

Algoritmos genéticos

Javier Campos



Dpto. Informática e Ingeniería de Sistemas
Universidad de Zaragoza



Jornadas de Primavera de Púlsar
Zaragoza
26-28 de abril de 2005

Un poco de historia

El porqué de esta charla...

1º: porque la organización quería que se hablase de esto

2º, mi 'tesis':

"Hay razones poderosas por las que todo alumno del CPS que haya estudiado en el edificio Ada Byron debería conocer los algoritmos genéticos."



Un poco de historia

◆ Edificio Ada Byron (1999-)



El nombre del edificio será por algo...



Un poco de historia

◆ Ada Byron, condesa de Lovelace (1815-1852)

Es la autora del que se considera el primer programa informático de la historia (programa para el cálculo de los números de Bernoulli)



1st Series.- Let $n=1$, and calculate (8.) for this value of n . The result is B_1 .

2nd Series.- Let $n=2$. Calculate (8.) for this value of n , substituting the value of B_1 just obtained. The result is B_3 .

3rd Series.- Let $n=3$. Calculate (8.) for this value of n , substituting the values of B_1 , B_3 before obtained. The result is B_5 . And so on, to any extent.

Un poco de historia



Edificio toma su nombre de Ada Byron



En un concurso de ideas realizado para dar nombre a los edificios de este campus, se propone el nombre de Ada Byron, primera programadora de la historia, para el edificio que alberga al departamento de informática y las aulas en las que mayoritariamente se imparte la titulación de ingeniería en informática.

Un poco de historia

◆ Charles Babbage (1791-1871)

Autor de la “máquina analítica”
(es “la abuela” de todos los
computadores actuales)

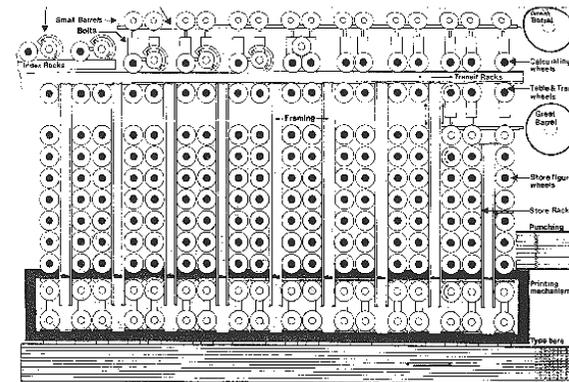
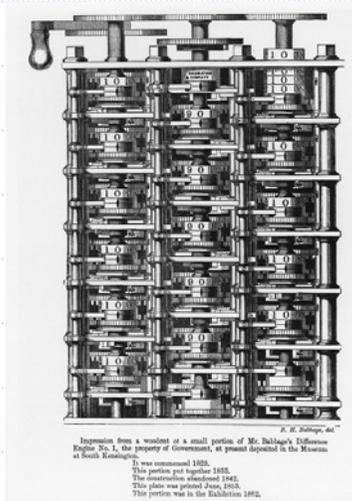


Fig. 2. Plan of Analytical Engine with grid layout, 1848. Rodzwa.



Un poco de historia



Byron diseñó programas para Babbage

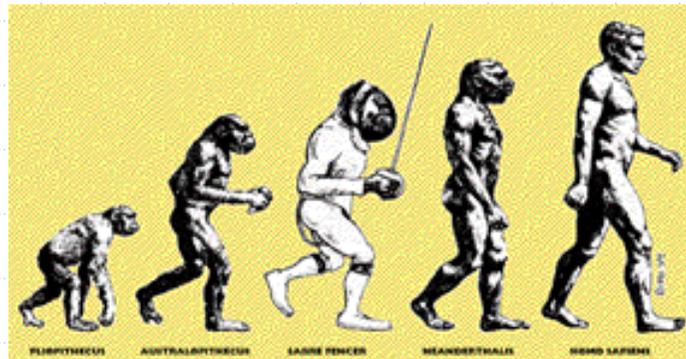
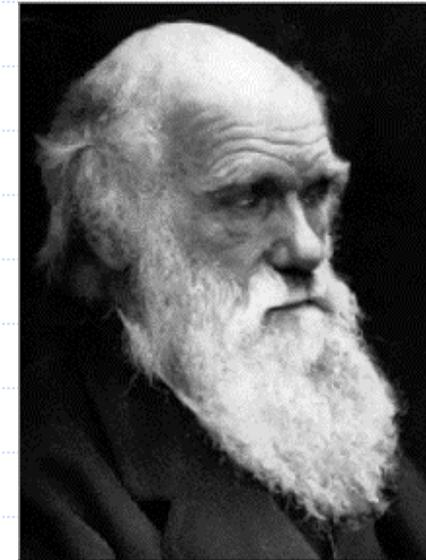


