Programación Evolutiva

Tema 4: Mejoras al esquema básico del AG

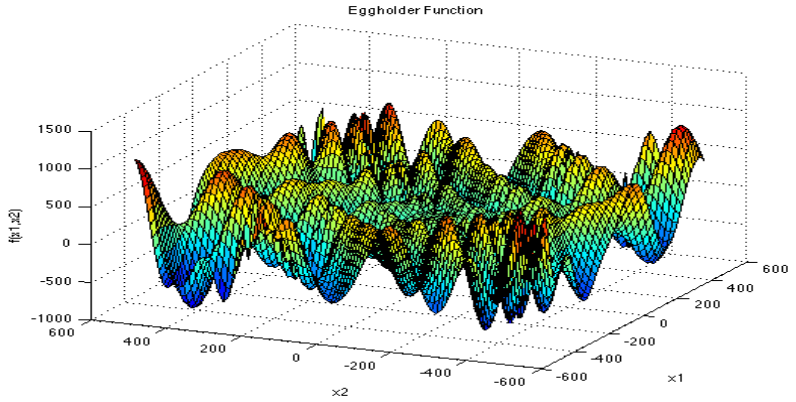
Carlos Cervigón, Lourdes Araujo 2023-2024.

Limitaciones de los AGs

- La representación: El espacio de búsqueda sólo se representa mediante cadenas binarias.
- La irrestricción: El mecanismo del AGS no considera las posibles restricciones o ligaduras que pudieran imponerse a la búsqueda.
- La opacidad: El AGS sólo está guiado por la aptitud de los individuos, y no incorpora ningún otro conocimiento específico del problema en cuestión.
- Ineficiencia: extravío o desorientación en la búsqueda

El problema de la ineficiencia

- **Extravío**: Tendencia al extravío de la búsqueda
  - El AG realiza la búsqueda de los mejores puntos utilizando únicamente la aptitud de los individuos
  - El AG no converge al óptimo global, desviándolo hacia un óptimo local (extravío).
- **Desorientación**:
  - A veces la información proporcionada es insuficiente para orientar la búsqueda del óptimo: desorientación (provocada por una mala codificación



Diversidad en los AGs

- Requisito de diversidad en los AGs:
  - **Diversidad de individuos**
  - **Diversidad de aptitudes.**
- Con poca variedad de individuos
  - El operador de cruce pierde la capacidad de intercambio de información útil entre individuos
  - La búsqueda se paraliza.
  - Todos tienen probabilidades similares de selección y todo recae en operadores genéticos (búsqueda aleatoria)

Convergencia prematura

- En algún momento de la evolución un individuo o un grupo obtiene una aptitud notablemente superior a los demás
- Evolución en avalancha:
  - **La diversidad disminuye**
  - En la siguiente generación se favorece aún más a los individuos más aptos hasta que dominan toda la población (superindividuos).
  - Los superindividuos sólo son los más aptos en cierto momento → convergencia prematura del AG hacia un subóptimo.
  - Este fenómeno puede ser deseable en la fase final.

Diversidad en el AG

- Diferencias demasiado grandes de los valores de adaptación impiden que el mecanismo de selección funcione adecuadamente:
  - Los de adaptación superior a la media consiguen una cantidad de copias de sí mismos en la siguiente generación muy superior al resto.
  - Esto hace que los restantes individuos tiendan a desaparecer, llegando nuevamente a una situación de falta de diversidad.

¿Mejoramos la selección?

¿Escalamos los valores de fitness?

Control de la diversidad: análisis de presión selectiva

PRESIÓN SELECTIVA:

- La mayor o menor tendencia a favorecer a los individuos más aptos
- Grado en el que se favorece a los individuos más adaptados.
- Cuantifica el número esperado de descendientes que se da al miembro más apto de la población

$$PresSel[t] := TamPob \cdot PuntMax[t]$$

$$PresSel[t] \equiv \frac{AptMax[t]}{AptMed[t]}$$

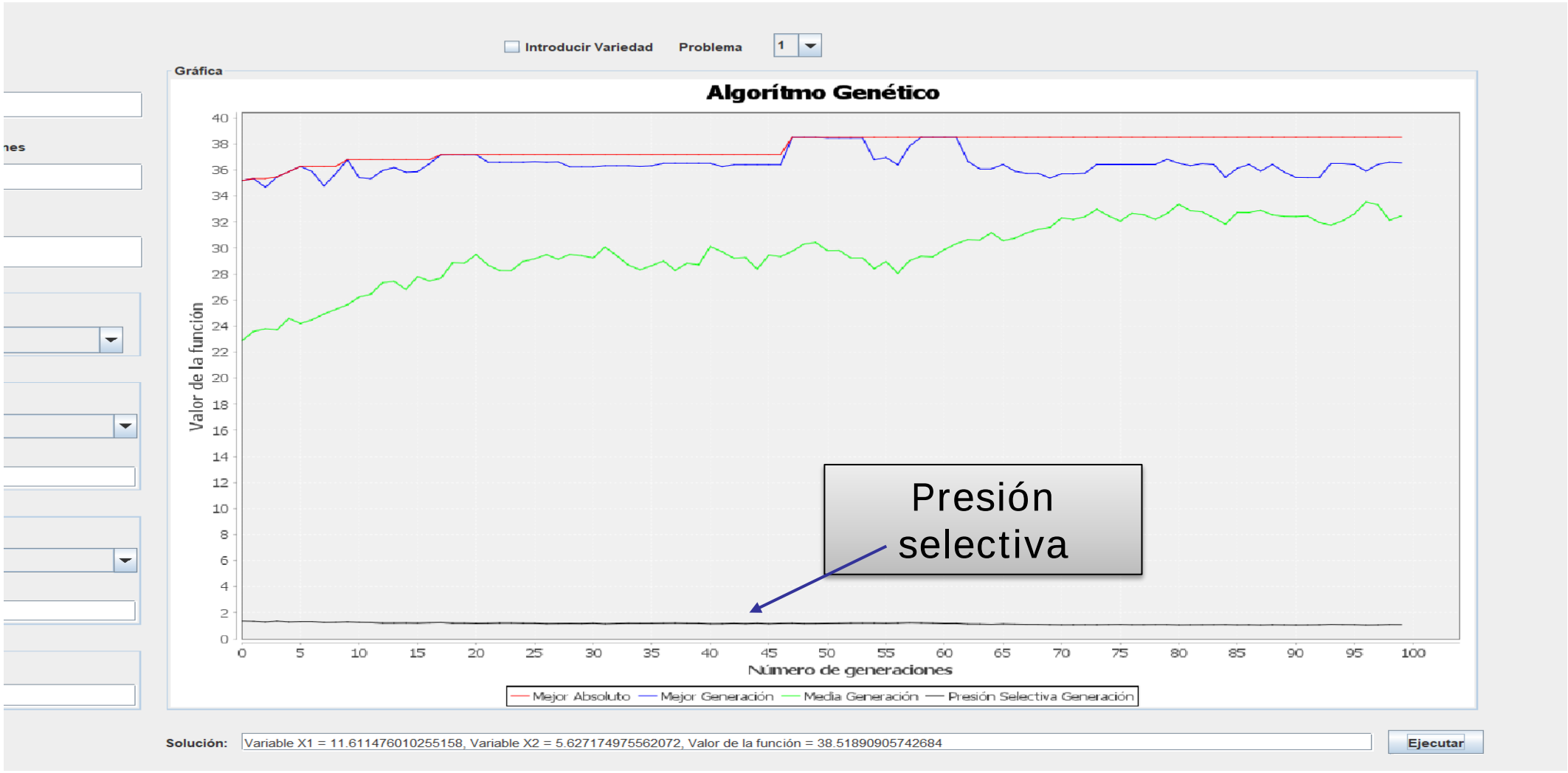
Control de la diversidad: análisis de presión selectiva

