Algoritmos genéticos

- **Introducción**
- Esquema básico
- El problema de la mochila
- Asignación de recursos
- El problema del viajante
- Variantes del esquema básico
- ¿Por qué funciona?
- Observaciones finales

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”.

Algoritmos genéticos. Introducción

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

Algoritmos genéticos

- Introducción
- **Esquema básico**
- El problema de la mochila
- Asignación de recursos
- El problema del viajante
- Variantes del esquema básico
- ¿Por qué funciona?
- Observaciones finales

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
```

Algoritmos genéticos. Esquema básico

Problema:

Maximizar $f(x) = x^2$ con x entero entre 0 y 31

Algoritmos genéticos. Esquema básico

Codificación:

Representación en binario:

Algoritmos genéticos. Esquema básico

Población inicial generada aleatoriamente

01101
11000
01000
10011

(tamaño 4)

Algoritmos genéticos. Esquema básico

Función de calidad:

$$f(x) = x^2$$

Cromosoma	x	f(x)
01101	13	169
11000	24	576
01000	8	64
10011	19	361

Algoritmos genéticos. Esquema básico

Población intermedia: cada individuo puede ser elegido con una probabilidad proporcional a su 'calidad'.

Cromosoma	x	f(x)	P(f(x))	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

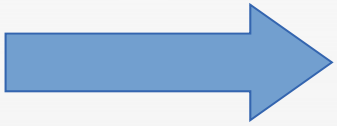
Algoritmos genéticos. Esquema básico

Combinación: parejas de la población intermedia de manera aleatoria

Cruce: se elige un punto intermedio y se intercambian los genes de los 'padres'.

1 1 0 | 0 0

1 0 0 | 1 1



1 1 0 1 1

1 0 0 0 0

Algoritmos genéticos. Esquema básico

Y además: **Mutación** (cambio aleatorio de algún bit elegido al azar con probabilidad pequeña)

Cromosoma	x	f(x)
110 11	27	729
100 00	16	259
011 00	12	144
110 01	25	625

Algoritmos genéticos. Esquema básico

Codificación:

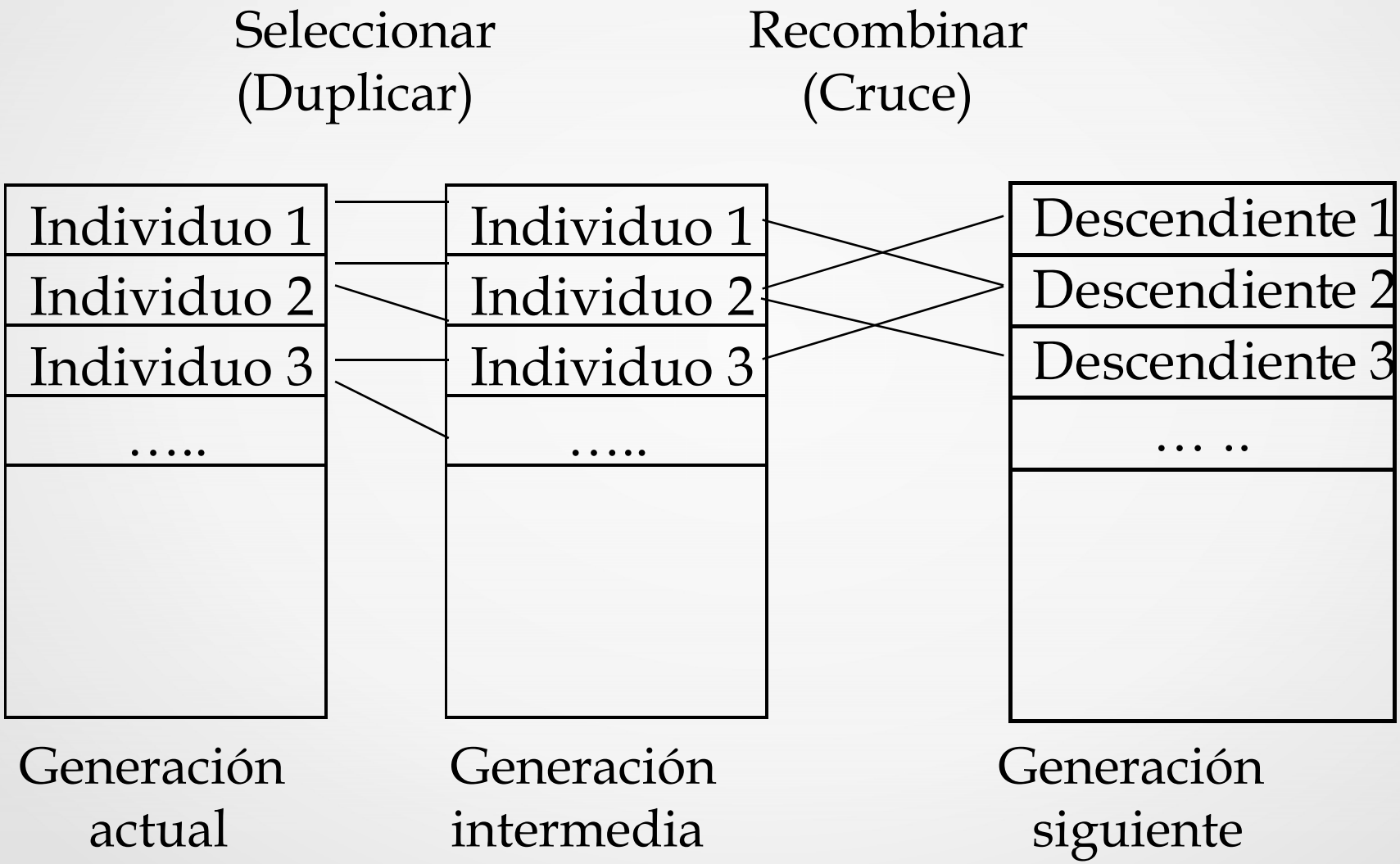
- Utilizar cadenas de bits para representar las soluciones
- Los bits pueden codificar números enteros, reales, conjuntos, …
