Tema 5: Algoritmos Genéticos

José Luis Ruiz Reina
Miguel A. Gutiérrez Naranjo

Departamento de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Inteligencia Artificial

Búsqueda Local

- Algoritmos de búsqueda de soluciones óptimas
  - Buscar la mejor solución dentro de un espacio de posibles soluciones
  - Maximizar o minimizar
- Mejoras iterativas
  - Empezar con un estado inicial cualquiera
  - Mejorar su calidad paso a paso
- Algoritmos:
  - Escalada
  - Escalada con reinicio aleatorio
  - Enfriamiento simulado
  - Algoritmos genéticos
- Ninguno de estos algoritmos ofrece completitud, pero a veces es la única aproximación en la práctica

Problemas de optimización

- Estado inicial: no está claramente definido
- Operadores:
  - Se puede definir cierta noción de nodo “sucesor” o “vecino”
  - En algunos casos, gran cantidad de vecinos
  - Introducimos cierta componente heurística y aleatoria, generando cada vez un único nodo “nodo vecino”
- Estados finales y soluciones:
  - Todos los estados son posibles soluciones, pero se trata de encontrar una solución “buena” (cuantificada por su función de valoración)
  - Si es posible, la mejor
  - Se busca el estado con un óptimo valor (máximo o mínimo)
  - No buscamos la secuencia de operadores (los estados contienen toda la información)

Algoritmos genéticos: evolución natural

- Optimización inspirada en los procesos evolutivos de la naturaleza:
  - La evolución ocurre en los cromosomas de los individuos
  - Las “buenas estructuras” sobreviven con más probabilidad que las demás
  - El nuevo material genético se obtiene mediante cruces y mutaciones
- Algoritmos genéticos:
  - Aplicación de estas ideas en la búsqueda de soluciones óptimas
  - No existe un único algoritmo genético
  - Es una denominación para este tipo de algoritmos evolutivos

Algoritmos genéticos: codificación del problema original

- Un primer paso es representar los estados del problema original como individuos de una población
  - Genes: material genético básico
  - Cromosomas: secuencia de genes que codifica a un estado del problema original
  - Población: conjunto de cromosomas (sólo un subconjunto de tamaño “manejable”)
  - Se trata de evitar los óptimos locales manejando una población de candidatos, en lugar de un único candidato
  - La población evoluciona en las distintas *generaciones*
  - Genotipo (el cromosoma) *vs* fenotipo (a quién representa el cromosoma)
- Bondad de los individuos
  - Según el valor de la función de valoración (usualmente llamada *fitness*)

Algoritmos genéticos: representación del problema

- Problemas de optimización: un ejemplo simple
  - Ejemplo: encontrar el mínimo de la función $f(x) = x^2$ en $[0, 2^{10}) \cap N$
- Variables `*GENES*` y `*LONGITUD-INDIVIDUOS*`
  - Ejemplo en la función cuadrado: `[0, 1]` y `10`, resp.
- Función `DECODIFICA(X)`, obtiene el *fenotipo*
  - En la función cuadrado: un cromosoma puede verse como un número binario de 10 dígitos (en orden inverso). El fenotipo de un cromosoma es dicho número (en notación decimal)
  - Ejemplo: `(0 1 1 0 0 1 0 0 0 0)` es un cromosoma que representa al 38
- Función `FITNESS(X)`, valor de de la función a optimizar (actuando sobre el genotipo)
  - Ejemplo en la función cuadrado: la función que recibiendo la representación binaria de un número, devuelve su cuadrado

Esquema general de un algoritmo genético

```
INICIAR población
EVALUAR cada individuo de la población

Repetir hasta CONDICIÓN_DE_TERMINACIÓN
    SELECCIONAR padres
    COMBINAR pares de padres
    MUTAR hijos resultantes
    EVALUAR nuevos individuos
    SELECCIONAR individuos para la siguiente generación

Devolver el mejor de la última generación
```

