Programación Evolutiva

Tema 4: Mejoras al esquema básico del AG

Carlos Cervigón, Lourdes Araujo 2009-2010.

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.
- La finitud: el AGS sólo puede trabajar con poblaciones finitas no muy grandes.

El problema de la ineficiencia

- Los AGs sólo se pueden considerar eficientes en comparación con otros métodos estocásticos, pero si se encuentra un método determinista para resolver un problema lo hará más eficientemente.
- Tendencia al extravío de la búsqueda: el AG realiza la búsqueda de los mejores puntos utilizando únicamente la aptitud de los individuos para recombinar internamente los bloques constructivos.
- A veces esta información proporcionada es insuficiente para orientar la búsqueda del óptimo: desorientación (deception).
- La presencia de desorientación suele estar fuertemente estimulada por una mala codificación

El problema de la ineficiencia

- En el peor de los casos puede ocurrir que el AG no converja al óptimo global, desviándolo hacia un óptimo local (extravío).
- Aunque un AG se desoriente no necesariamente se va a extraviar: para eso es necesario partir de una mala población inicial o plantear un problema especialmente diseñado para que se extravíe.
- El factor que más contribuye a la existencia de desorientación es la forma de la función de aptitud.
- La desorientación es frecuente en problemas en los que los puntos óptimos están aislados y rodeados de puntos de muy baja aptitud.

Diversidad en los AGs

- Requisito de diversidad en los AGs:
  - Diversidad de individuos
  - Diversidad de aptitudes.
- Con poca variedad de individuos
  - Poca variedad de esquemas
  - 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)

Diversidad en los AGs

- 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.

Convergencia prematura

- En algún momento de la evolución de un AG puede ocurrir que un individuo o un grupo de ellos obtengan una aptitud notablemente superior a los demás (fases tempranas de la evolución).
- Riesgo de que se produzca una evolución en avalancha: al incrementar los individuos más aptos su presencia en la población, la diversidad disminuye, y 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.

Control de la diversidad

- Para controlar la diversidad de las aptitudes se debe actuar sobre el mecanismo de asignación de probabilidades de supervivencia (o de descendientes, en su caso):
  - Cuanto más se favorezca a los más aptos menor variedad se obtendrá, y aumenta el riesgo de perder la información de valor desarrollada por los individuos moderadamente aptos.
- Si no se favorece especialmente a los individuos más aptos, se obtendrá más diversidad a costa de hacer las etapas de selección menos eficaces.
- A la mayor o menor tendencia a favorecer a los individuos más aptos se le llama **presión selectiva:** grado en el que se favorece a los individuos más adaptados.

Control de la diversidad

- Lo ideal es que en las primeras fases de la evolución hubiera poca presión selectiva con el fin de que la búsqueda fuera más global y en las últimas fases, se incremente la presión selectiva para encontrarla cuanto antes.

$$PresSel[t] := TamPob \cdot PuntMax[t] \qquad PresSel[t] \equiv \frac{AptMax[t]}{AptMed[t]}$$

- *PresSel* cuantifica el número esperado de descendientes que se da al miembro más apto de la población.
- Whitley recomienda trabajar con valores de presión próximos a 1.5 (valores mayores ocasionarían superindividuos, y valores menores frenarían la búsqueda sin ningún beneficio).
- El mínimo valor es 1, que corresponde a la situación en que todos los individuos son iguales (diversidad nula).

Métodos de control de la diversidad

- Control de la diversidad de aptitudes
  - Actuando directamente sobre el mecanismo de selección: mecanismos de muestreo insensibles a la distribución de aptitudes y a los errores de muestreo (métodos distintos de la ruleta)
  - Actuando sobre el mecanismo de asignación de aptitudes: la probabilidad de supervivencia deja de ser necesariamente proporcional a la aptitud bruta.
  - Controlando las edades de los individuos.
  - Introduciendo mecanismos de escisión.