- Ventaja: los operadores de cruce y mutación son simples.
- Inconveniente: no siempre resulta “natural”.

Algoritmos genéticos. Esquema básico

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.

Algoritmos genéticos. Esquema básico



Algoritmos genéticos. Esquema básico

- Operador de cruce de un punto
- Mutación:

  Hay una pequeña probabilidad de cambio de un bit.

Algoritmos genéticos

- Introducción
- Esquema básico
- **El problema de la mochila**
- Asignación de recursos
- El problema del viajante
- Variantes del esquema básico
- ¿Por qué funciona?
- Observaciones finales

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.

El problema de la Mochila

$$maximizar \sum_{1 \leq i \leq n} b_i.x_i$$

$$sujeto\ a : \sum_{1 \leq i \leq n} p_i.x_i \leq C$$

$$con : x_i \in \{0, 1\}, b_i > 0, p_i > 0, 1 \leq i \leq n$$

El problema de la Mochila

Representación:

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

(Observar que no se garantiza factibilidad)

Inicialización: Generar secuencias de ceros y unos

Operador de **cruce** de un punto

El problema de la Mochila

Función de calidad:

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

Se penaliza la no factibilidad (factibles serán mejores)

El problema de la Mochila

Se puede hacer de más formas:

"Una lista con los elementos que pertenecen a la mochila"

Problema: ¿qué operador de cruce utilizamos?

(2, 3, 4)
(1, 4, 5)

(2, 3, 5)
(1, 4, 4)

¿Eliminar los elementos duplicados?

Algoritmos genéticos

- Introducción
- Esquema básico
- El problema de la mochila
- **Asignación de recursos**
- El problema del viajante
- Variantes del esquema básico
- ¿Por qué funciona?
- Observaciones finales

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 $j=1,2,\ldots,m$

y para ser **óptima** debe **maximizar**: $B(\boldsymbol{x}) = \Sigma_{i=1} x_i b_i$

Asignación de recursos

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=100, la tasa de mutación pm=1/n, y la tasa de recombinación pc=0’6.