PRESIÓN SELECTIVA:

- Primeras fases de la evolución:
  - **Baja presión selectiva** con el fin de que la búsqueda sea más global
- Últimas fases de la evolución:
  - **Aumentar** la presión selectiva

- Whitley recomienda valores próximos a **1.5**
  - Valores mayores ocasionarían superindividuos
  - Valores menores frenarían la búsqueda sin ningún beneficio

Control de la diversidad



Ejemplo 1

- Dada estos datos de población

$$PresSel[t] := TamPob \cdot PuntMax[t]$$

$$PresSel[t] \equiv \frac{AptMax[t]}{AptMed[t]}$$

individuo	***aptitud i***	***P i***
I1	5	**0.116**
I2	12	**0.279**
I3	11	**0.255**
I4	8	**0.186**
I5	7	**0.162**
	43	

PreSel = 5 * 0.279 = 1.395

PreSel = 12/8.6 = 1.395

Ejemplo 2

- Dada estos datos de población

$$PresSel[t] := TamPob \cdot PuntMax[t]$$

$$PresSel[t] \equiv \frac{AptMax[t]}{AptMed[t]}$$

individuo	***aptitud*** *i*	***P i***
I1	1000	**0.994**
I2	1	**0.00099**
I3	2	**0.0019**
I4	3	**0.0029**
	1006	

PreSel = 4 * 0.994 = 3.976

PreSel = 1000/251.5 = 3.976

Métodos de control de la diversidad

- Nuevos métodos de selección
  - Selección por Ranking: mecanismo de muestreo insensible a la distribución de aptitudes
    - La ruleta es muy dependiente de las aptitudes y puede generar superindividuos muy rápido.
  - Modificar aptitudes
    - Desplazamiento de aptitud
    - Escalado de aptitud
- Otras opciones:
  - Controlar edades de los individuos.
  - Usar mecanismos de escisión.

Funciones de evaluación y adaptación

- Dada la función de evaluación original, la transformamos en función de adaptación o aptitud *revisada*.
  - Desplazamiento de la aptitud:
    - Tiene como finalidad hacer que la función de aptitud **devuelva valores positivos**
    - Nos permite **convertir un problema de minimización en otro de maximización**
  - Escalado de la aptitud:
    - Permite establecer una separación entre los valores de aptitud de cada individuo para que la selección funcione de forma adecuada

Desplazamiento de la aptitud

- Para poder aplicar correctamente algunos mecanismos de selección (por ejemplo ruleta) los valores de aptitud necesitan ser **positivos**
- Modificamos los valores de la función de aptitud, de forma que se obtengan valores positivos.
- Sea ***fmax*** el valor máximo de los valores de fitness o adaptación g_i de la población en una determinada generación.
- Entonces para cada individuo ***i***, la nueva adaptación o aptitud revisada f_i la definimos como:

$$f_i = fmax - g_i$$

- En lugar de utilizar ***fmax*** conviene utilizar valores ligeramente mayores (por ejemplo, un 105%) para prevenir que la adaptación revisada se haga nula.

Ejemplo minimización

$$f_i = fmax - g_i$$

Individuo	*Valor original g_i*	*Valor adaptado f_i $(f_{max}-g_i)$*	puntuación	
I1	-3	**10**	10/23	0.434
I2	-1	**8**	8/23	0.347
I3	2	**5**	5/23	0.217
I4	7	**0**	0/23	0
		23		

Desplazamiento de aptitud

fmax es el mejor valor de g_i *en t*

Desplazamiento de la aptitud o adaptación

```
public void corrigeMinimizar(double max) {
   //lo aplicaré en problemas de minimización
  for(int i = 0; i < this.tamPoblacion; i++)
     this.fitness[i] = (1.05 * max) - this.fitness[i];
  }
```

$$f_i = fmax - g_i$$

Ejemplo maximización

$$f_i = g_i + f\ \min$$

individuo	***Valor original*** g_i	***Valor adaptado*** f_i $(g_i + f_{mIN})$	**puntuación**	
I1	-3	**0**	0/17	0
I2	-1	**2**	2/17	0.117
I3	2	**5**	5/17	0.294
I4	7	**10**	10/17	0.588
		17		

Desplazamiento de aptitud

Fmin es el el peor valor absoluto de g(x) en t

Desplazamiento de la aptitud o adaptación

```
public void corrigeMaximizar(double min) {
//lo aplicaré en problemas de maximización
for(int i = 0; i < this.tamPoblacion; i++)
   this.fitness[i] = this.fitness[i] + Math.abs(min);
}
```

. . .

