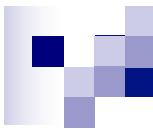


INTELIGENCIA ARTIFICIAL APLICADA AL CONTROL

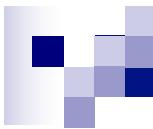
Tema 6: Introducción a los algoritmos genéticos

Dpto.: Arquitectura de Computadores y Automática
Autor: Matilde Santos

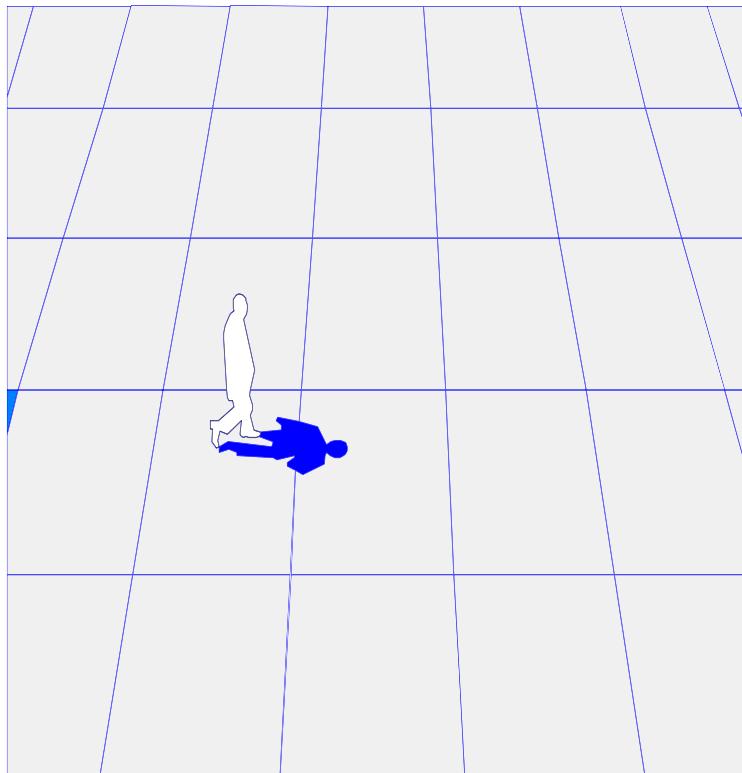


COMPUTACIÓN EVOLUTIVA

- La computación evolutiva (CE) es una estrategia de optimización que se basa en mecanismos de la evolución como la genética y la selección natural
- La CE tiene cuatro ámbitos principales:
 - Algoritmos genéticos
 - Programación evolutiva
 - Estrategias de evolución
 - Programación genética

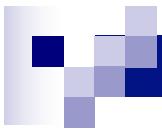


ALGORITMOS EVOLUTIVOS



“Los Algoritmos Genéticos son buenos para explorar espacios de búsqueda enormes y navegar por ellos, buscando combinaciones óptimas, soluciones que de otro modo no encontrarías en toda una vida”

- Salvatore Mangano
Computer Design, May 1995

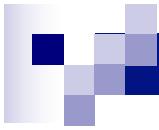


ALGORITMOS GENÉTICOS

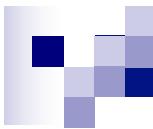
1.1. INTRODUCCIÓN

1.2. ALGORITMO GENÉTICO BÁSICO

1.3. EJEMPLOS

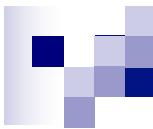


6.1 INTRODUCCIÓN



HISTORIA

- Fraser, 1950's, computadoras para simular sistemas naturales genéticos
- Bagley, 1967, Tesis doctoral, 1º en usar el término GA
- Fogel, 1960's, Programación evolutiva,
- Rechenberg, 1960's, Estrategias evolutivas
- Latane, Particle swarm optimization (Social impact theory)
- **Holland**, 1975, “Adaptación en sistemas naturales y artificiales”
- DeJong's Tesis: GAs, 1975
- **Goldberg**, “GA in search, optimization, and machine learning”, 1989
- Desde 1985, explosión de interés y utilización

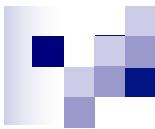


IDEA

Algoritmos de búsqueda basados en los principios de la evolución natural y adaptación genética

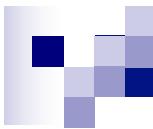
Un conjunto inicial de individuos evoluciona a lo largo de un número de generaciones mediante técnicas de reproducción y mutaciones para mejorar y así sobrevivir





