EA *Algoritmos genéticos*

4 Algoritmos genéticos: Introducción

- Inventados por John Holland a mitades de los 70.
- Inspirados en el modelo de evolución biológica.
- Utilizan el principio de selección natural para resolver problemas de optimización “complicados”.

- Idea:
  - Partiendo de una población inicial (soluciones factibles)
  - Seleccionar individuos (favorecer a los de mayor *calidad)*
  - Recombinarlos
  - Introducir mutaciones en sus descendientes
  - Insertarlos en la siguiente generación

4 Algoritmos genéticos
Esquema básico

```
algoritmo genético
principio
  t:=0;
  inicializa P(t);
  evalúa P(t);
  mq not termina hacer
    t:=t+1;
    P(t):=selecciona P(t-1);
    recombina P(t);
    muta P(t);
    evalúa P(t)
  fmq;
fin
```

4 Algoritmos genéticos Esquema básico

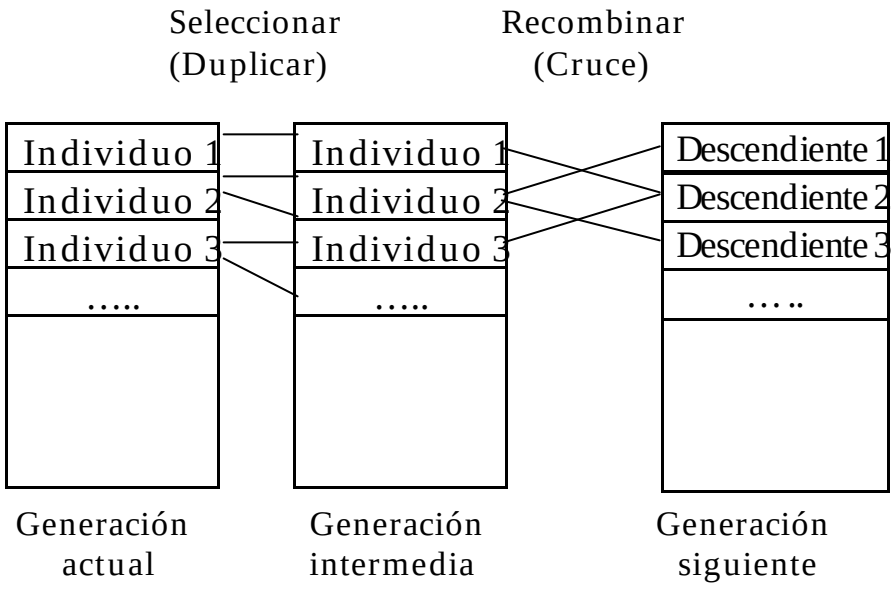
- Maximizar $f(x) = x^2$ con x entero entre 0 y 31
  - Representación en binario

0	1	1	0	1

  - Población inicial generada aleatoriamente, tamaño 4

01101
11000
01000
10011

  - Función de calidad $f(x) = x^2$
  - Selección:



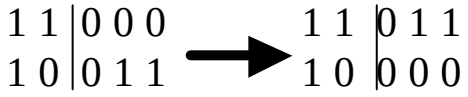
4 Algoritmos genéticos Esquema básico

– Generar una población intermedia. Para ello asignar a cada individuo una probabilidad de ser seleccionado directamente proporcional a su función de calidad.

Cromosoma	x	f(x)	prob.	copias
01101	13	169	0.14	1
11000	24	576	0.49	2
01000	8	64	0.06	0
10011	19	361	0.31	1

De la población intermedia se seleccionan parejas de forma aleatoria.

– Cruce: elegir un punto intermedio e intercambiar los genes de los padres a partir de ese punto.



– Mutación: cambio de un bit elegido aleatoriamente (probabilidad pequeña).

Cromosoma	x	f(x)
11011	27	729
10000	16	256
01100	12	144
11001	25	625

4 Algoritmos genéticos
Esquema básico

- Codificación:
  - Utilizar cadenas de bits para representar las soluciones
  - Los bits pueden codificar números enteros, reales, conjuntos, …
  - Ventaja: los operadores de cruce y mutación son simples.
  - Inconveniente: no siempre resulta “natural”.

