Programación Evolutiva

Tema 6: Algoritmos evolutivos

Carlos Cervigón, Lourdes Araujo. 2009-2010.

Algoritmos evolutivos

- Los AGs usan cadenas binarias con el fin obtener la máxima generalidad y el máximo grado de paralelismo implícito
- Esta representación es tan rígida que a veces es mejor considerar otras más flexibles aunque con ello se pierda algo de generalidad y de paralelismo implícito.
- Es posible compensar la pérdida de paralelismo implícito procesando más información útil (compensar la violación del principio del alfabeto de símbolos mínimo dando más importancia al principio de los bloques constructivos con significado).

Algoritmos evolutivos

- En 1994 Michalewicz propuso una estrategia de incorporación directa del conocimiento específico en la representación: Programas de Evolución (PEs).
- Siguen la estructura básica de los AGs, diferenciándose en:
  - Una representación natural y próxima al dominio del problema y que a la vez sea útil.
  - Se utilizan estructuras de datos como: vectores, reales, matrices, listas, conjuntos, árboles, etc.
  - Operadores genéticos específicos de la representación y lo más cerrados posible.
- Si se diseñan adecuadamente, los PEs pueden competir en eficiencia y calidad de las soluciones con métodos específicos (más costosos de implementar).

Ejemplo: Planificación de horarios

- Una posible formulación del problema es : disponemos de un colegio con *m* profesores {P1,…, Pm}, una serie de *n* turnos o de horas {H1,…,Hn} y una serie de *k* clases {C1,...,Ck}.
- Se trata de obtener el horario más adecuado (de acuerdo a unos objetivos especificados) que satisfaga una serie de restricciones más o menos severas

	H_1	H_2	...	H_n
P_1	C_{11}	C_{12}	...	C_{1n}
P_2	C_{21}	C_{22}	...	C_{2n}
...	...	...	...	...
P_m	C_{m1}	C_{m2}	...	C_{mn}

Algoritmos evolutivos

- Los PEs surgieron al resolver problemas de optimización paramétrica en los que se requería alta precisión.
- La representación con enteros binarios requiere utilizar individuos de tamaño muy grande (requerimientos computacionales y de almacenamiento excesivos).
- Los operadores genéticos clásicos son ineficaces para realizar el ajuste fino en las etapas finales de la evolución.
- Los resultados de los AGs mejoran significativamente si se usa una representación y unos operadores genéticos naturales.
- Se distinguen dos grandes grupos de PEs según la naturaleza de los objetos tratados:
  - PEs especializados en optimización numérica
  - PEs especializados en optimización combinatoria

Optimización numérica

- Numerosos resultados experimentales sobre estos tipos de problema indican la conveniencia de usar una representación y unos operadores más próximos al dominio del problema: vector de números en punto flotante (PF).
  - Los individuos son vectores de números en coma flotante en los que cada componente es asimilable a un gen.
  - La mutación altera ligeramente el valor del individuo.
  - El cruce proporciona dos individuos que "promedian" los valores de sus progenitores.
- Hay una gran variedad de operadores para esta representación.

Optimización numérica

- Al usar una representación sin codificación se debe comprobar que los resultados de los operadores genéticos están en sus correspondientes intervalos
- En los AGs no era necesario (implícito al diseñar el mecanismo de codificación).
- Se introduce el problema del tratamiento de las restricciones.
- Para problemas con restricciones muy sencillas la comprobación puede hacerse directamente, eliminando las soluciones no factibles. Para tratar restricciones no triviales se requieren mecanismos más complejos.

Optimización Combinatorial

- Los problemas de optimización combinatoria son muy variados y no existe un prototipo.
- En los problemas de optimización combinatoria el orden de las variables es inherente al propio problema
  - Estos problemas se adecuan a la representación mediante permutaciones
  - Si tenemos N variables su representación será una lista de N enteros en la que cada uno aparece una única vez
- Ejemplos
  - Problema del viajante de comercio (TSP)
  - El cuadrado mágico

Ejemplo: cuadrado mágico

- En un cuadrado es de n x n, el cuadrado tendrá n² casillas y los números que colocaremos serán del 1 a n² y la fórmula para encontrar la constante mágica de un cuadrado mágico de orden n es:

$$\frac{n(n^2+1)}{2}$$

- Ejemplo 3 x 3

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

- Individuo:

8	1	6	3	5	7	4	9	2

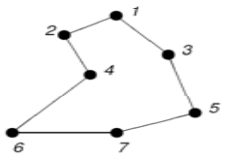
Representación de Permutaciones

- El problema del viajante de comercio (PVC) (Travelling Salesperson Problem o TSP) es un ejemplo destacado:

Dadas *m* ciudades (o vértices) {1,2,…,m} y los costes de viajar de unas a otras ***cij*** (*i,j* € {1,…,m}) se trata de calcular un recorrido (conjunto de aristas) completo (pasa por todas las ciudades), cerrado (que comience y termine en la misma ciudad) y conexo que haga mínimo el coste total del recorrido.

