DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Algoritmos Genéticos

Fernando Berzal, berzal@acm.org

Algoritmos Genéticos

La estructura básica de un algoritmo genético

Componentes de un algoritmo genético

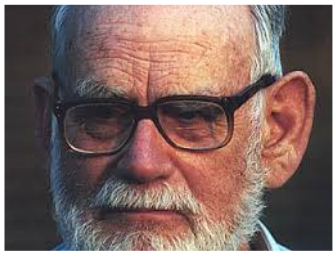
- Técnicas de representación
- Operadores de cruce
- Operadores de mutación
- Mecanismos de selección

Los parámetros de un algoritmo genético

Otras representaciones

- Representación entera y permutaciones
- Representación real
- Representación en árbol (programación genética)

Algoritmos genéticos



A finales de los 1950s y principios de los 1960s, el biólogo inglés Alex S. Fraser (1923-2002) publicó una serie de trabajos sobre la evolución de sistemas biológicos en una computadora digital, sirviendo de inspiración para los algoritmos genéticos:

Fraser, A. S., "**Simulation of genetic systems by automatic digital computers**. I. Introduction," Aust. J. Biol. Sci., vol. 10, pp. 484–491, 1957.

Algoritmos genéticos

Hans-Joachin Bremermann (1926-1996) fue el primero en ver la evolución como un proceso de optimización, además de realizar una de las primeras simulaciones con cadenas binarias que se procesaban por medio de reproducción, selección y mutación (predecesor de los algoritmos genéticos).

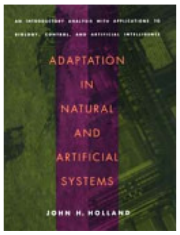
Chromosome 1	1 1 0 1 0 0 0 1 1 0 1 0
Chromosome 2	0 1 1 1 1 1 1 1 1 1 0 0

Bremermann, H.J. (1962): "**Optimization through evolution and recombination**". In: Self-Organizing systems 1962, edited M.C. Yovitts et al., Spartan Books, Washington, D.C. pp. 93–106.

Algoritmos genéticos

John Henry Holland (1929-) desarrolla a principios de los 1960s los "planes reproductivos" y "adaptativos" en un intento por hacer que las computadoras aprendan imitando el proceso de la evolución.

John H. Holland (1962):
"**Outline for a logical theory of adaptive systems**",
JACM 9(3):297–314. DOI 10.1145/321127.321128



John H. Holland (1975):
"**Adaptation in Natural and Artificial Systems**".
The University of Michigan Press, 1975
ISBN 0472084607

Algoritmos genéticos

Rasgos fenotípicos

- Diferencias físicas y de comportamiento que afectan a la respuesta de un individuo a su entorno.
- En la naturaleza, parcialmente determinadas por la herencia, parcialmente derivados por factores relacionados con el desarrollo de individuo.
- Si los rasgos fenotípicos incrementan las posibilidades de reproducción y pueden heredarse, tenderán a incrementarse en subsecuentes generaciones… y servir de base a nuevas combinaciones de rasgos.

Algoritmos genéticos

Evolución natural

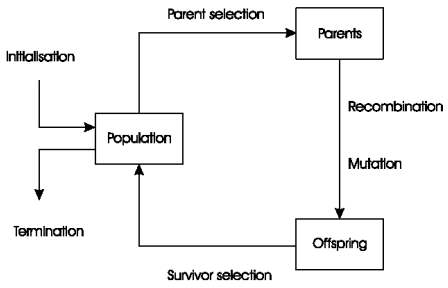
- La población consiste en un conjunto de individuos.
- Los individuos son seleccionados (combinaciones de rasgos mejor adaptados tienden a incrementarse en la población).
- La población evoluciona (la selección natural se combina con la variación introducida por cambios aleatorios [una fuente de diversidad]).

Algoritmos genéticos

- Se hace evolucionar una población de individuos (cada uno de los cuales representa una posible solución).
- La población se somete a acciones aleatorias semejantes a las de la evolución biológica (mutaciones y recombinaciones genéticas).
- Los individuos se seleccionan de acuerdo con una función de adaptación en función del cual se decide qué individuos sobreviven (los más adaptados) y cuáles son descartados (los menos aptos).

Algoritmos genéticos

Características esenciales
(de cualquier algoritmo evolutivo)



- Algoritmos de tipo "generar y probar" [generate & test].
- Algoritmos estocásticos basados en poblaciones (comportamiento no determinista).
- Operadores de variación (cruce y mutación) crean la diversidad necesaria.
- La selección reduce la diversidad (hacia soluciones de calidad).

Algoritmos genéticos

Características esenciales

- Inicialización (generación aleatoria de una población inicial)
- Variación (operadores de cruce y mutación)
- Evaluación (aptitud [fitness] de cada individuo)
- Selección (selección probabilística)

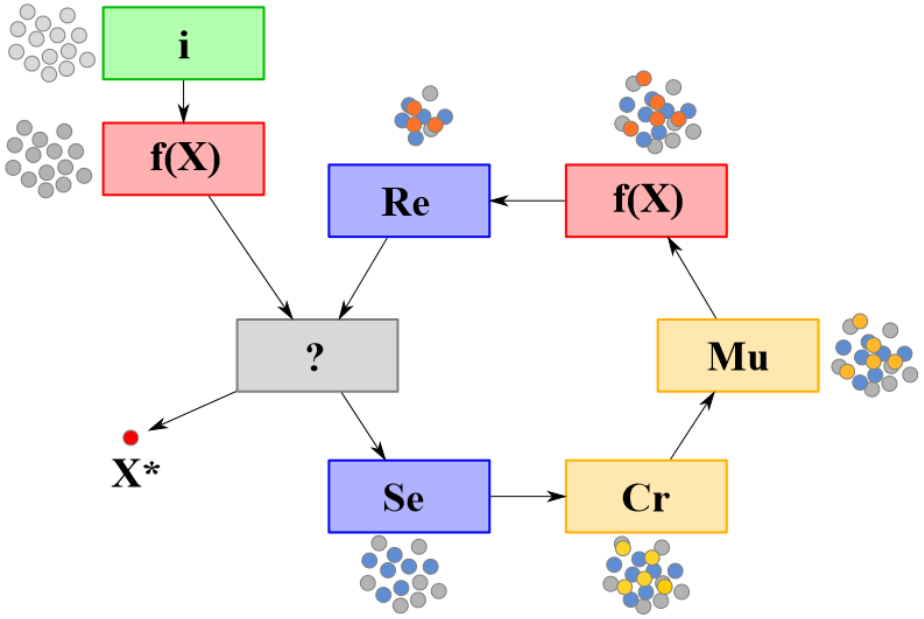
Algoritmos genéticos

Fases

- Inicialización
- Evaluación

Repetición…

- Selección
- Cruce
- Mutación
- Evaluación
- Reemplazo



Algoritmos genéticos

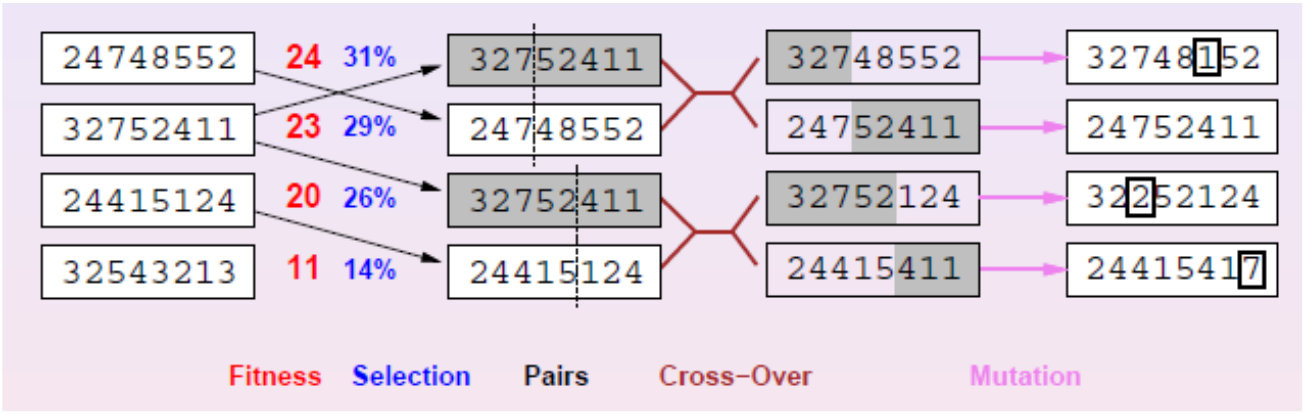
Algoritmo genético clásico

```
t ← 0
población(t) ← poblaciónInicial
EVALUAR(población(t))

while not (condición de terminación)
    t ← t + 1
    población(t) ← SELECCIONAR(población(t-1))
    población(t) ← CRUZAR(población(t))
    población(t) ← MUTAR(población(t))
    EVALUAR(población(t))

return población(t)
```

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Algoritmos genéticos

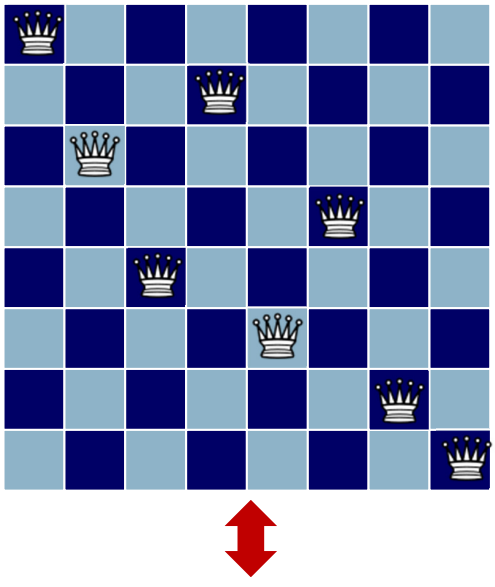
Selección, cruce & mutación



Algoritmos genéticos

Ejemplo: El problema de las N reinas

- Fenotipo:
  Configuración del tablero.



- Genotipo:
  Permutación de enteros.

1	3	5	2	6	4	7	8

Algoritmos genéticos

Ejemplo: El problema de las N reinas

- Función de evaluación:
  Número de parejas de reinas que no se atacan.

  Definición alternativa como problema de minimización:
  - Penalización de una reina =
    Número de reinas a las que ataca directamente.
  - Penalización del tablero =
    Suma de las penalizaciones de todas las reinas.
  - Fitness = - penalización del tablero.

Algoritmos genéticos

Ejemplo: El problema de las N reinas

- Operador de mutación:
  Pequeña variación en una permutación.

  Ejemplo:
  Intercambio de dos posiciones elegidas al azar.

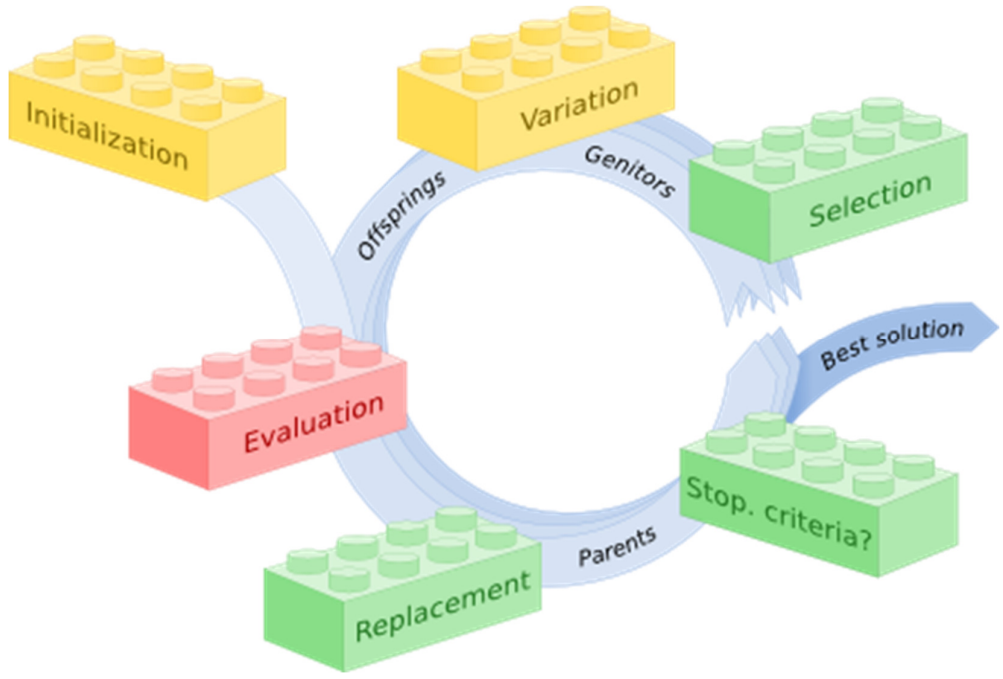
Algoritmos genéticos

Ejemplo: El problema de las N reinas

- Operador de cruce:
  Combinación de dos permutaciones
  - Elegir un punto de cruce al azar.
  - Copiar la primera parte de ambas.
  - Rellenar la segunda parte usando el otro padre: se añaden valores desde el punto de cruce, en el orden en el que aparecen y saltando los que ya están.

Algoritmos genéticos

Componentes de un algoritmo genético



http://eodev.sourceforge.net/

Algoritmos genéticos

Componentes de un algoritmo genético

- Representación de las soluciones del problema.
- Función de evaluación (fitness/aptitud).
- Mecanismo de creación de la población inicial.
- Operadores genéticos (cruce & mutación).
- Mecanismo de selección [probabilística].
- Valores para los distintos parámetros del algoritmo.