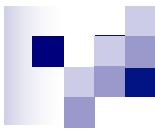
VENTAJAS

- Concepto fácil de entender
- Modular, separado de la aplicación
- Optimización multiobjetivo
- Bueno para entornos "ruidosos"
- En cualquier etapa da una solución
 - La respuesta mejora con el tiempo
 - Una buena solución aceptable en un tiempo razonable
- Inherentemente paralelo
- Fácilmente computación distribuida



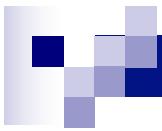
MÁS VENTAJAS

- Mantiene diversidad de soluciones
 - Fácil explotar soluciones anteriores o alternativas
- Procedimiento (de búsqueda) adaptativo
- No necesitan control directo del entorno
- No necesitan recorrer todo el espacio de búsqueda
- Técnica robusta
- Acelerar y mejorar con conocimiento sobre el dominio del problema
- Estrategias híbridas (Fuzzy+GA, GA+NN, etc)



INCONVENIENTES

- No garantía de encontrar la solución óptima
- Limitar el espacio de búsqueda a soluciones posibles (*feasibles*)
- Función de evaluación que refleje el problema a resolver
- No son algoritmos especializados en ningún problema en particular
- El resultado (satisfactorio) puede depender mucho del diseñador

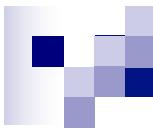


APLICACIONES

Para problemas cuya complejidad no permita una solución directa

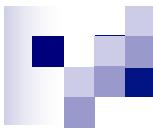
- y sea suficiente una solución aceptable -

- Optimización
- Problemas de búsqueda
- Identificación de parámetros
- Reconocimiento
- Aprendizaje máquina



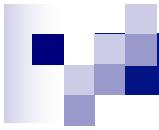
ÁMBITOS

- Robótica
- Planificación
- Optimización: conexionado de circuitos, de redes neuronales, etc.
- Simulación software
- Diseño e implementación de hardware
- Identificación
- Búsqueda en BBDD
- etc

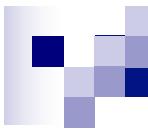


EJEMPLOS DE APLICACIÓN

Domain	Application Types
Control	gas pipeline, pole balancing, missile evasion, pursuit
Design	semiconductor layout, aircraft design, keyboard configuration, communication networks
Scheduling	manufacturing, facility scheduling, resource allocation
Robotics	trajectory planning
Machine Learning	designing neural networks, improving classification algorithms, classifier systems
Signal Processing	filter design
Game Playing	poker, checkers, prisoner's dilemma
Combinatorial Optimization	set covering, travelling salesman, routing, bin packing, graph colouring and partitioning