El problema del viajante

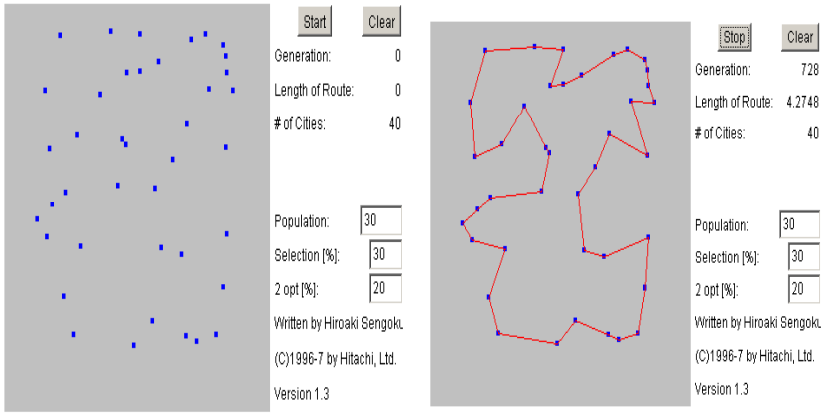
- Codificación
  - Etiquetar las ciudades: 1, …, N
  - Un camino completo será una permutación de los N enteros
- El espacio de búsqueda es muy grande
  - Para 50 ciudades tenemos 50! ≈10^{64}caminos posibles
    - El nº total de átomos del universo es de 10^{77}



El problema del viajante

- Interés del problema:
  - Valor teórico: problema NP-completo
  - Valor práctico: por ejemplo, en el trazado de circuitos VLSI
- Ejemplo arquetípico de problema basado en el orden: técnicas para resolverlo aplicables a otros muchos problemas basados en el orden y, en general, otros problemas combinatorios.
- Se han propuesto para resolverlo múltiples representaciones, cada una con sus operadores asociados.
- Son lo suficientemente generales como para ser útiles en otros problemas basados en el orden.

Demo TSP



El problema del viajante

- Espacio de búsqueda: permutaciones de las *m* ciudades o vértices.
- Cualquier permutación representa un recorrido completo, cerrado y conexo: candidato a solución.
- No existe un modo práctico de codificar el conjunto de las *m* permutaciones de elementos mediante cadenas binarias de bits de modo tal que los operadores genéticos clásicos sean eficientes en la resolución del PVC.
- La representación mediante cadenas de números enteros combina sencillez y eficiencia.
- Resulta fácil encontrar operadores de mutación cerrados.
- Los operadores de cruce son más complejos y específicos de cada representación.

Problema del viajante

- Representación: La manera más natural de codificar las soluciones del PVC consiste en enumerar las ciudades por orden de recorrido (sentido arbitrario, dada la simetría).
- Para un PVC 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

- El mismo recorrido comenzando por la ciudad 8:

8	9	4	6	2	3	5	1	7

- Y cambiando el sentido del recorrido:

3	2	6	4	9	8	7	1	5

Ejemplo

```
tipo TIndividuo = registro{
//identificadores de cada ciudad
 genes : vector de entero;
 real adaptación; //función de evaluación
//puntuación relativa:adaptación/sumadaptacion
 real puntuacion;
 real punt_acu; //puntuación acumulada
 entero longitudCromosoma = 27;
 . . .
 . . .
```

Cruce por emparejamiento parcial (PMX)

- Consiste en elegir un tramo de uno de los progenitores y cruzar preservando el orden y la posición de la mayor cantidad posible de ciudades del otro.