Técnicas de representación

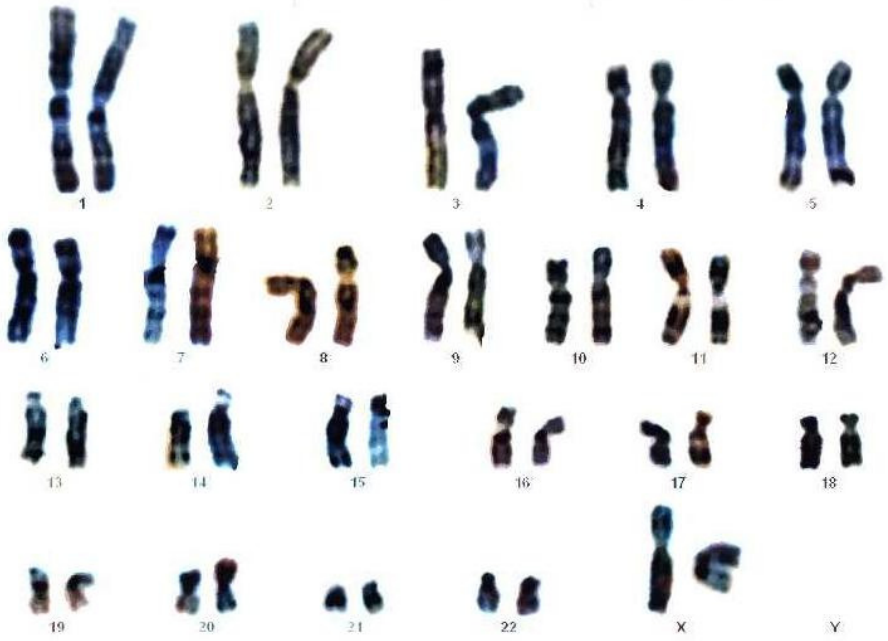
Genética básica

- La información necesaria para la "construcción" de un ser vivo viene codificada en su genoma (ADN).
- El genotipo determina el fenotipo.
- La correspondencia entre genes y rasgos fenotípicos es compleja:
  - Un gen puede afectar a muchos rasgos [pleiotropy].
  - Muchos genes pueden afectar a un rasgo [polygeny].
- Pequeños cambios en el genotipo pueden provocar pequeños cambios en el fenotipo del organismo (p.ej. altura, color de pelo, color de ojos…).

Técnicas de representación

Genética básica

Ejemplo: Homo sapiens



Técnicas de representación

Código genético

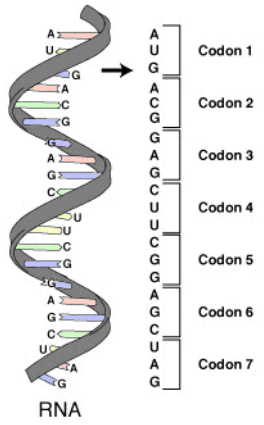
- Las proteínas son secuencias construidas a partir de 20 aminoácidos diferentes.
- El ADN es una doble hélice helicoidal formada por parejas de nucleótidos que contienen bases nitrogenadas:
  - 2 purínicas: adenina (A) y guanina (G)
  - 2 pirimidínicas: timina (T) y citosina (C)

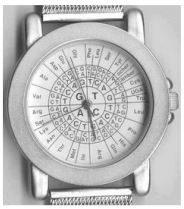
  NOTA: En el ARN, el uracilo (U) sustituye a la timina (T).
- Tripletas de nucleótidos forman codones, cada uno de los cuales codifica un aminoácido específico.
- Para todas la formas de vida en la Tierra, el código genético es el mismo…

Técnicas de representación

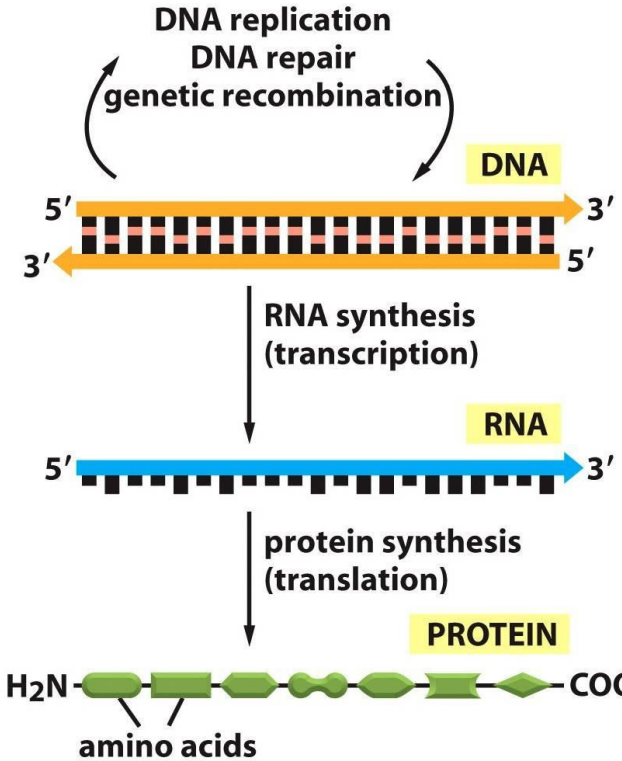
Second base of codon

First base of codon	U	C	A	G	Third base of codon
U	UUU Phenylalanine phe UUC UUA Leucine leu UUG	UCU Serine ser UCC UCA UCG	UAU Tyrosine tyr UAC UAA STOP codon UAG	UGU Cysteine cys UGC UGA STOP codon UGG Tryptonphan trp	U C A G
C	CUU Leucine leu CUC CUA CUG	CCU Proline pro CCC CCA CCG	CAU Histidine his CAC CAA Glutamine gin CAG	CGU Arginine arg CGC CGA CGG	U C A G
A	AUU Isoleucine ile AUC AUA AUG Methionine met (start codon)	ACU Threonine thr ACC ACA ACG	AAU Asparagine asn AAC AAA Lysine lys AAG	AGU Serine ser AGC AGA Arginine arg AGG	U C A G
G	GUU Valine val GUC GUA GUG	GCU Alanine ala GCC GCA GCG	GAU Aspartic acid asp GAC GAA Glutamic acid glu GAG	GGU Glycine gly GGC GGA GGG	U C A G



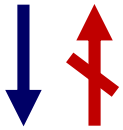


Técnicas de representación



Biología Molecular

Genotipo



Fenotipo

- Lamarck estaba equivocado

Técnicas de representación

Históricamente, diferentes tipos de algoritmos evolutivos se han asociado a diferentes técnicas de representación:

- Cadenas de bits (algoritmos genéticos)
- Vectores de números reales (estrategias de evolución)
- Máquinas de estados (programación evolutiva)
- Árboles LISP (programación genética)

Técnicas de representación

Selección de la representación adecuada

- Las diferencias entre una técnica de representación y otra no son demasiado relevantes en la práctica.
- Se ha de escoger la representación que mejor se adapte al problema que pretendemos resolver.
  - Los operadores de variación (cruce y mutación) se escogen de acuerdo a la representación escogida.
  - La selección se basa en el valor de fitness de las soluciones candidatas, por lo que es independiente de la representación.

Técnicas de representación

Selección de la representación adecuada

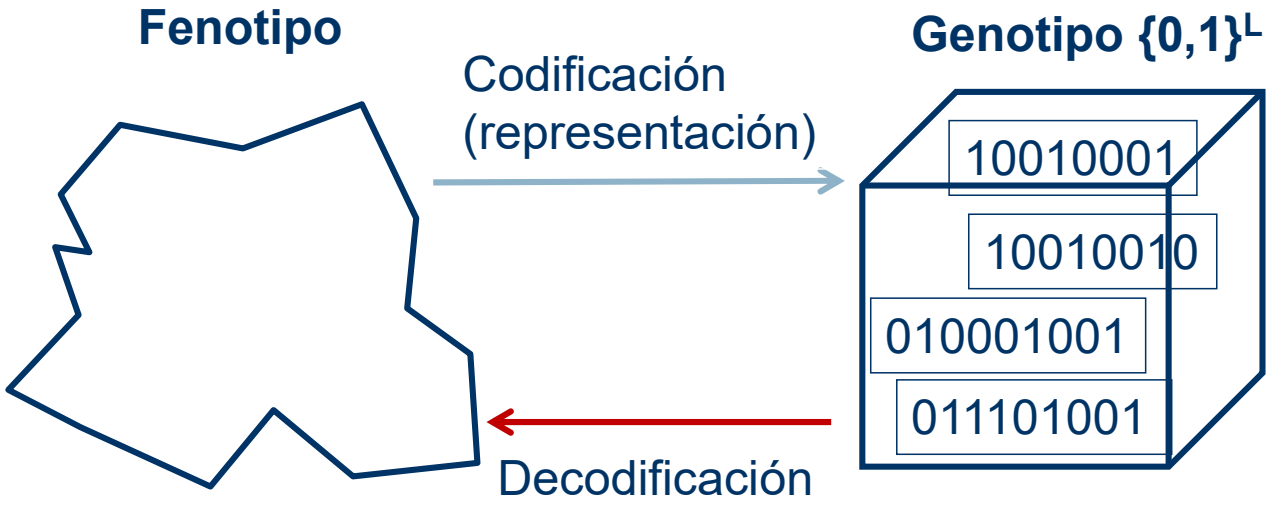
Las soluciones candidatas (individuos)…

- se evalúan de acuerdo a su fenotipo (comportamiento).
- se codifican de acuerdo a su genotipo (cromosomas).

- La decodificación es 1 a 1
  (la codificación, no necesariamente).
- Para ser capaces de encontrar la solución óptima, toda posible solución ha de poder representarse.

Técnicas de representación

Codificación binaria



Técnicas de representación

Codificación binaria

Representación tradicional

Chromosome 1	1 1 0 1 0 0 0 1 1 0 1 0
Chromosome 2	0 1 1 1 1 1 1 1 1 1 0 0

- **Cromosoma**: Cadena de bits que representa a cada individuo de la población.
- **Gen**: Cada posición de la cadena.
- **Alelo**: Valor de un gen.

Función de evaluación

La función de evaluación [fitness]…

- … representa la función objetivo de nuestro problema de optimización.
- … asigna valores reales a cada fenotipo.
- … se utiliza de base para realizar la selección
- Cuanto más discrimine, mejor (valores diferentes).
- Normalmente, pretenderemos maximizar el valor de fitness (aunque algunos problemas se plantean mejor como problemas de minimización y la conversión resulta trivial).

Población inicial

- La inicialización de la población suele ser aleatoria.
- Conviene asegurar una mezcla adecuada de todos los alelos posibles (esto es, incluir en la población inicial valores diferentes para todos los genes).
- Pueden utilizarse, como semilla, soluciones existentes y soluciones obtenidas utilizando heurísticas (dependientes del problema concreto).

Operadores genéticos

- Nos permiten generar nuevas soluciones candidatas.
- Se suelen clasificar en función de su aridad (esto es, de su número de entradas):
  - Aridad 1: Operadores de **mutación**.
  - Aridad>1: Operadores de **recombinación**.
  - Aridad=2: Operadores de **cruce**.
- La selección de operadores concretos depende del mecanismo de representación escogido.

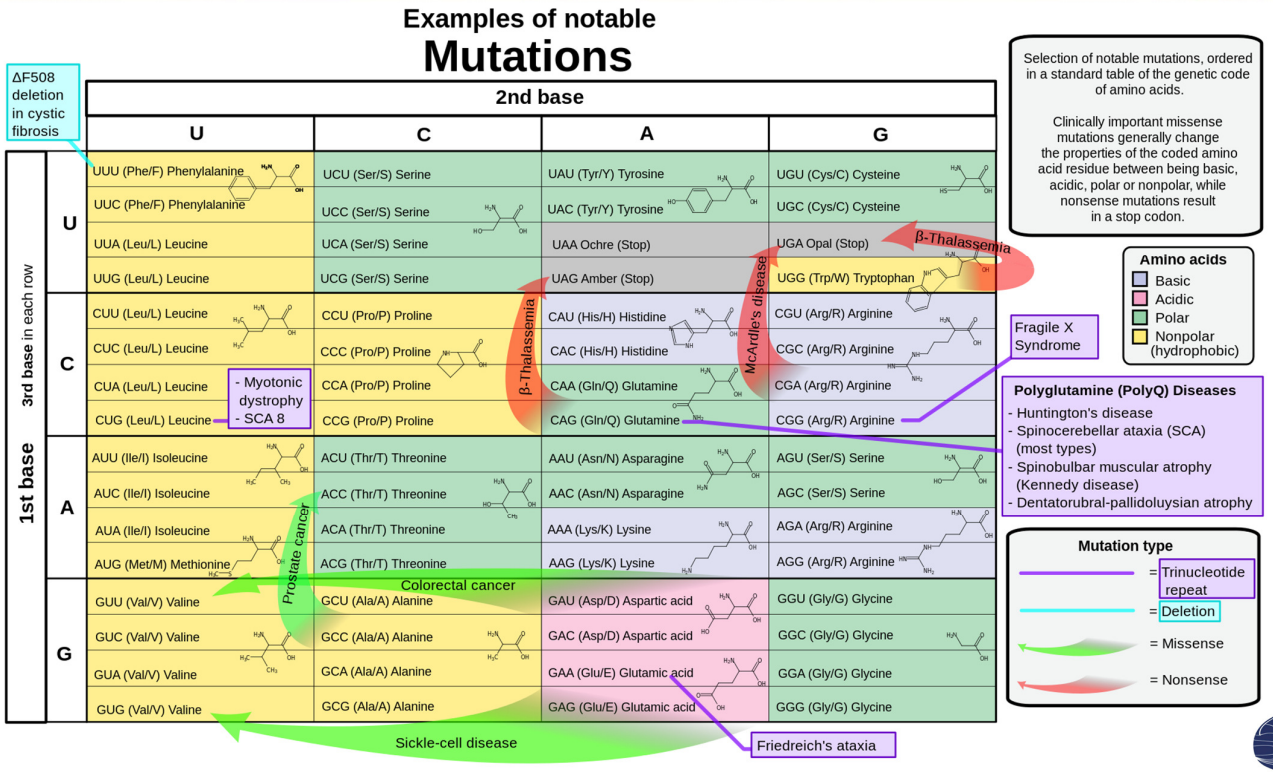
Operadores de mutación

Mutación (genética)

- Ocasionalmente, se producen errores en la replicación del material genético…
- Un hijo puede tener material genético que no proviene de ninguno de sus padres (no heredado)
- En la naturaleza, la mutación puede ser
  - Catastrófica (hijo no viable)
  - Neutral (no influencia su fitness)
  - Ventajosa (nueva característica beneficiosa)

NOTA: La redundancia en el código genético sirve de mecanismo de control de errores

Operadores de mutación



http://en.wikipedia.org/wiki/Mutation

Operadores de mutación

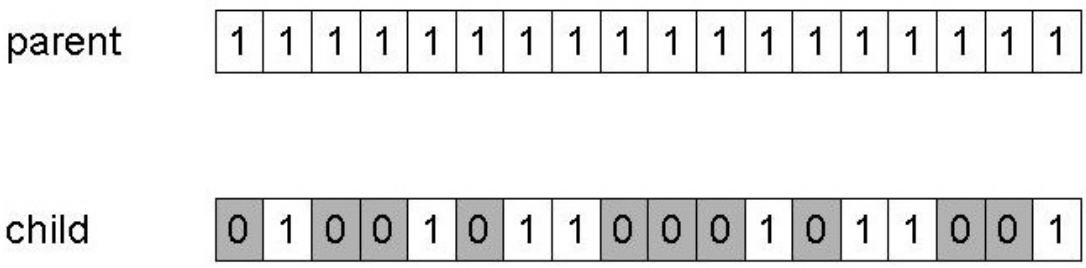
- Los operadores de mutación actúan sobre el genotipo.
- Es esencial su aleatoriedad (lo que los diferencia de otros operadores heurísticos unarios).
- Su importancia varía dependiendo del tipo de algoritmo evolutivo:
  - En los algoritmos genéticos, preservan e introducen diversidad en la población.
  - En programación genética, apenas se utilizan.
  - En programación evolutiva clásica, son los únicos operadores de variación utilizados.