Asignación de recursos

Resultados obtenidos (100 ejecuciones / 6 casos)

n=15, m=10		n=20, m=10		n=28, m=10		n=50, m=5		n=60, m=30		n=105, m=2	
$f_{5\,10^3}(x)$	N	$f_{10^4}(x)$	N	$f_{5\,10^4}(x)$	N	$f_{10^5}(x)$	N	$f_{10^5}(x)$	N	$f_{2\,10^5}(x)$	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	

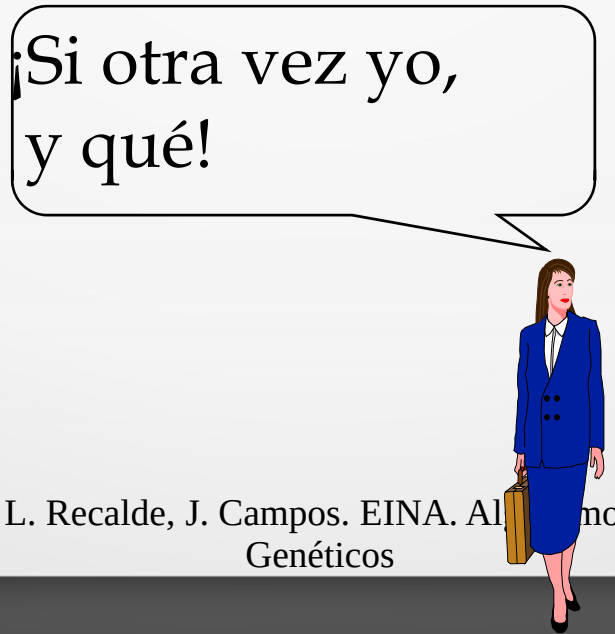
Algoritmos genéticos

- Introducción
- Esquema básico
- El problema de la mochila
- Asignación de recursos
- **El problema del viajante**
- Variantes del esquema básico
- ¿Por qué funciona?
- Observaciones finales

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."



El problema del viajante

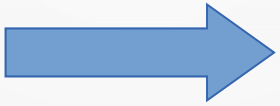
Codificación: vector siguiendo recorrido:

[3 , 2 , 5, 4, 1]

Cruce: (de un punto)

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

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



[3, 2, 1, 4, 2]

[3, 5, 5, 4, 1]

Ciudades repetidas // No se visitan todas

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

El problema del viajante

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] 

Cruce:

Cualquiera de los habituales, de un punto por ejemplo.

Algoritmos genéticos

- Introducción
- Esquema básico
- El problema de la mochila
