Computación Evolutiva

Eugenio Scalise
Centro de Ingeniería de Software y Sistemas (ISYS)

Inteligencia Artificial (CC 6021236)
Postgrado en Ciencias de la Computación
Junio 2019

Agenda

- Motivación
- Computación Evolutiva / Algoritmos Evolutivos
- Algoritmos Genéticos
  - Cuándo usarlos
  - Algoritmo Genético Simple
  - Representación, selección, reproducción, otros parámetros
- Permutaciones, Problema TSP
  - Representación directa de rutas
  - Operadores genéticos para permutaciones en representación directa
  - Otras representaciones
  - Ejemplo de problema de permutaciones (no TSP)
- Frameworks y bibliotecas para computación evolutiva
- Referencias

Motivación

- Las técnicas de resolución de problemas mediante búsqueda son insuficientes para problemas complejos (intratables, alta dimensionalidad de estados, imposibilidad de definir heurísticas o podas eficientes).

- Las *estrategias evolutivas* son algoritmos que imitan los principios de la evolución natural para problemas de optimización de parámetros.

- Las estrategias evolutivas o computación evolutiva incluye técnicas que generalmente son clasificadas como *búsquedas aleatorias o estocásticas*.

Computación Evolutiva

- Técnicas que usan como elementos claves conceptos de la teoría de la evolución de Darwin para su diseño e implementación.

- Es un mecanismo de selección de soluciones potenciales y construcción de nuevos candidatos por recombinación de características con base en otras ya existentes, de manera similar como ocurre en la naturaleza.

- La idea es aprovechar los conceptos que están detrás de la evolución de las especies, con el objetivo de abordar problemas de búsqueda y aprendizaje.

Bases biológicas

- **Evolución**
    - Proceso que conduce al mantenimiento o incremento de la habilidad de una población para sobrevivir y reproducirse en su ambiente.
    - Búsqueda entre un conjunto de secuencias genéticas (soluciones), donde son deseables aquellos individuos u organismos capaces de sobrevivir y reproducirse en su ambiente (aptitud/calidad).
- **Adaptación**
    - Medida de la habilidad de un organismo para anticipas cambios en su ambiente y responder adecuadamente.
- **Objetivo perseguido:** Generar una población de individuos con una buenta adaptación.

Bases biológicas

- **Procesos básicos:**

1. **Selección Natural:** determina qué individuos participarán en la reproducción

2. **Reproducción:** asegura la recombinación y mezcla de rasgos de los padres en los hijos.

Hechos básicos de la evolución natural

- La evolución ocurre a nivel de los cromosomas o información genética.
- Los cromosomas codifican la estructura de los individuos.
- La codificación/decodificación del material genético es esencial para la existencia.
- La selección natural relaciona los cromosomas con la calidad del individuo codificado.
- Durante la reproducción ocurre la evolución.
- Los mecanismos básicos de reproducción son: mutación y recombinación.
- Generalmente, el mecanismo de evolución natural no posee memoria. Todo lo que se conoce de la población está en la información genética.

Algoritmos evolutivos

- Conjunto de técnicas de resolución de problemas complejos basadas en la emulación de los procesos naturales de la evolución.

- Enfoque alternativo para abordar problemas de búsqueda y de aprendizaje complejos, a través de modelos computacionales de procesos evolutivos.

- Procedimientos de búsqueda basados en el principio de la evolución que guían la búsqueda haciendo evolucionar un conjunto de estructuras (información genética), seleccionando de manera iterativa a las mejores.

- **Idea general:** combinar características de buenas soluciones para obtener otras mejores.

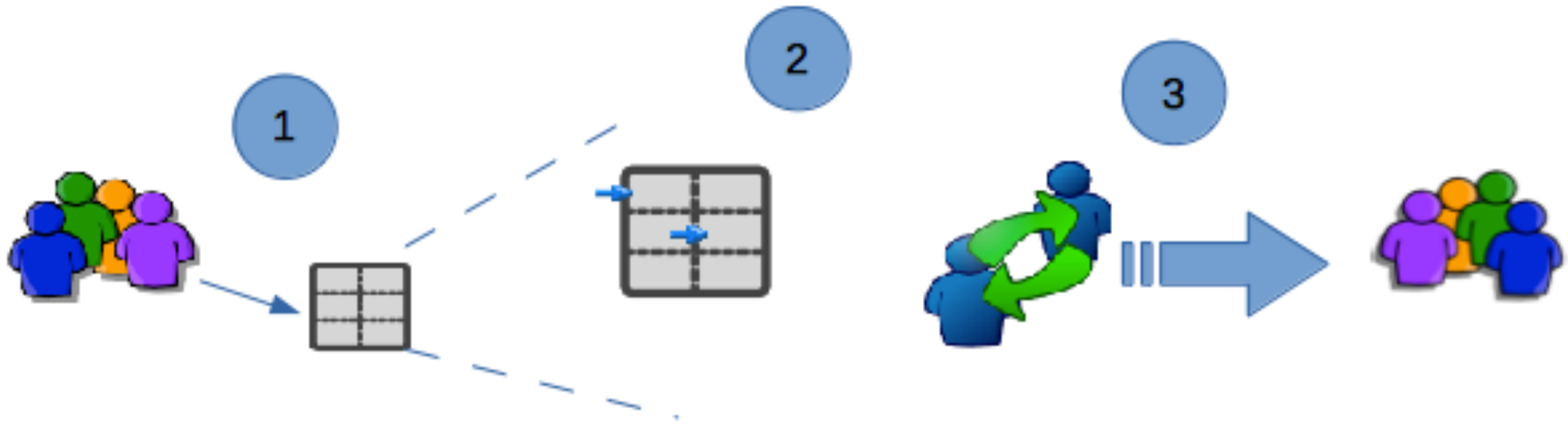
Algoritmos evolutivos

Para emular el proceso de evolución es necesario disponer de:

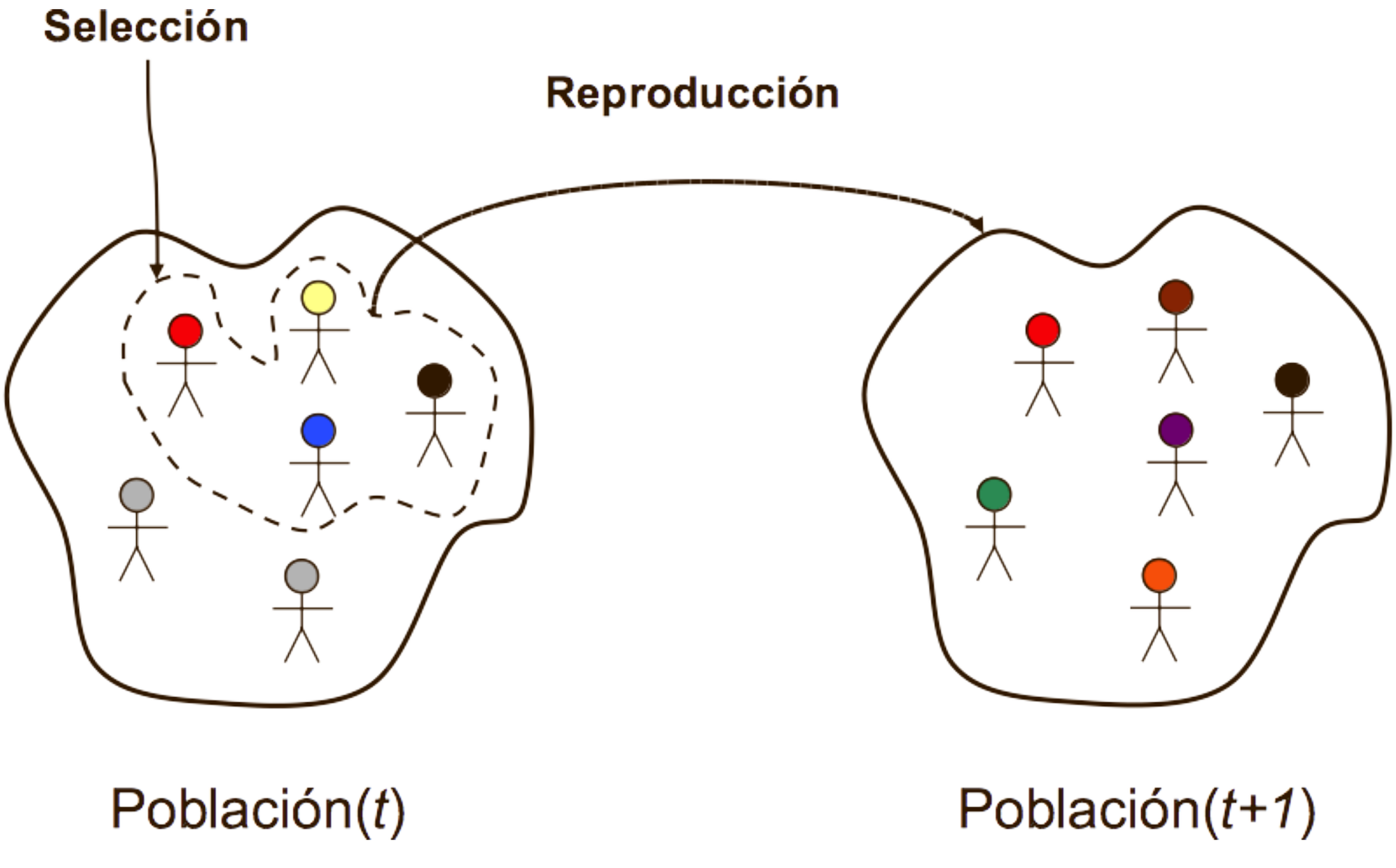
1. Una población de posibles soluciones codificadas como individuos.