Operadores de mutación

Mutación estándar

Dado un cromosoma, se altera el valor de cada gen con una probabilidad p_m

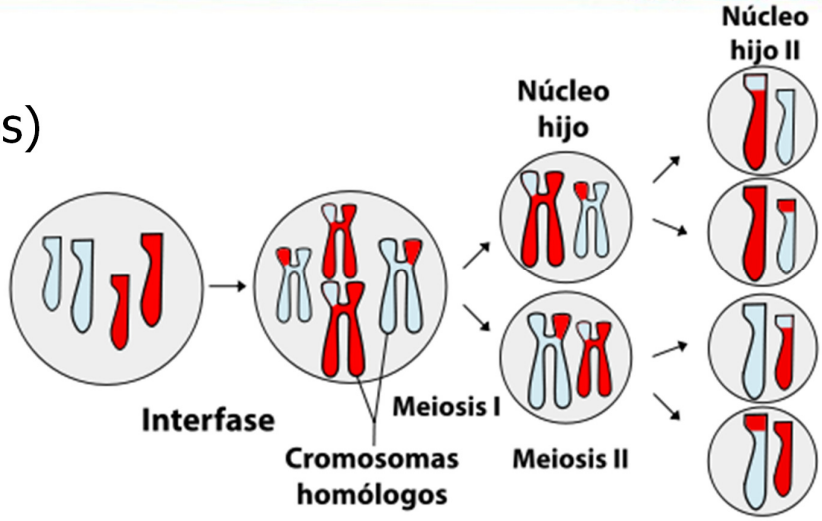
p_m se denomina probabilidad de mutación.



Operadores de cruce

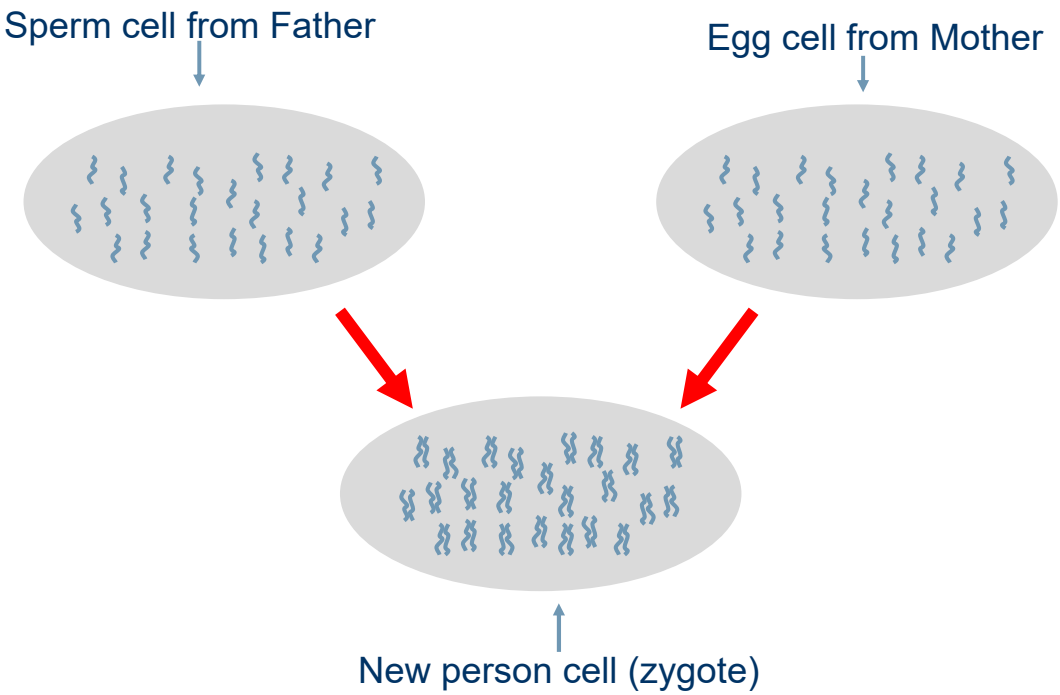
Meiosis

(producción de gametos)



- Las parejas de cromosomas se alinean y duplican.
- Los cromosomas homólogos se reparten en dos células hijas, se enlazan en un centrómero e intercambian partes de ellos (se cruzan).
- Como resultado, se obtienen 4 células haploides.

Operadores de cruce

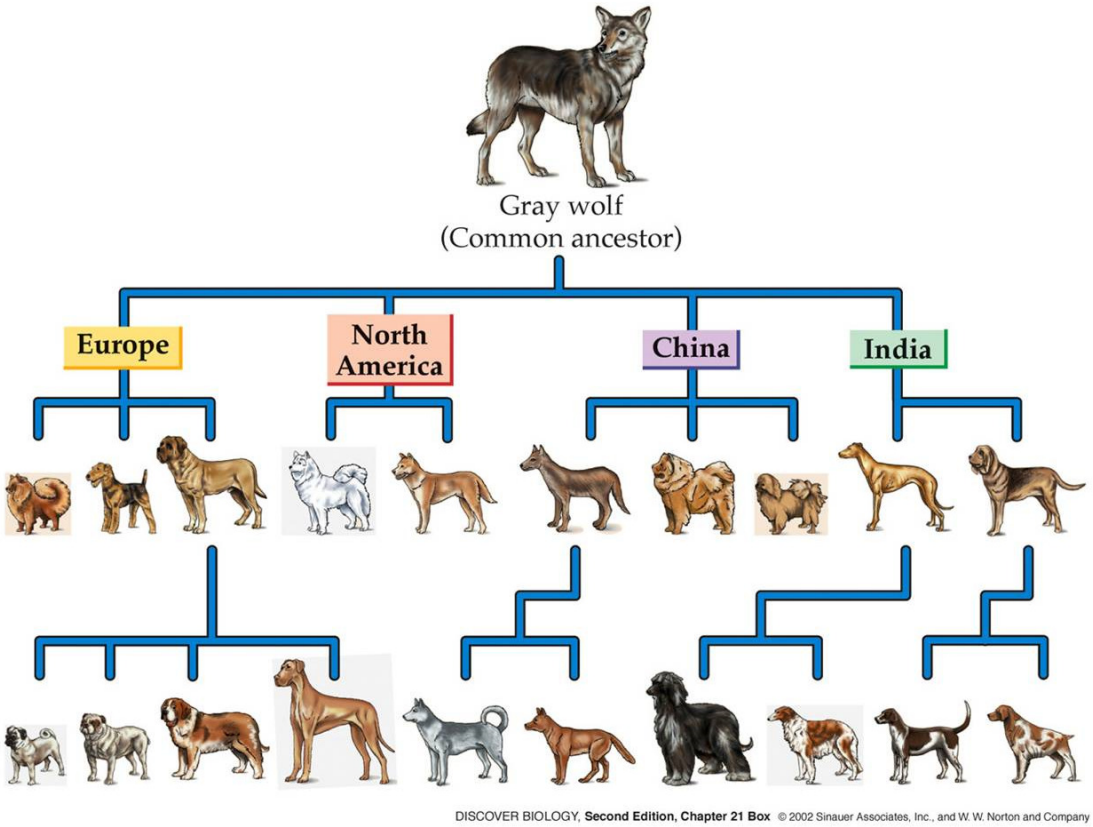


Operadores de cruce

- Combinan información de los padres para crear nuevos descendientes.
- La selección de qué información de los padres se combina es estocástica.
- Muchos descendientes pueden ser peores que los padres (en términos de la función de fitness).
- Se espera que algunos de ellos sean mejores al combinar los elementos de sus genotipos que conducen a la obtención de mejores fenotipos.

  NOTA: Este principio se ha utilizado durante milenios en agricultura y ganadería para la selección de variedades de plantas y razas de animales (incluidos los domésticos).

Operadores de cruce



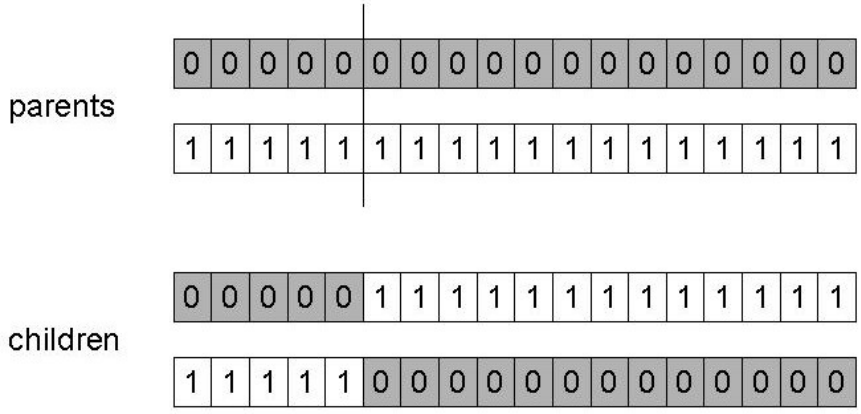
DISCOVER BIOLOGY, **Second Edition, Chapter 21 Box**

Operadores de cruce

Cruce en un punto

[1-point crossover]

- Se selecciona un punto de cruce aleatoriamente.
- Se dividen los padres en ese punto.
- Se crean hijos intercambiando partes de los cromosomas.



Operadores de cruce

Cruce en un punto

[1-point crossover]

Limitación: Depende del orden en el que aparecen los genes

- Es más probable que sigan juntos genes que estén cerca.
- Nunca mantiene juntos los genes de extremos opuestos.

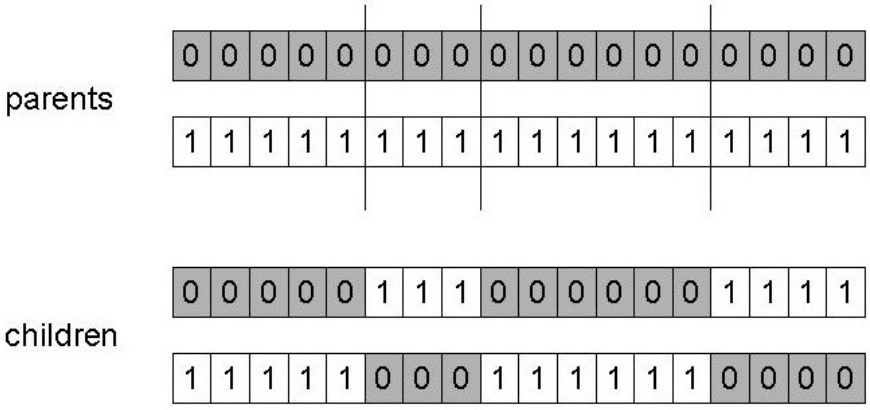
Este **sesgo posicional** puede aprovecharse si conocemos algo sobre la estructura de las soluciones de nuestro problema, aunque no suele ser el caso…

Operadores de cruce

Cruce en n puntos

[n-point crossover]

- Se eligen n puntos de cruce aleatoriamente.
- Se fragmentan los cromosomas en esos puntos.
- Se juntan fragmentos, alternando los padres.

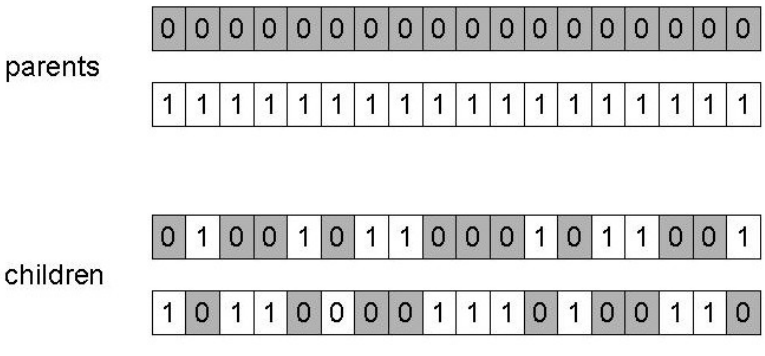


Operadores de cruce

Cruce uniforme

[uniform crossover]

- Se elige al azar el padre del que proviene cada gen para el primer hijo (el segundo hijo es el "inverso")
- La herencia de un gen es ahora independiente de su posición en el cromosoma (se eliminar el sesgo posicional del cruce en n puntos).