Ada Byron traduce al inglés las notas de una conferencia de Babbage en Turín y añade unas anotaciones que triplican la extensión del documento original de la conferencia. En esas notas se incluye el programa para calcular los números de Bernoulli con la máquina analítica. Desde ese momento colaborará con Babbage.

Un poco de historia

◆ Charles Darwin (1809-1882)

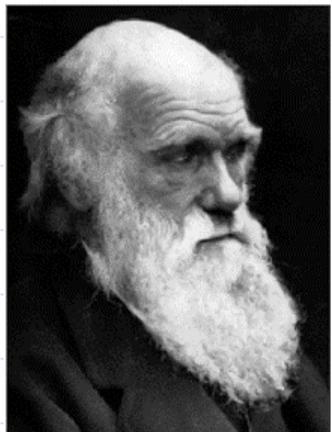
Naturalista, autor de la teoría de la Selección Natural para el origen de las especies



Un poco de historia



Babbage inspiró la teoría de Darwin

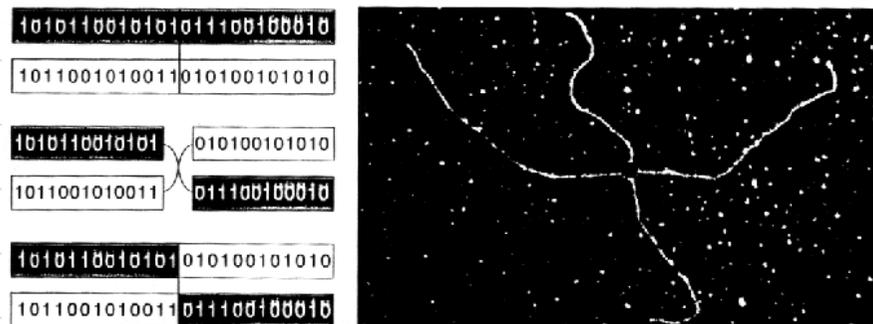


Babbage convenció a Darwin de que todo en la naturaleza funciona según unas reglas. Esa idea llevó a Darwin a la búsqueda de las reglas que rigen la evolución de las especies. Finalmente Darwin modificó algo las ideas de Babbage añadiendo a las reglas la posibilidad de la selección aleatoria para explicar los cambios de las especies.

Un poco de historia

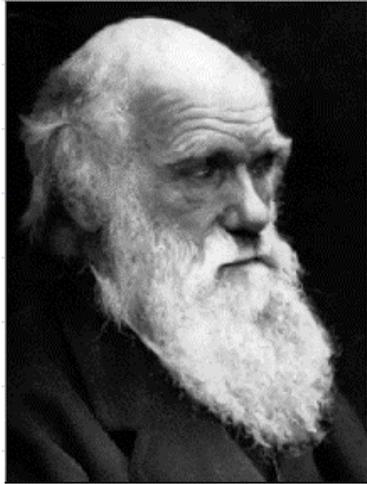
◆ John Holland (1929-)

Profesor de Ingeniería Eléctrica e Informática y profesor de Psicología de la Universidad de Michigan, autor de los algoritmos genéticos



CROSSOVER is the fundamental mechanism of genetic rearrangement for both real organisms and genetic algorithms. Chromosomes line up and then swap the portions of their genetic code beyond the crossover point.