2. Un procedimiento de selección basado en la aptitud del individuo.

3. Un procedimiento de transformación para construir nuevas soluciones a partir de las disponibles.



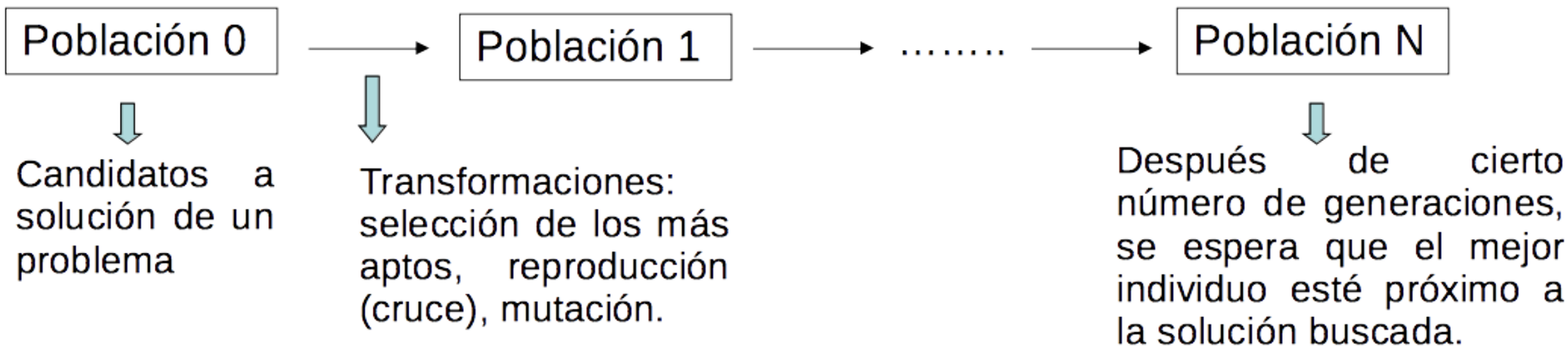
Algoritmos evolutivos



Algoritmos evolutivos

Ambiente → ***Problema***

Población → ***Conjunto de individuos (posibles soluciones al problema)***



Algoritmos evolutivos

Estructura General:

1. Generar una población inicial de manera aleatoria.

2. Evaluar la aptitud de cada individuo en la población.

3. Seleccionar los mejores individuos para la reproducción.

4. Crear nuevos hijos a través del cruce y la mutación.

5. Crear una nueva población formada por los padres y los hijos, eliminando aquellos individuos con la menor aptitud.

6. Volver al paso número 2 hasta que se consiga una buena solución o se haya iterado una cantidad determinada de veces.

Algoritmos evolutivos: componentes básicos

1	Representación genética	Problema
2	Población inicial (*)	
3	Función de aptitud	
4	Mecanismo de selección	Dinámica evolutiva
5	Operadores evolutivos	
6	Parámetros de control	

Algoritmos evolutivos: componentes básicos

(descripción de cada componente básico descrito en clases en pizarra)

Algoritmos Genéticos

- Es el tipo de algoritmo evolutivo más popular.
- Propuesto por John Holland en 1975.
- Son algoritmos de búsqueda basados en los mecanismos de evolución biológica.
- Enfocan el proceso de resolución a través de una dinámica evolutiva, emulando la selección de los más aptos y la reproducción de las especies.

Algoritmos Genéticos: cuándo usarlos

- Cuando sabemos poco sobre el espacio de búsqueda.
- Cuando el espacio de búsqueda es muy grande.
- Cuando es suficiente una buena aproximación a la solución (no necesariamente la óptima) en tiempo razonable para problemas que no pueden ser resueltos fácilmente con técnicas conocidas.
- Cuando el problema a resolver es similar a otros problemas resueltos exitosamente por un AG.
- Cuando se necesita una herramienta exploratoria para examinar nuevos enfoques para la solución de un problema.
- Pre-requisitos:
    - Es necesario “codificar” las soluciones candidatas del problema.
    - Debe existir una manera clara de definir y evaluar la función de ajuse o “fitness”.

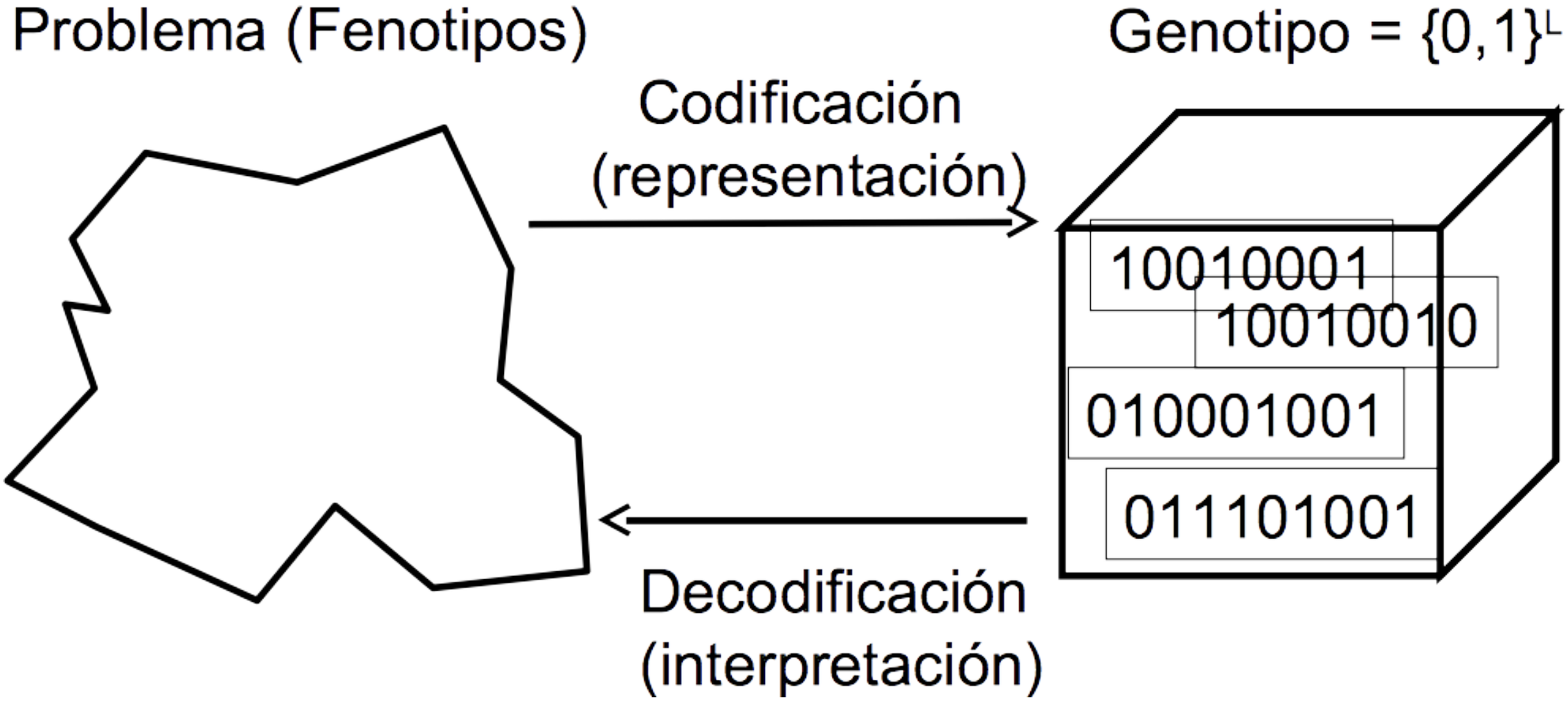
Algoritmo Genético Simple

- Algoritmo evolutivo sugerido por Holland (1975) que considera:
  - Representación binaria
  - Población de tamaño constante
  - Inicialización aleatoria
  - Selección por rueda de la ruleta
  - Operadores genéticos: cruce en un punto y mutación
  - Sustitución total entre generaciones
  - Parámetros usuales (aprox.): n = 100, pc = 0.8, pm = 0.002

Algoritmo Genético Simple

```
t = 0;
initialize(P(t=0));
evaluate(P(t=0));
while isNotTerminated() do
    P_p(t) = P(t).selectParents();
    P_c(t) = reproduction(P_p);
    mutate(P_c(t));
    evaluate(P_c(t));
    P(t+1) = buildNextGenerationFrom(P_c(t), P(t));
    t = t +1;
end
```

Representación



Representación

- Un individuo está caracterizado por un conjunto de parámetros: genes
- Los genes son combinados como una cadena/string: cromosoma
- El cromosoma forma el genotipo
- El genotipo contiene la información para construir un organismo/ individuo: fenotipo
- La reproducción es un proceso simple a nivel de los cromosomas (genotipo)
- La adaptación (fitness) es medida en el espacio del problema (fenotipo)

Representación: Codificación

- En los AE se puede utilizar cualquier alfabeto o representación, sin embargo en un AGS se usa un alfabeto binario

- El orden de los genes en el cromosoma suele ser importante

- En general, pueden haber muchas codificaciones de los parámetros/valores asociados a una solución o individuo

- Una buena codificación es probablemente el factor más importante en el desempeño del AGS

- En algunos casos, existen cromosomas que no representan soluciones válidas

Codificación/decodificación

- Utilizar tipos de datos del lenguaje de programación y usar operaciones a nivel de bit (bitwise operations)

- Determinar el número de bits necesarios por cada parámetro/valor del individuo.