- Selección:
  - Asignar una probabilidad de supervivencia proporcional a la calidad
  - Generar una población intermedia
  - Elegir parejas de forma aleatoria
  - No se pueden cruzar elementos de dos generaciones distintas.

- Operador de cruce de un punto

- Mutación:
  - Hay una pequeña probabilidad de cambio de un bit.

4 El problema de la mochila

- Recordar...
  - Se tienen n objetos y una mochila
  - El objeto i tiene peso p_i y la inclusión del objeto i en la mochila produce un beneficio b_i
  - El objetivo es llenar la mochila, de capacidad C, de manera que se maximice el beneficio.

$$\text{maximizar} \quad \sum_{1 \le i \le n} b_i x_i$$

$$\text{sujeto a} \quad \sum_{1 \le i \le n} p_i x_i \le C$$

$$\text{con} \quad x_i \in \{0,1\},\ b_i > 0,\ p_i > 0,\ 1 \le i \le n$$

- Problemas:
  - Codificación genética: cómo representar las soluciones
  - Calidad de las soluciones: cómo se mide

El problema de la mochila

- Representación:

$$x = (x_1, \ldots, x_n), \quad x_i \in \{0, 1\}$$

Observar que no se garantiza factibilidad

- Función de calidad:

$$f(x) = \begin{cases} C - \sum_{1 \le i \le n} b_i x_i & \text{si } \sum_{1 \le i \le n} b_i x_i > C \\ \sum_{1 \le i \le n} b_i x_i & \text{en otro caso} \end{cases}$$

Penalizar la no factibilidad. Obliga al algoritmo a elegir soluciones factibles porque son mejores.

- Inicialización:

Generar secuencias de ceros y unos

- Operador de cruce de un punto

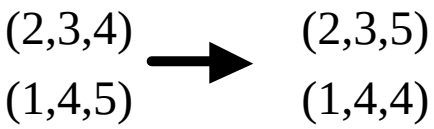
4 El problema de la mochila

- Otra posible representación:

  Una lista con los elementos que metemos en la mochila.

- Problema: qué operador de cruce utilizamos?

  Observar que el operador de un punto no sirve, es necesario adaptarlo.



  Por ejemplo, eliminar los elementos repetidos

Asignación de recursos

- Hay m recursos de capacidades $c_1, c_2, \ldots, c_m$ y n tareas a ejecutar que consumen parte de los recursos.
  La tarea i-ésima consume w_{ij} partes del recurso j.

- La ejecución de la tarea i-ésima produce un beneficio b_i.

- Se trata de decidir qué tareas se ejecutan de manera que se maximice el beneficio total.

Asignación de recursos

- Representación de un individuo:
  - $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, con $x_i \in \{0,1\}$
    (x_i=1 significa ejecutar la tarea i-ésima)
  - para ser factible debe verificar: $\Sigma_{i=1} w_{ij} x_i \leq c_j$, para j=1,2,…,n
  - y para ser óptima debe maximizar: $B(\boldsymbol{x}) = \Sigma_{i=1} x_i b_i$

- La función de calidad:

  $f(\boldsymbol{x}) = \Sigma_{i=1} b_i x_i - s \max\{b_i\}$

  donde $s = |\{j \mid \Sigma_{i=1} w_{ij} x_i > c_j\}|$, es decir, el número de recursos agotados.

- El tamaño de la población elegido es μ=50, la tasa de mutación $p_m = 1/n$, y la tasa de recombinación p_c=0'6.

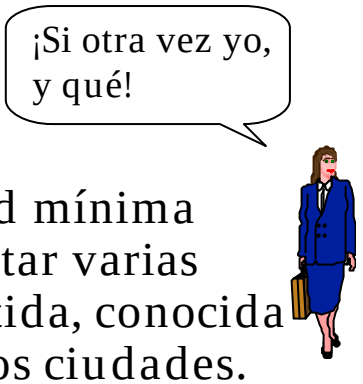
Asignación de recursos

- Resultados obtenidos tras 100 ejecuciones de 6 casos distintos:

