Programación Evolutiva

Tema 6: Algoritmos evolutivos

Carlos Cervigón, Lourdes Araujo. 2023-2024.

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.
- En 1994 Michalewicz propuso una estrategia de incorporación directa del conocimiento específico en la representación

Algoritmos evolutivos

Algoritmos evolutivos

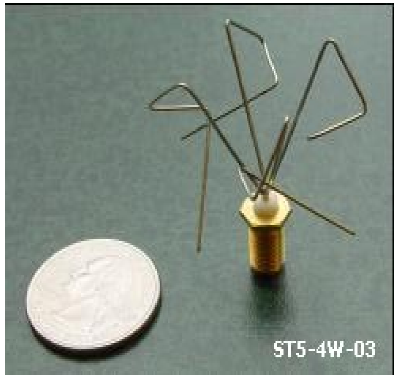
Siguen la estructura básica de los AGs, diferenciándose en:

- 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.

- Se usan en muchos problemas de optimización donde a veces una solución casi óptima es suficiente

Algoritmos evolutivos

- Planificación de horarios
- Búsqueda de rutas
- Cortado de patrones
- Asignación de pistas de aeropuerto
- Optimización de diseños de antenas (NASA)
- Asignación cuadrática
- Travelling Salesman Problem (TSP)
- Cualquier problema de optimización: Numérica, Combinatoria..



NASA Evolvable Systems Group

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}

Ejemplo: Planificación de horarios

- La función de adaptación o fitness “mide” la calidad de un horario teniendo en cuenta las restricciones
- Restricciones
  - En todo momento debe haber un solo profesor en cada clase.
  - Un profesor no puede estar en más de una clase a la vez.
  - El número de profesores desocupados debe ser lo menor posible.
  - Se deben distribuir los turnos a lo largo de toda la semana.
  - Se deben dejar las tardes libres a los profesores con dedicación parcial.
- Se pueden utilizar técnicas de reparación que gestionan las soluciones inconsistentes, por ejemplo un horario con duplicidades obtenido al realizar un cruce

Generalización Planificación de horarios

- **Función de aptitud**: Suma de las productividades. Se introducen penalizaciones para el tratamiento de restricciones.

- **Representación**: Matriz de enteros **R** := $[\boldsymbol{r}_{\mathbf{ij}}] \in R^{mxn}$ en la que el elemento $r_{\mathbf{ij}} \in \{C_1,\ldots, C_k\}$ indica el puesto que le ha correspondido al empleado $T_{\mathbf{i}}$ en el turno $H_{\mathbf{j}}$:

Turnos

Empleados

	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

Algunos 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.

Optimización Combinatoria

- 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
  - El cuadrado mágico
  - Colocación de bloques
  - Problema del viajante de comercio (TSP)

Ejemplo: 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:

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

- Ejemplo 3 x 3

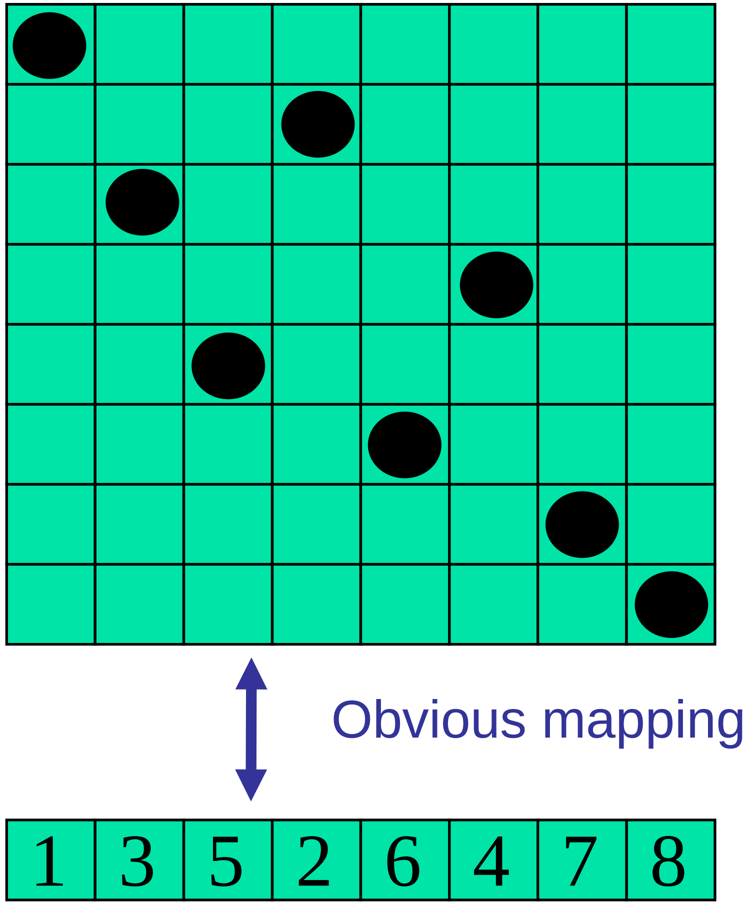
8	1	6	15
3	5	7	15
4	9	2	15
15	15	15	15

- Individuo: [8 1 6 3 5 7 4 9 2]