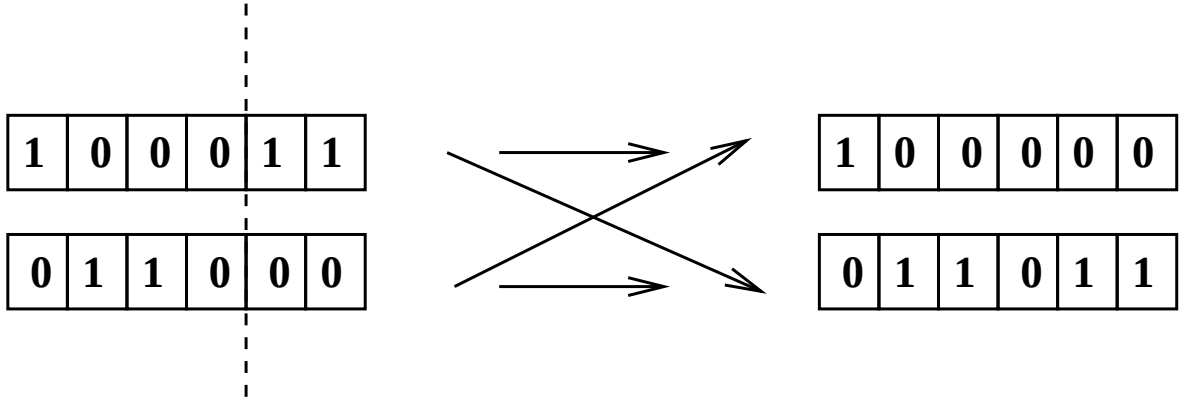
Ejemplo de ejecución (minimizando la función cuadrado)

```
>>> algoritmo_genetico_v2_con_salida(cuad_gen, 20, 10, 0.75, 0.6, 0.1)

 Generacion: 1. Media: 361954.9, Mejor: (0 1 1 0 0 1 0 0 0 0), Valor: 1444
 Generacion: 2. Media: 79730.6, Mejor: (0 1 1 0 0 1 0 0 0 0), Valor: 1444
 Generacion: 3. Media: 22278.6, Mejor: (0 1 1 0 0 1 0 0 0 0), Valor: 1444
 Generacion: 4. Media: 3537.7, Mejor: (1 1 1 1 0 0 0 0 0 0), Valor: 225
 Generacion: 5. Media: 1597.3, Mejor: (0 0 1 1 0 0 0 0 0 0), Valor: 144
 Generacion: 6. Media: 912.8, Mejor: (0 1 0 0 0 0 0 0 0 0), Valor: 4
 Generacion: 7. Media: 345.3, Mejor: (0 1 0 0 0 0 0 0 0 0), Valor: 4
 Generacion: 8. Media: 60.7, Mejor: (0 1 0 0 0 0 0 0 0 0), Valor: 4
 Generacion: 9. Media: 14.0, Mejor: (0 1 0 0 0 0 0 0 0 0), Valor: 4
 Generacion: 10. Media: 4.5, Mejor: (0 1 0 0 0 0 0 0 0 0), Valor: 4
 Generacion: 11. Media: 3.7, Mejor: (1 0 0 0 0 0 0 0 0 0), Valor: 1
 Generacion: 12. Media: 3.4, Mejor: (1 0 0 0 0 0 0 0 0 0), Valor: 1
 Generacion: 13. Media: 2.4, Mejor: (0 0 0 0 0 0 0 0 0 0), Valor: 0
 ....
```

Combinación de individuos

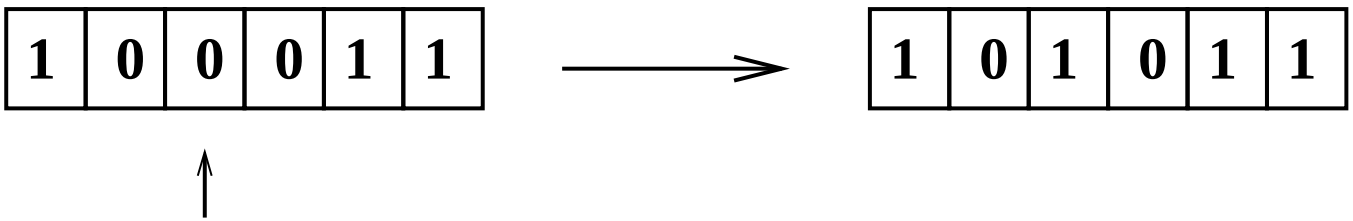
- Operadores que combinan la información de los *padres* para obtener nuevos *hijos*
- Cruce en un punto:



  - Aleatoriedad: al elegir el punto de cruce
- Otras posibilidades:
  - Cruce multipunto (varios segmentos de intercambio)
  - Cruce uniforme (para cada posición del hijo, *sorteamos* de quién hereda)
  - Cruces específicos de la representación (p.ej. permutaciones)

Mutación de individuos

- Mutaciones:



- Aleatoriedad:
  - Con una determinada probabilidad (usualmente baja) cambiar algunos genes
  - Si se cambia, elegir aleatoriamente a qué gen se cambia
- Variantes:
  - Específicas de la representación (p.ej. permutaciones)

Permutaciones

- En caso en que el cromosoma sea una permutación de genes, es necesario tener operadores específicos de mutación y cruce
- Mutación por intercambio:

- Mutación por inserción:

- Mutación por mezcla:

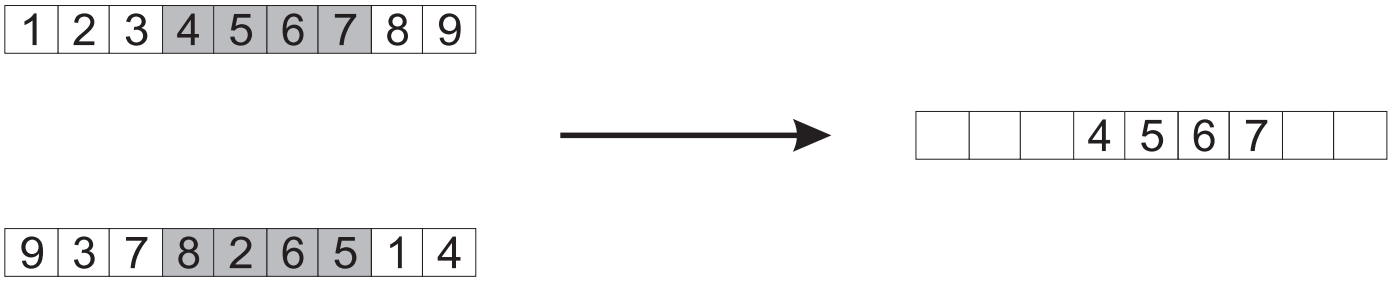
Permutaciones

Cruce basado en orden

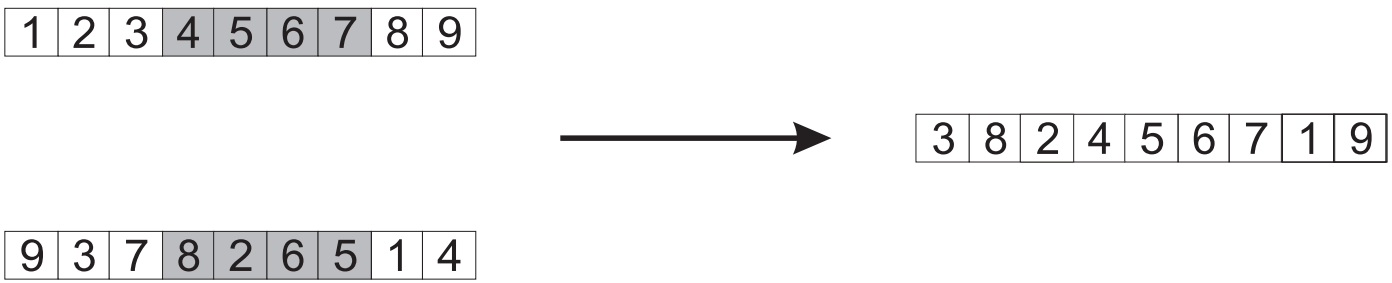
- Elige dos puntos de corte aleatoriamente del primer padre y copia el segmento entre ellos en el primer hijo.
- A partir del segundo punto de corte en el segundo padre, copia los genes no usados en el primer hijo en el mismo orden en que aparecen en el segundo padre, volviendo al principio de la lista si fuera necesario.
- Crea el segundo hijo de manera análoga, intercambiando el rol de los padres.