Operadores de cruce

Cruce uniforme

[uniform crossover]

No tiene sesgo posicional (no depende del orden en el que aparecen los genes y cada elección es independiente), pero…

Sesgo distributivo

Es más probable que elija el 50% de los genes de cada padre (en lugar del 90% de uno y el 10% de otro).

Operadores de recombinación

Se pueden diseñar operadores de recombinación que combinen información de más de dos padres (no estamos restringidos por lo que ocurre en la Naturaleza):
Poco frecuentes, aunque a veces son útiles

Tipos

- Basados en las frecuencias de los alelos (votación p-sexual, generalización del cruce uniforme)
- Basados en la segmentación de los padres (cruce diagonal, generalización del cruce en n puntos)
- Basados en operaciones numéricas sobre alelos con valores reales (p.ej. centro de masas)

Operadores de cruce

¿Cruce o mutación? El eterno debate

Depende del problema pero, en general:

- Suele ser bueno utilizar ambos operadores:
- Se puede utilizar un algoritmo evolutivo que sólo emplee mutación, pero no un algoritmo que sólo use recombinación.

Operadores de cruce

¿Cruce o mutación? El eterno debate

- El operador de cruce explora el espacio de búsqueda (descubre áreas prometedoras lejos de los padres).
- El operador de mutación explota lo que ya hemos descubierto (permanece cerca del área de los padres).

- Sólo el cruce puede combinar lo mejor de dos padres.
- Sólo la mutación puede introducir nuevos alelos en la población (el cruce no cambia la proporción de alelos en la población).
- Para llegar realmente al óptimo, puede que necesitemos una mutación afortunada…

Mecanismos de selección

Analogía con la evolución:

- El entorno dispone de recursos limitados (esto es, sólo puede dar soporte a un número finito de individuos).
- Los individuos tienen instintos básicos de reproducción.
- Es inevitable la existencia de algún tipo de selección:

  Aquellos individuos que compiten por los recursos de forma más efectiva tienen más posibilidades de reproducirse.

Mecanismos de selección

- La población…

  … contiene un conjunto de posibles soluciones.

  … suele tener un tamaño fijo.

- Algunos algoritmos evolutivos establecen una distribución espacial sobre la población (p.ej. grids o nichos).
- La selección suele considerar la población en su conjunto y la probabilidad de reproducción se define sobre la generación actual de la población.

Mecanismos de selección

- Se asignan probabilidades de selección a los individuos de la población (padres) en función de su fitness.
- Se suele emplear un mecanismo de selección estocástico:
  - Los mejores individuos es más probable que se seleccionen (aunque nada lo garantiza).
  - Incluso el peor individuo de la población puede ser seleccionado.
- La selección estocástica nos ayuda a escapar de óptimos locales.

Mecanismos de selección

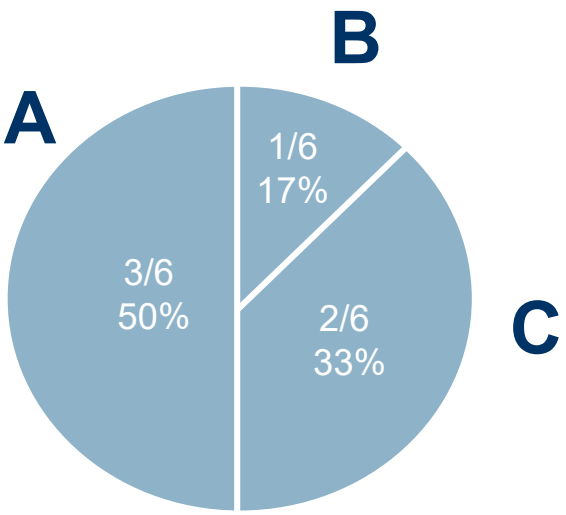
Selección proporcional ("ruleta")

- Idea: Los mejores individuos tienen más posibilidades de reproducirse (de manera proporcional a su fitness).
- Implementación: Ruleta
  - A cada individuo se le asigna una parte proporcional de la ruleta.
  - Se gira la ruleta n veces para seleccionar una población de n individuos.

Mecanismos de selección

Selección proporcional ("ruleta")

fitness(A)=3
fitness(B)=1
fitness(C)=2



Mecanismos de selección

Selección proporcional ("ruleta")

String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

[Goldberg 1989]

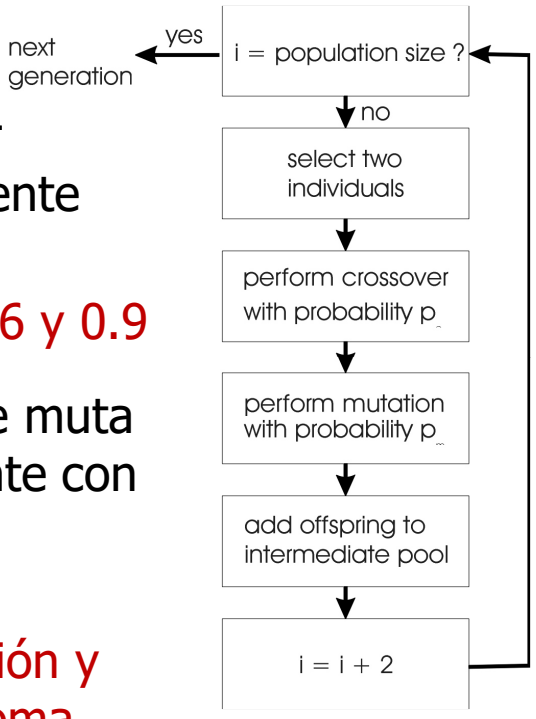
Mecanismos de selección

Algoritmo genético estándar

- Seleccionar dos padres.
- Con probabilidad p_c, cruzar los padres (si no, simplemente se copian).

  p_c típicamente entre 0.6 y 0.9
- Para cada descendiente, se muta cada bit independientemente con probabilidad p_m.

  p_m típicamente entre 1/tamaño de la población y 1/longitud del cromosoma



Mecanismos de selección

Algoritmo genético estándar

Cruce

String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

Mutación

String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	18	324
Sum				2354
Average				588.5
Max				729

Mecanismos de selección

Selección de supervivientes (a.k.a. reemplazo)

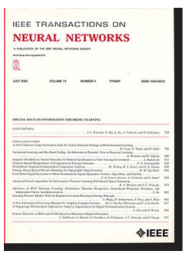
- La mayoría de los algoritmos evolutivos utilizan una población de tamaño fijo, por lo que hay que definir una forma de crear la siguiente generación a partir del conjunto de individuos disponibles (padres e hijos).

- La selección final suele ser determinística:
  - Basada en el fitness (sólo los mejores sobreviven).
  - Basada en la edad (la generación de descendientes sustituye por completo a sus padres).

Mecanismos de selección

- En ocasiones, se combinan fitness y edad: **elitismo** (el mejor individuo siempre sobrevive, sea padre o hijo).

- Ha sido demostrado que un algoritmo genético requiere elitismo para poder converger al óptimo global.



Günter Rudolph: "Convergence Analysis of Canonical Genetic Algorithms". IEEE Transactions on Neural Networks, 5:96-101, January 1994.

Mecanismos de selección

Modelos poblacionales

- **Modelo generacional** [SGA: standard GA]
  Cada individuo vive durante una única generación;
  i.e. el conjunto completo de padres es reemplazado.

- **Modelo estacionario** [SSGA: steady-state GA]
  En cada generación se crea un único descendiente,
  que reemplaza a un miembro de la población inicial
  (p.ej. peor fitness, FIFO o aleatorio).

Gap generacional = Fracción reemplazada de la población

gap(SGA) = 1 vs. gap(SSGA) = 1 / tamaño de la población

Mecanismos de selección

La competición entre individuos, en función de su fitness, por tanto, opera en dos sitios:

- Selección de padres (para crear descendientes)
- Selección de supervivientes (siguiente generación)

Los operadores de selección operan sobre individuos y son independientes de su representación.

Una vez determinadas las probabilidades de selección, se pueden utilizar distintos algoritmos para implementar el proceso de selección [de padres]…

Mecanismos de selección

FPS [Fitness Proportional Selection]

Número esperado de copias de un individuo i:

$$\mathbf{E(\ n_i\ ) = \mu \bullet f(i) \ / \ \langle\ f\ \rangle}$$

μ	Tamaño de la población
f(i)	Fitness del individuo I
$\langle f \rangle$	Fitness medio de la población

Algoritmo de la ruleta

Dada la distribución de pobabilidades de selección, girar la ruleta n veces para realizar n selecciones

- No garantiza que se obtenga el valor esperado de $\mathbf{n_i}$

Mecanismos de selección

FPS [Fitness Proportional Selection]

Número esperado de copias de un individuo i:

$$\mathbf{E(\ n_i\ ) = \mu \bullet f(i) \ / \ \langle\ f\ \rangle}$$

μ	Tamaño de la población
f(i)	Fitness del individuo I
$\langle f \rangle$	Fitness medio de la población

Algoritmo SUS de James Baker (1987) [Stochastic Universal Sampling]

Se usa la ruleta una única vez
y se seleccionan n elementos equiespaciados

- Garantiza que $\text{floor}(E(n_i)) \leq n_i \leq \text{ceil}(E(n_i))$

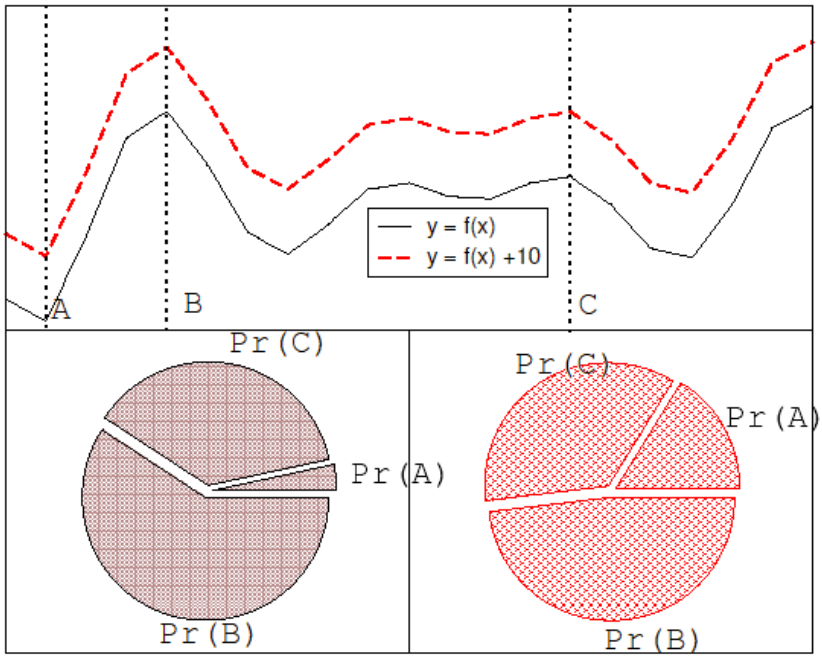
Mecanismos de selección

La selección proporcional presenta algunos problemas:

- Un individuo muy apto puede expandirse rápidamente si el resto de la población en mucho peor (convergencia prematura)
- Cuando los valores de fitness son similares, el proceso de selección pierde presión (selección prácticamente uniforme).
- Las probabilidades de selección son muy susceptibles ante la trasposición de la función de fitness.

Mecanismos de selección

Trasposición en la función de fitness
y su influencia en la selección proporcional



Mecanismos de selección

Cambiar la escala de la función de fitness
puede solucionar los dos últimos problemas:

- **Windowing**: $f'(i) = f(i) - \beta^t$
  donde β^t es el peor fitness en esta generación
  (o las últimas n generaciones)

- **Sigma Scaling**: $f'(i) = \max(f(i) - (\langle f \rangle - c \bullet \sigma_f), 0.0)$
  donde c es una constante, normalmente c=2.0

Mecanismos de selección

Selección basada en el ranking

Para eliminar los problemas de la selección proporcional, las probabilidades de selección se basan en valores relativos en vez de en los valores absolutos de fitness.

- Se ordena la población de acuerdo a su fitness y las probabilidades de selección se basan en el ranking (ranking μ-1 para el mejor, 0 para el peor).

- Se mantiene una presión selectiva constante.

NOTA:
Idea similar a los tests no paramétricos en Estadística.

Mecanismos de selección

Ranking lineal

$$P_{lin-rank}(i) = \frac{(2-s)}{\mu} + \frac{2i(s-1)}{\mu(\mu-1)}$$

- Parametrizado por el factor s: $1.0 < s \leq 2.0$
- El factor s mide la ventaja del mejor individuo (el número de descendientes que le tocan en un SGA).

Individual	Fitness	Rank	P_{selFP}	P_{selLR} $(s=2)$	P_{selLR} $(s=1.5)$
A	1	0	0.1	0	0.167
B	4	1	0.4	0.33	0.33
C	5	2	0.5	0.67	0.5
Sum	10		1.0	1.0	1.0

Mecanismos de selección

Ranking exponencial

$$P_{exp-rank}(i) = \frac{1-e^{-i}}{c}.$$

- El ranking lineal tiene una presión selectiva limitada.
- El ranking exponencial permite asignar más de dos desdendientes al mejor individuo de la población.
- La constante c se utiliza para normalizar de acuerdo al tamaño de la población.

Mecanismos de selección

Selección por torneo

Los métodos anteriores se basan en estadísticas de la población en su conjunto.

- Pueden suponer un cuello de botella en su paralelización.
- Dependen de la existencia de una función de fitness (que puede no existir en determinados problemas, e.g. estrategias en juegos y otros problemas en los que sólo se puede evaluar comparando individuos).

IDEA BÁSICA:

- Escoger k miembros al azar y seleccionar al mejor.
- Repetir las veces que haga falta.