Funciones de evaluación y adaptación

- Para que el operador de selección funcione correctamente, los valores de la función de adaptación deben tener una separación adecuada:
  - Si son muy próximos, la selección no podrá distinguir a los más adaptados para favorecerlos
  - Si se producen valores extremadamente mejores que el resto, la selección puede hacer que la población se sature con copias de estos superindividuos, convergiendo prematuramente.
  - Además, los valores de la adaptación necesitan ser **positivos** para poder aplicar muchos de los mecanismos de selección que hemos visto
- Hay que distinguir entre la función de adaptación y la de evaluación

Funciones de evaluación y adaptación

- Una vez elegida la función de evaluación, se construye la función de adaptación mediante una serie de transformaciones que facilitan su uso en el algoritmo.
- Las operaciones más habituales son desplazamiento y escalado.
- El desplazamiento tiene como finalidad hacer que la función de aptitud devuelva valores positivos y nos permite convertir un problema de minimización en otro de maximización.
- El escalado permite establecer una separación entre los valores que toma para distintos individuos que sea apropiada para el funcionamiento de la selección

Desplazamiento de la adaptación

- La solución está en realizar una modificación de los valores de la función de adaptación, de forma que se obtengan valores positivos y que cuanto menor sea el valor de la función (más cercano al óptimo) mayor sea el correspondiente valor revisado.
- Sea *cmax* el valor máximo de los valores de adaptación *g(x)* de la población en una determinada generación. Entonces la adaptación revisada *f(x)* correspondiente a la adaptación *g(x)* la definimos como:

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

- En lugar de utilizar *cmax* conviene utilizar valores ligeramente mayores (por ejemplo, un 105%) para prevenir que si toda la población converge a un mismo valor de adaptación, la adaptación revisada se haga nula.

Ejemplo

g(x)	f(x) (c_{max}-g(x))	puntuación	
-3	10	10/23	0.434
-1	8	8/23	0.347
2	5	5/23	0.217
7	0	0/23	0
	23		

Revisar suma de aptitudes : (tamaño x Fmax)-sumaAptitudes

Desplazamiento de la adaptación

```
funcion revisar_adaptacion_minimiza(var TPoblacion pob,
                TParametros param, var real cmax)
{
   cmax = -∞;
   // un valor por debajo de cualquiera que pueda
   // tomar la función objetivo
   para cada individuo desde 1 hasta param.tam_pob hacer{
     si (pob[i].adaptacion > cmax) entonces
        cmax = pob[i].x;
   }
   cmax = cmax * 1.05; //margen para evitar sumadaptacion = 0
                    // si converge la población
   para cada individuo desde 1 hasta param.tam_pob hacer{
      pob[i].adaptacion = cmax - pob[i].x;
   }
}
```

Desplazamiento de la adaptación

- Si el problema es de maximización donde pueden aparecer posibles valores negativos podemos hacer:

$$f(x) = g(x) + F_{min}$$

- F_{min} es el el peor valor absoluto de *g(x) en t*

-3 -1 2 7 ➔ 0 2 5 10

Escalado de la adaptación

- Una forma de controlar la diversidad de las aptitudes es definiendo una función de aptitud neta, distinta de la función de evaluación: escalado de la función de evaluación.
- Permite establecer una separación entre los valores que toma para 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

- Sin escalado, en las fases tempranas 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.
- Más tarde, cerca de la convergencia, la competición entre los miembros de la población es menos intensa y la simulación tiende a evolucionar erráticamente:
  - 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 a partir de los valores proporcionados por la función de evaluación del siguiente modo

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

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

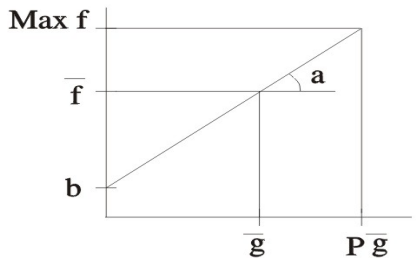
1. Deben coincidir los valores medios de la función de evaluación y de la función de aptitud en la población. $\bar{f} = \bar{g}$
2. La adaptación del mejor debe ser el producto por un cierto parámetro *P* por la adaptación media

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

Número esperado de copias del mejor individuo de la población

Escalado lineal

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



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

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

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

Mejora de la convergencia: Elitismo

- El elitismo consiste en asegurar la supervivencia de los mejores individuos de la población
- Se muestrea una élite de ***r*** miembros de entre los mejores de la población y se incorporan directamente a la población final, sin pasar por la intermedia.
- Durante el reemplazo los individuos de la élite se preservan, descartando si es necesario a alguno de sus hijos (siempre que tenga aptitud menor).
- En general, el porcentaje de la población que puede formar parte de la élite debe ser pequeño, no mayor de un 1 o un 2% del tamaño 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..