8 reinas

fenotipo: el tablero

genotipo:
Permutaciones del 1-8



8 reinas

- Fitness:
  - 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

1	3	5	2	6	4	7	8

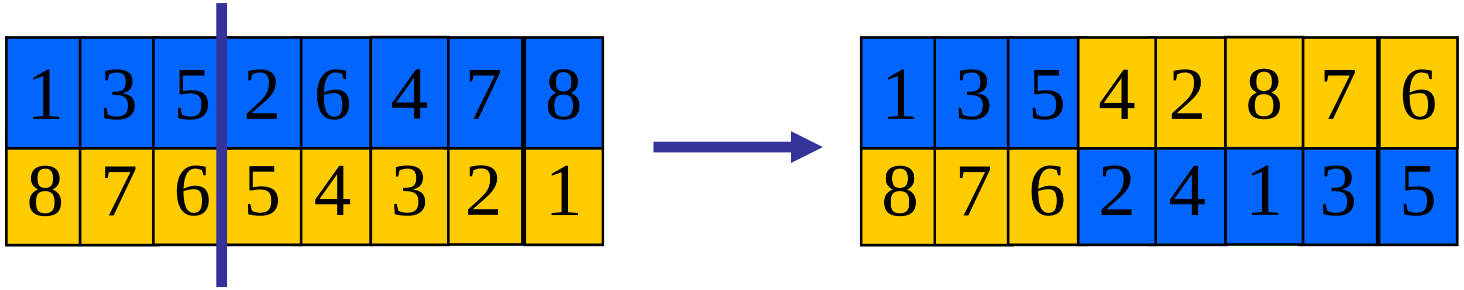
$\longrightarrow$

1	3	7	2	6	4	5	8

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



Ejemplo: colocación de bloques

- Consiste en la colocación de unos bloques rectangulares dentro de un contenedor de base rectangular aprovechando al máximo la superficie de dicho contenedor.
- Se dispone de un contenedor con una superficie rectangular de anchura A y de altura ilimitada, y una lista de *n* bloques $Ln=(R_1,R_2,...,R_n)$ con $n>=1$. Cada uno de ellos con una longitud menor o igual que A.
- El objetivo es colocar los bloques dentro del contenedor, minimizando la altura total alcanzada por los mismos y cumpliendo restricciones:
  - los rectángulos no se pueden solapar
  - no pueden sobrepasar los límites de la superficie
  - no se permiten las rotaciones de 90º.

Problema de colocación de bloques

- Representación con enteros: a cada uno de los bloques a colocar se le asigna un número entero
- El individuo es una secuencia de números enteros correspondientes a bloques, en un orden correspondiente a las posiciones en la superficie del contenedor.

1	**3**	**2**	**4**

- El bloque 1 es el primero en colocarse, luego el 3, luego el 2 y por último el 4.
- Con esto conocemos la secuencia de entrada de los bloques en el contenedor, pero necesitamos saber cómo se colocan dentro del contenedor; para ello se utilizan diferentes algoritmos.
- *Ejemplo*: **Algoritmos Evolutivos: teoría y casos prácticos.** Lourdes Araujo, Carlos Cervigón. Kindle Edition Amazon

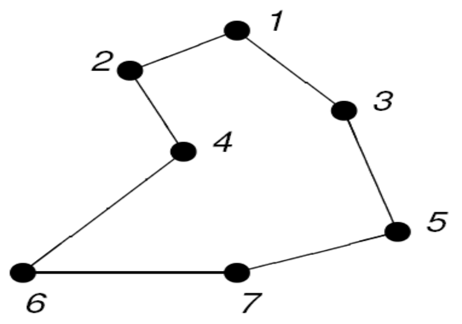
Representación de Permutaciones

- 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! $\approx 10^{64}$ caminos posibles
    - El nº total de átomos del universo es de 10^{77}