- Por ejemplo, representar valores reales en el rango [*a*,*b*] con *p* dígitos de precisión requiere:

$$nBits(a, b, p) = \left\lceil log_2 \left[(b - a + 1) 10^p \right] \right\rceil$$

- Para decodificar un valor binario *I* (genotipo) en su equivalente real *x* (fenotipo):

$$x = a + bin2dec(I) \frac{b - a}{2^{n+1} - 1}$$

con

$$I = (b_n b_{n-1} \ldots b_2 b_1 b_0)_2$$

Otras codificaciones (no AGS)

- Números reales (43.2 -33.1 ... 0.0 89.2)
- Permutaciones de elementos (E11 E3 E7 ... E1 E15)
- Listas de reglas/acciones (R1 R2 R3 ... R22 R23)
- Instrucciones de programas (genetic programming)
- …
- (cualquier estructura de datos)

Función de aptitud (fitness)

- Propósito:
  - Medir la calidad de las soluciones/individuos
  - Selección de parientes
  - Medida de convergencia
  - Si se alcanza un estado estacionario, permite seleccionar los individuos a eliminar/morir
- Debe reflejar de alguna forma el “valor” del cromosoma
- No es relevante cómo se evalúa (aunque en general debe ser simple) e inclusive puede incorporar las restricciones del problema
- Segundo factor más crítico en un AGS después de la codificación

Selección

- Idea general: los mejores individuos tienen mayor oportunidad de ser seleccionados (y reproducirse)
- Esta oportunidad es proporcional al ajuste/*fitness* de cada individuo
- Alternativas:
  - Selección por rueda de la ruleta
  - Selección por torneo

Selección por rueda de la ruleta

- Obtener T, la suma de las adaptaciones de todos los cromosomas
- Generar un número N aleatorio entre 0 y T
- Retornar el individuo cuya adaptación mas la suma total de sus predecesores es mayor o igual que N
- La probabilidad de ser seleccionado es proporcional a la adaptación

Cromosoma:	1	2	3	4	5	6
Adaptación:	8	2	17	7	4	11
Total:	8	10	27	34	38	49
N (1 ≤ N ≤ 49):			23			
Seleccionado:			3			

Selección por rueda de la ruleta (otra visión)

- Sea p_i la probabilidad de seleccionar al individuo *i*:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_i}$$

donde *n* es el tamaño de la población y f_i es la función de aptitud para el individuo *i*

- Calcular la probabilidad acumulada (q_i)
  $q_0 = 0$
  $q_i = q_{i-1} + p_i$ ($i = 1 \ldots n$)

- Se genera un valor aleatorio *r* en [0,1]

- Seleccionar el individuo *i* que cumpla la condición $q_{i-1} < r < q_i$

- Lo dos pasos anteriores se repiten *n* veces

Selección por torneo

- **Torneo binario:** dos individuos se seleccionan al azar; el que tiene mejor adaptación es seleccionado como padre para la reproducción

- **Torneo binario probabilístico:** dos individuos se seleccionan al azar; el más apto de los dos es seleccionado como padre con probabilidad p ($0.5 < p < 1$)

- **Grandes torneos:** se seleccionan m individuos de manera aleatoria; el mejor es seleccionado como padre; para seleccionar k individuos se repite el proceso k veces

- Se puede ensayar ajustando dinámicamente los valores p y m

Problemas con el rango de la función de adaptación

- **Convergencia prematura**
  - Individuos “superdotados” dominan la población
  - La población puede converger a un máximo local
  - Demasiada explotación, muy poca exploración
- **Finalización lenta**
  - Después de algunas generaciones, la adaptación promedio converge pero no se encuentra ningún máximo global. No hay diferencia significativa entre el mejor individuo y el promedio de la adaptación de la población
  - Poca explotación, mucha exploración

Soluciones a estos problemas

- Usar selección por torneo
- Modificar la función de ajuste para la rueda de la ruleta siguiendo estos esquemas:
  - Fitness scaling
  - Fitness windowing
  - Fitness ranking

Fitness scaling

- Los valores de la función de adaptación son escalados por sustracción y división de manera que el peor valor sea cercano a cero y el mejor valor sea cercano a un cierto valor (usualmente 2)

- Con esto, la probabilidad de seleccionar el mejor individuo es dos veces el promedio del ajuste y la probabilidad de seleccionar el peor individuo es cercana a cero

- Este enfoque tiene problemas cuando el mejor/peor individuo son muy extremos (se puede solventar definiendo un valor máximo/mínimo para la adaptación)

Fitness Scaling (truncamiento)

- Se evalúa mean_f_{adapt} (media de la función de adaptación) y std_f_{adapt} (desviación estándar de la función de adaptación)

- Se reajusta la adaptación de cada individuo mediante:

  $$f'(i) = f(i) - (mean\_f_{adapt} - c*std\_f_{adapt})$$

  donde c es un valor en [0,3]

- Si $f'(i) < 0$ entonces $f'(i) = 0$

- Aplicable sólo para adaptaciones f(i) positivas

Fitness windowing

- Enfoque que pretende incrementar la competición entre individuos similares
- La adaptación de cada individuo se calcula con la función de ajuste original
- El ajuste del peor individuo es sustraído de la adaptación de cada individuo

Fitness ranking

- Los individuos son numerados en orden creciente de adaptación

- El ranking en ese orden constituye la adaptación ajustada para cada individuo

- El número inicial del ranking y los incrementos pueden ser escogidos de distintas formas e influenciar los resultados (por defecto, ambos valores son 1)

- Este enfoque elimina los problemas de super individuos

- Suele ser mejor que fitness scaling y windowing scaling

Reproducción

- **Cruce**
  - Dos padres producen dos descendientes
  - Es posible que los cromosomas de los padres sean copiados sin modificaciones en los descendientes
  - Es posible que los cromosomas de los padres sean combinados aleatoriamente en los descendientes
  - Usualmente, la probabilidad de cruce se encuentra entre 0.6 y 0.9
- **Mutación**
  - Es posible que los genes de un descendiente sean modificados aleatoriamente
  - Usualmente, la probabilidad de mutación es baja (por ejemplo, 0.001)

Ciclo reproductivo (AGS)

1. Seleccionar los *n* candidatos a reproducirse (*mating pool*)

2. Barajar los candidatos (*shuffle*)

3. Para cada par consecutivo aplicar el operador de cruce con probabilidad *pc*; en caso contrario, copiar los padres.

4. Para cada descendiente, aplicar el operador de mutación (*bit-flip* con probablidad *pm* para cada bit/gen)

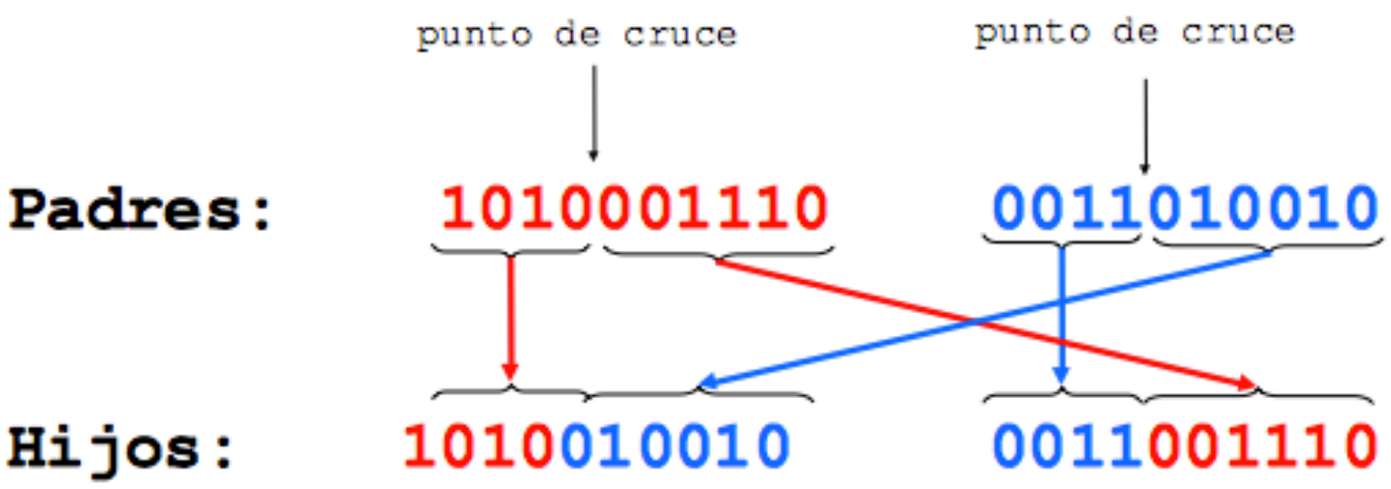
5. Reemplazar la población con los descendientes obtenidos

Reproducción: Cruce

- Cruce de un punto (AGS)
- Cruce de dos puntos
- Cruce uniforme

Cruce de un punto

- Seleccionar una posición aleatoria en los cromosomas de los padres
- Dividir los padres en el punto de cruce
- Crear los descendientes intercambiando las partes
- La probabilidad de cruce (pc) generalmente está en el rango [0.6, 0.9]