$$f_i = g_i + f \min$$

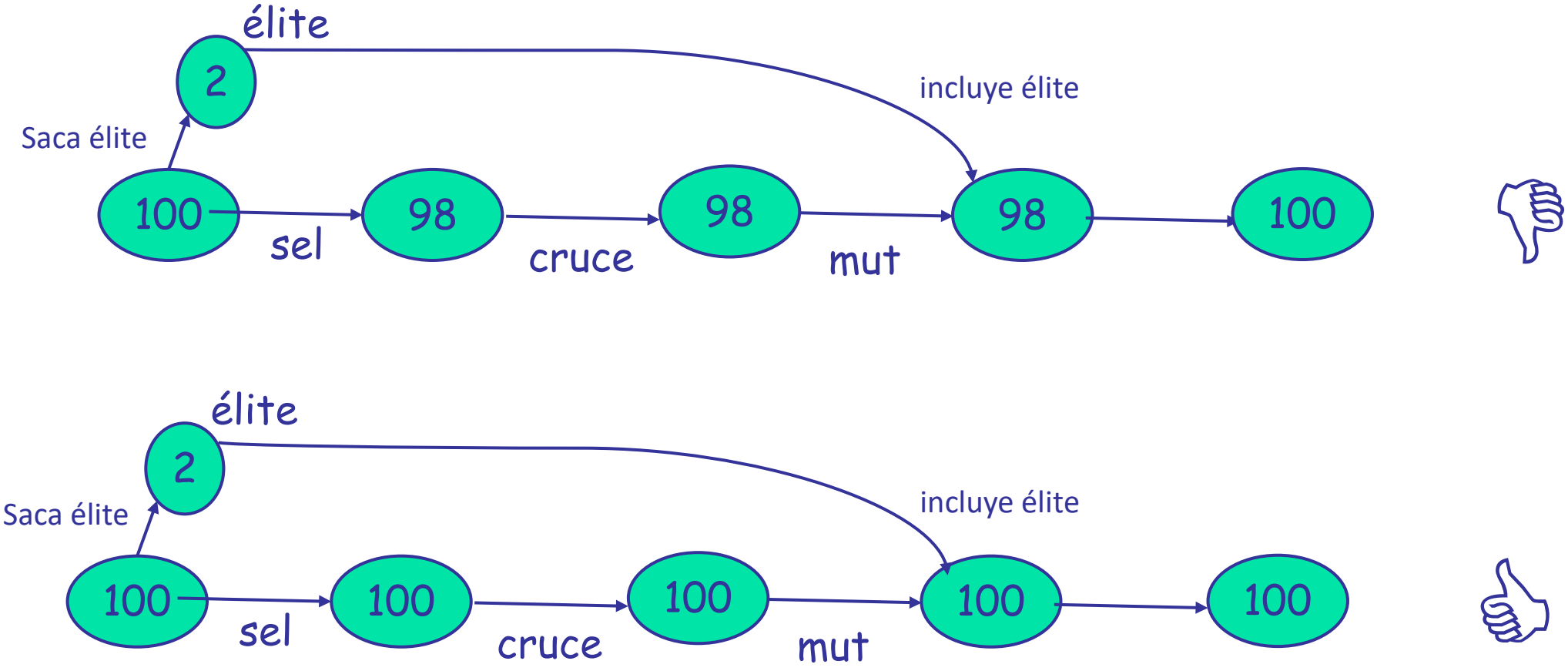
Elitismo

- El elitismo consiste en asegurar la supervivencia de un grupo con los mejores individuos de la población
- Mejora mucho la convergencia
- Los individuos de la élite ¿Se Cruzan? ¿Se mutan? Hay variantes
- Se muestrea una élite con los ***r*** mejores individuos de la población y se asegura su supervivencia, haciendo que se integren en la población para la siguiente generación.
- El porcentaje de la población que puede formar parte de la élite debe ser pequeño, en torno al 2% y 3% de la población

Elitismo

- La forma más sencilla de implementar el elitismo para una élite de *r* individuos es reservar los *r* mejores individuos de la generación anterior.
- Los restantes *N-r* individuos de la población se seleccionan de la forma usual.
  - Si se utiliza reemplazo inmediato, y los operadores de cruce y mutación se aplican sobre toda la población, incluida la élite, se pueden perder los mejores individuos.
  - Sin embargo, es importante que los individuos de la élite participen en las operaciones genéticas, ya que son los mejores.
- Dos variantes:

Elitismo


élite
2
Saca élite
incluye élite
100
sel
98
cruce
98
mut
98
100
élite
2
Saca élite
incluye élite
100
sel
100
cruce
100
mut
100
100

Elitismo

```
public void run() {
    iniciarPoblacion();
    while(this.generacionActual < this.maxGeneraciones) {
        generaElite();
        //Seleccion
        //Cruce
        //Mutación
        introducirElite();
        evaluar();
        . . .
        //Siguiente generacion
        generacionActual++;
     }}
```

Elitismo

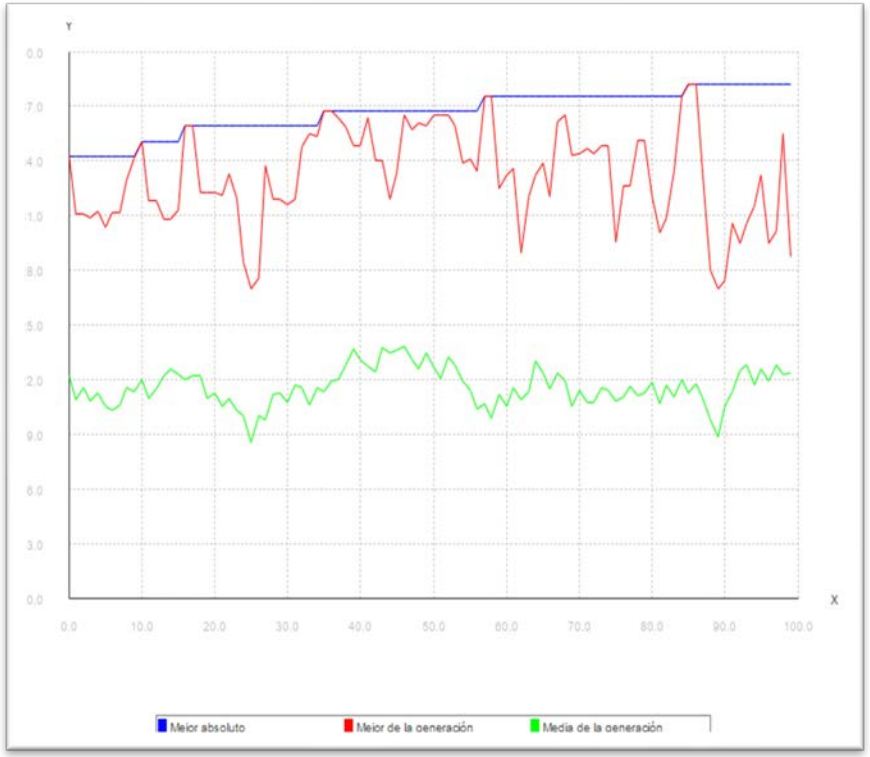
```
int tamElite = …
pob.evalua();
for(int i = 0; i < maxGeneraciones; i++ ){
   //Extraemos los mejores individuos de la población
   elite = pob.separaMejores(tamElite);

   //Aplicamos el proceso de seleccion/reproduccion/mutacion
    pob.selecciona();
    pob.reproduce(probCruce);
    pob.muta(probMutacion);
   //Volvemos a integrar a la élite
    pob.incluye(elite);
    delete(elite);
    pob.evalua();
 }
  return MejorIndividuo;
}
```

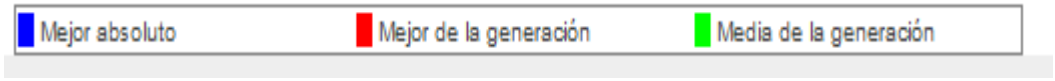
¡Con elitismo, la gráfica del mejor de la generación con la gráfica del mejor absoluto deben coincidir!