Algoritmo:

- **Elegir aleatoriamente dos puntos de corte.**
- **Intercambiar las dos subcadenas comprendidas entre dichos puntos en los hijos que se generan.**
- **Para los valores que faltan en los hijos se copian los valores de los padres:**
  - **Si un valor no está en la subcadena intercambiada, se copia igual.**
  - **Si está en la subcadena intercambiada, entonces se sustituye por el valor que tenga dicha subcadena en el otro padre.**

Cruce por emparejamiento parcial (PMX)

- Ejemplo:

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

- Se selecciona un recorrido parcial eligiendo al azar dos puntos de corte (Por ejemplo 3 y 7):

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

Cruce por emparejamiento parcial (PMX)

- Se intercambian los segmentos situados entre los puntos de corte:

x	x	x	1	8	7	6	x	x
x	x	x	4	5	6	7	x	x

- Ese intercambio define también un conjunto de emparejamientos que servirán para despejar las x:
  - 1 con el 4
  - 8 con el 5
  - 7 con el 6
  - 6 con 7

Cruce por emparejamiento parcial (PMX)

- Se especifican las x de los progenitores originales que no planteen conflicto:

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

- Las x que plantean conflicto se reemplazan por su pareja. Por ejemplo, el primer elemento del primer progenitor está repetido, por lo que se sustituye por 4:

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

- Se obtienen soluciones factibles.

Cruce por orden (OX)

- El cruce por orden consiste en copiar en cada uno de los hijos una subcadena de uno de los padres mientras se mantiene el orden relativo de las ciudades que aparecen en el otro padre.

Algoritmo:

- **Elegir aleatoriamente dos puntos de corte.**
- **Copiar los valores de las subcadenas comprendidas entre dichos puntos en los hijos que se generan.**
- **Para los valores que faltan en los hijos se copian los valores de los padres comenzando a partir de la zona copiada y respetando el orden:**
  - **Si un valor no está en la subcadena intercambiada, se copia igual.**
  - **Si está en la subcadena intercambiada, entonces se pasa al siguiente posible.**

Cruce por orden (OX)

- Se elige para cada descendiente un tramo de uno de los progenitores y a la vez se preserva el orden relativo de todas las ciudades del otro.
- Ejemplo: Selecciona un recorrido parcial (dos puntos de corte elegidos al azar):

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

- Intercambio de segmentos:

x	x	x	1	8	7	6	x	x
x	x	x	4	5	6	7	x	x

Cruce por orden (OX)

- Para cada progenitor se parte de uno de los puntos de corte y se copian las ciudades del otro progenitor conservando el orden relativo y omitiendo las que ya estén presentes
- Al llegar al final de la cadena se continúa por el principio hasta retornar al punto de partida.
- En el ejemplo se parte del segundo punto de corte.
- Para el primer progenitor se obtiene el descendiente:

3	4	5	1	8	7	6	9	2

Cruce por orden (OX)

- Para el segundo progenitor se obtiene el descendiente:

2	1	8	4	5	6	7	9	3

- Este cruce sólo considera el orden relativo de las ciudades, no su posición.
- Existen dos variantes de este cruce que toman en cuenta las posiciones (muy usados en planificación de tareas).

Variantes al cruce por orden

- Cruce por orden con posiciones prioritarias
  - No se elige un tramo para intercambiarlo entre los progenitores, sino un conjunto de posiciones al azar. (El resto no cambia).
- Cruce por orden con orden prioritario:
  - Los individuos no intercambian ciudades, sino el orden relativo existente entre ellas.
- Ejemplo: Se eligen para el intercambio de orden las posiciones 3, 4, 6 y 9.
- En el primer progenitor el orden de esas ciudades es

$$3 \rightarrow 4 \rightarrow 6 \rightarrow 9$$

Variantes al cruce por orden

- En el segundo progenitor esas ciudades ocupan las posiciones 9, 1, 7 y 8 (el primer descendiente será una copia del segundo progenitor en todas las posiciones salvo en ésas):

x	5	2	1	8	7	x	x	x

- En ellas se colocarán las ciudades seleccionadas conservando su orden relativo:

3	5	2	1	8	7	4	6	9

- El segundo se obtiene repitiendo el proceso para el segundo progenitor:

2	1	7	4	5	6	3	8	9

Cruce por ciclos

- Cada ciudad hereda sucesivamente la posición de alguno de los progenitores, de acuerdo con sus posiciones en un ciclo.

v =

1	2	3	4	5	6	7	8	9

w =

4	1	2	8	7	6	9	3	5

- Se opera completando "ciclos de sucesión". Para el primer descendiente se parte de la primera ciudad del primer progenitor

1	x	x	x	x	x	x	x	x

Cruce por ciclos

- Obliga a darle a la ciudad *4* el 4º puesto:

1	x	x	4	x	x	x	x	x

- Esto selecciona la ciudad 8 (ciudad bajo la 4 en w).
- Análogamente, se incluyen la ciudades 3 y 2, lo que lleva a la 1 (que completa el ciclo)

1	2	3	4	x	x	x	8	x

- La ciudades restantes se rellenan con el otro padre:

1	2	3	4	7	6	9	8	5

- El segundo descendiente se obtiene análogamente:

4	1	2	8	5	6	7	3	9

Cruce por recombinación de rutas

- La descendencia se construye combinando las rutas que interconectan las ciudades de los progenitores.
- Se construye la tabla de conectividades entre ciudades de uno u otro progenitor:

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

- Se construye el primer descendiente tomando la ciudad inicial de uno de los progenitores, (por ejemplo, la del segundo, *4*).