Permutaciones

Cruce basado en orden (Paso 1):



Cruce basado en orden (Paso 2):



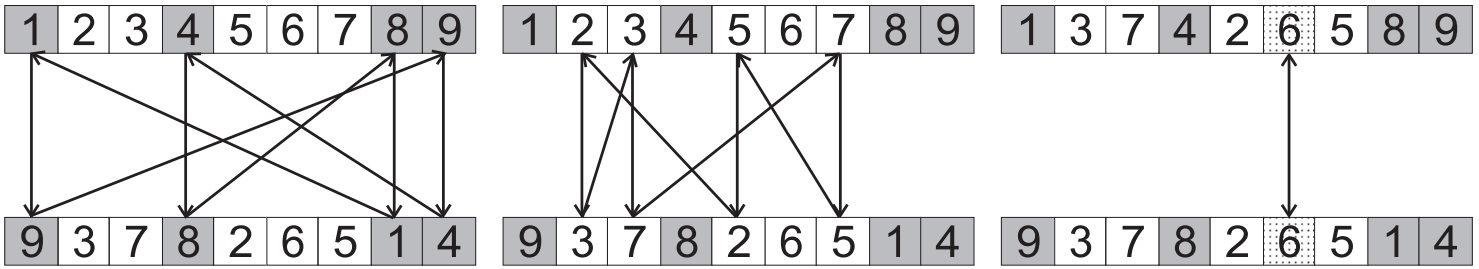
Permutaciones

Cruce basado en ciclos: Dividimos la permutación en ciclos y alternamos los ciclos de cada padre.
Para construir ciclos:

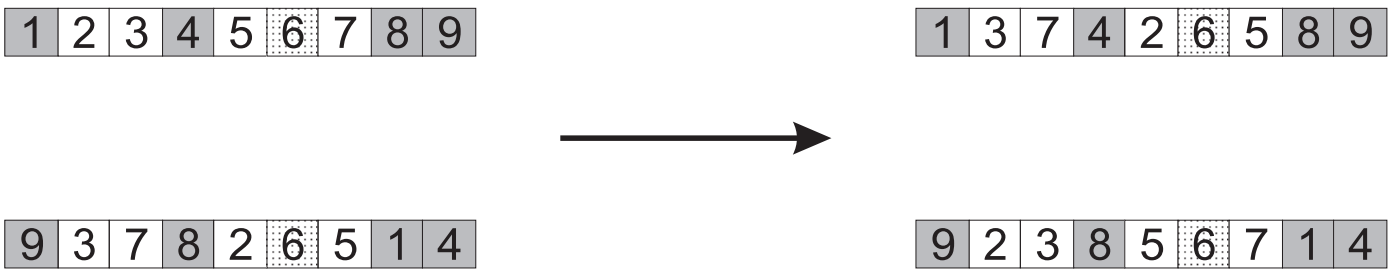
- Tomamos la primera posición *nueva* en el padre $\boldsymbol{P_1}$.
- Buscamos el gen en la misma posición de $\boldsymbol{P_2}$.
- Vamos a la posición con el mismo gen en $\boldsymbol{P_1}$.
- Añadimos este gen al ciclo.
- Repetimos los apsos del 2 al 4 hasta que lleguemos al primer gen de $\boldsymbol{P_1}$.

Permutaciones

Cruce basado en ciclos (Identificación de ciclos):



Cruce basado en ciclos (Construcción de hijos):



Mecanismos de selección

- Un algoritmo genético debe tener un método para seleccionar individuos de una población:
  - Para obtener aquellos individuos que van a ser usados como padres
  - A veces, también para decidir qué hijos pasan a la siguiente generación
- La selección debe:
  - Favorecer a los mejores (según su valoración))
  - Permitir la diversidad
  - Usualmente tiene una componente aleatoria
- Ejemplos de selección:
  - Proporcional a su valoración
  - Torneo
  - Élite + aleatoriedad

Selección proporcional a la valoración

- Idea:
  - Seleccionar aleatoriamente, pero de manera que cada individuo tenga una probabilidad de ser seleccionado proporcional a su valoración respecto de las valoraciones del total de la población
  - Los mejores individuos se seleccionarán más frecuentemente
- La probabilidad de que cada individuo i sea seleccionado es

$$P(i) = \frac{F(i)}{\sum_{j=1}^{n} F(j)}$$

  - Importante: con este método de selección sólo podemos resolver problemas de maximización
  - Si es de minimización habría que transformar la función de *fitness*
- Variante: selección por *ranking*
  - La probabilidad asignada es proporcional a su posición en la población (ordenada por *fitness*)