Gráfica maximización

- Sin elitismo



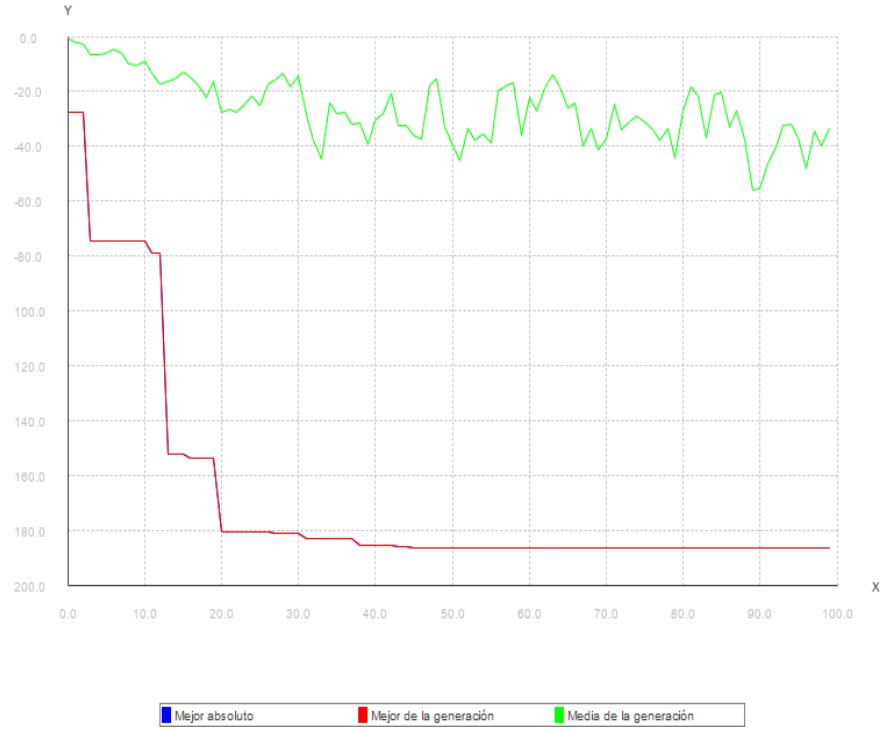
- Con elitismo

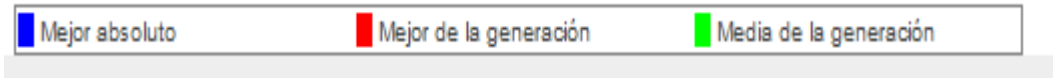


Gráfica minimización

- Sin elitismo

- Con elitismo





Elitismo

- Con tamaños de población pequeños se consiguen efectos similares a los del elitismo introduciendo reinicializaciones periódicas en los AGs:
  - Cada vez que el AG converge se salvan los mejores individuos, se reinicializan los demás y se vuelve a comenzar.
  - La reinicialización mejora la diversidad.
  - Se puede hacer una reinicialización periódica de toda la población cada ciertas generaciones

Escalado de la adaptación o fitness

- Permite controlar la diversidad de las aptitudes
- Permite establecer una separación entre los valores de adaptación de distintos individuos que sea apropiada para el funcionamiento de la selección
- Es una contracción o dilatación de la función de aptitud que tiene por objetivo que los puntos del espacio de búsqueda estén separados por una distancia adecuada.
- Tipos:
  - Lineal
  - Sigma
  - Potencial
  - Basado en el orden...

Escalado de la adaptación

- Al principio hay una tendencia a que unos pocos superindividuos dominen el proceso de selección:
  - Los valores de la función objetivo deben ser ***subescalados*** para evitar que los superindividuos ocupen la población.
- Al final, la competición entre los miembros de la población es menos intensa:
  - los valores de la función objetivo deben ser ***sobreescalados*** para acentuar las diferencias entre los miembros de la población con el fin de continuar favoreciendo a los mejores miembros.

Escalado lineal

- Se calcula la adaptación o fitness a partir de los valores proporcionados por la función de evaluación del siguiente modo

Valor escalado $\longrightarrow$ $f(x_i) = a * g(x_i) + b$

donde los parámetros *a* y *b* se calculan para cada generación de forma que verifiquen.

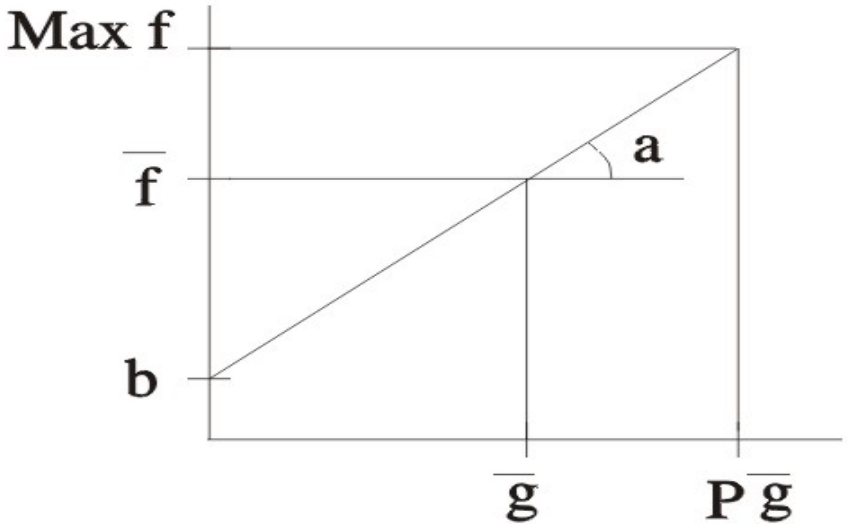
1. Deben coincidir los valores medios de la función de evaluación y de la función de aptitud en la población. $\overline{f} = \overline{g}$
2. La adaptación del mejor debe ser el producto por un cierto parámetro *P (entre 1.2 y 2)* por la adaptación media

$$f_{max} = P * \overline{g}$$

Número esperado de copias del mejor individuo de la población

Escalado lineal

Es decir, tenemos la situación mostrada en la siguiente figura:



$$b = (1-a)\frac{\sum g(x_i)}{TamPob} = (1-a)\overline{g} \qquad a = \frac{(P-1)\overline{g}}{g_{max} - \overline{g}}$$

Escalado lineal

- Dependiendo del rango de valores que tome la función objetivo del problema, este tipo de escalado puede producir valores de adaptación negativos, que tenemos que evitar.
- En este caso podemos sustituir la segunda condición por otra que especifique que el valor mínimo de la adaptación sea 0:

$$f_{min} = 0$$

- con lo que se llega al siguiente valor de

$$a = \frac{\overline{g}}{\overline{g} - g_{min}}$$

Ejercicio

- Dados los siguientes valores de aptitud, calcula los valores escalados *f* de la aptitud *g*. Utiliza un valor de P = 2

$$f(x_i) = a * g(x_i) + b$$

a = 0.666

b = 4.666

Individuo xi	*Aptitud original g (xi)*	*Aptitud escalada f (xi)*
I1	1	5.33
I2	2	5.98
I3	18	16.54
I4	35	27.76

- Dibuja la gráfica de los valores originales y los escalados

Otras mejoras y alternativas

- Se puede introducir conocimiento específico del problema a resolver
  - Codificación adaptada al problema
  - Mecanismo de tratamiento de individuos no factibles
  - Introducción de restricciones a las soluciones
  - Construcción de nuevos operadores
  - Creación de algoritmos híbridos con otras técnicas heurísticas

Ejemplo codificación adaptada al problema: TSP

- TSP (Travelling Salesman Problem). Representación: La manera más natural de codificar las soluciones consiste en enumerar las ciudades por orden de recorrido (sentido arbitrario, dada la simetría).