- Asignación de recursos
- El problema del viajante
- **Variantes del esquema básico**
- ¿Por qué funciona?
- Observaciones finales

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

Variantes del esquema básico

Tener en cuenta

Factibilidad: los cromosomas pueden codificar soluciones **no factibles** del problema.

Solución: penalizar en la función de calidad

descartar

reparar

Variantes del esquema básico

Tener en cuenta:

Unicidad de la codificación:

- Uno a uno
- Uno a *N*
- *N* a uno

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

Variantes del esquema básico

Cambio de generación:

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.

Variantes del esquema básico

Selección:

Asignar a cada individuo una probabilidad de ser elegido definida como:

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

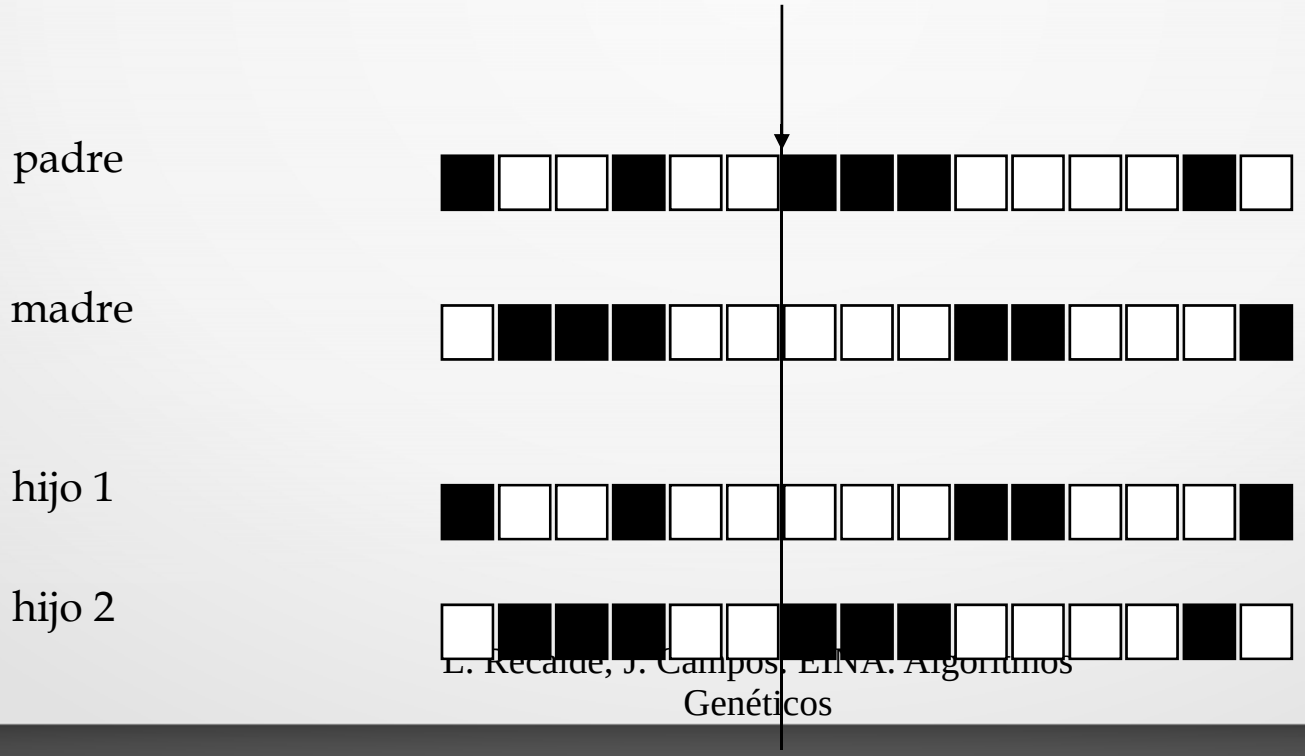
donde f puede ser

- La función de calidad (quizás escalada o centrada)
- La posición de la solución si se ordenan según su calidad

Variantes del esquema básico

Cruce

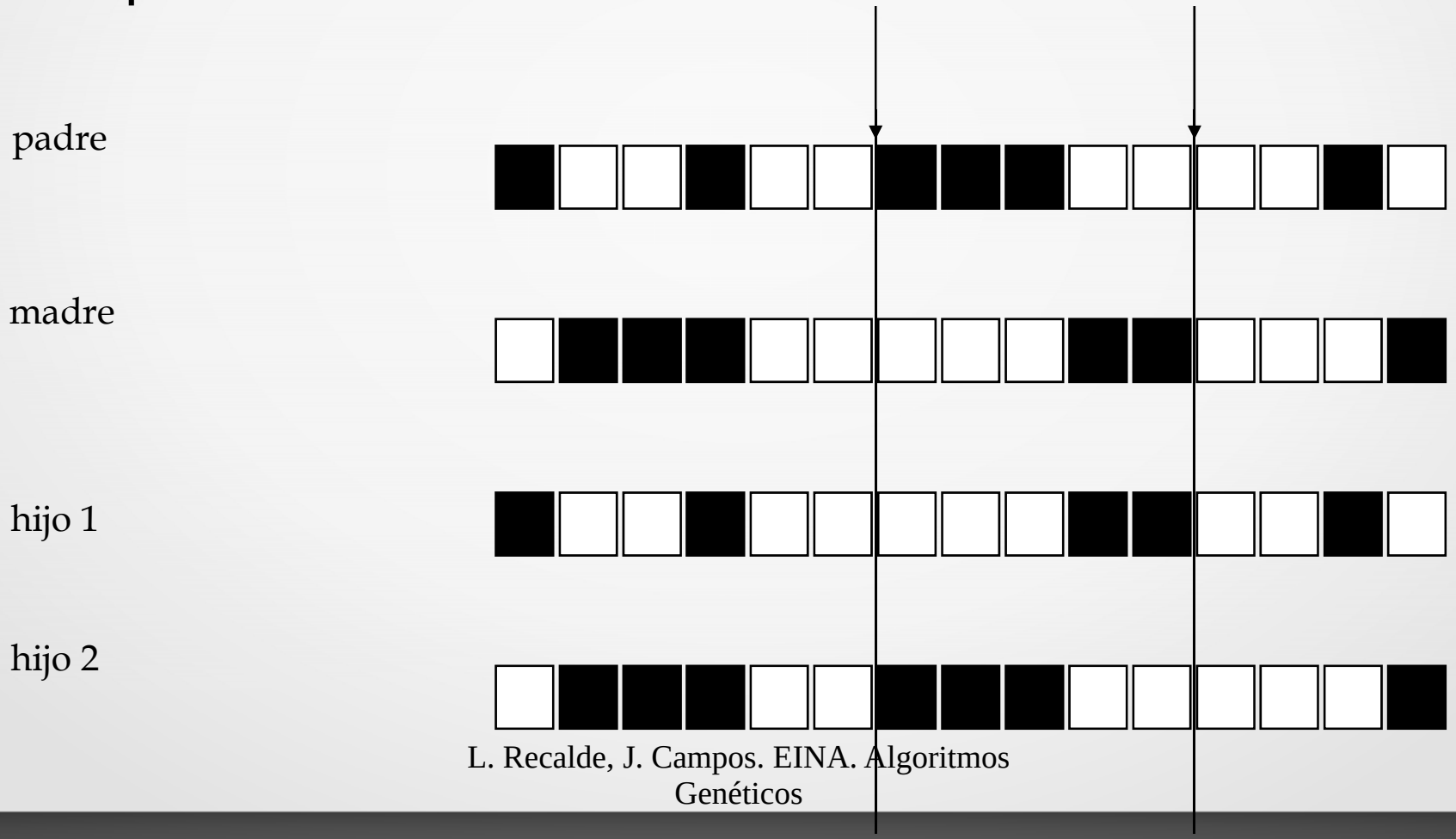
De un punto: seleccionar **aleatoriamente** un punto en el cromosoma e intercambiar el final de cada cromosoma a partir de dicho punto.