n=15, m=10		n=20, m=10		n=28, m=10		n=50, m=5		n=60, m=30		n=105, m=2	
$f_{5\cdot10^3}(x)$	N	$f_{10^4}(x)$	N	$f_{5\cdot10^4}(x)$	N	$f_{10^5}(x)$	N	$f_{10^5}(x)$	N	$f_{2\cdot10^5}(x)$	N
4015	83	6120	33	12400	33	16537	1	7772	5	1095445	–
4005	16	6110	20	12390	30	16524	1	7761	4	1095382	10
3955	1	6100	29	12380	10	16519	2	7758	11	1095357	3
		6090	11	12370	1	16518	5	7741	7	1095266	1
		6060	3	12360	19	16499	1	7739	1	1095264	9
		6050	1	12330	5	16497	1	7738	3	1095206	3
		6040	3	11960	1	16494	1	7725	1	1095157	2
				11950	1	16473	1	7719	1	1095081	1
						16472	1	7715	1	1095035	2
						16467	1	7711	2	1095035	8
						16463	1	7706	1	1094965	1
f=4012’7		f=6102’3		f=12374’7		f=16378		f=7626		f=1093897	

4 *El problema del viajante*

- Recordar



Encontrar un recorrido de longitud mínima para un viajante que tiene que visitar varias ciudades y volver al punto de partida, conocida la distancia existente entre cada dos ciudades.

- Codificación: en forma de vector siguiendo el orden del recorrido

  Ejemplo:

  [3, 2, 5, 4, 1] ⟶ 3 → 2 → 5 → 4 → 1

- Cruce:
  - De un punto:

    [3, 2, 5, 4, 1] ⟶ [3, 2, 1, 4, 2]
    [3, 5, 1, 4, 2] ⟶ [3, 5, 5, 4, 1]

    - Pueden aparecer ciudades repetidas
    - No siempre visitamos todas.

4 El problema del viajante

- Heurística:
  - Elegir una ciudad , *i*, aleatoriamente
  - Suponer que en el padre 1 de la ciudad *i* vamos a la *j* y en el padre 2 de *i* vamos a *k*
    - Si *j,k* ya están incluidos, elegir una nueva ciudad.
    - Si no, añadir la ciudad que no esté incluida más próxima a *i*.
  - Repetir mientras queden ciudades sin recorrer

❖ Otra codificación:

Asignar a cada ciudad un valor entre 0 y 1 aleatoriamente. El recorrido se obtiene al ordenar estos números de mayor a menor.

Ejemplo:

[0.2, 0.8, 0.4, 0.7, 0.9] ⟶ 5⟶2⟶4⟶3⟶1

❖ Cruce:

Cualquiera de los habituales, de un punto por ejemplo.

EA Variantes del esquema básico

- Codificación: ¿cómo se representan las soluciones en forma de "cromosomas"?
  - Cadenas de 0's y 1's (algoritmos clásicos)
  - Números enteros y reales
  - Otros
- Cuestiones a tener en cuenta:
  - Factibilidad: los cromosomas pueden codificar soluciones no factibles del problema.
    - Solución: penalizar en la función de calidad
      descartar
      reparar
  - Legalidad: los cromosomas pueden no ser decodificables a una solución.

    Ejemplo: problema de la mochila

    (2,3,4) → (2,3,5)
    (1,4,5) → (1,4,4)
  - Unicidad de la codificación:
    - Uno a uno
    - Uno a *N*
    - *N* a uno

EA Variantes del esquema básico

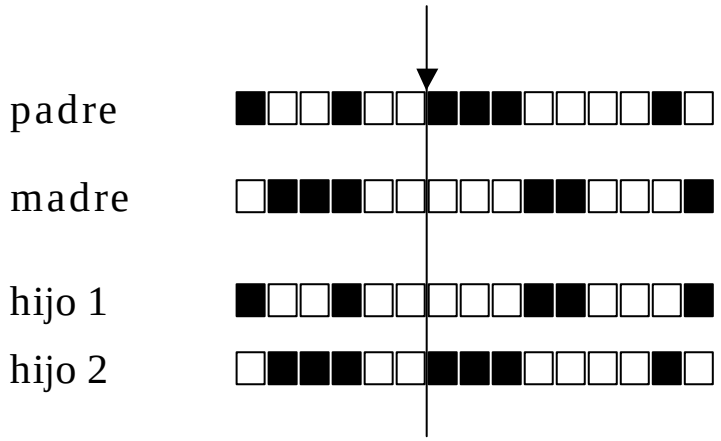
- Cambio de generación:
  - Manteniendo el tamaño de la población
    - Reemplazar padres por hijos
    - Reemplazar un par de individuos elegidos aleatoriamente por los hijos
    - Otros
  - Aumentando el tamaño de la población
    - Crear una población temporal formada por los padres y los hijos y seleccionar de ahí los mejores para formar la nueva generación
    - Dados n padres generar m hijos ($m>n$) y de ahí seleccionar los n mejores.
- Selección:
  - Asignar a cada individuo una probabilidad de ser elegido definida como