Cruce por recombinación de rutas

- De entre todas las ciudades a que está conectada la anterior (*5* y *3*) se toma la menos conectada y en caso de empate se toma una de ellas al azar (la *5*) (así se incrementa la probabilidad de completar un recorrido con éxito).
- Se repite el paso anterior considerando todas las nuevas ciudades conectadas a la última (es decir, *6* y *2*).
- Se reitera el paso anterior hasta terminar con éxito o no poder continuar (en este caso se comienza de nuevo).
- En este caso se termina con éxito, resultando como primer descendiente

4	5	6	7	8	1	2	3	9

Cruce por recombinación de rutas

- Para obtener el segundo descendiente se repite el proceso comenzando por la otra ciudad inicial. Ahora existe la posibilidad de llegar a un bloqueo:

4	5	6	7	8	1	2	3	9

entonces se comienza de nuevo hasta lograr un individuo factible, como:

1	2	5	4	3	9	8	7	6

Codificación ordinal

Nos planteamos la existencia de una codificación del PVC para la que el cruce clásico sea un operador cerrado.

Codificación ordinal:

- Se ordenan todas las ciudades en una lista dinámica de referencia según cierto criterio.
- Para construir un individuo se van sacando una a una las ciudades recorridas, codificando en el j-ésimo gen del individuo la posición que tiene la j-ésima ciudad en la lista dinámica.
- Ese número es siempre un entero entre 1 y $m-j+1$

Codificación ordinal

Ejemplo :

Lista dinámica L = { 1,2,3,4,5,6,7,8,9}

El recorrido

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

Se representa

4	4	2	1	4	3	2	2	1

El recorrido

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

Se representa

1	1	1	1	1	1	1	1	1

Codificación ordinal

Aplicando el cruce clásico en el cuarto gen se obtienen los individuos:

4	4	2	1	1	1	1	1	1

1	1	1	1	4	3	2	2	1

Que corresponden a los recorridos

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

y

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

factibles

Mutación por inversión

- En este tipo de mutación se aplica con una determinada probabilidad y consiste en seleccionar dos puntos del individuo al azar e invertir los elementos que hay entre dichos puntos.

1	2	6	9	4	7	3	8	5

1	2	6	3	7	4	9	8	5

Mutación por intercambio

- En este tipo de mutación se seleccionan dos puntos al azar y se intercambian los valores. Por ejemplo, los valores de las posiciones 3 y 7:

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

1	2	**3**	3	7	4	**6**	8	5

Mutación por inserción

- En este tipo de mutación se inserta una o varias ciudades elegidas al azar en unas posiciones también elegidas al azar. El caso más simple es con una sola inserción, por ejemplo, seleccionamos la ciudad 4 y la insertamos en la tercera posición, tal y como se muestra

1	2	6	9	4	7	3	8	5

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

- Un ejemplo con varias inserciones (también llamada con desplazamiento):

1	2	6	9	4	7	3	8	5

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

Mutación heurística

- En este tipo de mutación se seleccionan **n** ciudades al azar. A continuación se generan todas las permutaciones de las ciudades seleccionadas. De todos los individuos que se generan con dichas permutaciones se selecciona el mejor.
- Por ejemplo, si seleccionamos al azar las ciudades 6, 4 y 3:

7	2	6	9	4	1	3	8	5

- Las permutaciones son: 643, 634, 463, 436, 346 y 364

Mutación heurística

- Los individuos que se pueden obtener con las permutaciones de esa tres ciudades son 6 y entre ellas elegiremos la mejor.

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

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

. . .

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

Planificación de horarios

- El problema de la planificación de horarios es fundamental en Investigación Operativa, tanto a nivel práctico como teórico.
- Ejemplo destacado de PEs con representación matricial.
- Formulación:
  - En una empresa se dispone de m empleados $\{T_1,\ldots, T_m\}$ que deben atender k puestos $\{C_1, \ldots, C_k\}$ durante n turnos $\{H_1,\ldots,H_n\}$ a lo largo de la semana (H_1 representa el turno *miércoles por la tarde*).