Variantes del esquema básico

Cruce

De dos puntos.



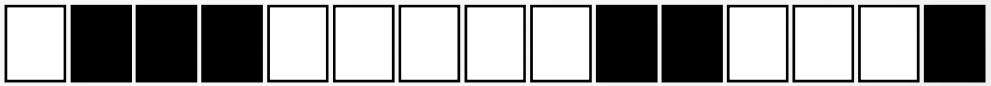
Variantes del esquema básico

Cruce

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

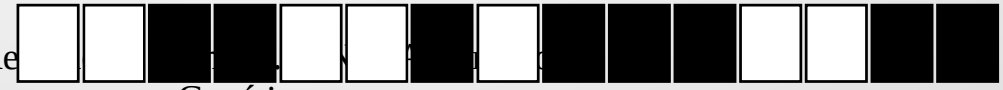
padre

madre



hijo 1

hijo 2



Variantes del esquema básico

Mutación

Evita que solo se considere un subconjunto de las posibles soluciones

(Diversidad)

Algoritmos genéticos

- Introducción
- Esquema básico
- El problema de la mochila
- Asignación de recursos
- El problema del viajante
- Variantes del esquema básico
- **¿Por qué funciona?**
- Observaciones finales

¿Por qué funciona?

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

Ejemplo: 00*1*0=

{000100,

000110,

001100,

001110 }

¿Por qué funciona?

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.

$$\langle Ns(g+1) \rangle = Ns(g) * ms(g)/m(g)$$

¿Por qué funciona?

Teorema del esquema:

$$\langle Ns(g+1) \rangle = Ns(g) * ms(g)/m(g)$$

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

¿Por qué funciona? (crítica EM)

Teorema del esquema:

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

- El teorema del esquema dice que cada generación es mayor (¿mejor?) que la anterior
- No da ninguna idea de que vayamos a alcanzar el óptimo en un **número razonable de generaciones**
- Tampoco da ninguna prueba de que no estemos acercándonos a un **óptimo local** (fijaros que habla de calidad media)

Algoritmos genéticos

- Introducción
- Esquema básico
- El problema de la mochila
- Asignación de recursos
- El problema del viajante
- Variantes del esquema básico
- ¿Por qué funciona?
- **Observaciones finales**

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

Observaciones finales

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

Observaciones finales

Los algoritmos genéticos funcionan **peor** cuando:

- La representación no recoge las características de las soluciones
- Los operadores no generan candidatos “interesantes”

Observaciones finales

. Tener en cuenta

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)

Mi resumen (EM)

- **Las heurísticas** nos proporcionan buenas intuiciones sobre cómo recorrer un espacio de soluciones.
- Las heurísticas no tienen en general ninguna garantía.
- El ciclo de trabajo es: mantener un buen conjunto de casos (benchmarks) y probar la heurística sobre ellos; actualizar el conjunto de casos.
- En el caso de los algoritmos genéticos tiene **cierta dificultad seleccionar**: representación de las soluciones, operador de cruce, mutación.
- El teorema del esquema proporciona **cierta justificación** (asintótica, es decir, en el infinito) de la heurística.