Elitismo

```
int tamElite = tamaño / 25;
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;
}
```

Elitismo

- El elitismo mejora la velocidad de convergencia de los AGs cuando la función de evaluación es unimodal (no existen subóptimos), pero puede empeorar con funciones fuertemente multimodales.
- 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.

Otras mejoras y alternativas

- Se mantiene la estructura fundamental, introduciendo simplemente variantes y parametrizaciones en las configuraciones por defecto de los métodos y procedimientos de un AG.
- 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 : Problema del viajante

- 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

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 habitual no se 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

Con cadenas binarias:

- Podemos discretizar el rango de cada variable fijando una precisión deseable (por ejemplo 2 decimales) y posteriormente se tratan esos valores como si fueran enteros

Variable: $-1.5 \le x \le 2.0$

Precisión: 3 decimales

Tamaño: $\lfloor \log_2 \lfloor (l_{sup} - l_{inf}) * 10^{precisión} \rfloor + 0.9 \rfloor$ = 12 bits

- Problemas : Alta dimensionalidad (largos cromosomas → lenta evolución) y poca precisión para algunos problemas

Representación de valores reales

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 en coma flotante en los que cada componente se corresponde con un gen.

3.24	2.71	0.87	2.34	4.55

Operadores sobre reales

- Operadores de cruce **discretos**:
  - simple, uniforme, dos puntos.
- Operadores de cruce basados en **agregación**:
  - aritmético, lineal, geométrico.
- Operadores de cruce basados en **entornos**:
  - SBX, BLX.
- Operadores de mutación

Cruce discreto (simple o varios puntos)

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

Ejemplo:

3.2	2.7	0.8	2.4
2.5	3.7	0.5	4.3

Los hijos serían:

3.2	2.7	0.5	4.3
2.5	3.7	0.8	2.4

Cruce discreto uniforme

- Se define una probabilidad **Pi**, que permite sortear cada gen para ver si se intercambian los genes de los padres o se quedan sin cambiar en los hijos. Por ejemplo si **Pi** es 0,4 y obtenemos los siguientes valores de probabilidad en cada gen:

0.03	0.02	0.6	0.01	0.7	0.8
3.2	2.7	0.8	2.4	3.1	2.8

2.5	3.7	0.5	4.3	2.3	3.3

2.5	3.7	0.8	4.3	3.1	2.8

3.2	2.7	0.5	2.4	2.3	3.3

Otros cruces

CRUCE ARITMÉTICO

- En este cruce se 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 del siguiente modo: hi = (p1i +p2i) / 2

CRUCE MEDIA GEOMÉTRICA

- Para generar el cromosoma hijo se utiliza la siguiente fórmula: $hi = (p1i + p2i)^{1/2}$
- Tiene el inconveniente de que se pierde diversidad ya que tiende a llevar los individuos hacia la mitad del intervalo permitido $[a,b]$.

Mutación

MUTACIÓN UNIFORME

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

MUTACIÓN NO UNIFORME

- Normalmente se aplican pequeños cambios, añadiendo a cada variable 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.

El operador de inversión

- Mejora evolutiva de la codificación.
- Existe una forma de estimular la formación de buenos bloques constructivos sometiendo a la propia distribución de posiciones al proceso de evolución: operador de inversión o también mutación estructural.
- La inversión es un operador unitario que selecciona dos puntos en una cadena e invierte el orden de las posiciones intermedias recordando el significado de cada una.
- Los bits de la cadena se identifican asignando a cada alelo una etiqueta con su posición.

El operador de inversión

- Por ejemplo, al invertir la cadena

$$\langle 0_1 0_2 0_3 | 1_4 1_5 0_6 1_7 | 0_8 0_9 0_{10} 1_{11} \rangle$$

entre los puntos señalados se obtiene

$$\langle 0_1 0_2 0_3 | 1_7 0_6 1_5 1_4 | 0_8 0_9 0_{10} 1_{11} \rangle$$

- El contenido de la cadena no se altera, tan sólo se reorganiza su estructura (tras la inversión la cadena sigue representando al mismo individuo, con otra estructura).
- Se introduce este operador con la idea de que el AG busque no sólo el mejor fenotipo (contenido) sino el mejor genotipo (estructura).