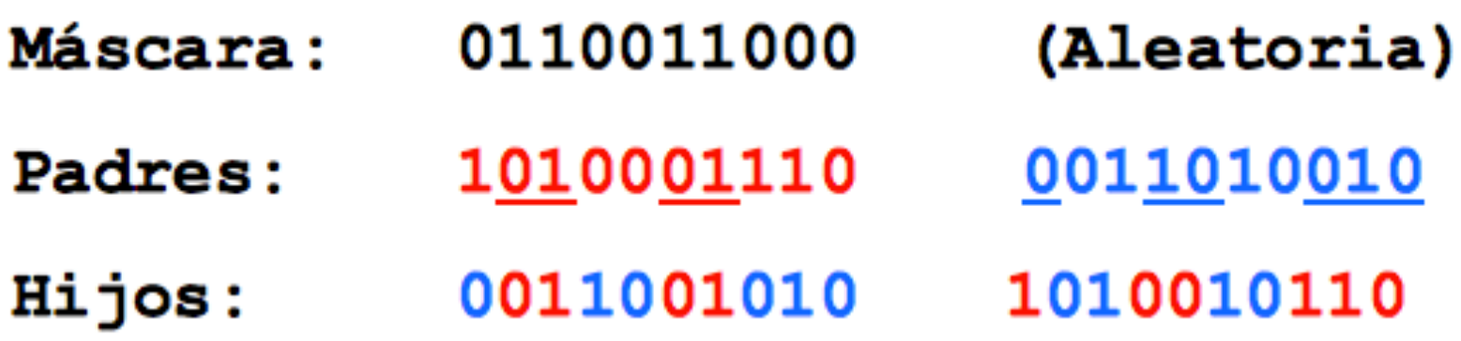


Cruce de dos puntos

- Se seleccionan dos posiciones al azar en los cromosomas de los padres
- Intercambiar los genes entre los dos puntos de cruce
- Evita que los genes al principio y al final del cromosoma sean intercambiados siempre cuando se recombinan los padres
- Hay una generalización para *n* puntos

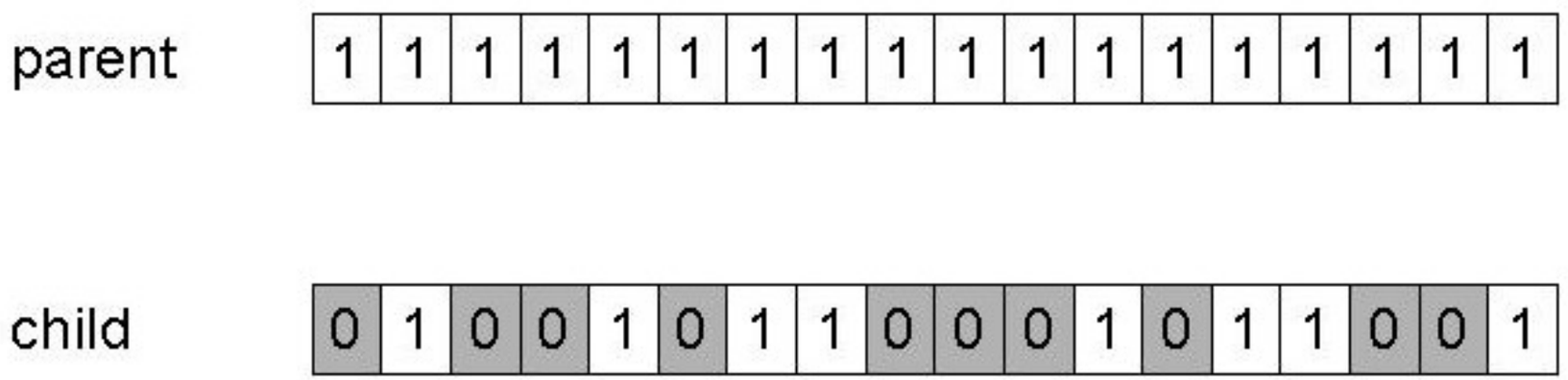
Cruce uniforme

- Se genera una máscara aleatoria
- La máscara determina qué genes/bits se copian de cada padre
- La densidad de la máscara determina cuánto material genético es tomado del otro padre



Reproducción: mutación

- Alterar cada gen de manera independiente con una probabilidad pm (tasa de mutación)
- Algunos valores usuales para pm pueden ser:
  - 1 / n (n = tamaño de la población)
  - 1 / k (k = tamaño del cromosoma)



Consideraciones

- La representación y codificación debe:
  - Representar todos los objetos del dominio (completitud)
  - Fácil de aplicar (sencillez)
  - Representar únicamente objetos del dominio (coherencia)
- Un AGS sólo puede operar en dominios discretos

Consideraciones

- La representación puede generar individuos que no pertenecen al dominio del problema. Esto se puede solventar mediante:
  - Penalización: reducir la aptitud de individuos no factibles
  - Reparación: corregir cualquier individuo no factible
  - Utilizar una codificación donde resulte improbable generar individuos no factibles (esto incluye modificar los operadores genéticos)

Otras variantes

- **Selección**
  - **Generacional con elitismo:** un número fijo de los mejores individuos son copiados sin modificar en la nueva generación
- **Criterios de parada**
  - Número de generaciones/iteraciones (orientado a costo)
  - Establecer un criterio de calidad en la función de aptitud del más apto o de la población (orientado a calidad)
  - Número de nuevos cromosomas

Otras variantes

- **Otros operadores**
  - Inversión
  - Mutación por barajeo (shuffle)
  - Mutación por intercambio
- **Medidas a calcular**
  - Mejor individuo de la población
  - Promedio del ajuste de la población

AGS: virtudes y defectos

- Permite una amplia aplicación en diversos dominios de problemas
- Es de fácil implementación
- Es el programa evolutivo estudiado con mayor profundidad
- Es relativamente fácil de entonar a un problema particular
- Los operadores genéticos son muy simples y de fácil comprensión
- Posee pocos parámetros
- Con ligeras modificaciones se puede aumentar su potencial significactivamente

AGS: virtudes y defectos

- Su eficiencia es muy dependiente del diseño de la función de adaptación
- El esquema de reproducción puede ocasionar la pérdida de la mejor solución alcanzada
- Convergencia prematura ante la presencia de “super individuos”
- La acción conjunta de los operadores evolutivos no es natural y puede no ser lo más conveniente

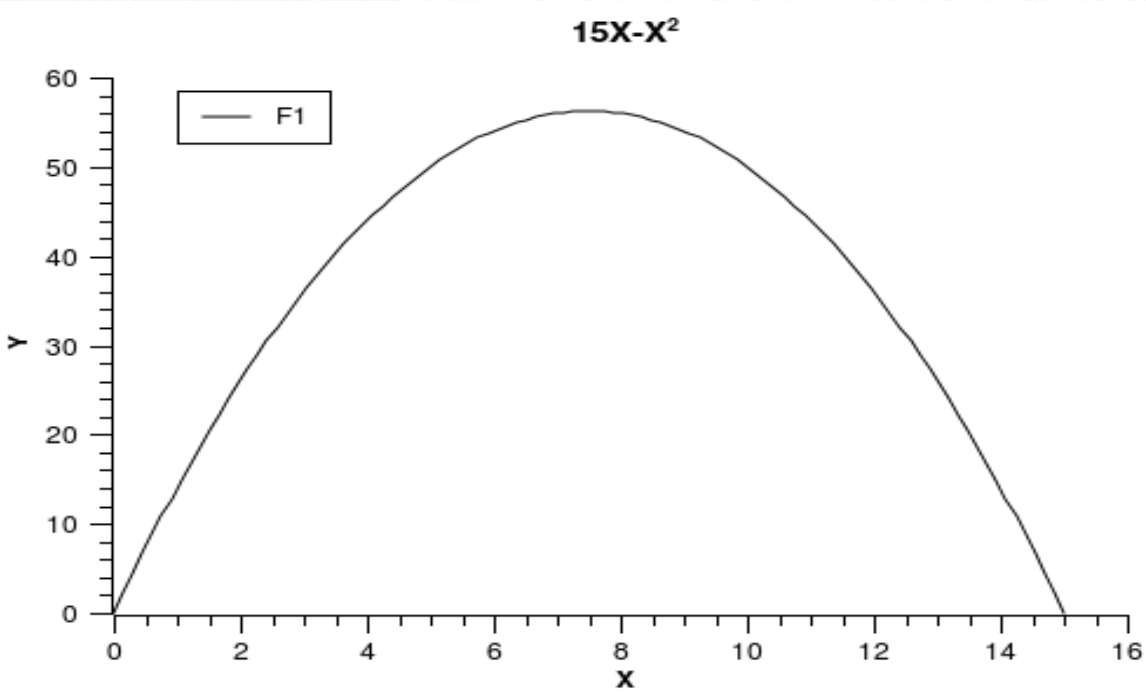
Ejemplo

(ver ejemplo en láminas aparte…)

Ejemplo de AG

Ejemplo 1

Maximizar la función $F(X) = 15x - x^2$ en el intervalo $[0\text{-}15] \in N$



Ejemplo de AG

1.Representar el dominio del problema como un cromosoma de longitud fija.