Mecanismos de selección

Selección por torneo

La probabilidad de seleccionar un individuo dependerá de

- El ranking del individuo.
- El tamaño del torneo, i.e. la muestra k.
  (cuanto mayor sea k, mayor presión de selección).
- El método de muestreo
  (el muestreo sin reemplazo aumenta la presión selectiva)
- Si el mejor individuo siempre gana (torneo determinista) o si esto sucede con probabilidad p (torneo estocástico).

Si k=2, el tiempo necesario para que el mejor individuo cope la población es el mismo que con ranking lineal usando $s = 2 \cdot p$

Mecanismos de selección

Selección por torneo round-robin

Como una liga…

- Cada individuo de la población se evalúa frente a q rivales elegidos aleatoriamente.
- Se le asigna una victoria cada vez que es mejor que su oponente.
- Tras realizar todos los torneos, se seleccionan los μ individuos con más victorias.

p.ej. Programación evolutiva (q=10)

Mecanismos de selección

Sobreselección [over-selection]

Para poblaciones grandes (>1000 individuos).

p.ej. Programación genética

- La población se ordena por fitness y se divide en dos:
  - Grupo 1: Mejor x%.
  - Grupo 2: Restante (100-x)%.

Population size	Proportion of population in fitter group (x)
1000	32%
2000	16%
4000	8%
8000	4%

- A la hora de seleccionar padres:
  - 80% del primer grupo.
  - 20% del segundo grupo.

Parámetros

Ejemplo

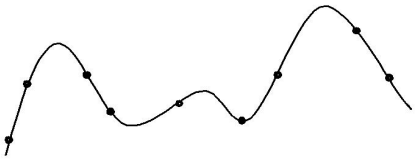
El problema de las N reinas

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

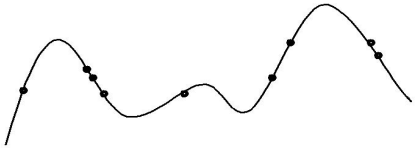
Una posible elección de
los parámetros de un algoritmo genético.

Parámetros

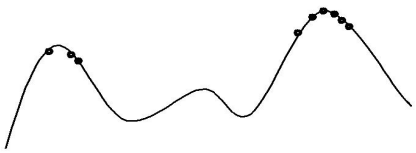
Fases de la ejecución de un algoritmo genético "ideal":



Generaciones iniciales:
Distribución aleatoria de la población.



Generaciones intermedias:
Población cerca de las colinas.

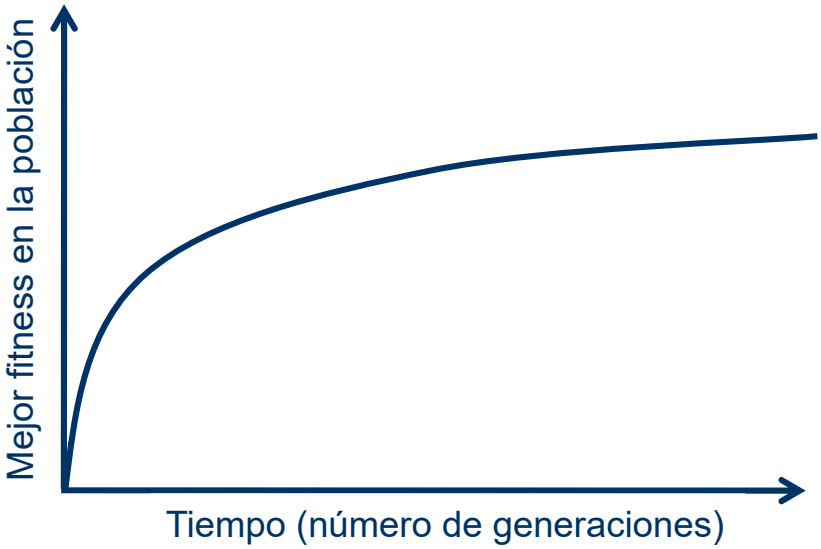


Generaciones finales:
Población concentrada en las
cimas de las colinas más altas.

Parámetros

Evolución típica del fitness

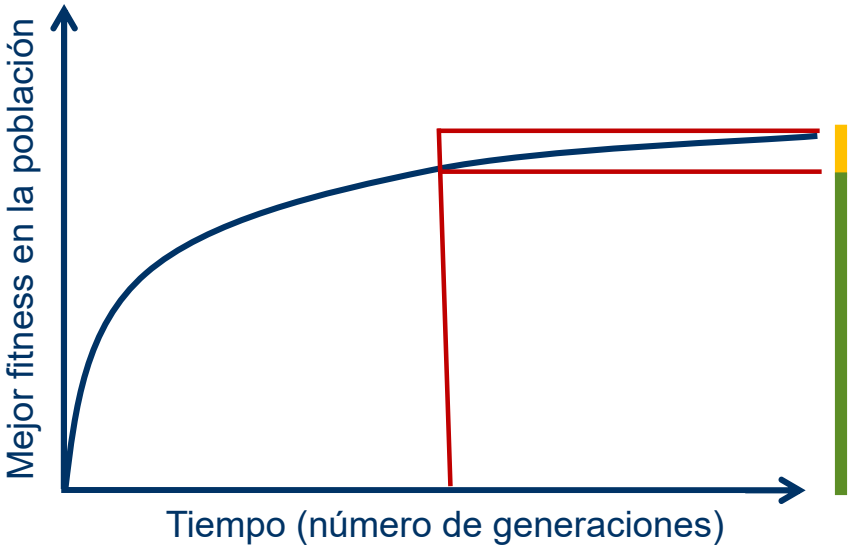
durante la ejecución de un algoritmo genético:



Parámetros

¿Merece la pena una ejecución larga?

Depende (a veces, pueden obtenerse mejores resultados con varias ejecuciones más cortas).



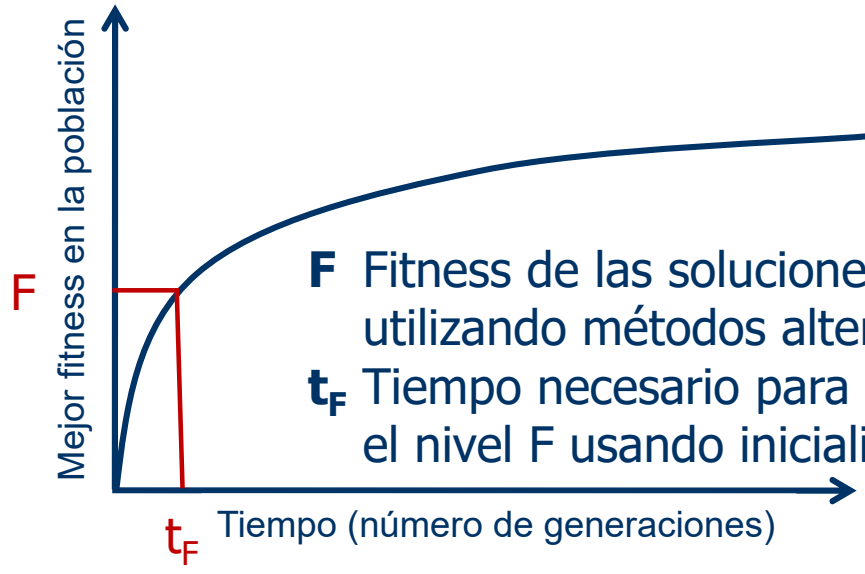
Progreso en la segunda mitad de la ejecución

Progreso en la primera mitad de la ejecución

Parámetros

¿Merece la pena invertir en la inicialización?

Depende (puede que sí, si existen técnicas adecuadas para encontrar buenas soluciones iniciales).



F Fitness de las soluciones encontradas utilizando métodos alternativos.

t_F Tiempo necesario para alcanzar el nivel F usando inicialización aleatoria.

Parámetros

Estudios teóricos

(limitados en cuanto a su aplicabilidad)

- Tiempo de toma de control [takeover time]: Tiempo que tarda el operador de selección en que toda la población esté formada por copias del mejor individuo.
- Tiempo de mezcla [mixing time]: Tiempo que tardan los operadores de recombinación en combinar bloques presentes inicialmente en distintos individuos.

$t_{takeover} < t_{mixing} \rightarrow$ Convergencia prematura

Otras representaciones

- **Codificación Gray** (sigue siendo codificación binaria)

  Motivación: Pequeños cambios en el genotipo causan pequeños cambios en el fenotipo.

- **Vectores de números** (reales o enteros)

  Codificación habitual cuando las variables de nuestro problema son numéricas (codificación directa, aunque habrá que definir operadores genéticos adecuados).

Representación entera

- Los operadores de cruce binarios (en un punto, en n puntos y uniforme) se pueden seguir utilizando.

- El operador de mutación se modifica en función de la situación:
  - Valores ordinales:
    Se hace que sea más probable cambiar a valores cercanos en la escala [creep mutation].
  - Valores categóricos (sin orden natural):
    Selección aleatoria uniforme [random resetting].
    EJEMPLO: Coloreado de un grafo con k colores.

Permutaciones

- Los problemas de ordenación/secuenciación son especiales (p.ej. TSP, QAP…): lo importante es establecer qué elementos aparecen junto a otros.

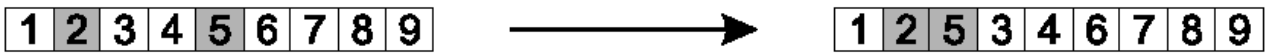
- Los operadores de cruce y mutación habituales dan lugar a soluciones inadmisibles, por lo que los operadores de mutación deben cambiar al menos dos valores y los de cruce han de diseñarse específicamente para problemas de este tipo.

NOTA: La probabilidad de mutación ahora se referirá a cromosomas completos, no a genes individuales

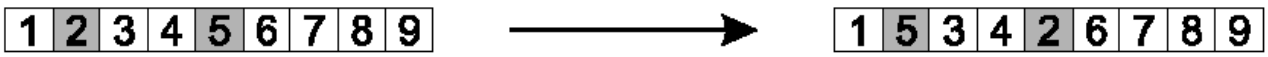
Permutaciones

Operadores de mutación (1/2)

- **Inserción [insert]**: Elegir dos alelos aleatoriamente y colocar el segundo justo después del primero.



- **Intercambio [swap]**: Seleccionar dos alelos aleatoriamente e intercambiarlos.

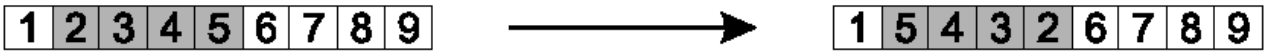


NOTA: Observe como cada operador de mutación afecta de forma diferente al orden relativo y a las relaciones de adyacencia.

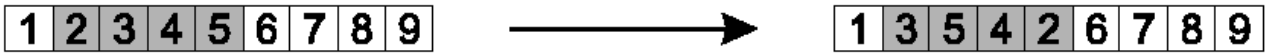
Permutaciones

Operadores de mutación (2/2)

- **Inversión [inversion]**: Seleccionar dos alelos aleatoriamente e invertir la cadena entre ellos.



- **Revuelto [scramble]**: Seleccionar un subconjunto de genes y reordenar aleatoriamente los alelos (el subconjunto no tiene por qué ser contiguo).



NOTA: Observe como cada operador de mutación afecta de forma diferente al orden relativo y a las relaciones de adyacencia.

Permutaciones

Operadores de cruce

Cruce de orden

(se preserva el orden relativo de los elementos)

- Seleccionar una parte arbitraria del primer padre.
- Copiar esta parte en el hijo.
- Copiar los números que no están en la primera parte…
  … empezando desde el punto de cruce.
  … utilizando el orden en el que aparecen en el 2º padre
  … volviendo al principio del cromosoma cuando hayamos llegado al final.

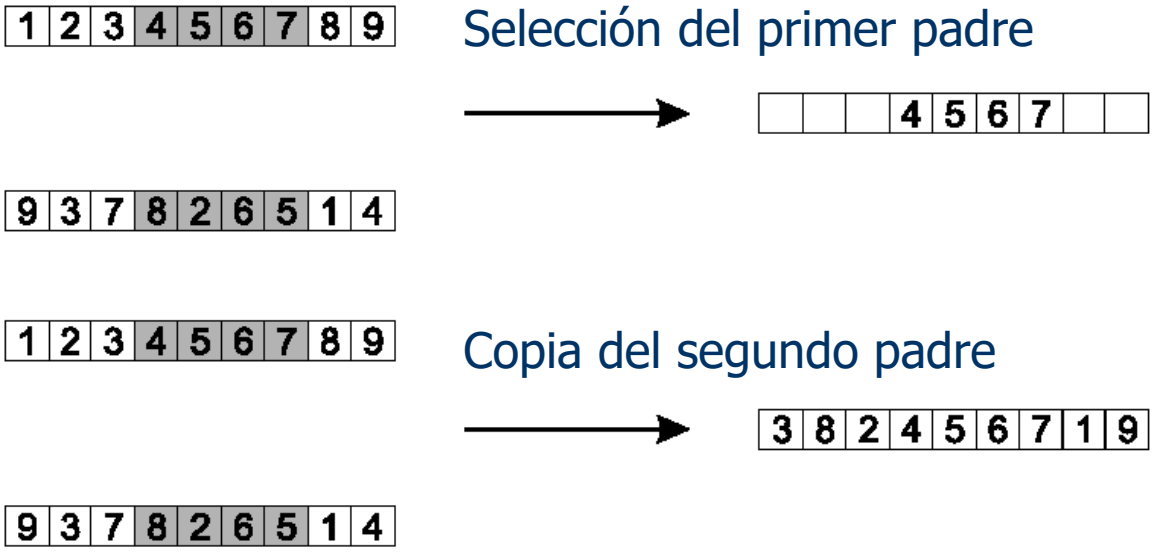
El segundo hijo se crea intercambiando los papeles de los padres.