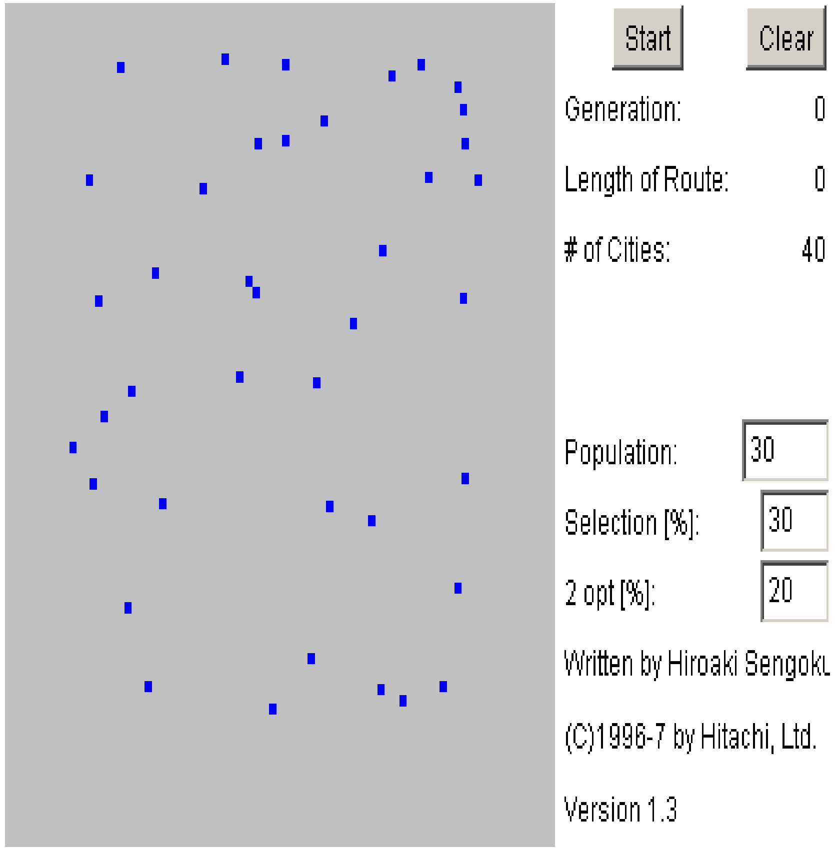
El problema del viajante

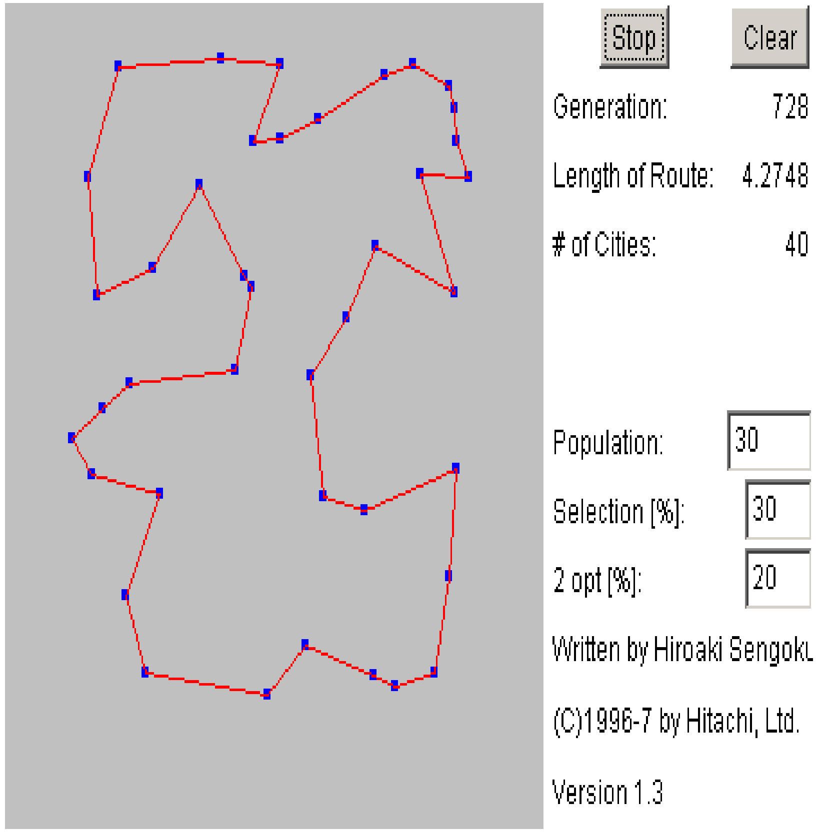
- Interés del problema:
  - Valor teórico: problema NP-completo
  - Valor práctico: por ejemplo, como aplicación al trazado de circuitos VLSI

- 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.