Como los números son enteros se determina el No. de bits del cromosoma

Primero determinar el número de enteros a representar

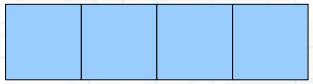
→ $x_{max} - x_{min} + 1 = M$

15-0 +1 = **16**

Luego, el número de bits necesario será:

→ Número de bits = $\log_2 M$

Log_2 16 = **4**

Cromosoma = 

4 Bits

Gen = 1 bit del cromosoma

Alelos (Alfabeto) = 0,1

Ejemplo de AG

2. Definir la función de adaptación f(x).

Como se desea maximizar una función, la función de adaptación es la misma función, evaluando cada individuo de la población en la función.

$F(X) = 15x - x^2$

3. Generar, de manera aleatoria, la población inicial de cromosomas

X1	1100	12
x2	0100	4
x3	0001	1
x4	1110	14
x5	0111	7
x6	1001	9

Se considera que el tamaño de la población será 6.

Ejemplo de AG

4. Calcular la adaptación de cada cromosoma: $f(x_1)$, $f(x_2)$,…., $f(x_N)$.

			$f(x_i)$
X1	1100	12	36
x2	0100	4	44
x3	0001	1	14
x4	1110	14	14
x5	0111	7	56
x6	1001	9	54

$F(X) = 15x - x^2$

5. Seleccionar los cromosomas de la población actual. Se utilizara muestreo por ruleta.

$$p_i = \frac{f_i}{\sum_{j=0}^{n} f_i}$$

n = tamaño de la población
f = función de aptitud

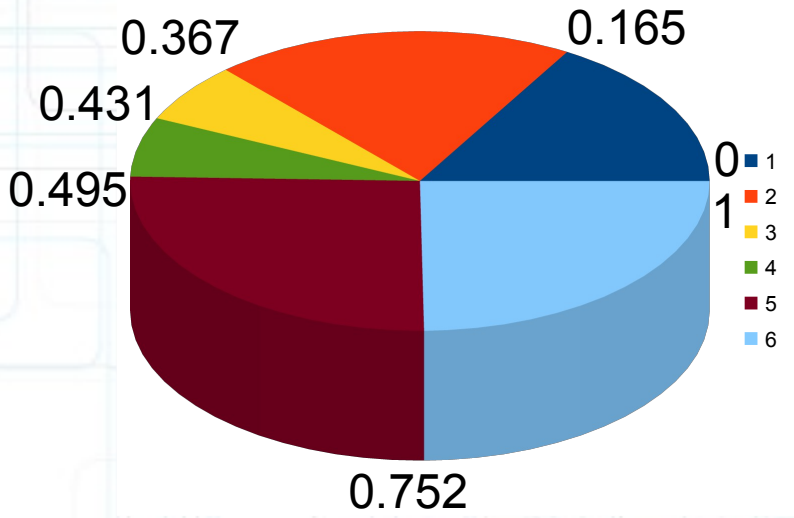
			$f(x_i)$	p_i	p_i
X1	1100	12	36	36/218	0,165
x2	0100	4	44	44/218	0,202
x3	0001	1	14	14/218	0,064
x4	1110	14	14	14/218	0,064
x5	0111	7	56	56/218	0,257
x6	1001	9	54	54/'218	0,248

Ejemplo de AG

			$f(x_i)$	p_i	p_i	q_i
X1	1100	12	36	36/218	0,165	0,165
x2	0100	4	44	44/218	0,202	0,367
x3	0001	1	14	14/218	0,064	0,431
x4	1110	14	14	14/218	0,064	0,495
x5	0111	7	56	56/218	0,257	0,752
x6	1001	9	54	54/'218	0,248	1,000

qo = 0
qi = qi+1 + pi (i=1...n)

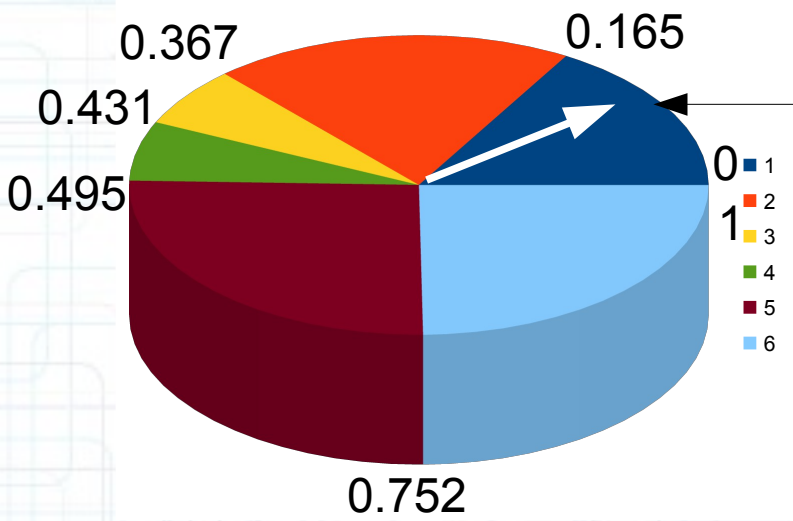
Representación de la Probabilidad Acumulada en un Gráfico de círculo para aplicar el método de la ruleta.

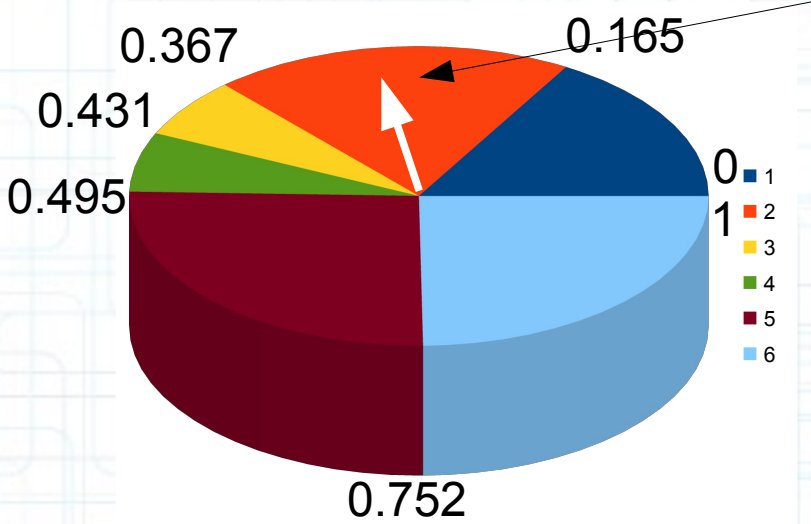


Ejemplo de AG

X1	1100	12
x2	0100	4
x3	0001	1
x4	1110	14
x5	0111	7
x6	1001	9

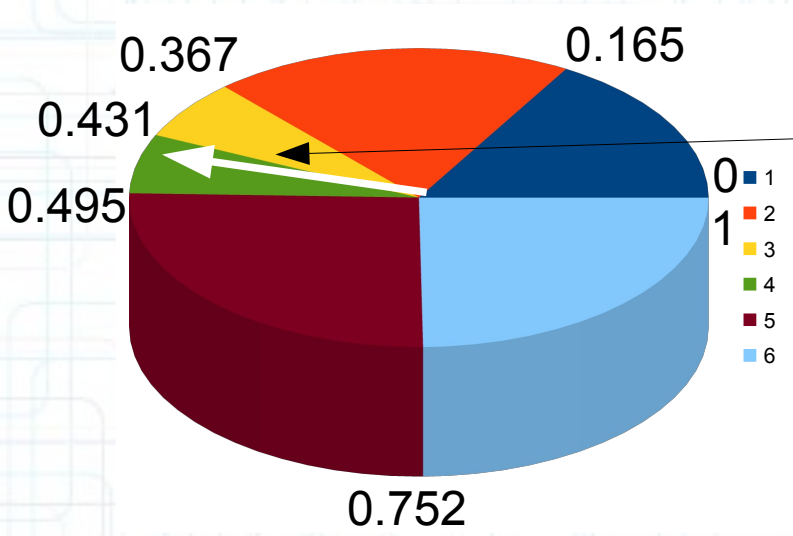
$f(x_i)$	p_i	p_i	q_i	a
36	36/218	0,165	0,165	0,100
44	44/218	0,202	0,367	0,267
14	14/218	0,064	0,431	0,433
14	14/218	0,064	0,495	0,600
56	56/218	0,257	0,752	0,767
54	54/'218	0,248	1,000	0,933