Permutaciones

Operadores de cruce

Cruce de orden

(se preserva el orden relativo de los elementos)



Permutaciones

Operadores de cruce

PMX [Partially-Mapped Crossover]

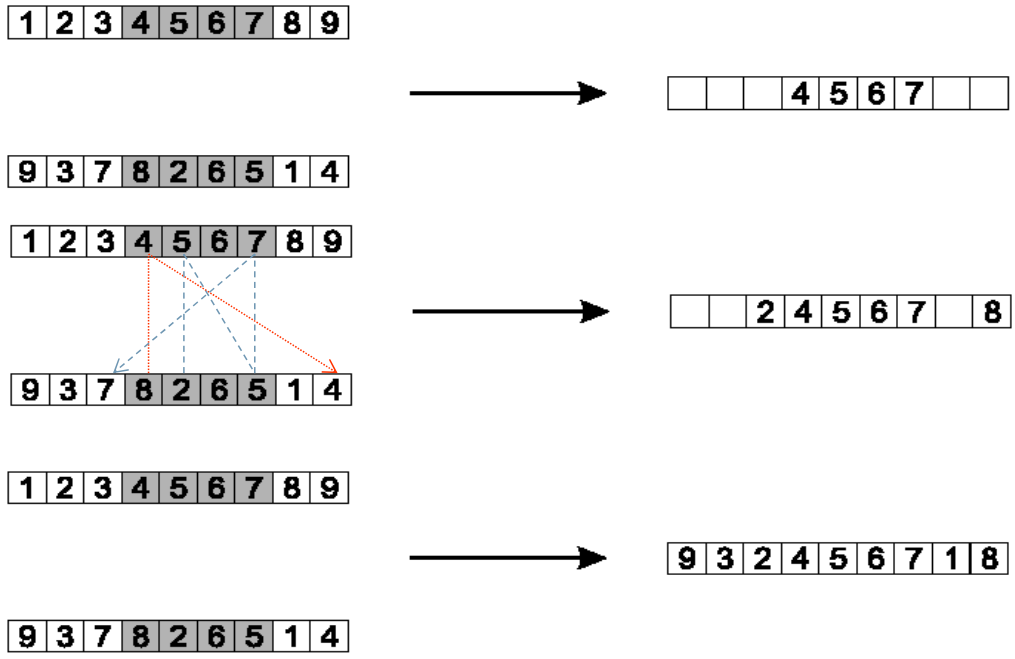
A partir de los padres P1 y P2:

1. Elegir un segmento aleatorio y copiar desde P1.
2. Desde el primer punto de cruce, buscar elementos de ese segmento en P2 que no se han copiado.
3. Para cada uno de esos elementos i, buscar qué elemento j se ha copiado en su lugar desde P1.
4. Colocar i en la posición ocupada j de P2 (sabemos que no estará ahí, ya que lo hemos copiado ya).
5. Si el lugar ocupado por j en P2 ya se ha rellenado en el hijo (k), poner i en la posición ocupada por k en P2.
6. El resto de elementos se rellena de P2.

Permutaciones

Operadores de cruce

PMX [Partially-Mapped Crossover]



Permutaciones

Operadores de cruce

Cruce de ciclos [cycle crossover]

Cada alelo viene de un padre con su posición:

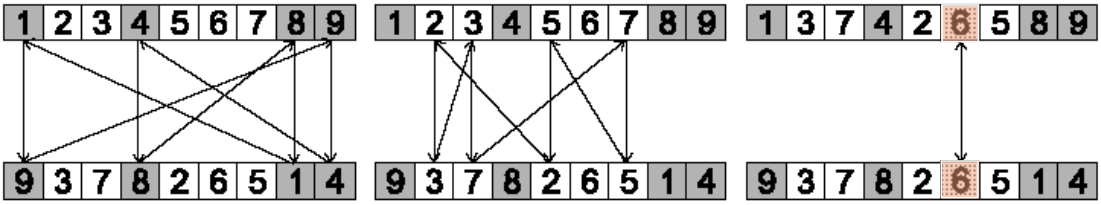
1. Formar un ciclo de alelos de la siguiente forma:
   a) Comenzar con el primer alelo de P1.
   b) Mirar el alelo que está en la misma posición en P2.
   c) Ir a la posición con el mismo alelo en P1.
   d) Añadir este alelo al ciclo.
   e) Repetir b)-d) hasta que lleguemos al primer alelo.
2. Poner los alelos del ciclo en las posiciones que tienen en el primer padre.
3. Coger el siguiente ciclo del otro padre.

Permutaciones

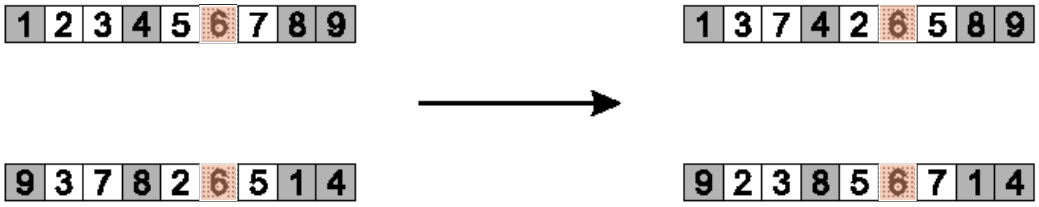
Operadores de cruce

Cruce de ciclos [cycle crossover]

1. Identificación de ciclos:



2. Copia de ciclos alternativos en el hijo:



Permutaciones

Operadores de cruce

Recombinación de aristas [edge recombination]

Se construye una tabla con las aristas presentes en los dos padres, marcando las comunes.

Ejemplo: [1 2 3 4 5 6 7 8 9] y [9 3 7 8 2 6 5 1 4]

Element	Edges	Element	Edges
1	2,5,4,9	6	2,5+,7
2	1,3,6,8	7	3,6,8+
3	2,4,7,9	8	2,7+, 9
4	1,3,5,9	9	1,3,4,8
5	1,4,6+		

Permutaciones

Operadores de cruce

Recombinación de aristas [edge recombination]

Choices	Element selected	Reason	Partial result
All	1	Random	[1]
2,5,4,9	5	Shortest list	[1 5]
4,6	6	Common edge	[1 5 6]
2,7	2	Random choice (both have two items in list)	[1 5 6 2]
3,8	8	Shortest list	[1 5 6 2 8]
7,9	7	Common edge	[1 5 6 2 8 7]
3	3	Only item in list	[1 5 6 2 8 7 3]
4,9	9	Random choice	[1 5 6 2 8 7 3 9]
4	4	Last element	[1 5 6 2 8 7 3 9 4]

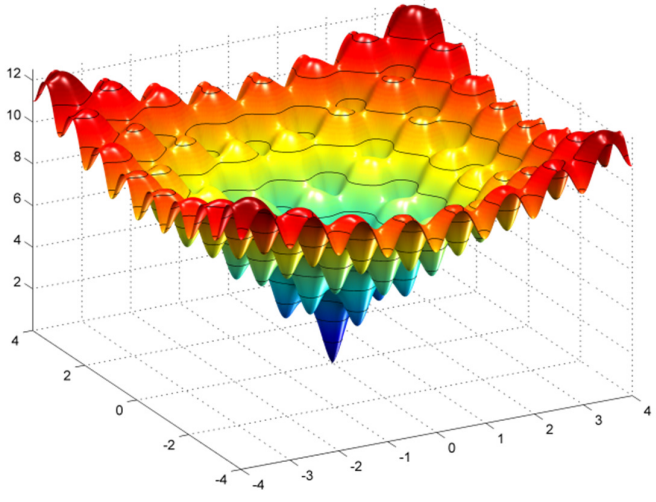
Element	Edges	Element	Edges
1	2,5,4,9	6	2,5+,7
2	1,3,6,8	7	3,6,8+
3	2,4,7,9	8	2,7+, 9
4	1,3,5,9	9	1,3,4,8
5	1,4,6+		

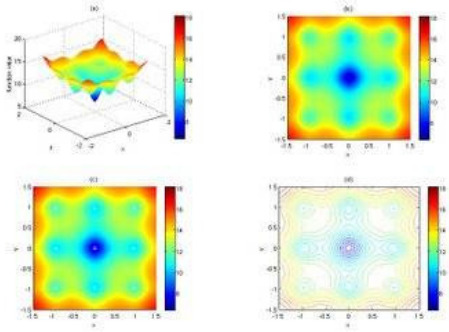
Representación real

La típica en muchos problemas de optimización.

Ejemplo: Función de Ackley

$$f(x) = -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i) \right) + 20 + e$$





Representación real

- Dado que los números reales se representan en binario en el ordenador, el número de bits utilizado para representar cada valor determina la precisión máxima de la solución.

- Si usamos L bits, cualquier z $\in$ [x,y] $\subseteq \Re$ se representa por una cadena de L bits $(a_1,\ldots,a_L) \in \{0,1\}^L$

  Γ: $\{0,1\}^L \rightarrow$ [x,y] define la representación

$$\Gamma(a_1,\ldots,a_L) = x + \frac{y-x}{2^L-1} \cdot \left(\sum_{j=0}^{L-1} a_{L-j} \cdot 2^j\right) \in [x,y]$$

- Sólo se representan 2^L valores (de un conjunto infinito)

Representación real

Operadores de mutación

$$\bar{x} = \langle x_1, \ldots, x_l \rangle \rightarrow \bar{x}' = \langle x_1', \ldots, x_l' \rangle$$
$$x_i, x_i' \in [LB_i, UB_i]$$

- Mutación uniforme (análoga a la mutación binaria o a la inicialización aleatoria de valores enteros).

$$x_i' \text{ elegido aleatoriamente del intervalo } [LB_i, UB_i]$$

- Mutación no uniforme (muchas alternativas posibles), p.ej. añadir una variación aleatoria para cada variable utilizando una distribución normal N(0,σ)
  y truncar al intervalo válido para cada variable.

Representación real

Operadores de mutación autoadaptativos

- Se incluyen los parámetros de la mutación (desviación σ, desviación para cada variable σ_i o, incluso, matriz de covarianza), a.k.a. mutation step sizes, en el genoma para que también puedan evolucionar.

- El usuario no fija los parámetros de la mutación, sino que coevolucionan con las propias soluciones:

  p.ej. Cromosomas $\langle x_1,\ldots,x_n, \sigma \rangle$

  - $\sigma' = \sigma \cdot \exp(\tau \cdot N(0,1))$
  - $x'_i = x_i + \sigma' \cdot N_i(0,1)$

    Típicamente, "tasa de aprendizaje" $\tau \propto 1/\sqrt{n}$

Representación real

Operadores de mutación autoadaptativos

Matriz de covarianza C

- Cromosomas $\langle x_1,\ldots,x_n, \sigma_1,\ldots, \sigma_n, \alpha_1,\ldots, \alpha_k \rangle$, k=n(n-1)/2
  - $c_{ii} = \sigma_i^2$
  - $c_{ij} = 0$ si i y j no están correladas
  - $c_{ij} = \frac{1}{2} (\sigma_i^2 - \sigma_j^2) \tan(2\alpha_{ij})$ si i y j están correladas.
- Mutación
  - $\sigma'_i = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1))$
  - $\alpha'_j = \alpha_j + \beta \cdot N_j(0,1)$
  - $\mathbf{x}' = \mathbf{x} + \mathbf{N}(\mathbf{0},\mathbf{C}')$

$\tau' \propto 1/(2n)^{1/2}$, $\tau \propto 1/(2n^{1/2})^{1/2}$, $\beta \approx 5°$

Representación real

Operadores de mutación autoadaptativos

Matriz de covarianza C

Covariance Matrix Adaptation Evolution Strategy (CMA-ES)
puede que sea el mejor algoritmo evolutivo para optimización numérica.

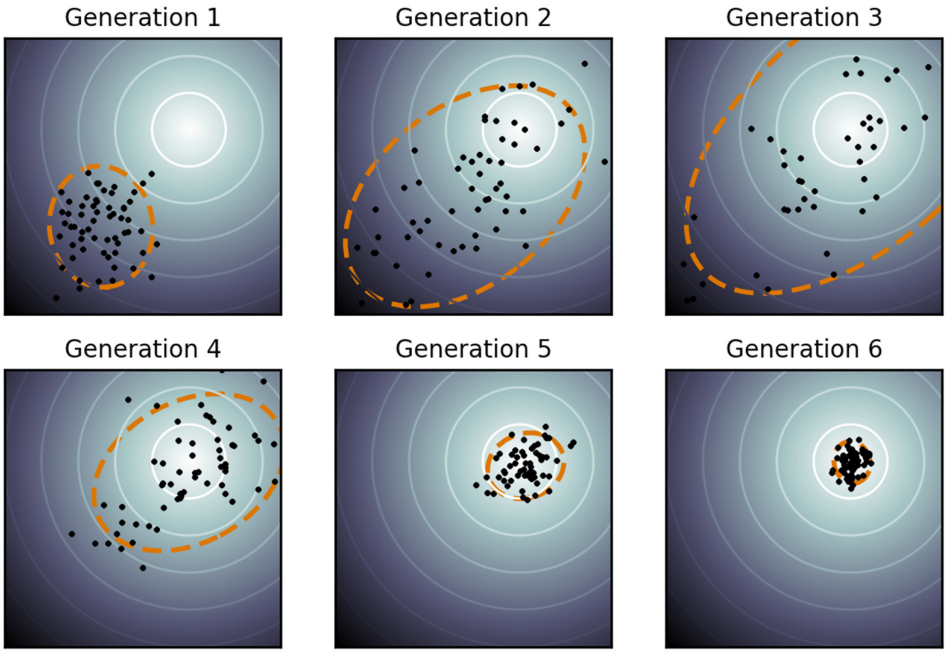
https://en.wikipedia.org/wiki/CMA-ES

CEC'2005 Competition
IEEE Congress on Evolutionary Computation
https://cma-es.github.io/cec2005/cec2005compareresults.pdf

Representación real

Operadores de mutación autoadaptativos

CMA-ES (matriz de covarianza C)



Representación real

Operadores de cruce

- **Recombinación discreta** (valores discretos):
  - Cada alelo proviene de uno de sus padres.
  - Se puede usar el cruce uniforme o en n puntos.