TSP: muchas demos por la Web

http://joneisen.me/d3-tsp-demo/





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 problema.
- La representación mediante **cadenas de números enteros** combina sencillez y eficiencia.

5	1	7	8	9	4	6	2	3

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
{
 Lista de enteros : genes; //identificadores de cada ciudad
 real fitness; //fitness o adaptación
 real puntuacion; //puntuación relativa:adaptación/sumadaptacion
 real punt_acu; //puntuación acumulada
 entero longitudCromosoma = 27;
 . . .
 . . .
```

3	2	6	4	23	12	7	...	5

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

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

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

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).

Cruce por orden con posiciones prioritarias (OXPP)

- No se elige un tramo para intercambiarlo entre los progenitores, sino un conjunto de posiciones al azar. (El resto no cambia).

P1: 1 2 $\boxed{3}$ $\boxed{4}$ 5 $\boxed{6}$ 7 8 9

P2: 4 5 $\boxed{2}$ $\boxed{1}$ 8 $\boxed{7}$ 6 9 3

P1: _ _ 2 1 _ 7 _ _ _

P2: _ _ 3 4 _ 6 _ _ _

H1:	3	4	2	1	5	7	6	8	9
H2:	2	1	3	4	8	6	7	9	5

Cruce por orden con orden prioritario (OXOP): cálculo H1

P1: 1 2 $\boxed{3}$ $\boxed{4}$ 5 $\boxed{6}$ 7 8 $\boxed{9}$

P2: 4 5 $\boxed{2}$ $\boxed{1}$ 8 $\boxed{7}$ 6 9 $\boxed{3}$

Cruce por orden con orden prioritario:

Los individuos no intercambian ciudades, sino el orden relativo existente entre ellas.

Selecciono unas posiciones al azar, por ejemplo 3ª 4ª 6ª 9ª

Miro en en P1 las ciudades de esas posiciones son

3 4 6 9

Miro en en P2 en que posición están dichas ciudades son

9ª 1ª 7ª 8ª

Cruce por orden con orden prioritario: cálculo H1

Para generar el primer hijo: P2: 4 5 2 1 8 7 6 9 3

Copio los elementos de P2 excepto los de las posiciones anteriores (*9º 1º 7º 8º*)

H1: _ 5 2 1 8 7 _ _ _

Y los huecos se rellenan con las ciudades que había seleccionado inicialmente de P1, es decir 3 4 6 9 quedando

H1: 3 5 2 1 8 7 4 6 9

Universidad Complutense Madrid

Cruce por orden con orden prioritario: cálculo H2

P1:	1	2	$\boxed{3}$	$\boxed{4}$	5	$\boxed{6}$	7	8	$\boxed{9}$
P2:	4	5	$\boxed{2}$	$\boxed{1}$	8	$\boxed{7}$	6	9	$\boxed{3}$

Selecciono unas posiciones al azar, por ejemplo 3ª 4ª 6ª 9ª

Miro en en P2 las ciudades de esas posiciones son

2 1 7 3

Miro en en P1 en que posición están dichas ciudades son

2ª 1ª 7ª 3ª

Cruce por orden con orden prioritario : cálculo H2

Para generar el segundo hijo:

P1: 1 2 3 4 5 6 7 8 9

Copio los elementos de P1 excepto los de las posiciones anteriores (*2ª 1ª 7ª 3ª*)

H2: _ _ _ 4 5 6 _ 8 9

Y los huecos se rellenan con las ciudades que había seleccionado inicialmente de P2, es decir 2 1 7 3 quedando

H2: 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.

P1 =

1	2	3	4	5	6	7	8	9

P2 =

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

1	2	3	4	5	6	7	8	9

4	5	2	1	8	7	6	9	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 repite 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

Universidad Complutense Madrid

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

Ejemplo: Sudoku Evolutivo

Sudoku evolutivo

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

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

Figura 1. *Configuración inicial de un tablero y su solución*

Sudoku evolutivo

- El Sudoku no parece un problema adecuado para resolver utilizando un algoritmo genético (pues es problema de solución única)