Selección probabilística

- Implementación de sorteos con probabilidad: ruleta
  - Calcular la suma total acumulada de los valores de la función de *fitness* de todos los miembros de la población
  - Generar un número aleatorio ***x*** entre 0 y la suma total anterior
  - Recorrer la población, nuevamente acumulando los valores de la función *fitness* y seleccionando el primer cromosoma cuya suma acumulada sea mayor que ***x***

Ejemplo del método de la ruleta

- Población de 5 individuos $\{\boldsymbol{i1}, \boldsymbol{i2}, \boldsymbol{i3}, \boldsymbol{i4}, \boldsymbol{15}\}$ con valores $\boldsymbol{F(i1) = 2}$, $\boldsymbol{F(i2) = 7}$, $\boldsymbol{F(i3) = 1}$, $\boldsymbol{F(i4) = 4}$, $\boldsymbol{F(i5) = 6}$.
- Calculamos las umas acumuladas:
  $\boldsymbol{Ac(i1) = 2}$,
  $\boldsymbol{Ac(i2) = 2 + 7 = 9}$,
  $\boldsymbol{Ac(i3) = 2 + 7 + 1 = 10}$,
  $\boldsymbol{Ac(i4) = 2 + 7 + 1 + 4 = 14}$,
  $\boldsymbol{Ac(i5) = 2 + 7 + 1 + 4 + 6 = 20}$.
- Para seleccionar cuatro individuos tomamos cuatro valores aleatorios entre 0 y 20: 7, 13, 17, 5
- Seleccionamos los individuos $\boldsymbol{i2}$, $\boldsymbol{i4}$, $\boldsymbol{i5}$ e $\boldsymbol{i2}$ (nótese que se pueden repetir).

Selección por torneo y élite

- Selección por torneo:
  - Para cada selección, extraer aleatoriamente ***k*** individuos y seleccionar el mejor
  - Ventajas: no depende de la magnitud de la función de la valoración y se puede aplicar tanto a minimización como a maximización
  - Cuanto mayor el ***k*** usado, mayor es la *presión evolutiva*
- Selección elitista:
  - Escoger directamente un porcentaje de los mejores
  - Para introducir diversidad, el resto, seleccionarlos aleatoriamente de entre el resto

Otras componentes de un algoritmo genético

- Población inicial
  - Usualmente se crean ***N*** individuos de manera completamente aleatoria
- Terminación del algoritmo:
  - Hasta completar un número dado de generaciones
  - Cuando se hayan completado un número dado de generaciones sin mejorar
  - Hasta un valor prefijado de la función de valoración
- Diversos parámetros:
  - Tamaño de la población
  - Proporción de padres
  - Probabilidades de cruce y/o mutación

Ejemplo de algoritmo genético

```
t := 0
Inicia-Población P(t)
Evalúa-Población P(t)

Mientras t < N-Generaciones hacer
   P1   := Selección por torneo de (1-r)·p  individuos de P(t)
   P2   := Selección por torneo de (r·p) individuos de P(t)
   P3   := Cruza P2
   P4   := Union de P1 y P3
   P(t+1) := Muta P4
   Evalua-Población P(t+1)
   t:= t+1
Fin-Mientras

Devolver el mejor de P(t)
```

- Selección mediante torneo
- Parámetros del algoritmo: tamaño de la población, número de generaciones, proporción de padres (**r**), probabilidad de mutación

Otro ejemplo de algoritmo genético

```
t := 0
Inicia-Población P(t)
Evalúa-Población P(t)

Mientras t < N-Generaciones hacer
   P1   := Selecciona-Mejores P(t)
   P2   := Selecciona-aleatorio (P(t) - P1)
   P3   := Cruza (P1 U P2)
   P4   := Muta P3
   Evalua-Población P4
   P(t+1) := Selecciona-Mejores P4,P(t)
   t:= t+1
Fin-Mientras

Devolver el mejor de P(t)
```

- Selección combinada élite y aleatoriedad
- Para la siguiente generación, se toman los mejores de entre los hijos y los individuos originales
- Parámetros del algoritmo: tamaño de la población, número de generaciones, proporción de padres, proporción de mejores entre los padres, probabilidad de mutación