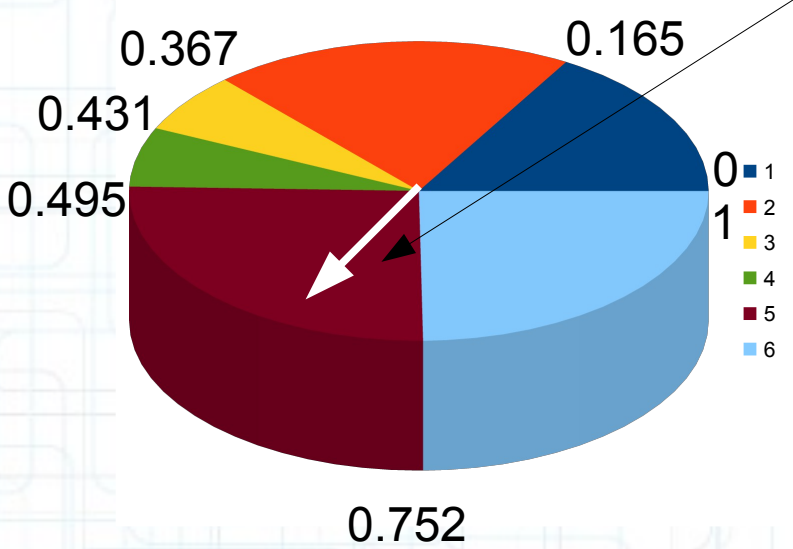




Ejemplo de AG

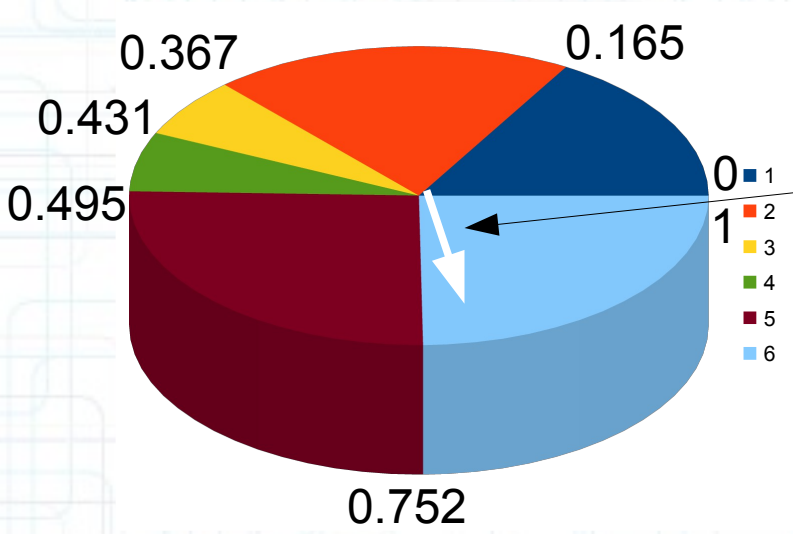
			$f(x_i)$	p_i	p_i	q_i	a
X1	1100	12	36	36/218	0,165	0,165	0,100
x2	0100	4	44	44/218	0,202	0,367	0,267
x3	0001	1	14	14/218	0,064	0,431	0,433
x4	1110	14	14	14/218	0,064	0,495	0,600
x5	0111	7	56	56/218	0,257	0,752	0,767
x6	1001	9	54	54/'218	0,248	1,000	0,933

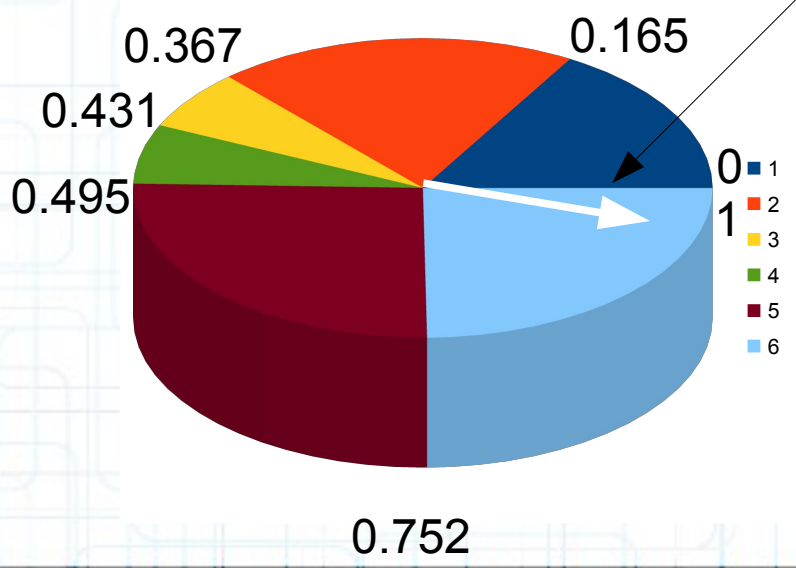




Ejemplo de AG

			$f(x_i)$	p_i	p_i	q_i	a
X1	1100	12	36	36/218	0,165	0,165	0,100
x2	0100	4	44	44/218	0,202	0,367	0,267
x3	0001	1	14	14/218	0,064	0,431	0,433
x4	1110	14	14	14/218	0,064	0,495	0,600
x5	0111	7	56	56/218	0,257	0,752	0,767
x6	1001	9	54	54/'218	0,248	1,000	0,933





Ejemplo de AG

Individuos seleccionados
x1, x2, x4,x5,x6, x6

X1	1100	12
x2	0100	4
x4	1110	14
x5	0111	7
x6	1001	9
x6	1001	9

6. Con probabilidad de cruce P_c, intercambiar partes de los cromosomas seleccionados.

			c
X1	1100	12	0.4
x2	0100	4	0.6
x4	1110	14	0.8
x5	0111	7	0.3
x6	1001	9	0.9
x6	1001	9	0.1

Si $c \leq P_c$ **entonces** Cruzar, **donde** $P_c = 0.7$

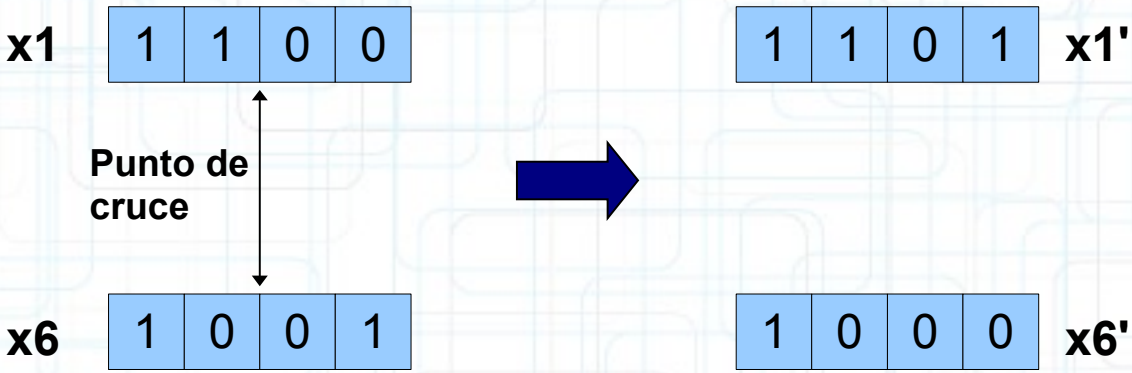
Ejemplo de AG

Cromosomas a cruzar

			c
X1	1100	12	0.4
x2	0100	4	0.6
x5	0111	7	0.3
x6	1001	9	0.1

Se selecciona aleatoriamente que cromosomas se cruzan:

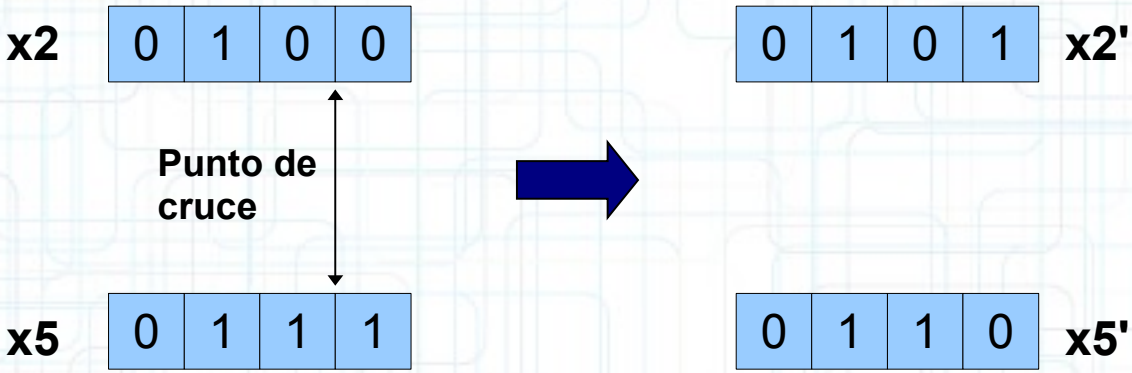
Cruce 1, cruzamos **x1** y **x6**, generando **x1'** y **x6'**.



Punto de cruce= 2

Ejemplo de AG

Cruce 2, cruzamos **x2** y **x5**, generando **x2'** y **x5'**.



Punto de cruce= 3

Población Temporal

X1'	1101	13
x2'	1010	5
x4	1110	14
x5'	0110	6
x6	1001	9
x6'	1000	8

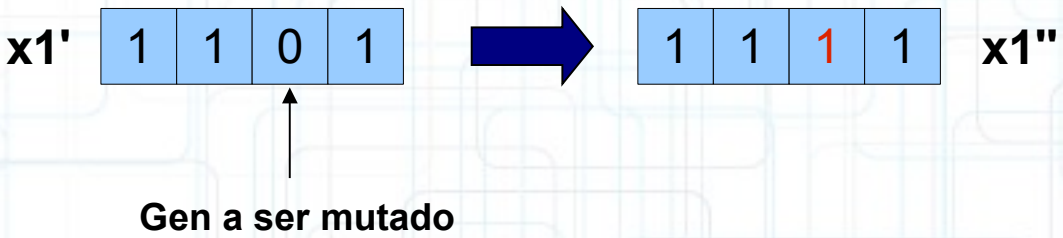
Ejemplo de AG

7. Con probabilidad de mutación P_M, aleatoriamente cambiar un gen de los cromosomas generados.

			m_1	m_2	m_3	m_4
X1'	1101	13	0.32	0.54	**0.005**	0.12
x2'	1010	10	0.91	0.2	0.69	0.56
x4	1110	14	0.02	0.3	0.47	0.8
x5'	0110	6	0.81	0.1	0.06	0.9
x6	1001	9	0.35	0.42	0.38	0.54
x6'	1000	8	0.7	0.23	0.03	0.2

Si $m_i \leq P_m$ **entonces** Mutar Gen i, **donde** $P_m = 0.01$

Mutación 1, mutamos **x1'**, generando **x1"**.