- Es un problema muy útil para experimentar con diferentes operadores genéticos y con diferentes parámetros.
- El Sudoku puede considerarse un problema de optimización con restricciones, a nivel de fila, columna y subcuadrícula.

Sudoku evolutivo

- **Representación 2**: El individuo se representa mediante un vector compuesto por 9 genes de 9 elementos cada uno.
- Cada gen se puede corresponder con una fila, una columna o una subcuadrícula del tablero.
- El individuo es un vector de 81 posiciones donde las posiciones fijas del tablero se respetan y las posiciones vacías se representan como 0.

Universidad Complutense Madrid

Sudoku evolutivo

- Tablero

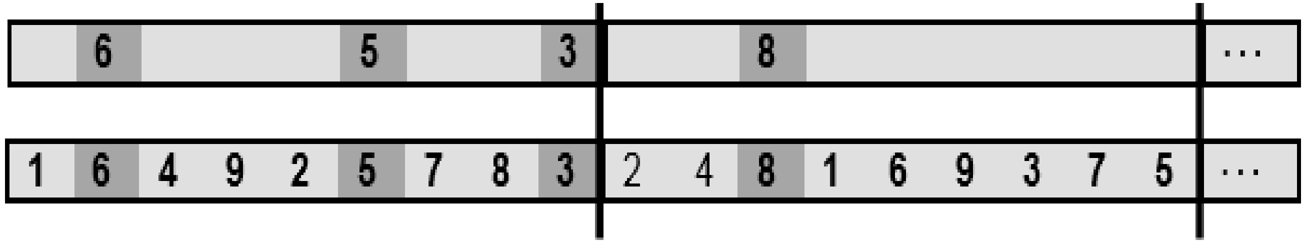
	6				5			3
		8						
2					1			9
8			4		7			6
		6				3		
7			9		1			4
5								2
		7	2		6	9		
	4		5		8		7	

- Individuo o cromosoma

0	6	0	0	0	5	0	0	3	0	0	8	0	0	0	0	0	0	..

Sudoku evolutivo

- La población inicial: Los huecos del cromosoma se inicializan con valores aleatorios entre 1 y 9, respetando las posiciones fijas y sin generar elementos repetidos dentro de la fila, de forma que al calcular la adaptación del individuo no tengamos que preocuparnos por esta restricción.



Sudoku evolutivo

- Función de adaptación:

F(x) = 5 * RS + RP + 20 * REA

- **RS: Restricción de sumas**: se basa en que en un tablero solución la suma de los valores de cada cuadrícula y cada columna debe ser 45. Se calcula para las columnas y las cuadrículas:
  - Restricción de columna *i*: |45 – suma de los valores de la columna *i* |
  - Restricción cuadrícula *i* : |45 – suma de los valores de la cuadrícula *i* |

RS = Suma de restricciones de las 9 columnas + suma de restricciones de las 9 cuadrículas

Sudoku evolutivo

- Función de adaptación:

$$F(x) = 5 * RS + RP + 20 * REA$$

- **Restricción de productos (RP): s**e basa en que en un tablero solución el producto de los valores de cada cuadrícula y cada columna debe ser 9!.
  - Restricción de columna *i*: | 362880 – producto de valores de la columna i |
  - Restricción cuadrícula *i* : | 362880 – producto de valores de la cuadrícula i |

RP = Suma de la raíz cuadrada de las restricciones de cada una de las 9 columnas + suma de la raíz cuadrada de las restricciones de las 9 cuadrículas

Sudoku evolutivo

- Función de adaptación:

$$F(x) = 5 * RS + RP + 20 * REA$$

- **REA: Restricción de elementos ausentes**
  - Restricción de columna *i*: número de valores del conjunto de dígitos del 1 al 9 que faltan en la columna i.
  - Restricción de cuadrícula *i*: número de valores del conjunto de dígitos del 1 al 9 que faltan en la cuadrícula i.

REA = Suma de restricciones de las 9 columnas + suma de restricciones de las 9 cuadrículas

Sudoku evolutivo

Operador de cruce:

- Los puntos de corte deben mantener intactas las filas, intercambiando filas completas.

0	6	0	0	0	5	0	0	3	0	0	8	0	0	0	0	0	0	..
1	0	0	6	0	0	0	0	3	0	0	0	0	4	0	0	0	0	..

- Con este punto de corte y cruce de un punto), uno de los hijos contendrá la primera fila del primer padre y el resto de las filas del segundo padre, siempre respetando las posiciones fijas que aparecen sombreadas.
- Otros cruces de filas: pmx, ox

Sudoku evolutivo

Mutación a nivel de fila

- El esquema más utilizado es el de mutación por intercambio: se seleccionan dos posiciones aleatorias de la fila hasta encontrar dos valores que no sean fijos y a continuación se intercambian.