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?
- Sí es posible conseguir convergencia independientemente de la población inicial si añade un mecanismo llamado 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).

Elección de la población inicial

- 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.
- A pesar de este resultado, la elección de la población inicial es fundamental para la velocidad de convergencia.

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.
  - Son los más sencillos, pero no siempre tienen significado (pueden fallar por exceso o defecto).
- Criterios de terminación enfocados a la calidad: (criterios incrementales de terminación o tipo TOL)
  - El algoritmo para al cumplirse ciertos requisitos de convergencia.
  - Enfocados a la estructura
  - Enfocados al contenido

Criterios de terminación

- Enfocados a la estructura:
  - Se evalúa la convergencia de la población verificando la cantidad de genes que han convergido.
  - Se considera que un gen del genotipo ha convergido cuando cierto porcentaje de la población (alrededor del 90 %) tiene los alelos iguales en dicho gen (o parecidos, si la representación 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.

Criterios de terminación

- Enfocados al contenido:
  - Miden el progreso hecho por el algoritmo en cierto número de iteraciones: si es menor que cierta tolerancia *TOL* (parámetro), la búsqueda termina.
  - Se suele basar en la aptitud total, la máxima o la mínima.
- Se suele usar un criterio de terminación híbrido que consta de una tolerancia y un mecanismo de seguridad del tipo MAX.

Mejoras a los operadores genéticos

Limitaciones de los operadores genéticos clásicos:

- El cruce clásico no puede realizar ciertas combinaciones de bloques constructivos:
  - Sean los esquemas 001*****01 y ****11**** con alta aptitud
  - Mediante el cruce clásico es imposible obtener 001*11***01.
- Asimetría entre la aplicación del cruce y la mutación (la unidad básica de mutación es el gen, la del cruce es el individuo):
  - la probabilidad de que un individuo se cruce no depende de su longitud, pero si la probabilidad de que contenga genes mutados.

Alternativas al cruce clásico

Se han propuesto diversas alternativas que a veces mejoran las prestaciones del cruce clásico.

- Cruce multipunto:
  - Los individuos se cruzan en torno a dos o más puntos. Este cruce mejora la capacidad de procesado de esquemas, a costa de perder velocidad de convergencia.
- 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 p_{unif} de qué progenitor heredará su valor en esa posición.
  - El segundo descendiente recibe el correspondiente gen del otro progenitor.
  - Intercambia genes en lugar de bloques constructivos: se usa para combinar atributos específicos, con independencia de la posición en que han sido codificados.

Mecanismos adicionales al cruce:

- Cruces anulares, Cruces externos, Cruces con barajado, Cruces con adaptación, Cruces solitarios.

Mecanismos adicionales al cruce

- Cruces anulares: Se basan en una representación en anillo de la cadena de genes.
- Cruces externos: El cruce en los AGs se realiza en el nivel de bits, no en el de genes : se trabaja con la mayor cantidad de esquemas.
  - Sin embargo, en algunos problemas (la mochila) es más natural obligar a que los cruces se produzcan exclusivamente en los puntos de separación entre genes: cruces externos.
  - Equivalente a incrementar el alfabeto.
- 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.

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.
  - El procedimiento concreto de alteración depende de la codificación utilizada.
  - 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.

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
  - 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 suele definir $\mathcal{X}$ implícitamente a través de un conjunto de $\boldsymbol{p}$ restricciones expresadas como inecuaciones

$$g_i(\mathbf{x}) \geq 0$$

AGs y restricciones

- Grado de violación de la i-ésima restricción:

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

- Los h_i se incluyen en la función de evaluación:

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

  - 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

- La función $Penalty(\ldots)$ es una función positiva y creciente que se especifica para cada problema.