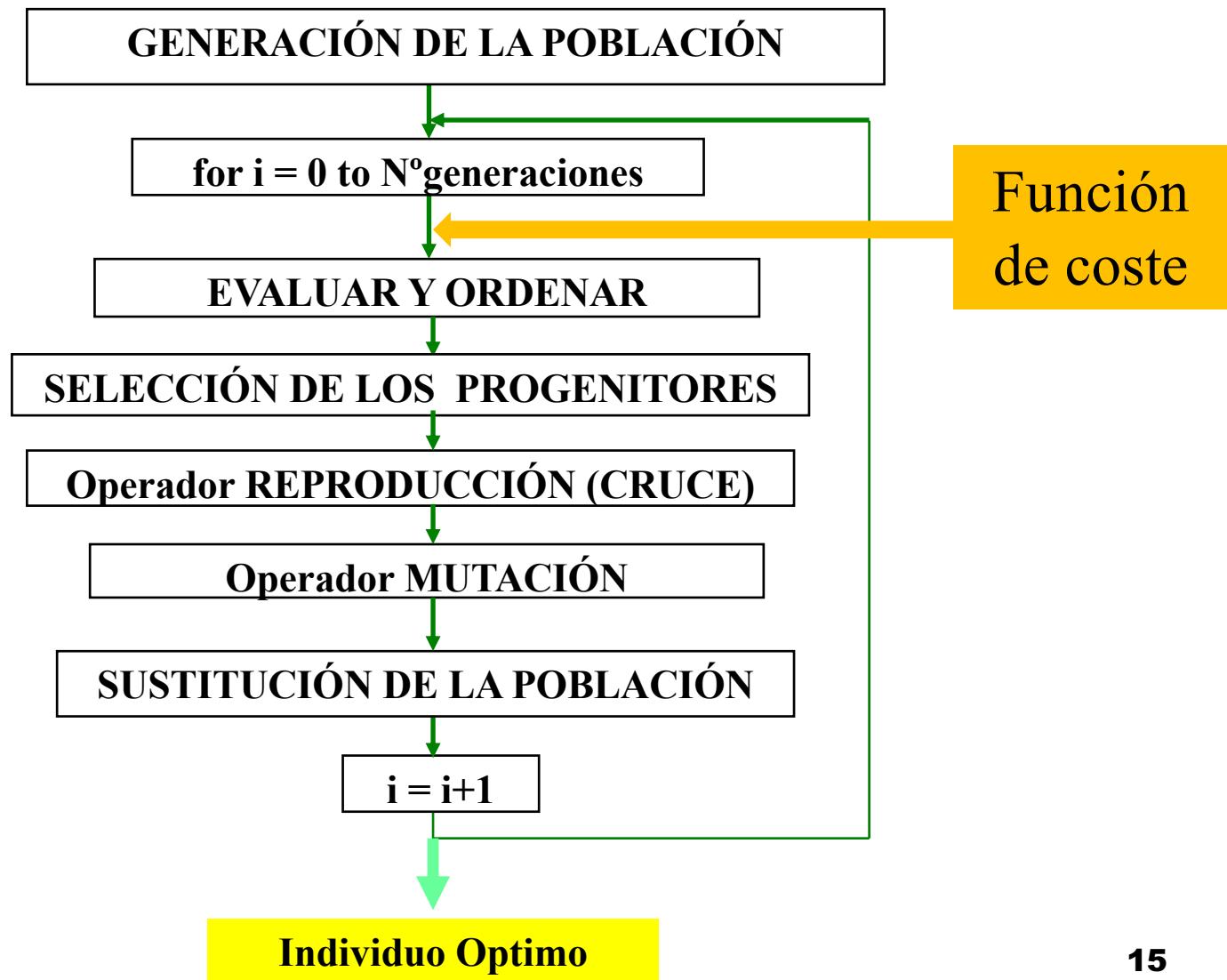


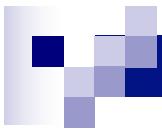
6.2 ALGORITMO GENÉTICO BÁSICO



ALGORITMO GENÉTICO BÁSICO

Es un algoritmo:
trozo de código
con instrucciones para la resolución de un problema





GA PSEUDO-CÓDIGO

{

 initialize population;

 evaluate population;

 while TerminationCriteriaNotSatisfied

 {

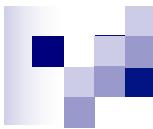
 select parents for reproduction;

 perform recombination and mutation;

 evaluate population;

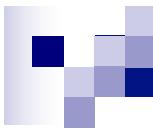
 }

}



ALGORITMO GENÉTICO SIMPLE

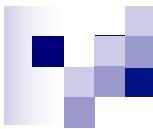
- El GA de Holland se conoce como “Simple Genetic Algorithm” (SGA)
- Otros GAs utilizan diferentes:
 - Representaciones
 - Mutaciones
 - Cruces
 - Mecanismos de selección



COMPONENTES DE UN GA

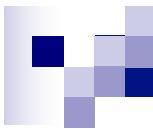
- Un problema a resolver, y ...

- Codificación (gen, cromosoma)
- Inicialización (creación)
- Función de evaluación (dominio)
- Selección de los padres (reproducción)
- Operadores genéticos (mutación, recombinación)
- Ajustes de parámetros (práctica y arte)



POBLACIÓN INICIAL

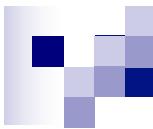
- Población de n individuos
 - cada individuo es una posible solución
- Individuo = estructura de datos (cadenas de caracteres o cromosomas de un alfabeto Φ)
 - CROMOSOMA $A = a_1 \ a_2 \ a_3 \dots a_m, \ a_i \in \Phi$
 - GEN: cada elemento del cromosoma, a_i
 - Alelo: valor del gen
 - Locus: posición en la cadena
- Tamaño: suficiente para cubrir el espacio de soluciones



REPRESENTACIÓN

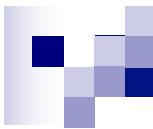
- Codificación en cadenas binarias
 - Facilidad de operación
 - Mayor descomposición de las características del problema
 - Alfabeto de menor cardinalidad (búsqueda más sencilla)
 - Convergencia de los algoritmos probada
 - Variantes (BCD, código Gray, etc)