- **Recombinación aritmética** (valores intermedios):
  Se recombinan los valores de los padres x e y:

$$z_i = \alpha\, x_i + (1 - \alpha)\, y_i$$

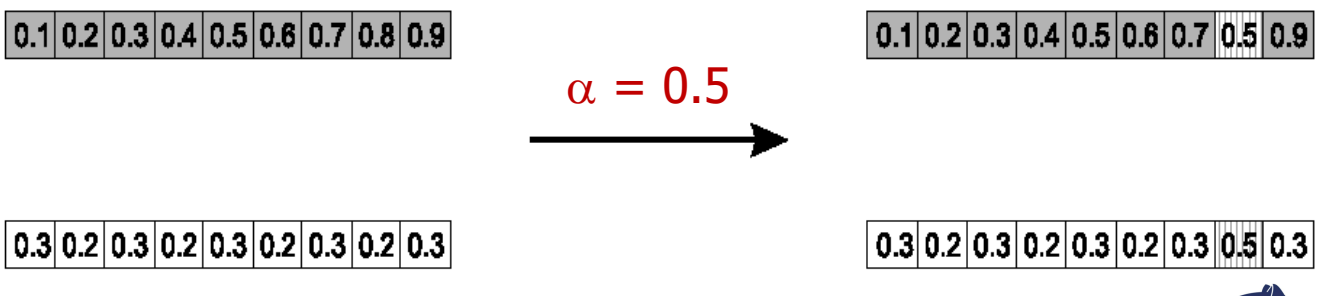
  donde el parámetro α ($0 \leq \alpha \leq 1$) puede ser constante (cruce aritmético uniforme), variable (p.ej. en función de la generación en la que nos encontremos) o, incluso, aleatorio.

Representación real

Cruce aritmético único

[single arithmetic crossover]

- Seleccionar un gen concreto al azar (k).
- Primer hijo: $\langle x_1, ..., x_{k-1}, \alpha \cdot y_k + (1-\alpha) \cdot x_k, ..., x_n \rangle$
- Segundo hijo: $\langle y_1, ..., y_{k-1}, \alpha \cdot x_k + (1-\alpha) \cdot y_k, ..., y_n \rangle$



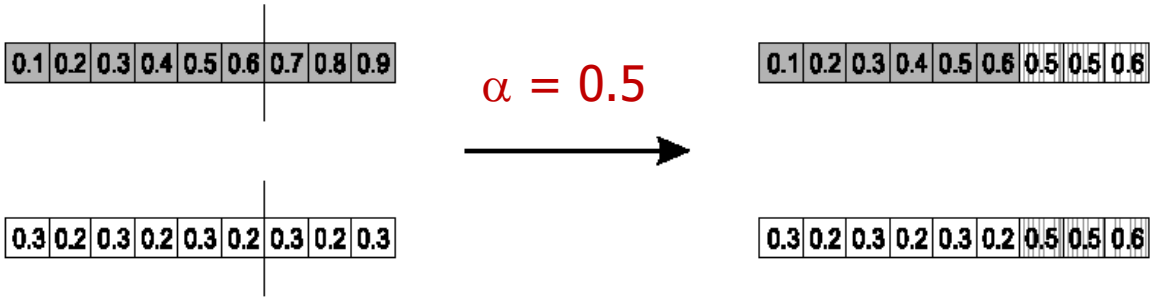
Representación real

Cruce aritmético simple
[simple arithmetic crossover]

- Seleccionar un gen concreto al azar (k).
- Combinar a partir del gen k:

$$\langle x_1,...,x_k,\alpha\cdot y_{k+1}+(1-\alpha)\cdot x_{k+1},...,\alpha\cdot y_n+(1-\alpha)\cdot x_n\rangle$$

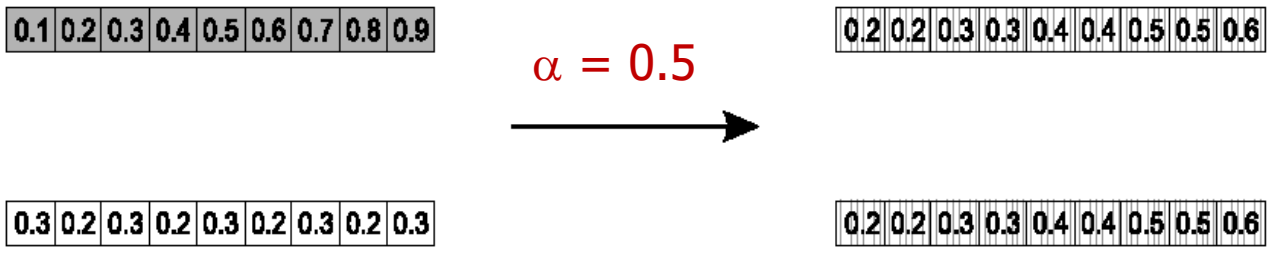
- A la inversa para el otro hijo.



Representación real

Cruce aritmético completo
[whole arithmetic crossover]

- Variante más utilizada
- El primer hijo es $\alpha\cdot\vec{x}+(1-\alpha)\cdot\vec{y}$
- El segundo hijo es $(1-\alpha)\cdot\vec{x}+\alpha\cdot\vec{y}$



Representación real

Cruce por mezcla BLX-α
[blend crossover]

- Diseñado para crear descendientes en una región más amplia que el rectángulo n-dimensional definido por los padres.
- Para cada variable, dados dos padres x_i e y_i (con $x_i < y_i$)
  - Se muestrea un número uniforme $u \in [0,1]$
  - Se calcula $\gamma = (1-2\alpha)u - \alpha$
  - Se fija el valor del descendiente $z_i = (1-\gamma)\, x_i + \gamma\, y_i$

El rango para el hijo z_i es $[x_i - \alpha d_i, y_i + \alpha d_i]$, $d_i = y_i - x_i$
p.ej. α=0.5 equilibra exploración y explotación

Programación genética

Programación genética

Aunque los primeros intentos por hacer evolucionar programas se remontan a los 1950s y 1960s, hasta los 1980 no se obtuvieron resultados satisfactorios.

- Joseph F. Hicklin (1986) usó reproducción y mutación para generar programas en LISP.
- Cory Fujiki y John Dickinson (1987) hicieron evolucionar sentencias condicionales para representar estrategias en problemas de teoría de juegos.
- Nichael Lynn Cramer (1985) y, posteriormente, John R. Koza (1989) propusieron, de forma independiente, el uso de una representación en árbol sobre la que se implementó un operador de cruce que permitía intercambiar subárboles entre los diferentes programas de una población generada al azar.

Programación genética

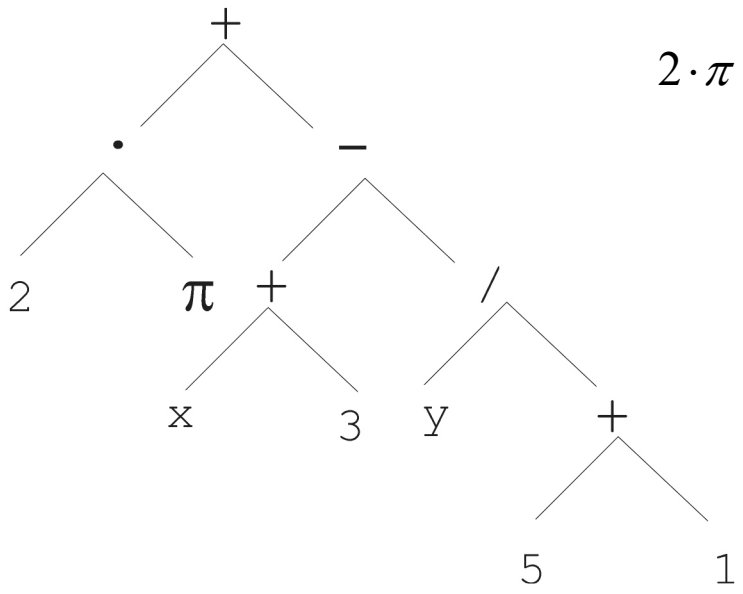
La propuesta de Koza fue la que se acabó imponiendo y, más tarde, se denominó Programación Genética.

- Nichael Lynn Cramer (1985): **Representation for the Adaptive Generation of Simple Sequential Programs**. 1st International Conference on Genetic Algorithms and their Applications, CMU, Pittsburgh, PA, July 24-26, 1985. http://homepages.sover.net/~nichael/nlc-publications/icga85/index.html

- John R. Koza (1989): **Hierarchical genetic algorithms operating on populations of computer programs.** IJCAI'89, 11th International Joint Conference on Artificial Intelligence, volume 1, pages 768-774, Detroit, Michigan, August 20-25, 1989. http://www.genetic-programming.com/jkpdf/ijcai1989.pdf

Programación genética

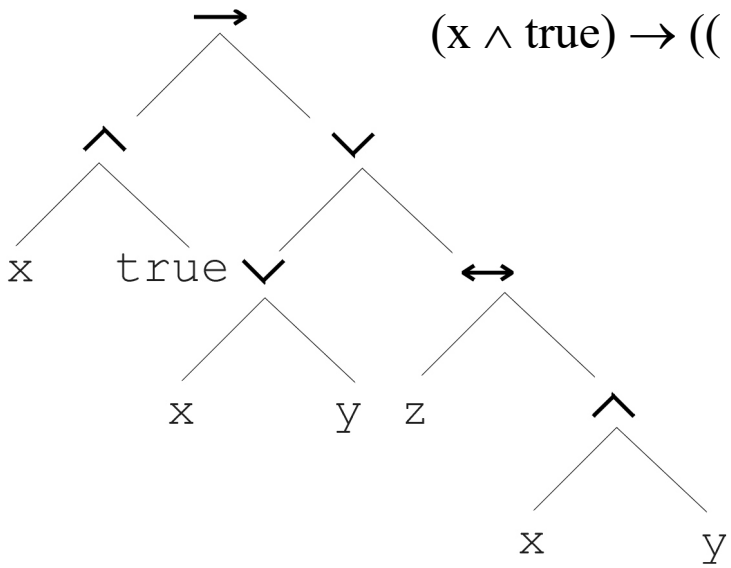
Idea básica: representación basada en **árboles**.



$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$

Programación genética

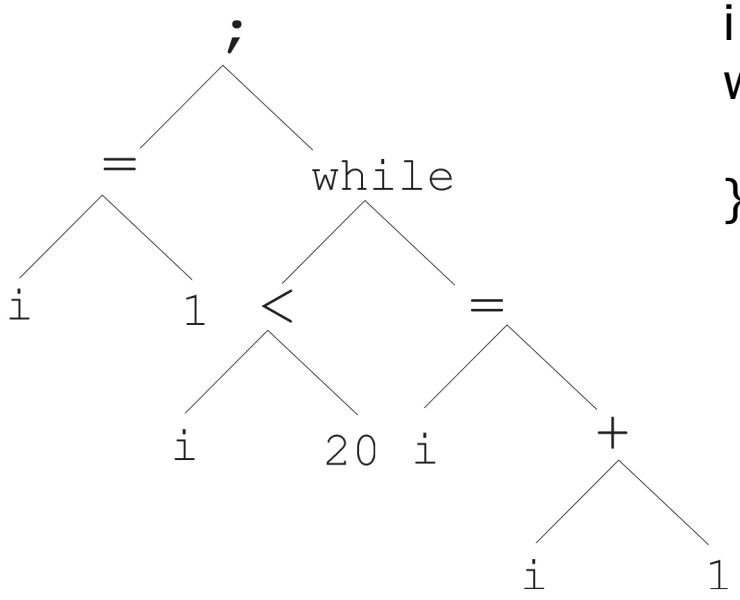
Idea básica: representación basada en **árboles**.



$$(x \wedge \text{true}) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$$

Programación genética

Idea básica: representación basada en **árboles**.




```
i =1;
while (i < 20) {
    i = i +1;
}
```

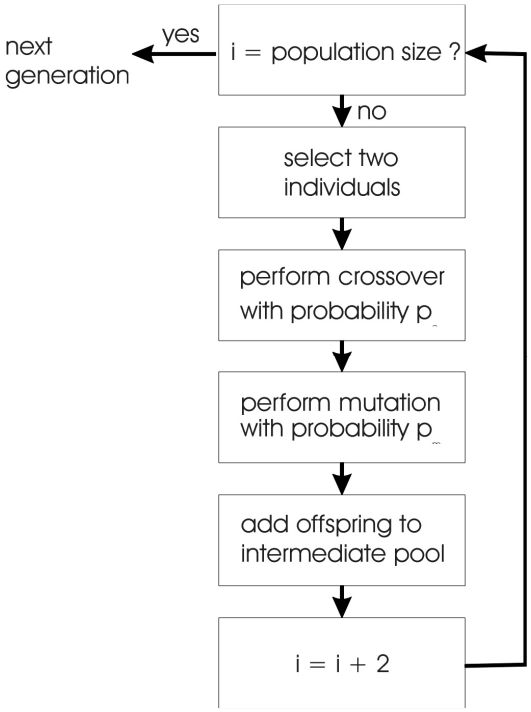

Programación genética

Características

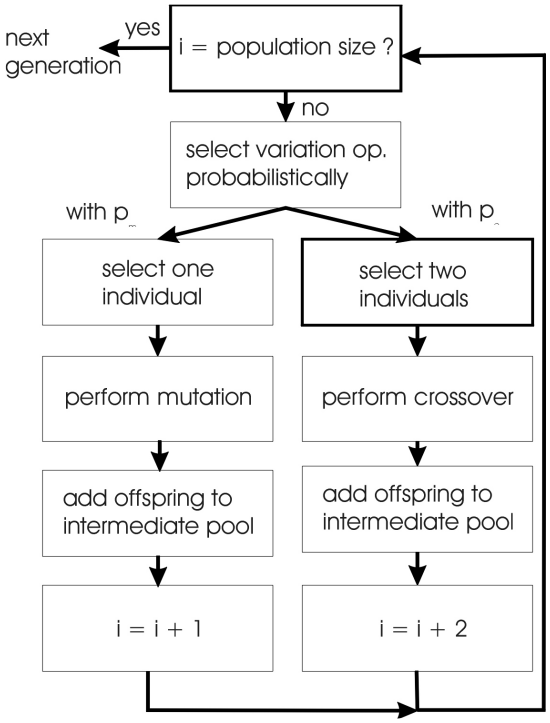
- Cromosomas representados por estructuras no lineales.
- Cromosomas de tamaño variable.
- Necesita poblaciones enormes (miles de individuos).
- Lentitud.
- Mutación posible pero no necesaria (discutible).