- Para un problema con nueve ciudades el recorrido

$$5 \rightarrow 1 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 4 \rightarrow 6 \rightarrow 2 \rightarrow 3$$

- Se representa con el individuo:

5	1	7	8	9	4	6	2	3

- Los operadores de cruce son más complejos y son específicos de cada representación

Ejemplo codificación adaptada al problema: cuadrado mágico

- En un cuadrado es de n x n, el cuadrado tendrá n^2 casillas y los números que colocaremos serán del 1 a n^2 y la fórmula para encontrar la constante mágica de un cuadrado mágico de orden n es:

$$\frac{n(n^2+1)}{2}$$ = suma de valores de fila, columna o diagonal

- Ejemplo 3 x 3

8	1	6	15
3	5	7	15
4	9	2	15
15	15	15	15

(8 1 6 3 5 7 4 9 2)

- Individuo:

$$\frac{n(n^2+1)}{2}$$ = suma de valores de fila, columna o diagonal

Representación y codificación

- Una representación es perfecta respecto a los objetos representados si cumple:
  1. Completitud: Todos los objetos deben poder ser representados.
  2. Coherencia: Únicamente debe representar objetos del problema.
  3. Uniformidad: Todos los objetos deben estar representados por la misma cantidad de codificaciones.
  4. Sencillez: Debe ser fácil de aplicar el mecanismo de codificación objeto <->individuo.
  5. Localidad: Pequeños cambios en los individuos se han de corresponder con pequeños cambios en los objetos.

Mejoras en el AG

- Requisito de localidad: La codificación binaria no cumple el requisito de localidad: puntos muy próximos tienen codificaciones muy distintas (acantilados de Hamming)
- Una mutación en un individuo puede producir cambios drásticos en los puntos representados.
- El problema de localidad se puede corregir utilizando otras codificaciones: Codificación de Gray.
- Asocia a cada entero de la sucesión 0,1, …, 2^N-1 una cadena de bits de longitud N que se construye:
  - Comenzando por la cadena **0…0**, en cada iteración se conmuta el bit menos significativo que produzca una nueva cadena.

Codificación de Gray

- Por ejemplo, la sucesión 0,1,2……,7 = 2^3-1 tiene la siguiente codificación Gray:

000 001 011 010 110 111 101 100

- Tienen la propiedad de que 2 valores consecutivos entre sí en el espacio de búsqueda difieren solo 1 bit.

```
BeginAlgorithm{Bin2Gray}
BIN[0] = 0;
 for( i = N-1; i >= 0; i--)
  GRA[i+1]=BIN[i+1] xor BIN[i];
EndAlgorithm
```

```
BeginAlgorithm{Gray2Bin}
BIN[0] = 0;
 for( i = 1; i <= N; i++)
   BIN[i]=BIN[i-1] xor GRA[i];
EndAlgorithm
```

- Donde *i=N* indica el bit más significativo de la cadena y *i=1* el bit menos significativo.

Representación de valores reales

Genes con valores reales:

- En muchos problemas de optimización numérica se ha comprobado la utilidad de utilizar un cromosoma representado como un **vector de números reales**
- Esta representación es más cercana al dominio del problema que queremos resolver y permite una mayor precisión numérica.
- El esquema de codificación preferido es la representación de los individuos mediante vectores de números reales en los que cada componente se corresponde con un gen.

3.24	2.71	0.87	2.34	4.55

Operadores genéticos con reales

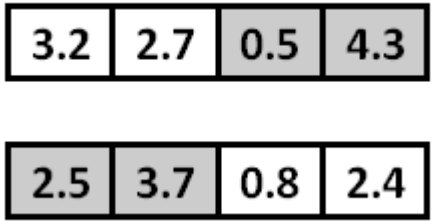
- Operadores de cruce **discretos**:
  - Simple (uno o varios puntos)
  - Uniforme
- Operadores de cruce basados en **agregación**:
  - Aritmético
  - Lineal
  - geométrico.
- Operadores de cruce basados en **entornos**:
  - SBX: Simulated Binary CrossOver
  - BLX
- Operadores de mutación

Cruce discreto (simple o varios puntos)

- Se trata realmente del cruce normal aplicado a vectores de números reales:

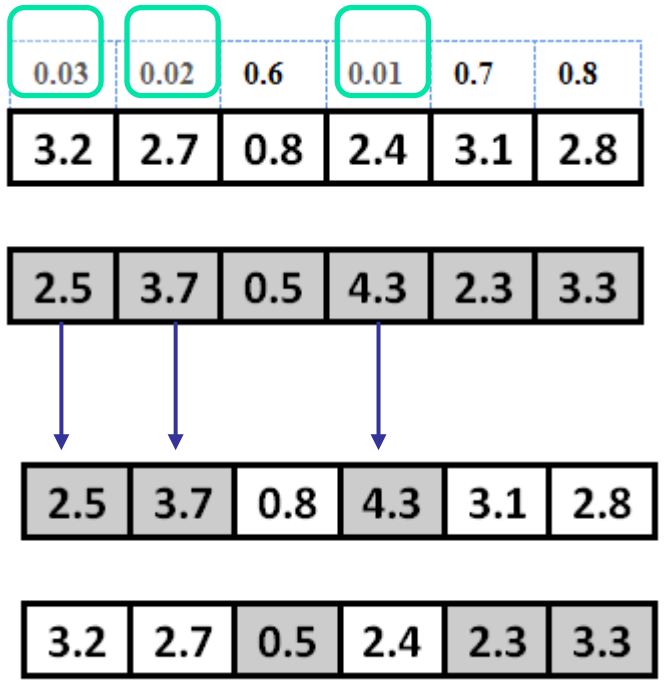
Ejemplo:

Los hijos serían:



Cruce discreto uniforme

- Se define una probabilidad $\mathbf{P_i}$, que permite decidir en cada gen si se intercambian los genes de los padres o se quedan sin cambiar en los hijos. Por ejemplo si $\mathbf{P_i}$ es 0,4 y obtenemos los siguientes valores de probabilidad en cada gen:



Cruce aritmético

- Realiza una combinación lineal entre los cromosomas de los padres. El caso más simple es el cruce de media aritmética, donde el hijo se genera de alguno de los siguientes modos:
  - $hi = (p1_i + p2_i) / 2$
  - $hi = \alpha p1_i + (1-\alpha)p2_i \quad 0<=\alpha<=1$

Ejemplo para $\alpha=0.6$

- P1= (3.2 2.7 0.8) … h1=(0.6 * 3.2+ 0.4 * 2.5,...)
- P2= (2.5 3.7 0.5) … h2=(0.6 *2.5+0.4 * 3.2, …)

Cruce media geométrica

CRUCE MEDIA GEOMÉTRICA

- Para generar el cromosoma hijo se utiliza la siguiente fórmula:

$$hi = (p1_i + p2_i)^{1/2}$$

- Tiene el inconveniente de que se pierde diversidad ya que tiende a llevar los individuos hacia la mitad del intervalo permitido $[a,b]$.