AGs y restricciones

- Michalewicz propone la siguiente función de penalización para resolver problemas optimización paramétrica:

$$\text{Penalty}(\mathbf{x}[t]) = k \left(\frac{t}{Num\_Max\_Iter}\right)^r \text{EvalMed}(P[t]) \sum_{i=1}^{NrRes} h_i(\mathbf{x}[t]) \text{ donde}$$

$NrRes$ es el número de restricciones

Num_Max_Iter es el número máximo de iteraciones

$k \simeq 1/NrRes$

$r \simeq 1$

h_i es violación de la rest. i-ésima

- Al avanzar la búsqueda la penalización se va haciendo cada vez más rígida.

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:
  - Consisten en habilitar un procedimiento con el que corregir cualquier solución no factible que se genere.
  - Inconveniente: suele ser difícil encontrar un procedimiento de corrección lo suficientemente general y, si se encuentra, normalmente consume muchos recursos de computación.
  - Este procedimiento es necesariamente específico para cada problema.

AGs y restricciones

- Técnicas de decodificación: Consisten en realizar la codificación de modo tal que sea imposible generar soluciones no factibles.
- Se modifica X a través de la codificación para que se aproxime lo máximo posible a χ.
- Suelen requerir modificar los operadores genéticos para que conserven la factibilidad de los individuos.
- Inconvenientes similares a las técnicas de reparación.
- De modo indirecto todas estas técnicas están añadiendo conocimiento específico.

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 tal 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 :

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

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

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

Se define el vector de valores específicos $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_m)$ a través de $\rho_i := p_i / w_i \quad (\forall i = 1, \ldots, m)$.

Problema de la mochila

AG para el problema:

- 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).
- Si se ignora la restricción de capacidad existe una correspondencia biunívoca entre los individuos y los puntos del dominio (representación perfecta).
- Función de aptitud: $u(\mathbf{x}) := f(\mathbf{x}) = \mathbf{p} \cdot \mathbf{x} = p_1 x_1 + \cdots + p_m x_m$
- Para considerar la restricción de capacidad limitada **w.x <= C** se pueden usar las tres técnicas vistas anteriormente.

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

- Se han utilizado las siguientes funciones de penalización:

Proporciona buenos resultados →

$$\begin{aligned} Penalty_{log}(\mathbf{x}) &:= \log_2(1 + \rho_{max}\, h(\mathbf{x})) \\ Penalty_{lin}(\mathbf{x}) &:= \rho_{max}\, h(\mathbf{x}) \\ Penalty_{sq}(\mathbf{x}) &:= (\rho_{max}\, h(\mathbf{x}))^2 \end{aligned}$$

siendo $\rho_{max} := \text{máx}_{i=1,\ldots,m}\{p_i / w_i\}$.

Se define el vector de valores específicos $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_m)$ a través de $\rho_i := p_i / w_i \quad (\forall i = 1, \ldots, m)$.

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 $\mathbf{w.\ x_f} <= C$
  - 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.
- También existen los algoritmos de decodificación que cambian la codificación de los individuos.

Ejemplo mochila

```
class CromosomaMochila extends Cromosoma {
      static private int MAX_C = 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_C) return valor-MAX_C;
     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-penalty(totalPeso);
  }
```


Parametrización de los AGs

- No existen criterios definitivos para dar valores a los parámetros: Tamaño de la población, Longitud de los individuos, Probabilidades de mutación y cruce, Número máximo de iteraciones, etc..
- Los AGs son técnicas de búsqueda paramétricamente robustas: funcionan con un amplio rango de valores de sus parámetros.
- Determinan la eficiencia del algoritmo.
- Numerosos experimentos realizados han permitido disponer de algunas orientaciones.

Parametrización de los AGs

- El tamaño de la población ***Tam_Pob*** suele ser mayor de 50 (valores menores suelen plantear problemas de convergencia prematura).
- El tamaño de los individuos ***Len*** 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% (también se trabaja con otros valores).
- 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

- Consiste en utilizar el AG para la búsqueda global y usar métodos específicos para la búsqueda local.
- La hibridación se puede realizar de forma modular. La conexión entre módulos se puede hacer secuencialmente o en paralelo.
- En la implementación secuencial se deja que el AG converja suficientemente y después se aplica el método local sobre los mejores puntos obtenidos.
- En la implementación en paralelo varios procesadores subordinados realizan la búsqueda local y evalúan los individuos obtenidos y un procesador central conectado con todos ellos hace evolucionar la población proporcionada.