Programación genética

Algoritmo genético



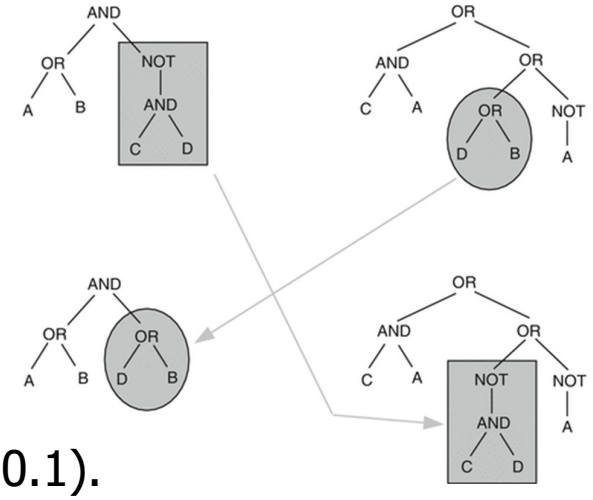
Programación genética



Programación genética

Cruce

Intercambio de subárboles



- Se selecciona un nodo aleatoriamente de cada uno de los padres (nodos internos con P=0.9, cualquiera con probabilidad P=0.1).

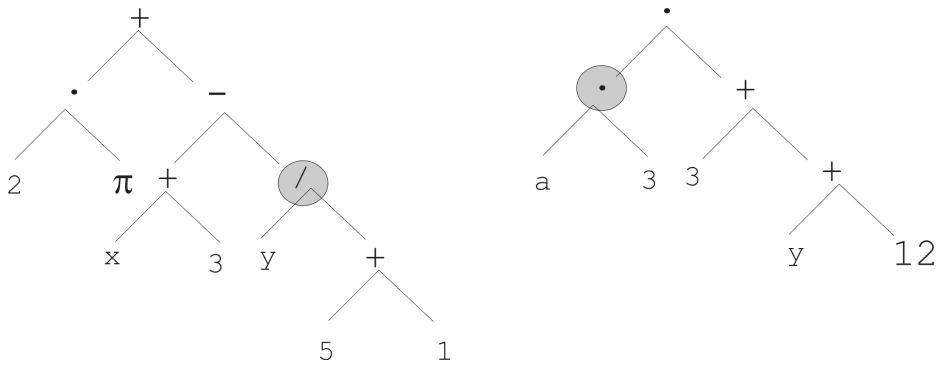
- Se intercambian los subárboles que tienen como raíces los nodos seleccionados.

NOTA: Siempre genera árboles sintácticamente válidos.

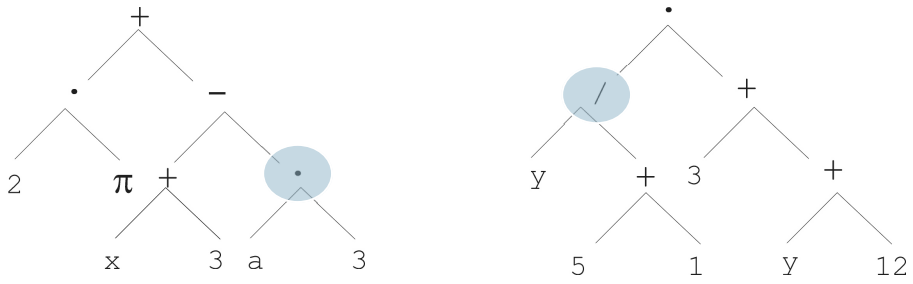
Programación genética

Cruce

Padres



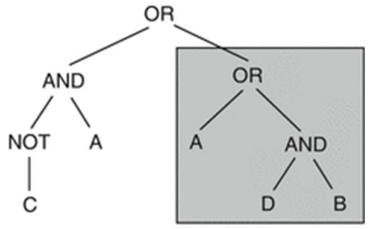
Hijos



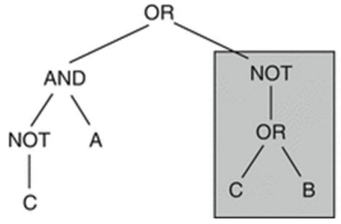
Programación genética

Mutación

Mutación de subárboles



- Se selecciona un nodo al azar (con probabilidad uniforme).

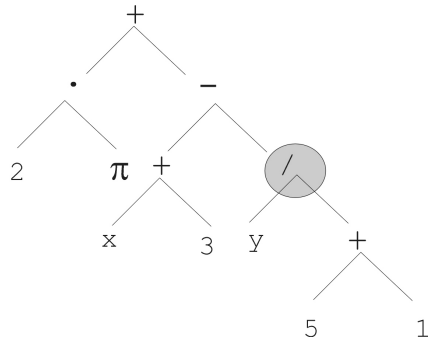


- Se elimina el subárbol que tiene como raíz el nodo seleccionado y se inserta un nuevo subárbol generado aleatoriamente.
- Un parámetro controla la profundidad máxima del subárbol generado aleatoriamente (el descendiente puede tener una profundidad mayor que el padre).

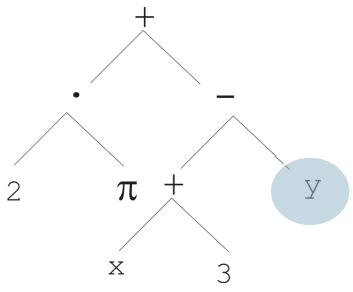
Programación genética

Mutación

Parámetros



- Probabilidad p_m de escoger mutación frente a recombinación.
  - $p_m = 0$ [Koza'1992]
  - $p_m = 0.05$ [Banzhaf et al. 1998]
- Probabilidad de escoger un nodo interno como raíz del subárbol reemplazado.



Programación genética

Selección

- Proporcional al fitness de cada individuo.
- Selección no uniforme en poblaciones grandes
  - Población dividida en dos grupos
    - Grupo 1: Mejor x% de la población
    - Grupo 2: El otro (100-x)% de la población

    x% distinto para poblaciones de distinto tamaño
    1000, 2000, 4000, 8000 → 32%, 16%, 8%, 4%
  - 80% de las operaciones de selección sobre el grupo 1 (20% restante del grupo 2).

Programación genética

Inicialización

- Árboles de profundidad máxima D_{max}
- 50% de árboles completos
  (cada rama del árbol con profundidad D_{max}).
  - Nodos a profundidad < D_{max} : Funciones.
  - Nodos a profundidad = D_{max} : Terminales.
- 50% de árboles de profundidad variable
  (ramas del árbol de profundidad $\leq D_{max}$).
  - Nodos a profundidad < D_{max} : Funciones y terminales.
  - Nodos a profundidad = D_{max} : Terminales.

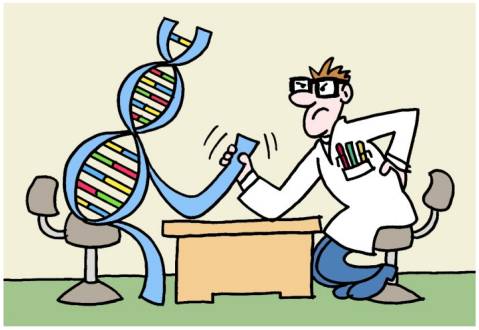
Programación genética

Abotargamiento [bloat]: "survival of the fattest"

- El tamaño de los árboles tiende a crecer
  a lo largo del tiempo.
- Se necesitan contramedidas para evitarlo:
  - Prohibición de operadores genéticos que den como resultado hijos demasiado grandes.
  - Presión para mantener la parsimonia, p.ej. penalización por ser demasiado grande.

Programación genética

Aplicaciones



- Regresión simbólica
- Controladores para robots
  - Sólo algunos árboles son realmente ejecutables.
  - La ejecución puede cambiar el entorno (fitness).
  - Cálculo del fitness mediante simulación (costosa).
  - Pese a todo, suele funcionar…

Programación genética

MetaGP

- Idea: Hacer evolucionar un sistema de programación genética usando programación genética.

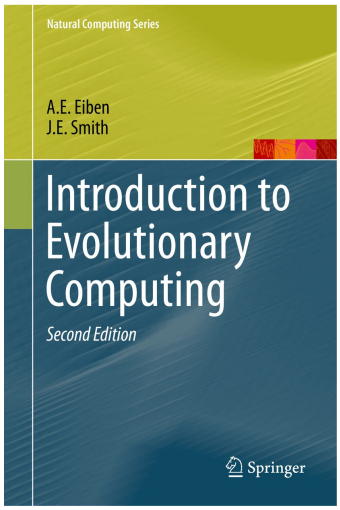
  Jürgen Schmidhuber, Dirk Dickmanns & Andreas Winklhofer (1987): Der genetische Algorithmus: Eine Implementierung in Prolog. Tech. Univ. Munich, 1987.
  http://people.idsia.ch/~juergen/geneticprogramming.html

- EURISKO ya hacía algo similar en 1983…

  Douglas Lenat (1983):
  "EURISKO: A program that learns new heuristics and domain concepts". Artificial Intelligence 21:61–98.
  DOI 10.1016/S0004-3702(83)80005-8

Bibliografía

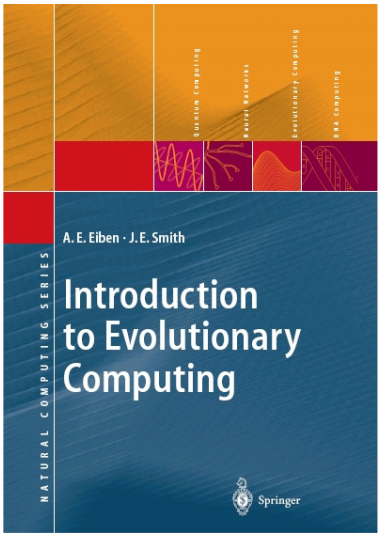
Lecturas recomendadas



- A.E. Eiben & J.E.Smith:
  Introduction to Evolutionary Computing
  Springer, 2nd edition, 2015
  ISBN 3662448734
  http://www.evolutionarycomputation.org/

Bibliografía

Lecturas recomendadas



- A.E. Eiben & J.E.Smith:
  Introduction to Evolutionary Computing
  Springer, 2nd printing, 2007
  ISBN 3540401849
  http://www.cs.vu.nl/~gusz/ecbook/ecbook.html

[Capítulo 3: Genetic algorithms]

Lecturas opcionales

[Capítulo 6: Genetic programming]

[Capítulo 8: Parameter control]

Bibliografía

Bibliografía en castellano

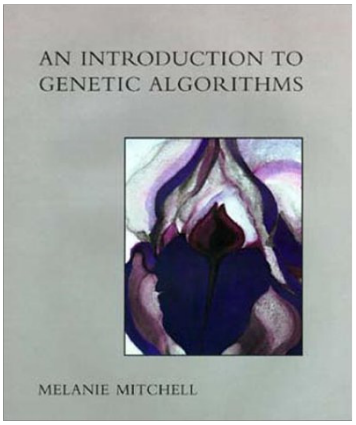
- Carlos Artemio Coello Coello:
  Introducción a la Computación Evolutiva.
  CINVESTAV-IPN, 2014.
  http://delta.cs.cinvestav.mx/~ccoello/compevol/apuntes.pdf

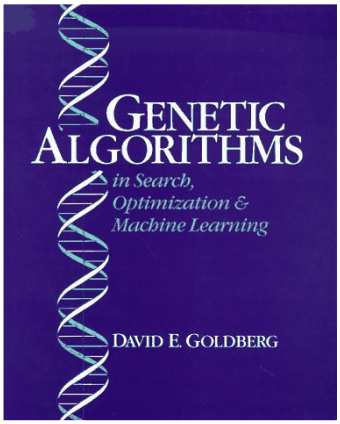
  [Capítulo 5: La importancia de la representación]
  [Capítulo 6: Técnicas de selección]
  [Capítulo 7: Técnicas de cruza (sic)]
  [Capítulo 8: Mutación]
  [Capítulo 9: Ajuste de parámetros]

Bibliografía

Bibliografía complementaria

- Melanie Mitchell:
  An Introduction to Genetic Algorithms
  MIT Press, 1996.
  ISBN 0262133164

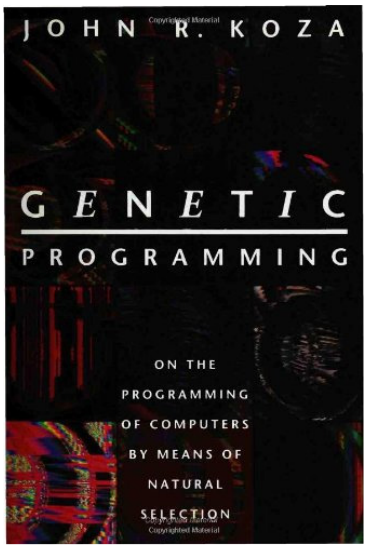




- David E. Goldberg:
  Genetic Algorithms in Search, Optimization & Machine Learning.
  Addison-Wesley, 1989.
  ISBN 0201157675

Bibliografía

Bibliografía complementaria



- John R. Koza:
  Genetic Programming:
  On the Programming of Computers by Means of Natural Selection.
  MIT Press, 1992.
  ISBN 0-262-11170-5