- Otros tipos de representaciones
 - Alfabetos de distinta cardinalidad



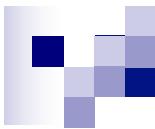
INICIALIZACIÓN

- Tamaño:
 - Comenzar con una población de tamaño moderado (50-500)
 - El tamaño de la población tiende a aumentar linealmente con la longitud del individuo (no exponencialmente)
- Aleatoria:
 - Población lo más dispersa posible, cubrir todo el espacio
 - Prueba del funcionamiento del algoritmo
- Heurística (incluir valores prometedores):
 - Garantizar la diversidad de las soluciones (no sesgar)
 - Evitar la convergencia prematura del algoritmo
 - Incorporar restricciones



EVALUACIÓN

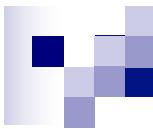
- Función objetivo, de coste, de ajuste o evaluación (*fitness*)
 - Cada individuo es evaluado (cómo es de bueno)
 - Grado de acercamiento de ese individuo a la solución (aptitud, bondad o mérito)
 - Probabilidad de supervivencia de ese individuo
 - Directamente relacionada con el problema que se quiere resolver
 - Función propia para cada problema particular
 - Normalización
 - Mejor discriminación



REPRODUCCIÓN

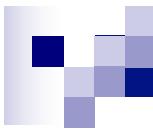
- SELECCIÓN DE LOS PADRES
- OPERADOR GENÉTICO:
 - CRUCE
 - MUTACIÓN

Los operadores genéticos mejoran significativamente la búsqueda



SELECCIÓN DE LOS PADRES

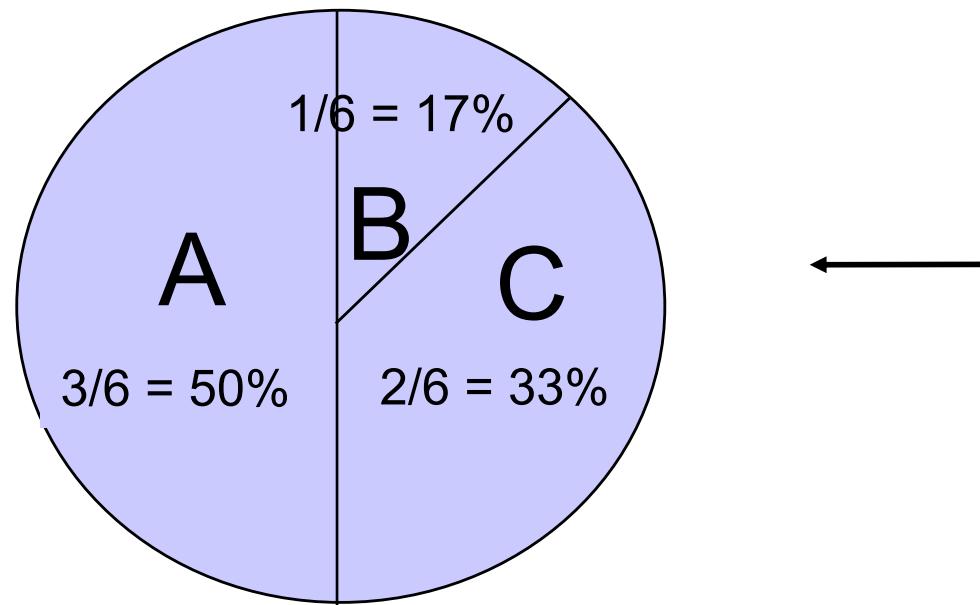
- Mantener las características más ventajosas de la población
 - Selección aleatoria, según la aptitud de cada parente
 - Los mejores individuos, mayor probabilidad
 - Probabilidades proporcionales a la aptitud
 - Métodos:
 - Valor de un individuo/suma de los valores objetivos de la población
 - Método de la ruleta (proporcional a la aptitud)



