecuaciones de igualdades de la siguiente manera :

$$(x_{i1}, x_{i2}, ..., x_{ip}) \subseteq (x_1, x_2, ..., x_q)$$

$$A_1\vec{x}^1 + A_2\vec{x}^2 = b$$

$$\vec{x}^1 = -A_1^{-1}A_2\vec{x}^2 + A_1^{-1}b$$

Tras eliminar las igualdades se tienen las siguientes ecuaciones :

1) Restricciones de dominio originales:

$$\vec{l}_2 \le \vec{x}_2 \le \vec{u}_2$$

2) Nuevas desigualdades:

$$\vec{l}_1 \le A_1^{-1}b - A_1^{-1}A_2\vec{x}^2 \le \vec{u}_1$$

3) Desigualdades originales:

$$\left(C_2 - C_1A_1^{-1}A_2\right)\vec{x}_2 \le \vec{d} - C_1A_1^{-1}b$$

Ejemplo:

$f(x_1,x_2,x_3,x_4,x_5,x_6)$ sujeta a las restricciones :

$$2x_1 + x_2 + x_3 = 6$$
$$x_3 + x_5 - 3x_6 = 10$$
$$x_1 + 4x_4 = 3$$
$$x_2 + x_5 \leq 120$$
$$\vec{l} = [-40 \quad 50 \quad 0 \quad 5 \quad 0 \quad -5]$$
$$\vec{u} = [20 \quad 75 \quad 10 \quad 15 \quad 20 \quad 5]$$

Existen tres incógnitas que se pueden expresar en función de las otras tres:

$$\begin{bmatrix} 2 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -3 \\ 0 & 4 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 6 \\ 10 \\ 3 \end{bmatrix}$$
$$A_1\vec{x}^1 + A_2\vec{x}^2 = b$$

6.2 Operadores para representación en coma flotante.

6.2.1 Mutación Uniforme

Se selecciona una componente k del cromosoma padre aleatoriamente. Y se genera el valor de x_k' aleatoriamente utilizando una distribución de probabilidad uniforme dentro del rango {left(k),rigth(k)}.

PADRE : $(x_1, x_2, ..., x_k, ..., x_q)$ => HIJO : $(x_1, x_2, ..., x_k', ..., x_q)$

Entre las características que presenta están:

- Juega un papel muy importante en las fases iniciales del proceso de evolución.
- En las últimas fases del proceso de evolución el operador permite moverse lejos de un óptimo local en busca de un punto mejor.

6.2.2 Mutación frontera

Se selecciona una componente k del cromosoma padre aleatoriamente. Y se cambia su valor a left(k) o a rigth(k) aleatoriamente, ambos valores deben tener la misma probabilidad de obtenerse.

PADRE: $(x_1, x_2, ..., x_k, ..., x_q)$ => HIJO : $(x_1, x_2, ..., x_k', ..., x_q)$

con x_k'=left(k) o rigth(k).

Este operador es útil si el conjunto C de restricciones sea no vacio, y la posible solución caiga en la frontera del espacio de busqueda. En el caso de que C sea el conjunto vacio este operador no es útil.

6.2.3 Mutación no uniforme.

Es el responsable de conseguir ajustes finos en el proceso de evolución.

$$\text{PADRE} : (x_1, x_2, ..., x_k, ..., x_q) \Rightarrow \text{HIJO} : (x_1, x_2, ..., x_k', ..., x_q)$$

con:

$$x_k' = \begin{cases} x_k + \Delta(t, rigth(k) - x_k) & si \quad digito \quad aleatorio\ 0 \\ x_k - \Delta(t, x_k - legth(k)) & si \quad digito \quad aleatorio\ 1 \end{cases}$$

La función Δ(t,y) da un valor dentro del rango [0,y], la probabilidad de que Δ(t,y) este cercano al valor 0 aumenta cuando t aumenta, siendo t el número de generación.

Esta propiedad produce que este operador busque en el espacio unifórmente inicialmente, cuando t pequeña, y muy localmente cuando t aumenta.

$$\Delta(t, y) = y \cdot r \cdot \left[1 - \frac{t}{T}\right]^b$$

Donde:

r ∈ [0.,1.]
T número máximo de generaciones.
b parámetro del sistema para determinar el grado de no uniformidad.(por ejemplo b=2).

6.2.4 Cruce Aritmético

Los sistemas que utilizan el cruce aritmético son más estables.

PADRES : $\mathbf{x_1}=(x_1^1,\ldots\ldots\ldots\ldots\ldots,x_q^1)$ $\mathbf{x_2}=(x_1^2,\ldots\ldots\ldots\ldots\ldots,x_q^2)$

HIJOS : $\mathbf{x_1}'=a\cdot\mathbf{x_1}+(1-a)\,\mathbf{x_2}$ $\mathbf{x_2}'=a\cdot\mathbf{x_2}+(1-a)\,\mathbf{x_1}$

Donde a $\in$ [0.,1.] elegido aleatoriamente.

6.2.5 Cruce simple.

Se elige un punto de cruce k, aleatoriamente :

$$\text{PADRES}: \begin{matrix}\vec{x}_1=(x_1,\ldots\ldots\ldots,x_q)\\ \vec{x}_2=(y_1,\ldots\ldots\ldots,y_q)\end{matrix}$$

$$\text{HIJOS}: \begin{matrix}\vec{x}_1'=\left(x_1,\ldots,x_k,y_{k+1}\cdot a+x_{k+1}\cdot(1-a),\ldots\ldots\ldots,y_q\cdot a+x_q\cdot(1-a)\right)\\ \vec{x}_2'=\left(x_1,\ldots,x_k,x_{k+1}\cdot a+y_{k+1}\cdot(1-a),\ldots\ldots\ldots,x_q\cdot a+y_q\cdot(1-a)\right)\end{matrix}$$

El problema que se plantea con este operador es encontrar el valor de a que asegure el mayor intercambio de información posible.

Se puede comenzar con un valor de a=1 y si $\mathbf{x_1}'$ o $\mathbf{x_2}'$ no pertenecen a D, se puede decrementar a multiplicándola por un factor 1/ρ. Si ρ aumenta a tiende a cero y los cromosomas hijo será los mismos que los cromosomas padre.

Algunas propiedades del cruce simple son:

- El cruce simple produce un efecto similar al aritmético.
- La inestabilidad aumenta más utilizando este operador que el de cruce aritmético.

6.2.6 Cruce heuristico

Este operador se caracteriza porque:

- Contribuye a un ajuste fino.
- Produce una búsqueda en la dirección adecuada.
- Utiliza valores de la función de coste para determinar la dirección de búsqueda.
- Produce un único hijo.
- Puede suceder que no produzca ningún hijo.

PADRES : $\mathbf{x_1}$, $\mathbf{x_2}$

HIJOS : $\mathbf{x_3} = r\cdot(\mathbf{x_2}-\mathbf{x_1})+\mathbf{x_2}$ con r$\in$ [0.,1.] elegido aleatoriamente y $f(\mathbf{x_2}) \geq f(\mathbf{x_1})$ para problemas de maximización y $f(\mathbf{x_2}) \leq f(\mathbf{x_1})$ si es de minimización..

Este operador puede generar hijos que no sean válidos en ese caso se prueba con otro valor de r. Si después de w intentos no genera ningún hijo válido se sale de este operador.

6.2.7 Comprobando GENOCOP.

Se hicieron comprobaciones de GENOCOP con los siguientes valores de los parámetros:

- pop_size=70.
- Número de padres en cada generación 28.
- b=2
- t=500 o 1000.
- Se ejecuta el algoritmo 10 veces y se toma la media.