Un poco de historia

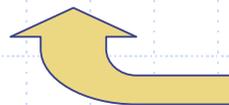
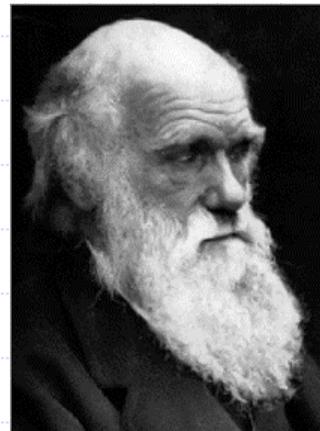
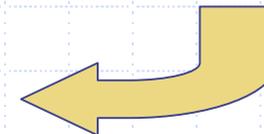
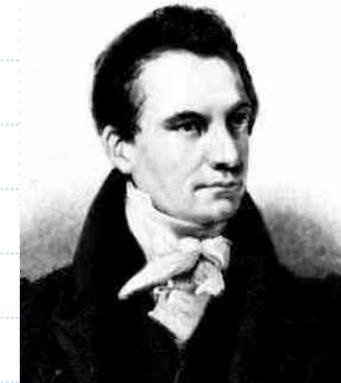
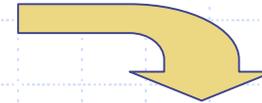
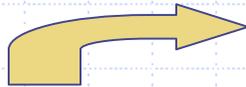


Darwin inspiró los algoritmos genéticos



Holland leyó el libro *La teoría genética de la selección natural* del biólogo evolucionista R.A. Fisher y aprendió que la evolución era una forma de adaptación más potente que el simple aprendizaje, y tomó la decisión de aplicar estas ideas para desarrollar programas adaptativos para resolver problemas.

Un poco de historia



“Hay razones poderosas por las que todo alumno del CPS que haya estudiado en el edificio Ada Byron debería conocer los algoritmos genéticos.”



Conceptos básicos

- ◆ Inventados por John Holland en los 70.
- ◆ Inspirados en el modelo de evolución biológica.
- ◆ Utilizan el principio de selección natural para resolver problemas de optimización combinatorios.
- ◆ Son una heurística probabilista de búsqueda multidireccional (según algunos autores, son una "meta-heurística evolutiva").



Conceptos básicos

◆ Terminología:

- **Población de individuos:** un conjunto de soluciones potenciales (o factibles) del espacio de búsqueda.
- **Evolución:** la población se va modificando iterativamente mediante los mecanismos de...
 - ◆ **selección natural:** favorece que los individuos con mayor *calidad* (es decir, mejor valor de la función objetivo) pervivan y se reproduzcan,
 - ◆ **recombinación:** permite la mezcla de varios individuos para crear nuevos individuos de la población,
 - ◆ **mutación:** permite introducir innovaciones en la población de forma aleatoria.



El esquema algorítmico

Un esquema algorítmico es un método genérico de resolución, válido para una familia grande de problemas.

Otros ejemplos de esquemas algorítmicos:

- Divide y vencerás
- Búsqueda con retroceso
- Algoritmos voraces
- Programación dinámica

...

```
algoritmo genético
```

```
principio
```

```
t:=0;
```

```
inicializa(P(t));
```

```
evalúa(P(t));
```

```
mientras que not termina hacer
```

```
t:=t+1;
```

```
P(t):=selecciona(P(t-1));
```

```
recombina(P(t));
```

```
muta(P(t));
```

```
evalúa(P(t))
```

```
fmq;
```

```
fin
```



El esquema algorítmico

Cada individuo de $P(t)$
se codifica con n bits:

$\mathbf{x} = (x_1, x_2, \dots, x_n)$,
con $x_i \in \{0, 1\}$.

algoritmo genético

principio

$t := 0$;

inicializa($P(t)$);

evalúa($P(t)$);

mientras que not termina hacer

$t := t + 1$;

$P(t) := \text{selecciona}(P(t-1))$;

recombina($P(t)$);

muta($P(t)$);

evalúa($P(t)$)

fmq;

fin



El esquema algorítmico

La función evalúa
calcula el valor de una
función de calidad para
todos los miembros de
la población $P(t)$.



La función de calidad es
la única parte específica
de cada problema en
este esquema.

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



El esquema algorítmico

La selección dirige el proceso de búsqueda, eligiendo individuos mejores para ser copiados en la siguiente generación, donde serán sujetos a recombinación y mutación.

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



El esquema algorítmico

Debe definirse una función de **calidad** para cada individuo \mathbf{x}_j : $f(\mathbf{x}_j)$, que estará relacionada con la función objetivo (a maximizar).