SELECCIÓN

■ Método de la ruleta

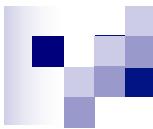
- Asignar a cada individuo una porción de la ruleta proporcional a su aptitud
- Girar la rueda n veces para seleccionar n individuos



$$\text{fitness}(A) = 3$$

$$\text{fitness}(B) = 1$$

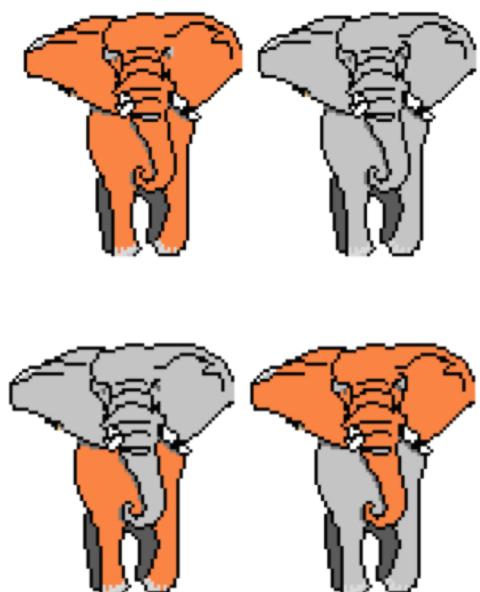
$$\text{fitness}(C) = 2$$

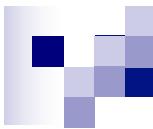


REPRODUCCIÓN: CRUCE

■ El cruce es una característica crítica de los AG

- Acelera la búsqueda en la evolución de una población
- Combinación efectiva de esquemas (sub-soluciones de diferentes cromosomas)
- A menudo se comienza con una tasa de cruce relativamente alta y se reduce durante la ejecución

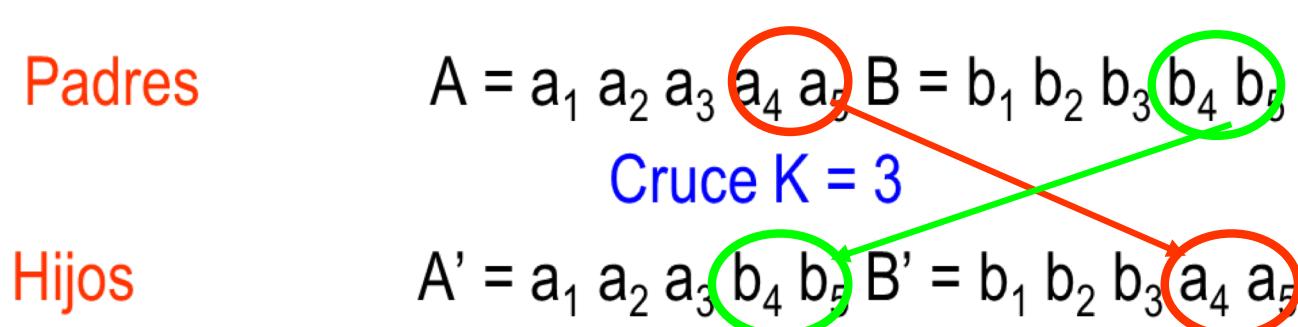


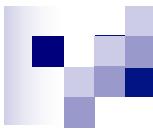


CRUCE: RECOMBINACIÓN

OPERADOR CRUCE DE UN PUNTO

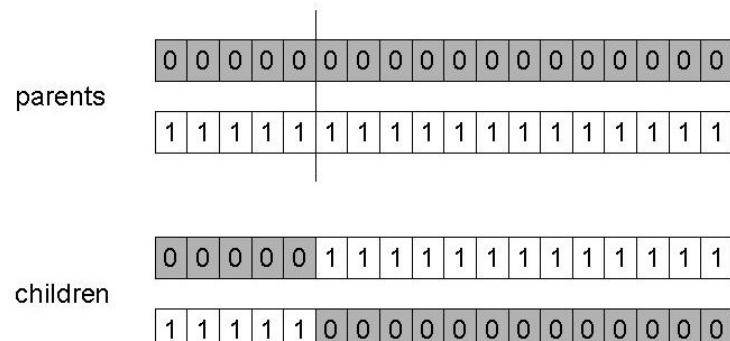
- Selección aleatoria de pares de individuos (dos a dos) cada par con una probabilidad de cruce, P_C
- A cada par seleccionado se le aplica un intercambio de contenido desde una posición aleatoria K hasta el final, con $K \in [1, m-1]$

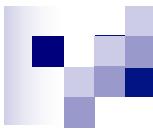




OTRAS IMPLEMENTACIONES DEL OPERADOR CRUCE

- Cruce de dos puntos:
 - Intercambio entre dos posiciones intermedias
- Cruce uniforme
 - En cada bit se elige al azar un parente para que contribuya con su bit al del hijo
 - El 2º hijo recibe ese bit del otro progenitor