Cruce SBX

CRUCE SBX (Simulated Binary CrossOver) Deb and Agrawal (1995)

Dados los padres

Xp1 = {x1, ….. , xi, …., xn)

Yp2 = {y1, ….. , yi, …., Yn)

y los hijos que se desean obtener

Xh1 = {xh1, ….. , xhi, …., xhn)

Yh2 = {yh1, ….. , yhi, …., yhn)

Cruce SBX

1. Se genera un número aleatorio *r* (0,1).
2. Se calcula β mediante la ecuación

$$\beta = \begin{cases} 2r^{\frac{1}{\eta+1}} & r \leq 0.5 \\ \left[\frac{1}{2(1-r)}\right]^{\frac{1}{\eta+1}} & r > 0.5 \end{cases}$$

3. Una vez determinado β, los hijos se calculan mediante la ecuación

$$Xh1 = 0.5\,[(1+\beta)\,Xp1 + (1-\beta)\,Yp2]$$

$$Xh2 = 0.5\,[(1-\beta)\,Xp1 + (1+\beta)\,Yp2]$$

Cruce SBX

Ejemplo

$r=0.4 \quad \eta = 1 \quad \beta = 1.27$

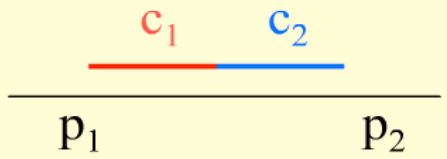
$X_{p1} = \{0, 1.0, 2.0, 3.0, 4.0, 5.0, 6\}$	$X_{h1} = \{0, 0.0, 2.0, 3.0, 4.0, 5.0, 7\}$
$Y_{p2} = \{0, 6.0, 5.0, 4.0, 3.0, 2.0, 1\}$	$X_{h2} = \{0, 7.0, 5.0, 4.0, 3.0, 2.0, 0\}$

- Valores de η pequeños generan hijos muy alejados de sus padres
- Se suele utilizar el valor $\eta = 1$

Cruce SBX

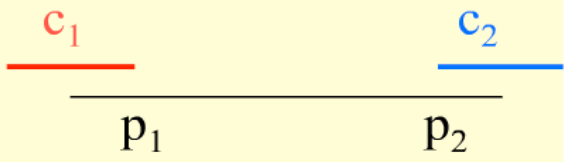
$$\beta = \left| \frac{c_1 - c_2}{p_1 - p_2} \right|$$

- **Contracting Crossover** $\beta < 1$



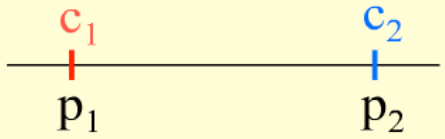
The offspring points are enclosed by the parent points.

- **Expanding Crossover** $\beta > 1$



The offspring points enclose the parent points.

- **Stationary Crossover** $\beta = 1$



The offspring points are the same as parent points

Cruce BLX- α

Dados 2 cromosomas

$$C_1 = (c_{11},\ldots, c_{1n}) \text{ y } C_2 = (c_{21},\ldots, c_{2n}) ,$$

BLX genera dos descendientes

$$H_k = (h_{k1},\ldots, h_{ki},\ldots, h_{kn}) , k = 1,2$$

donde h_{ki} se genera aleatoriamente en el intervalo:

$$[C_{min} - I\cdot\alpha, C_{max} + I\cdot \alpha]$$

$$C_{max} = \max \{c_{1i} , c_{2i}\}$$

$$C_{min} = \min \{c_{1i} , c_{2i}\}$$

$$I = C_{max} - C_{min} , \alpha \in [0,1]$$

Mutación

MUTACIÓN UNIFORME

- Dado un individuo, se modifica alguno de los valores cambiándolo por un valor aleatorio entre los posibles del intervalo

$$\bar{x} = \langle x_1, \ ..., x_l \rangle \rightarrow \bar{x}' = \langle x_1', ..., x_l' \rangle$$
$$x_i, x_i' \in [LB_i, UB_i]$$

MUTACIÓN NO UNIFORME (Gaussiana)

- Normalmente se aplican pequeños cambios, añadiendo a cada variable un valor tomado como una pequeña desviación aleatoria de una distribución normal N (0, σ), donde el valor de σ (desviación estándar) es el que controla el grado de cambio aplicado.

Elección de la población inicial

- La población inicial debe tener diversidad.
- La distribución de aptitudes debe ser uniforme para evitar la convergencia prematura.
- Si no se dispone de información se **elige la población inicial al azar.**
- El conocimiento específico puede ayudar para elegir una población inicial factible y/o próxima al óptimo.
- Conviene que la población inicial contenga la **mayor cantidad posible de puntos factibles**.

Elección de la población inicial

- Para una población inicial dada P[0] ¿Es posible modificar el AG de modo que converja siempre, independientemente de la población inicial?
- Prueba de contractividad:
  - Una iteración del AG es una transformación de una población en otra (bucle básico), adaptada para cumplir las condiciones del teorema del punto fijo de Banach (garantía de convergencia).
- Se puede demostrar que el bucle básico del AG es contractivo.
- Se necesita una imposición adicional al bucle: una iteración es válida si y sólo si hay una mejora absoluta en la aptitud media de la población

$$AptMed(P[t]) < AptMed(P[t+1])$$

- En caso de que no se produzca la mejora, no se cuenta la iteración.

Elección de la población inicial

```
P[0] = Poblacion_inicial();
AptP[0] = Evaluacion(P[0]);
t = 0;
mientras (t<Num_max_gen) y no CondTermina(P[t],AptP[t])
 {
   P[t] = Seleccion(P[t]);
   P[t] = Reproducción(P[t]);
   P[t] = Mutacion(P[t]);
   AptP[t] = Evaluacion(P[t]);
   si( AptP[t] >AptP[t-1]) //Contractividad
              t++;
 }
```

Criterios de terminación

- Criterios de terminación enfocados al coste: (tipo MAX)
  - Limitan el número máximo de iteraciones o el máximo número de evaluaciones.
  - Pueden fallar por exceso o defecto.
- Criterios de terminación enfocados a la calidad: (criterios incrementales de terminación o tipo TOL)
  - El algoritmo se detiene al cumplirse ciertos requisitos de convergencia:
    - Un gen del genotipo ha convergido cuando un porcentaje de la población (90 %) tiene los alelos iguales en dicho gen (o parecidos, si no es binaria).
    - La búsqueda termina cuando el número de genes que han convergido excede cierto porcentaje (alrededor del 95 %) del total de genes de que consta el genotipo.

Alternativas al cruce clásico

Cruce segmentado

- Versión del cruce multipunto que introduce variabilidad en el número de puntos de cruce.
- Consiste en reemplazar el número fijo de puntos de cruce por una probabilidad de segmentación $Pseg$ que da cuenta de la posibilidad de que se produzca un cruce al llegar a cierto punto de la cadena.
- Si $Pseg=0,20$ el promedio esperado de puntos de cruce será $0,2Len$ en cada par de progenitores.

Alternativas al cruce clásico

- Cruce uniforme:
  - Para cada bit del primer descendiente se decide, con probabilidad ***prob*** de qué progenitor heredará su valor en esa posición.
  - El segundo descendiente recibe el correspondiente gen del otro progenitor.
  - Normalmente con ***prob***=0.5 se decide si en los descencientes se intercambian los genes o se quedan igual
- Otros cruces:
  - Cruces anulares
  - Cruces externos
  - Cruces con barajado
  - Cruces solitarios.