Ejemplos: problema del viajante

- Genes y longitud de los individuos en el problema del viajante
  - `*GENES* = [almeria, cadiz, cordoba, granada, huelva, jaen, malaga, sevilla]`
  - `*LONGITUD-INDIVIDUOS* = 8`
- Decodificación en el problema del viajante:
  - `DECODIFICA(X)= X`
- `FITNESS(X)`:
  - En principio, distancia del circuito que representa el cromosoma
  - Si usamos cruces y mutaciones estándar: habría que introducir una penalización en los individuos con genes repetidos
  - Una opción mejor: diseñar cruces y mutaciones que a partir de permutaciones obtengan permutaciones

Ejemplos: problema del cuadrado mágico

- Colocar en un cuadrado $n \times n$ los números naturales de **1** a n^2, de tal manera que las filas, las columnas y las diagonales principales sumen los mismo
- Casos $n = 3$ y $n = 4$:

4	3	8
9	5	1
2	7	6

7	14	9	4
16	2	5	11
1	15	12	6
10	3	8	13

- Soluciones representadas como permutaciones de números entre **1** y n^2
- Genes y longitud de los individuos:
  - `*GENES* = (1 2 3 ...n`^{2}`)`
  - `*LONGITUD-INDIVIDUOS* = n`^{2}

Ejemplos: cuadrado mágico

- Función de decodificación, `DECODIFICA(X)`:
  - Un cromosoma representa al cuadrado cuya concatenación de filas es igual al cromosoma
- Nótese que la suma común se puede calcular:
  `SUMA=(n·(n`^{2} `+ 1))/2`
- **`FITNESS(X)`:**
  - Suma de las diferencias (en valor absoluto) entre la suma de los números de cada hilera (filas, columnas y diagonales) y **`SUMA`**
  - Si usamos cruces y mutaciones estándar: introducir una penalización en los individuos con genes repetidos
  - Mejor: cruces y mutaciones que a partir de permutaciones obtengan permutaciones

Ejemplos: problema de la mochila

- Problema:
  - Dados n objetos de pesos p_i y valor v_i ($i = 1, \ldots, n$), seleccionar cuáles se meten en una mochila que soporta un peso P máximo, de manera que se maximice el valor total de los objetos introducidos
- Genes y longitud de los individuos en el problema de la mochila
  - `*GENES* = [0, 1]`
  - `*LONGITUD-INDIVIDUOS* = n`

Ejemplos: problema de la mochila

- Función de decodificación, **`DECODIFICA(X)`**:
  - 1 ó 0 representan, respectivamente, si el objeto se introduce o no en la mochila
  - Tomados de izquierda a derecha, a partir del primero que no cabe, se consideran todos fuera de la mochila, *independientemente del gen en su posición*
  - De esta manera, todos los individuos representan candidatos válidos
- **`FITNESS(X)`**:
  - Suma de valores de los objetos, que según la representación dada por el **`DECODIFICA`** anterior, están dentro de la mochila
  - Nótese que en este caso no es necesaria ninguna penalización

Conclusión

- Algoritmos genéticos como proceso de búsqueda local
  - Mejora iterativa
  - Los cruces, las mutaciones y la diversidad tratan de evitar el problema de los óptimos locales
- Existen otras muchas implementaciones de algoritmos genéticos
  - Pero todas se basan en las mismas ideas de reproducción, mutación, selección de los mejores y mantenimiento de la diversidad
- Fácil de aplicar a muchos tipos de problemas:
  - optimización, aprendizaje automático, planificación,...
- Resultados aceptables en algunos problemas
  - Aunque no son mejores que algoritmos específicos para cada problema

Bibliografía

- Eibe, A.E. y Smith, J.E. *Introduction to Evolutianary Computing* (Springer, 2007).
  - Cap. 3: “Genetic Algorithms”.
- Russell, S. y Norvig, P. *Artificial Intelligence (A Modern Approach) 3rd edition* (Prentice–Hall, 2010).
  - Cap. 4 “Beyond classical search”.
- Mitchell, T.M. *Machine Learning* (McGraw-Hill, 1997)
  - Cap. 9: “Genetic Algorithms”
- Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs* (Springer, 1999).
  - Cap. 2 “GAs: How Do They Work?”.