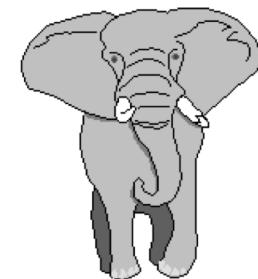


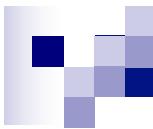


REPRODUCCIÓN: MUTACIÓN

■ Mutación es una ayuda al AG

- La población puede no evolucionar hacia la solución buscada
- Causa movimiento en el espacio de búsqueda (local o global)
- Introducir diversidad en la población
- Restaura la información perdida a la población
- Aumenta el espacio de exploración de soluciones
- Evitar óptimos locales





MUTACIÓN: LOCAL

- Alteración aleatoria de cada gen de un individuo con una probabilidad P_m
 - P_m no muy alta para evitar oscilaciones en el promedio
 - Constante o se aumenta cuando la variabilidad de la aptitud cae por debajo de cierto umbral

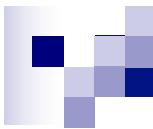
parent

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

child

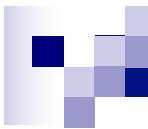
0	1	0	0	1	0	1	1	0	0	0	1	0	1	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

En la representación binaria, inversión: si es 1 pasa a ser 0 y viceversa

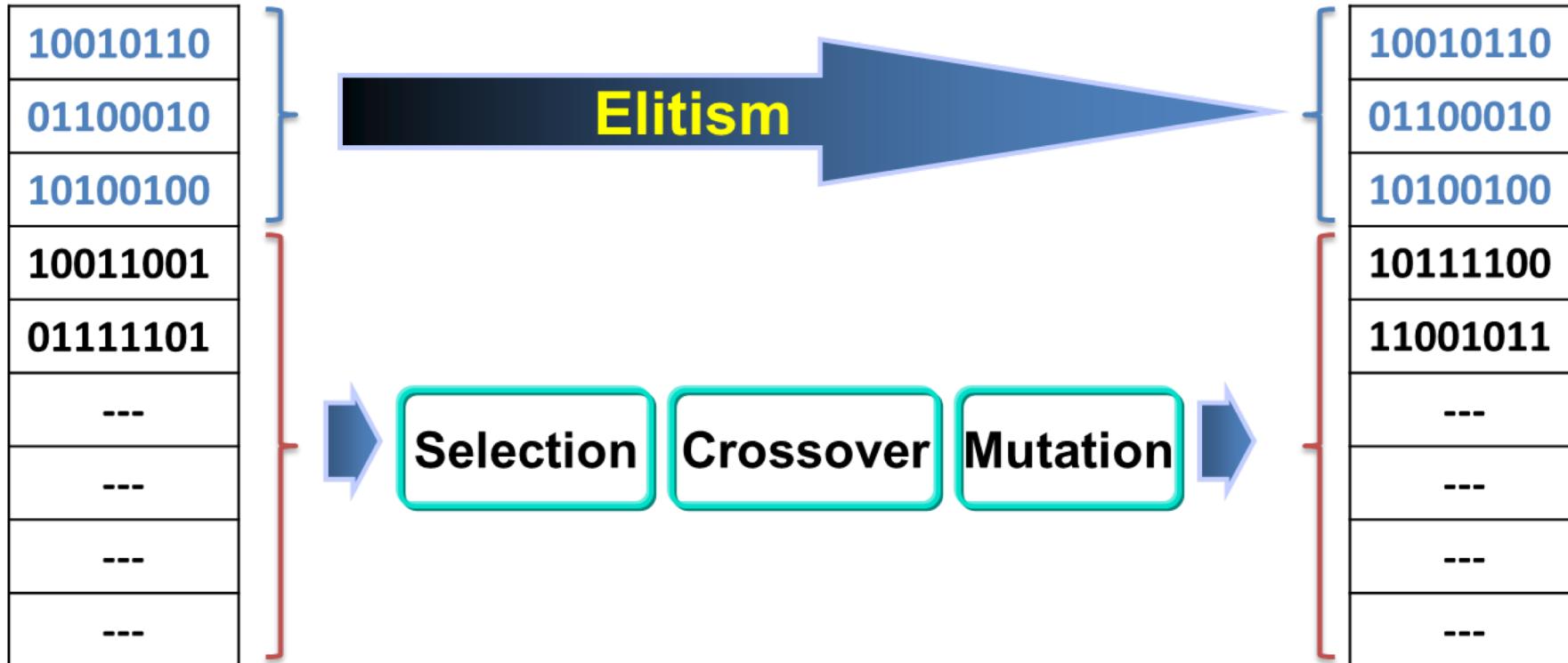


NUEVA POBLACIÓN

- Proceso iterativo (cada generación)
- Nueva población
 - Sustitución generacional
 - Toda la población por los hijos
 - Generational gap: reemplazar un % (los peores)
 - Elitismo:
 - Mejor/mejores seleccionados directamente
 - Nuevos individuos generados aleatoriamente
 - Para completar la población inicial y mantener la diversidad