Entonces, el proceso de selección determina μ (=50, p.e.) individuos para la nueva población, de acuerdo con la distribución de probabilidad (de supervivencia):

$$p_s(\mathbf{x}_j) = f(\mathbf{x}_j) / \sum_{j=1..μ} f(\mathbf{x}_j).$$

```
algoritmo genético
```

```
principio
```

```
t:=0;
```

```
inicializa(P(t));
```

```
evalúa(P(t));
```

```
mientras que not termina hacer
```

```
t:=t+1;
```

```
P(t):=selecciona(P(t-1));
```

```
recombina(P(t));
```

```
muta(P(t));
```

```
evalúa(P(t))
```

```
fmq;
```

```
fin
```



El esquema algorítmico

La recombinación se define con los parámetros:

- probabilidad de que un individuo sea combinado:
 p_c (=0'6, p.e.),
- la elección de 2 individuos (padres) de la población:
 $\mathbf{x} = (x_1, x_2, \dots, x_n)$
 $\mathbf{y} = (y_1, y_2, \dots, y_n)$,
- la elección aleatoria de un punto de cruce
 $\chi \in \{1, \dots, n-1\}$,

y produce los dos nuevos siguientes individuos:

$$\mathbf{x}' = (x_1, \dots, x_{\chi-1}, x_{\chi}, y_{\chi+1}, \dots, y_n)$$
$$\mathbf{y}' = (y_1, \dots, y_{\chi-1}, y_{\chi}, x_{\chi+1}, \dots, x_n)$$

algoritmo genético

principio

$t := 0;$

inicializa($P(t)$);

evalúa($P(t)$);

mientras que not termina hacer

$t := t + 1;$

$P(t) := \text{selecciona}(P(t-1));$

recombina($P(t)$);

muta($P(t)$);

evalúa($P(t)$)

fmq;

fin



El esquema algorítmico

La mutación es una operación probabilista: cambia aleatoriamente (p.e., con probabilidad $1/n$) el valor de un bit de la representación de un individuo.

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



Un ejemplo: asignación de recursos

- ◆ Un problema de asignación de recursos
 - Hay m recursos de capacidades c_1, c_2, \dots, 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.
 - Es un problema *NP*-completo y muy conocido porque:
 - ◆ desde un punto de vista teórico es un subproblema de otros problemas muy interesantes y más complejos, y
 - ◆ desde un punto de vista práctico sirve para modelar problemas reales muy diversos.



Un ejemplo: asignación de recursos

- Representación de un individuo: $\mathbf{x} = (x_1, x_2, \dots, x_n)$, con $x_i \in \{0, 1\}$ ($x_i = 1$ significa ejecutar la tarea i -ésima)
 - ◆ para ser factible debe verificar: $\sum_{i=1..n} w_{ij} x_i \leq c_j$ para $j=1, 2, \dots, m$
 - ◆ y para ser óptima debe maximizar: $B(\mathbf{x}) = \sum_{i=1..n} x_i b_i$
- La función de calidad elegida es la siguiente:
 $f(\mathbf{x}) = \sum_{i=1..n} b_i x_i - s \max\{b_j\}$
donde $s = |\{j \mid \sum_{i=1..n} w_{ij} x_i > c_j\}|$, es decir, el número de recursos agotados.
- El tamaño de la población elegido es $\mu=50$, la tasa de mutación $p_m=1/n$, y la tasa de recombinación $p_c=0'6$.



Un ejemplo: asignación de recursos

- Resultados obtenidos tras 100 ejecuciones de 7 casos distintos:

$n=15, m=10$		$n=20, m=10$		$n=28, m=10$		$n=39, m=5$		$n=50, m=5$		$n=60, m=30$		$n=105, m=2$	
$f_{5 \cdot 10^3}(x)$	N	$f_{10^4}(x)$	N	$f_{5 \cdot 10^4}(x)$	N	$f_{10^5}(x)$	N	$f_{10^5}(x)$	N	$f_{10^5}(x)$	N	$f_{2 \cdot 10^5}(x)$	N
4015	83	6120	33	12400	33	10618	4	16537	1	7772	5	1095445	—
4005	16	6110	20	12390	30	10605	1	16524	1	7761	4	1095382	10
3955	1	6100	29	12380	10	10604	8	16519	2	7758	11	1095357	3
		6090	11	12370	1	10601	1	16518	5	7741	7	1095266	1
		6060	3	12360	19	10588	5	16499	1	7739	1	1095264	9
		6050	1	12330	5	10585	5	16497	1	7738	3	1095206	3
		6040	3	11960	1	10582	1	16494	1	7725	1	1095157	2
				11950	1	10581	2	16473	1	7719	1	1095081	1
						10570	6	16472	1	7715	1	1095065	2
						10568	2	16467	1	7711	2	1095035	8
						10561	1	16463	1	7706	1	1094965	1
$f=4012'7$		$f=6102'3$		$f=12374'7$		$f=10536'9$		$f=16378$		$f=7626$		$f=1093897$	