Alternativas al cruce clásico

- Cruces anulares: Se basan en una representación en anillo de la cadena de genes.
- Cruces externos: Cruces a nivel de genes
- Cruces con barajado: Antes del cruce se desordenan aleatoriamente las posiciones de ambos progenitores a la par. Se realiza el cruce y se deshace el barajado.
- Cruces solitarios: Se prohíbe la aplicación de mutaciones a individuos procedentes de cruces.
- Cruce adaptativo:
  - La distribución de los puntos de cruce se va adaptando a lo largo de la evolución del algoritmo.
  - Se registran los puntos de cruce con los que se ha realizado el cruce en cada individuo y se favorecen los puntos de cruce **correspondientes a los mejores individuos**

Alternativas a la mutación clásica

- Mutaciones sobre genes:
  - Análogas al cruce externo: afectan a un gen completo, no a un bit.
  - Dado que un gen puede tomar más de dos valores (alelos), la mutación se realiza sustituyendo el valor actual por otro cercano.
  - Un mecanismo sencillo consiste en añadir o sustraer, con igual probabilidad, una cantidad fija al entero con que se codifica el correspondiente alelo.
  - Se presupone que alelos cercanos se codifican mediante enteros contiguos.

Alternativas a la mutación clásica

- Mutaciones no estacionarias:
  - Se reduce la tasa de mutación P_{mut} según evoluciona el AG.
  - Un mecanismo sencillo consiste en multiplicar la tasa de mutación por un factor de reducción constante cada cierto número de generaciones.

- Mutaciones no uniformes:
  - Consisten en dar distintas probabilidades de mutación a cada gen (o a cada bit dentro de un gen) dependiendo de su significado.

Mutación variable

- Tasa de mutación variable que se vaya reduciendo a medida que avanza la evolución [FOG89].
- De esta forma la búsqueda es más amplia en las generaciones iniciales y se va reduciendo a medida que avanza la evolución.
- Bäck y Schütz [BAC96B] han utilizado un porcentaje de mutación *p(t)*, que se va actualizando de generación en generación de acuerdo con la fórmula:

$$p(t) = \left(2 + \frac{n-2}{T-1}t\right)^{-1}$$

siendo *T* el número máximo de generaciones, *t* la generación en curso y *n* la longitud del cromosoma.

Parametros de los AGs

- El **tamaño de la población** suele ser mayor de 50 (valores menores suelen plantear problemas de convergencia prematura).
- El **tamaño de los individuos** suele venir dado como una consecuencia de los criterios de representación y codificación del problema. Se sugiere no alargar innecesariamente dicha cantidad.
- Las tasas de cruce suelen variar entre el 20% y el 60% y las de mutación entre el 0.1% y el 5%
- El número máximo de iteraciones varía mucho de un problema a otro: aproximadamente 50 para problemas sencillos y 1000 para problemas complejos.

Parametrización de los AGs

Ajuste automático de los parámetros de un AG

- El método de los parámetros evolutivos consiste en someter a los parámetros de aplicación de los operadores genéticos a un proceso de evolución análogo al de los AGs.
- Se trata de utilizar con más frecuencia los operadores que más contribuyan a la evolución.
- A cada operador se le asigna una aptitud operacional en función del incremento de aptitud que proporcione a la población su aplicación en un instante dado y se actualiza periódicamente.

Hibridación

- En algunos dominios es necesario incluir conocimiento del problema dentro del algoritmo de optimización
- Esta inclusión de conocimiento del problema se denomina **hibridación**
- Consiste en utilizar el AG para la búsqueda global y usar métodos específicos para la búsqueda local.
  - Un enfoque secuencial deja que el AG converja suficientemente y después se aplica el método local sobre los mejores puntos obtenidos.
  - Un enfoque paralelo usa varios procesadores para la búsqueda local y evalúar los individuos obtenidos y un procesador central conectado con todos ellos hace evolucionar la población proporcionada.

Selección por ranking

- Se basa en el ranking según la ordenación de los individuos por fitness decreciente. El valor asignado a cada individuo depende sólo de su posición en el ranking y no en su valor objetivo.

Posición i	*Aptitud i*
1ª	**90**
2º	**11**
3º	**7**
4º	**3**

- La probabilidad de seleccionar el individuo i-ésimo en el ranking se define como:

$$p(i) = \frac{1}{n}\left[\beta - 2(\beta - 1)\frac{i-1}{n-1}\right], 1 \le \beta \le 2$$

Donde β se puede interpretar como la tasa de muestreo del mejor individuo o **Presión selectiva**

Selección por ranking

```
public class SelectionRanking implements SelectionMethod {
        static private final double _beta = 1.5;
        public SelectionRanking() {      }

        public List<Chromosome> apply(List<Chromosome> pop)
           {
                rankingPunctuation(pop);
                SelectionRoulette roulette = new SelectionRoulette();
                return roulette.apply(pop);
        }
```

Selección por ranking

```
private void rankingPunctuation(List<Chromosome> pop) {
        double accPunc = 0.0;
        for (int i = 0; i < pop.size(); ++i) {
                double probOfIth = (double)i/pop.size();
                probOfIth *= 2*(_beta-1);
                probOfIth = _beta - probOfIth;
                probOfIth = (double)probOfIth * ((double)1/pop.size());
                pop.get(i).setAccPunc(accPunc);
                pop.get(i).setPunc(probOfIth);
                accPunc += probOfIth;
           }
        }

}
```

$$p(i) = \frac{1}{n}\left[\beta - 2(\beta - 1)\frac{i-1}{n-1}\right], 1 \le \beta \le 2$$

Ejemplo

- Si suponemos un BETA o presión selectiva de 2
- N = número de individuos
- i = posición del individuo en el ranking
- Se ordenan de mayor a menor valor de aptitud

$$p(i) = \frac{1}{n}\left[\beta - 2(\beta - 1)\frac{i-1}{n-1}\right], 1 \le \beta \le 2$$

Posición i	***Aptitud i***	***Pi original***	***Pi ranking***
1ª	**90**	**0.81**	¼(2-2 *(0/3)) = **0.5**
2º	**11**	**0.099**	¼(2-2 *(1/3)) = **0.383**
3º	**7**	**0.063**	¼(2-2 *(2/3)) = **0.166**
4º	**3**	**0.027**	¼(2-2 *(3/3)) = **0**
	111		

Otras mejoras de la convergencia

- **Cruce Uniforme HUX.** Intercambia exactamente la mitad de los alelos que son distintos en los padres. Garantiza que los hijos tengan una distancia Hamming máxima a sus dos padres.

- **Prevención de Incesto**.
  - Se forman N/2 parejas con los elementos de la población.
  - Sólo se cruzan las parejas cuyos miembros difieren en un número determinado de bits (umbral de cruce). El umbral se inicializa a L/4 (L es la longitud del cromosoma).
  - Si durante un ciclo no se crean descendientes mejores que los de la población anterior, al umbral de cruce se le resta 1.