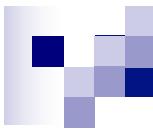
REEMPLAZO POBLACIÓN



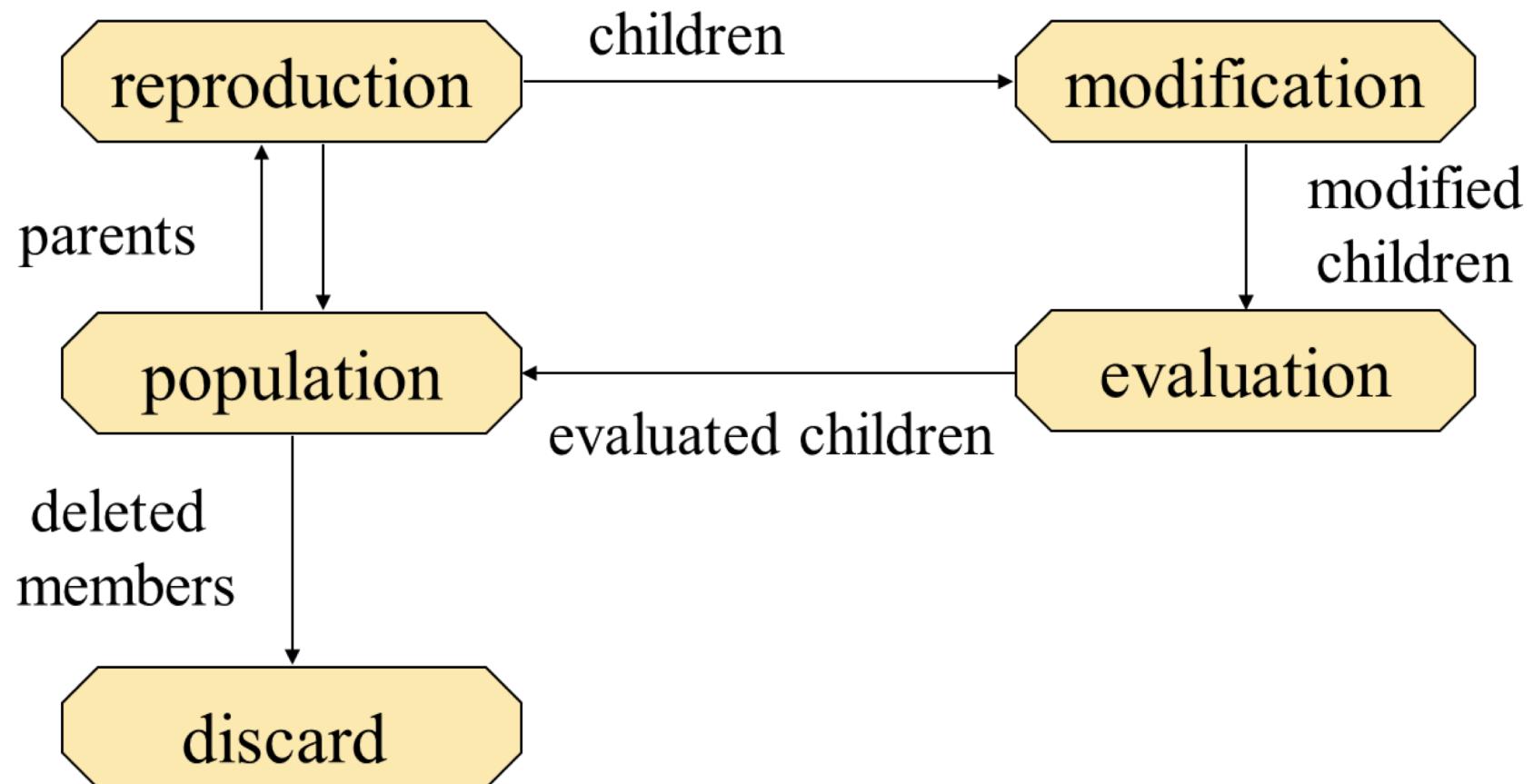
**Current
generation**

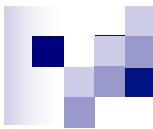
**Next
generation**

- La población suele mantener el mismo tamaño



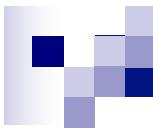
CICLO DE REPRODUCCIÓN





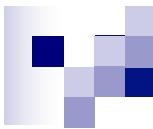
CONDICIÓN DE PARADA

- Tiempo computacional limitado
- Número limitado de generaciones (iteraciones)
 - Depende de la complejidad del problema
- Valor suficientemente bueno de la solución
 - Convergencia: un gen ha convergido cuando el 95% de la población comparte el mismo valor
 - Cuando la solución converge a un valor suficientemente bueno ($>$ aptitud especificada)
 - No hay cambios en el valor de la aptitud durante m generaciones



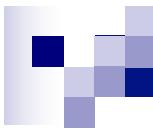
RESUMEN SGA

Representación	Cadenas binarias
Recombinación	N-punto o uniforme
Mutación	Bit-bit-flipping con probabilidad fija
Selección de padres	Proporcional a la aptitud
Selección de supervivientes	Todos los hijos reemplazan a los padres
Característica	Énfasis en el cruce



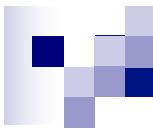
RESUMEN PROCESO GA

1. Creación e inicialización (aleatoria) de la población inicial
2. Seleccione probabilidad de cruce y mutación
3. Evaluar (aptitud) cada individuo de la población
4. Normalizar los valores de aptitud y usarlos para determinar las probabilidades de reproducción
5. Emparejar los padres para cruzarlos aleatoriamente
6. Seleccione los puntos de cruce (a menudo 2) aleatoriamente para cada par
7. Reproducir la nueva generación con el mismo número de miembros según las probabilidades
8. Mutación gen a gen
9. Para la siguiente generación ir al paso 2
10. Si se cumple terminación, mostrar los resultados



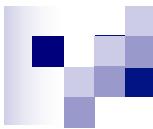
CUESTIONES

- Representación
- Tamaño de la población, tasa de mutación, ...
- Selección, políticas de eliminación
- Crossover, operadores de mutación