$$f(x_i) \Big/ \sum_{\text{población}} f(x_j)$$

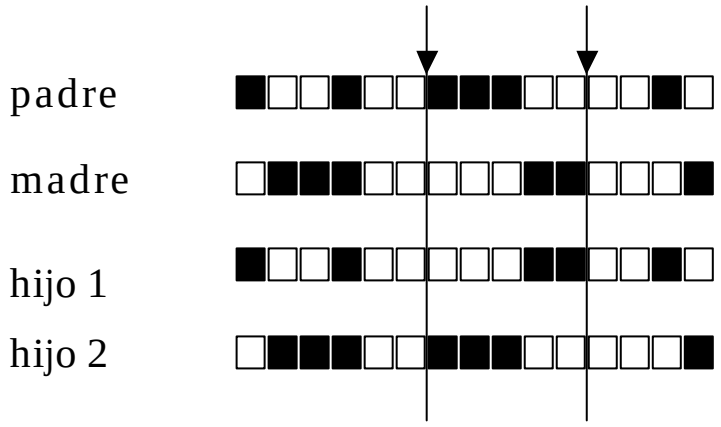
    donde f puede ser
    - la función de calidad (quizás escalada o centrada)
    - la posición de la solución si se ordenan según su calidad

Variantes del esquema básico

- Cruce
  - De un punto: seleccionar aleatoriamente un punto en el cromosoma e intercambiar el final de cada cromosoma a partir de dicho punto.

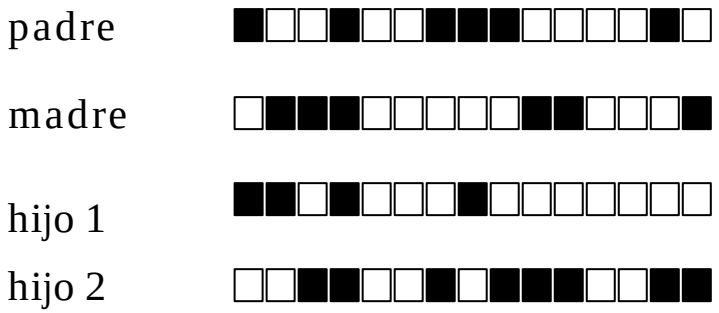


  - De dos puntos:



Variantes del algoritmo básico

– Uniforme: cada gen se hereda de un padre elegido aleatoriamente.



- Mutación

  Evita que solo se considere un subconjunto de las posibles soluciones

¿Por qué funciona?

- Un esquema es el conjunto de cromosomas que siguen un patrón.

  Ejemplo: 00*1*0={000100, 000110, 001100, 001110}

- Teorema del esquema:

  Relaciona la calidad de los miembros de un esquema en una generación con el número esperado de miembros en la siguiente generación.

  $$<Ns(g+1)>= Ns(g)* ms(g) / m(g)$$

  - $Ns(g)$ es el número de elementos del esquema s en la generación g
  - $m(g)$ la calidad media de los cromosomas en la generación g
  - $ms(g)$ una estimación de la calidad media de los cromosomas de la generación s que pertenecen al esquema s
  - $<x>$ es el valor esperado

4 Observaciones finales

- Observaciones:
  - La evolución está dirigida por la calidad relativa
  - Existe un paralelismo implícito, las operaciones se hacen implícitamente sobre todo un esquema.
  - Encontrar un equilibrio entre explotación / exploración

- Los algoritmos genéticos funcionan mejor cuando:
  - Las soluciones potenciales pueden representarse de forma que quede explícita la composición
  - Existen operadores para mutar y recombinar estas representaciones

- Los algoritmos genéticos funcionan peor cuando:
  - La representación no recoge las características de las soluciones
  - Los operadores no generan candidatos "interesantes"