AGs y restricciones

- Una de las limitaciones del AG básico es que carece de mecanismos para considerar posibles restricciones
- ¿Qué hacemos con las soluciones no factibles?
  - Técnicas de penalización
    - Se resuelve el problema sin restricciones, **penalizando a los individuos no factibles**.
  - Técnicas de reparación
  - Técnicas de decodificación

AGs y restricciones

Penalización:

- Se resuelve el problema sin restricciones, penalizando a los individuos no factibles.
- Sólo se necesita una medida del grado de no-factibilidad, para penalizar más a los puntos más alejados del dominio del problema.
- En problemas de optimización paramétrica se identifica un conjunto de p restricciones. h_i es el grado de violación de la i-ésima restricción

$$Eval(\mathbf{v}) \leftarrow Eval(\mathbf{v}) - K \cdot Penalty(h_1(\mathbf{v}), \ldots, h_p(\mathbf{v}))$$

AGs y restricciones

$$Eval(\mathbf{v}) \leftarrow Eval(\mathbf{v}) - K \cdot Penalty(h_1(\mathbf{v}), \ldots, h_p(\mathbf{v}))$$

- p es el número total de restricciones
- h_i es el grado de violación de la i-ésima restricción
- K es un coeficiente global de penalización

- Grado de violación de la i-ésima restricción:

$$h_i(\mathbf{x}) = \begin{cases} -g_i(\mathbf{x}) & \text{sii} \quad g_i(\mathbf{x}) < 0 \\ 0 & \text{sii} \quad g_i(\mathbf{x}) \geq 0 \end{cases} \qquad (\forall i = 1, \ldots, p)$$

- La función *Penalty(…)* es una función positiva y creciente que se especifica para cada problema.

AGs y restricciones

- Las funciones de penalización muy severas no son recomendables en problemas fuertemente restringidos:
  - El AG gasta mucho tiempo en procesar individuos no factibles.
  - Cuando se encuentra un individuo factible éste puede destacar demasiado sobre los demás (convergencia prematura).
- Las funciones de penalización muy flexibles tienen el riesgo de dejar prosperar a individuos que aun siendo no factibles tengan una aptitud neta mayor que otros factibles.

AGs y restricciones

- Técnicas de gratificación:
  - Premian a los individuos factibles en vez de penalizar a los no factibles
- Técnicas de reparación o corrección:
  - Procedimiento con el que corregir cualquier solución no factible que se genere.
  - Suele ser difícil encontrar y suele consumir muchos recursos de computación. Específico para cada problema.
- Técnicas de decodificación:
  - Consisten en realizar la codificación de modo tal que sea imposible generar soluciones no factibles.

AGs y restricciones : ejemplo

El problema binario de la mochila:

- Se trata de elegir un grupo de objetos de entre un conjunto de m con volúmenes $w_1,\ldots,w_m$ y valores $p_1,\ldots,p_m$ de manera que quepan en una mochila de capacidad C maximizando el valor total de su contenido.
- Este problema admite múltiples formulaciones según se dé uno u otro significado a las variables w_i y p_i

Problema de la mochila

Dado un vector de valores: $\mathbf{p} := (p_1, \ldots, p_m)$

Dado un vector de volúmenes: $\mathbf{w} := (w_1, \ldots, w_m)$

Encontrar un vector binario de selección :

$$\mathbf{x} := (x_1, \ldots, x_m) \longrightarrow (010011100001110001111111)$$

x_j = 1 si se ha seleccionado el j-ésimo objeto

$$\begin{cases} \text{maximizar} & f(\mathbf{x}) & = & \mathbf{p} \cdot \mathbf{x} \\ \text{sujeto a} & \mathbf{w} \cdot \mathbf{x} & \leq & C \\ \text{siendo} & x_i & \in & \{0,1\} \quad (\forall i = 1, \ldots, m) \end{cases}$$

Problema de la mochila

- Los individuos **v** del espacio de búsqueda coinciden con los puntos **x** del dominio del problema.
- La j-ésima posición del genotipo señala la presencia o ausencia de j-ésimo objeto en la mochila (representación natural).

Función de aptitud:

- Para considerar la restricción de capacidad limitada **w.x** <= **C** se pueden usar las tres técnicas vistas anteriormente.

$$u(\mathbf{x}) := f(\mathbf{x}) = \mathbf{p} \cdot \mathbf{x} = p_1 x_1 + \cdots + p_m x_m$$

Problema de la mochila

- Funciones de penalización: Se modifica la función de aptitud

$$u(x) \leftarrow u(\mathbf{x}) - Penalty(h(\mathbf{x}))$$

donde $h(\mathbf{x})$ es el grado de violación de la restricción, definido como

$$h(\mathbf{x}) = \begin{cases} \mathbf{w} \cdot \mathbf{x} - C & \text{sii} \quad \mathbf{w} \cdot \mathbf{x} > C \\ 0 & \text{sii} \quad \mathbf{w} \cdot \mathbf{x} \leq C \end{cases}$$

Problema de la mochila

- Algoritmos de reparación: se extraen objetos de la mochila cuando esté demasiado llena.
  - Reparación aleatoria: Se construye $\mathbf{x_f}$ extrayendo elementos de la mochila al azar hasta que se vuelva a verificar la restricción de volumen $\mathbf{w}.\ \mathbf{x_f} <= C$
  - Reparación codiciosa: Se extraen los objetos en orden creciente de valor específico, comenzando por el menos valioso y terminando cuando se vuelva a verificar la restricción de volumen.
- El algoritmo codicioso proporciona mejores resultados.

Ejemplo mochila

```
class CromosomaMochila extends Cromosoma {
      static private int MAX_P = 700;
      static private int ITEM_VALOR = 0;
      static private int ITEM_PESO = 1;
      static  int[][] mochila_items = {

   {49,21},{23,1},{21,4},{37,2},{28,28},{27,47},{21,2},{6,23},{38,12},
   {7,12},{11,45},{23,29},{27,36},{15,43},{17,1},{17,38},{5,4},{41,33}
   ,{32,18},{32,3},{21,38},{28,11},{0,35},{25,13},{30,29},{5,3},{11,32
   },{41,26},{28,21},{42,47},{22,9},35,29},{13,26},{14,47},{46,46},{2,
   19} , {34,30} ,{32,34}, {15,8},{48,48}, {39,5}, {3,0},
   {40,45},{28,20}, {16,22},{17,40},{31,4},{19,49}
};
```

Ejemplo mochila

```
double penalty(int valor){
  if (valor > MAX_P) return valor-MAX_P;
     else return 0;
}
  double eval() {
        int totalValor = 0, totalPeso = 0;
        for (int i = 0; i < chromosomeLength; i++) {
            if (bits[i]){
                totalValor += mochila_items[i][ITEM_VALOR];
                totalPeso += mochila_items[i][ITEM_PESO];
            }
       return totalValor - p_max * penalty(totalPeso);
  }
```

Universidad Complutense Madrid

Acerca de *Creative Commons*

- Licencia CC (Creative Commons)

  Este tipo de licencias ofrecen algunos derechos a terceras personas bajo ciertas condiciones.

  Este documento tiene establecidas las siguientes:

  - Reconocimiento (*Attribution*):
    En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.
  - No comercial (*Non commercial*):
    La explotación de la obra queda limitada a usos no comerciales.
  - Compartir igual (*Share alike*):
    La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

  Pulsa en la imagen de arriba a la derecha para saber más.