- Criterios de terminación
- Rendimiento, escalabilidad
- La solución es tan buena como lo es la función de evaluación (a menudo la parte más difícil)



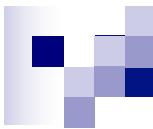
PARÁMETROS

- Estructura inicial de la población:
 - Tamaño de la población: número de individuos
 - Número de generaciones
 - Longitud de los individuos (tamaño cromosomas)
- Planificar la evolución
- Operadores genéticos
 - Probabilidad de cruce
 - en torno a 0.5; $P_c = 0.6-0.9$
 - Probabilidad de mutación
 - Típicamente entre 1/tamaño población y 1/longitud cromosoma (0.01)



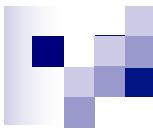
A.G. DISTRIBUIDOS

- Problemas complejos
 - Tamaño de población muy grande (tiempo de ejecución del algoritmo muy alto)
- Mutación adaptativa:
 - Incrementarla cuando la población es muy homogénea y disminuirla al decrecer la homogeneidad
- A.G. Distribuidos
 - Dividir una población grande en subpoblaciones



An example after Goldberg '89

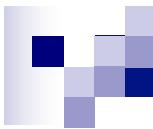
- Simple problem: $\max x^2$ over $\{0,1,\dots,31\}$
- GA approach:
 - Representation: binary code, e.g. 01101 \leftrightarrow 13 (2^5)
 - Population size: 4 individuals
 - 1-point xover, bitwise mutation
 - Roulette wheel selection
 - Random initialization
- We show one generational cycle done by hand



χ^2 Example: Selection

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

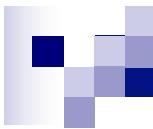
*Expected count: probabilidad de cruzarse (redondea)



X² example: crossover

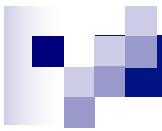
String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 1	4	0 1 1 0 0	12	144
2	1 1 0 0 0	4	1 1 0 0 1	25	625
2	1 1 0 0 0	2	1 1 0 1 1	27	729
4	1 0 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

*Crossover point: $k_1 = 4$, $k_2 = 2$



χ^2 example: mutation

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



X² example: una generación

	Inicial	Cruce	Mutación
Sum	1170	1754	2354
Average	293	439	588,5
Max	576	729	729