Planificación de horarios

- Se trata de obtener el horario más adecuado (de acuerdo a unos objetivos especificados) que satisfaga restricciones:
  - En todo momento debe haber un solo empleado en cada puesto.
  - Un empleado no puede estar en más de un puesto en el mismo turno.
  - El número de empleados desocupados debe ser lo menor posible.
- Y otras restricciones más flexibles:
  - Cada empleado tiene asignado un número de horas a la semana, que se desea alcanzar y no exceder.
  - Preferencias de horario de los empleados.
- Es un caso particular del problema general de planificación de tareas.

Planificación de horarios

- Función de aptitud: Suma de las productividades. Se introducen penalizaciones para el tratamiento de restricciones.
- Representación: Matriz de enteros **R** := $[\boldsymbol{r}_{ij}] \in R^{mxn}$ en la que el elemento $r_{ij} \in \{C_1,\ldots, C_k\}$ indica el puesto que le ha correspondido al empleado T_i en el turno H_j:

	H_1	H_2	$\ldots$	H_n
T_1	r_{11}	r_{12}	$\ldots$	r_{1n}
T_2	r_{21}	r_{22}	$\ldots$	r_{2n}
$\vdots$			$\ddots$	$\vdots$
T_m	r_{m1}	r_{m2}	$\ldots$	r_{mn}

Planificación de horarios

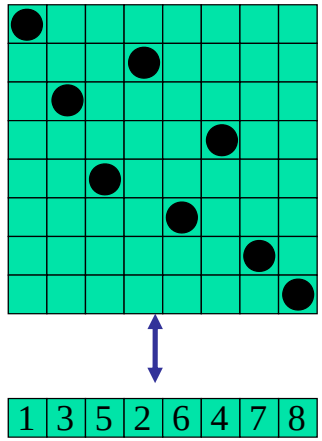
Operadores genéticos:

- Mutación de orden **p**: Para un solo empleado (fila) se toman dos sucesiones contiguas de $\boldsymbol{p}$ turnos y se intercambian.
- Mutación completa de orden **p**: Mutación de orden **p** aplicada a todos los empleados.
- Cruce heurístico de orden **q:** En cada progenitor se calcula para cada empleado (fila) la aptitud local asociada al horario asignado y se intercambian los $\boldsymbol{q}$ mejores con el otro progenitor. El parámetro $\boldsymbol{q}$ suele ir variando a lo largo de la búsqueda.
- Algoritmos de reparación: Tratan las soluciones inconsistentes. Su diseño es costoso y específico.

Las 8 reinas

fenotipo: el tablero

genotipo:
Permutaciones del 1-8



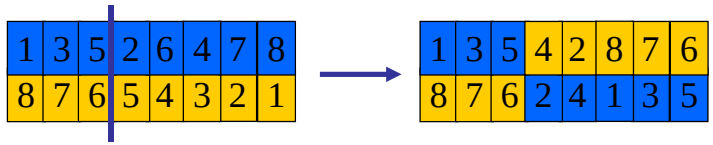
8 reinas

- aptitud:
  - Penalty de una reina: el número de reinas que come
  - Penalty de una configuración: la suma de los penalties de todas la reinas. Minimizar el penalty.
- Mutación :
  - pequeña variación en una permutación
  - intercambiar los valores de dos posiciones aleatorias

8 reinas

Cruce:

- Combinar dos permutaciones en dos nuevas permutaciones:
  - Elegir punto de cruce
  - Copiar la primera parte en los hijos
  - Crear la segunda parte insertando valores de otro padre:
    - En el orden en el que aparecen
    - Comenzando detrás del punto de cruce
    - Saltando los valores que ya están en el hijo



8 reinas

- Selección de padres:
  - Coger 5 padres y coger los dos mejores a cruzar
- Reemplazo
  - Al insertar un nuevo hijo en la población, elegir el miembro a reemplazar:
    - Ordenando la población por fitness decreciente
    - Enumerando la lista de arriba a abajo
    - Reemplazar el primero con fitness menor que el hijo dado

8 reinas: resumen

Representation	Permutations
Recombination	"Cut-and-crossfill" crossover
Recombination probability	100%
Mutation	Swap
Mutation probability	80%
Parent selection	Best 2 out of random 5
Survival selection	Replace worst
Population size	100
Number of Offspring	2
Initialisation	Random
Termination condition	Solution or 10,000 fitness evaluation