Un ejemplo: asignación de recursos

- Los valores de la primera fila corresponden con los valores óptimos “globales”.
- Los valores de la última fila son los valores óptimos medios calculados en las 100 ejecuciones.
- Por ejemplo, 33 ejecuciones del problema 20:10 dieron la solución óptima 6120, mientras que 20 ejecuciones dieron un valor 6110 (sólo aparecen en la tabla los 11 mejores resultados).
- El subíndice en $f_t(\mathbf{x})$ representa el n° total de individuos procesados (es decir, t es el producto del n° de individuos por generación por el n° de generaciones por ejecución).
- Sólo un porcentaje extremadamente pequeño de los 2^n puntos del espacio de búsqueda es procesado por el algoritmo genético.
 - ◆ Por ejemplo, para el problema 28:10, $t=5 \cdot 10^4$ y por tanto sólo un 0'018% del espacio de búsqueda se ha explorado.
 - ◆ Para el problema 105:2, se reduce a un $4'9 \cdot 10^{-25}\%$.



¿Por qué funciona?

- ◆ Hay intentos de explicación de dos tipos:
 - Teoremas matemáticos (teorema del esquema)
 - Leyes empíricas (estudios estadísticos de las poblaciones genéticas)
- ◆ En mi opinión, ninguna de ambas aproximaciones ha dado respuesta convincente... de hecho, ¿funciona?

Diccionario de la R.A.E.

Heurística: En algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc.



Otro ejemplo: el problema del viajante

◆ El problema del viajante

- Encontrar un recorrido de longitud mínima para un viajante que tiene que visitar n ciudades y volver al punto de partida, conocida la distancia existente entre cada dos ciudades
- El problema es NP-completo (no se conoce solución de coste en tiempo polinomial en n)
- Es probablemente el problema de optimización combinatoria más estudiado



Otro ejemplo: el problema del viajante

◆ Solución directa:

- Fuerza bruta, calcular las permutaciones y elegir la mejor (coste $n!$)
- Si podemos calcular la longitud de cada recorrido en un microsegundo...
 - ◆ $10! = 3.628.800 \rightarrow 3,6$ segundos para resolverlo
 - ◆ $20! \approx 2,43 \times 10^{18} \rightarrow 77.146$ años para resolverlo

◆ Algoritmos genéticos:

- no encuentran soluciones exactas,
- pero permiten encontrar aproximaciones suficientemente buenas y
- se pueden aplicar a conjuntos de ciudades muy grandes (millones) con tiempos de ejecución razonables en un supercomputador (semanas o meses).



Otro ejemplo: el problema del viajante

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

$[3,2,5,4,1]$ \rightarrow $3 \rightarrow 2 \rightarrow 5 \rightarrow 4 \rightarrow 1$

◆ Cruce: de un punto

$[3,2,5,4,1]$ \rightarrow $[3,2,1,4,2]$
 $[3,5,1,4,2]$ \rightarrow $[3,5,5,4,1]$

- Pueden aparecer ciudades repetidas
- No siempre visitamos todas



Otro ejemplo: 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



Otro ejemplo: el problema del viajante

- ◆ Caso particular: 20 ciudades en un círculo
($20! \approx 2,43 \times 10^{18} \rightarrow 77.146$ años)
- ◆ Otro: 29 ciudades de Baviera
($29! \approx 8,84 \times 10^{30} \rightarrow 2,8 \times 10^{17}$ años)
- ◆ Otro: 48 ciudades de USA
($48! \approx 1,24 \times 10^{61} \rightarrow 3,9 \times 10^{47}$ años)

GA Playground - Genetic Algorithms Toolkit
(<http://www.aridolan.com/ga/gaa/gaa.html>)



Para saber más...

- ◆ The Genetic Algorithms Archive
(<http://www.aic.nrl.navy.mil/galist/>)