Ejemplo de AG

Nueva población a evaluar desde el paso 4

X1"	1111	15
x2'	1010	10
x4	1110	14
x5'	0110	6
x6	1001	9
x6'	1000	8

Adaptación de la nueva población

			$f(x_i)$
X1"	1111	15	0
x2'	1010	10	50
x4	1110	14	14
x5'	0110	6	54
x6	1001	9	54
x6'	1000	8	56

Mejor individuo:
x6' : 1000 : 8 : 56

Promedio de la Población:
38

Adaptación de la vieja población

			$f(x_i)$
X1	1100	12	36
x2	0100	4	44
x3	0001	1	14
x4	1110	14	14
x5	0111	7	56
x6	1001	9	54

Mejor individuo:
x5 : 0111 : 7 : 56

Promedio de la Población:
36.33

Otras representaciones: Permutaciones

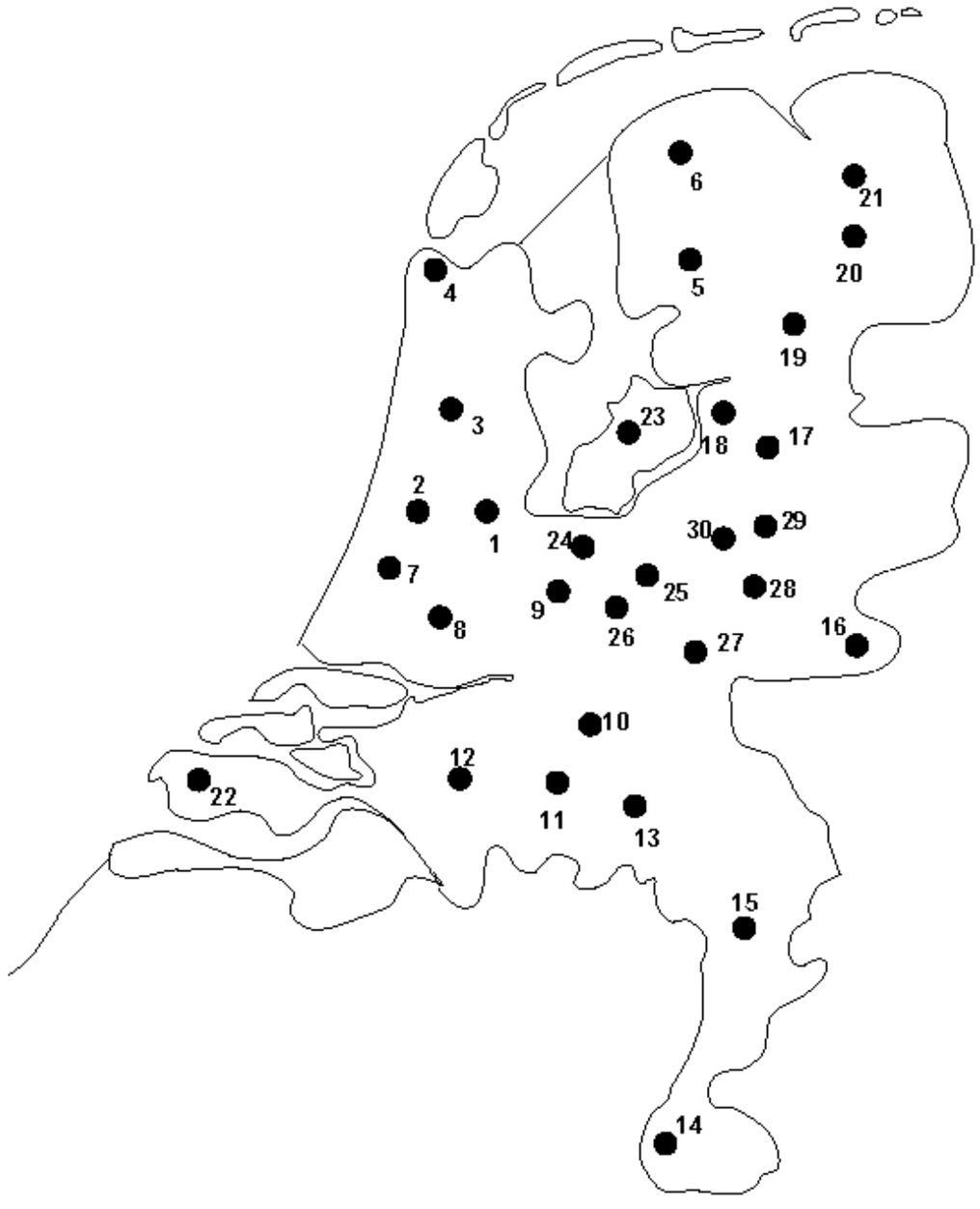
- Las permutaciones son útiles en problemas donde se requiere ordenar/secuenciar elementos.

- TSP (Traveling Salesman Problem, Problema del Agente Viajero): determinar un circuito/camino que partiendo de una ciudad arbitraria, visite todas las ciudades -una sola vez- siguiendo la ruta de menor longitud/costo.

- Problema de optimización combinatoria (NP-Completo): optimizar una función de costo que depende del orden de un conjunto finito de elementos.

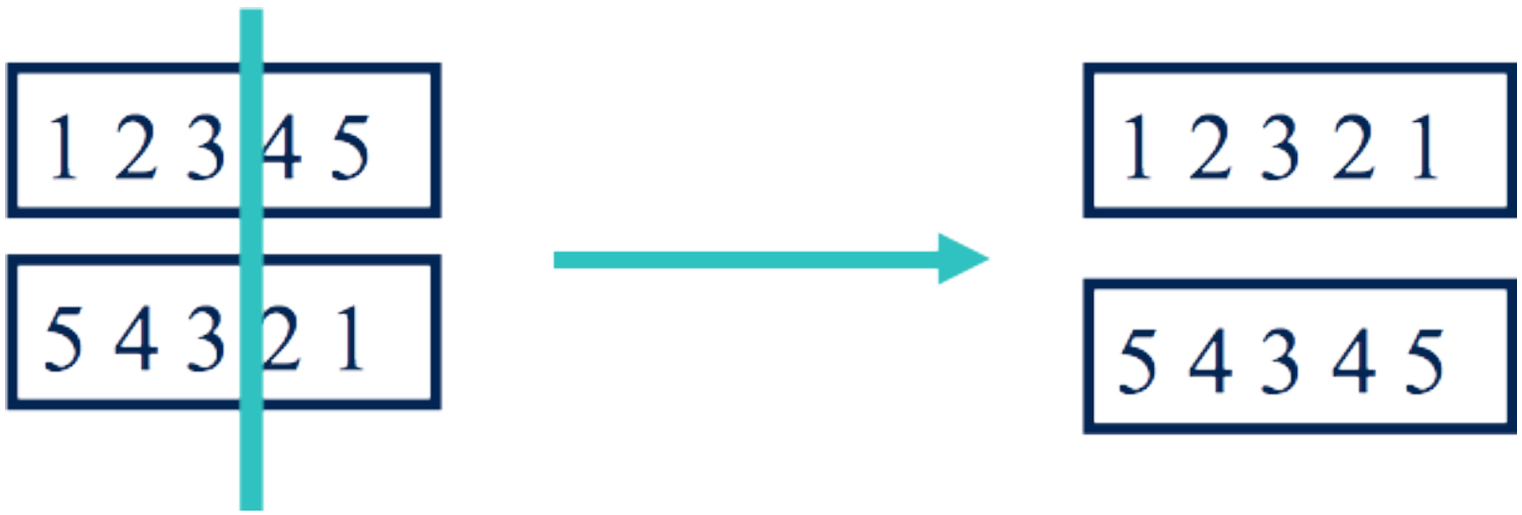
Permutaciones: TSP

- Problema: Dadas n ciudades, encontrar un circuito/tour con costo mínimo
- Codificación:
  - Etiquetar las ciudades: 1, 2, …, n
  - Un tour es una permutación. Por ejemplo, para n = 4, [1,2,3,4], [3,4,2,1],[1,3,4,2] son posibles soluciones
  - La representación binaria **no** es apropiada para este problema
- El espacio de búsqueda es grande (n!). Por ejemplo, para 30 ciudades es 30! ≈ 10^{32}



Operadores de cruce para permutaciones

- Los operadores de cruce tradicionales generan soluciones inválidas



- Se han propuesto muchos operadores especializados, que se enfocan en combinar el orden o información de adyacencia de los dos padres

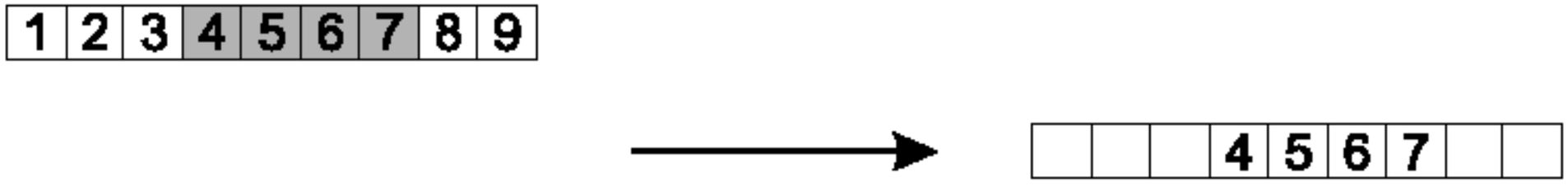
Cruce de primer orden

- La idea es preservar el orden relativo en el que ocurren los elementos.
- Procedimiento informal:

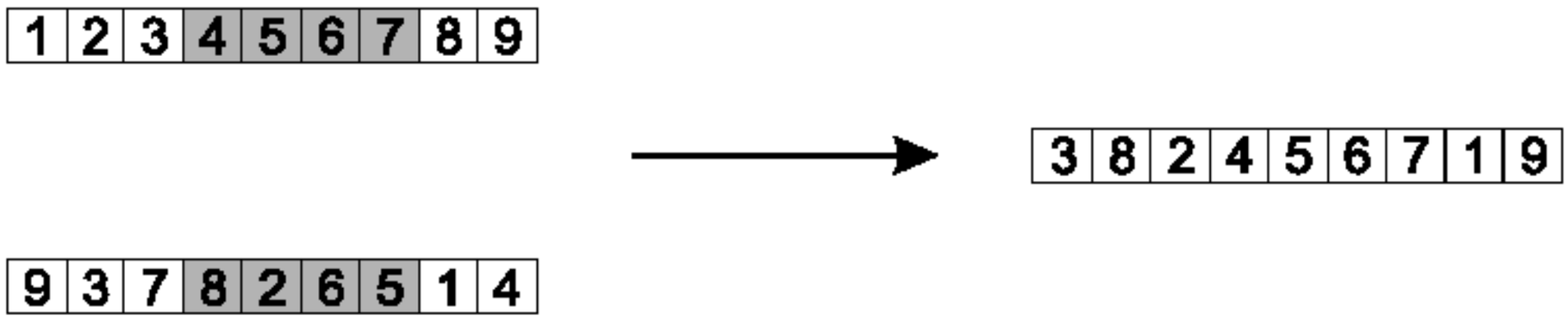
1. Escoger una parte arbitraria del primer padre
2. Copiar esta parte en el primer hijo
3. Copiar los números que no están en la primera parte al primer hijo comenzando desde el punto de corte derecho de la parte copiada, utilizando el orden del segundo padre y rellenando circularmente
4. Realizar un proceso análogo para el segundo hijo, con los roles de padre invertidos

Cruce de primer orden

- Copiar el bloque aleatorio del primer padre



- Copiar el resto del segundo padre en el orden 1, 9, 3, 8, 2



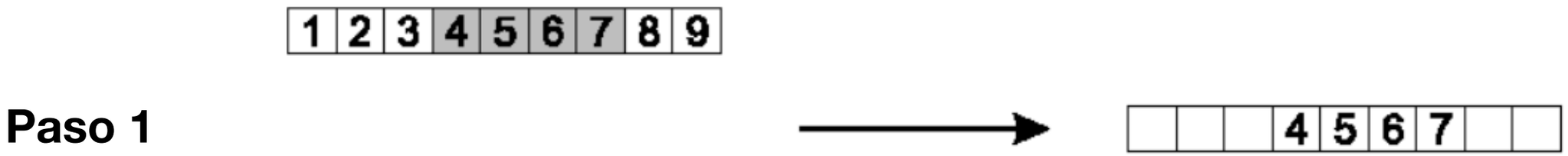
Partially Mapped Crossover (PMX)

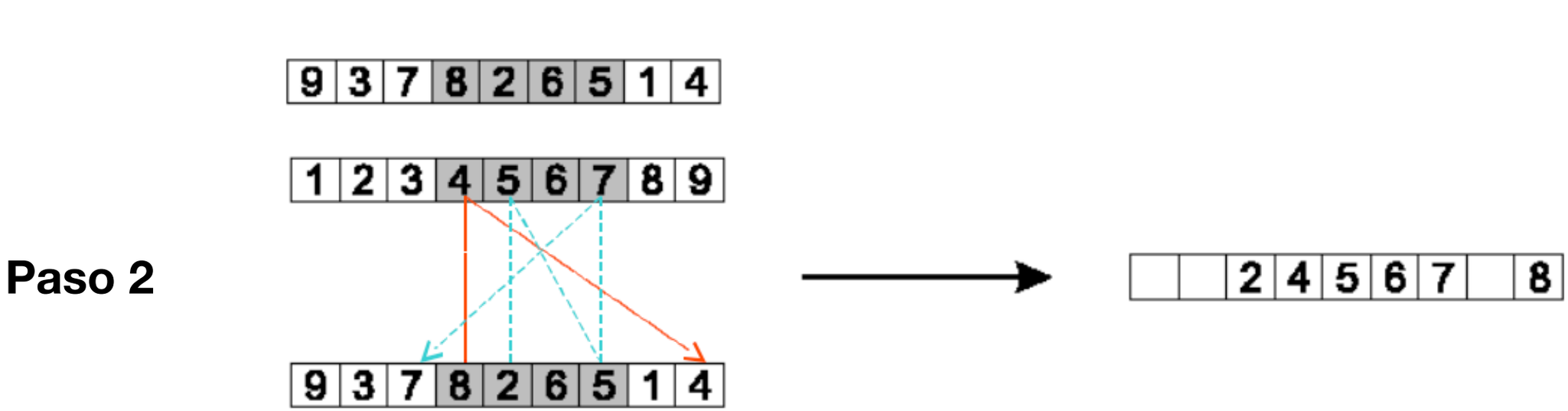
Procedimiento informal para padres P1 y P2:

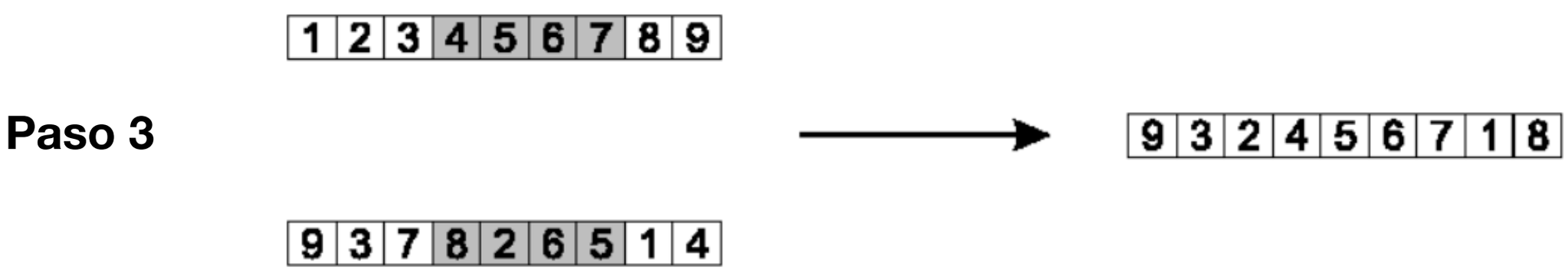
- Escoger un segmento aleatorio y copiarlo de P1
- Comenzando en el primer punto de cruce, buscar elementos en el segmento de P2 que no han sido copiados
- Para cada uno de esos i buscar en la descendencia para ver qué elemento j ha sido copiado en su lugar desde P1
- Colocar i en la posición que ocupa j en P2, sabiendo que no colocaremos a j en esa posición (porque ya está en la descendencia)
- Si el lugar ocupado por j en P2 ya ha sido ocupado en la descendencia k, colocar i en la posición ocupada por k en P2
- Luego de tratar los elementos del segmento aleatorio, el resto del descendiente se llena desde P2

El segundo hijo se crea de manera análoga.

Partially Mapped Crossover (PMX)







Cruce circular

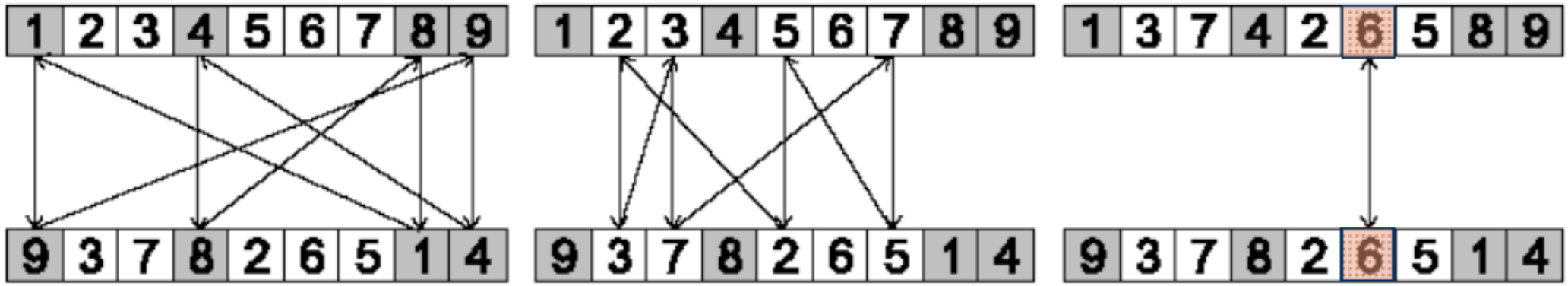
Idea general: cada alelo proviene de un padre junto con su posición.

Procedimiento informal:

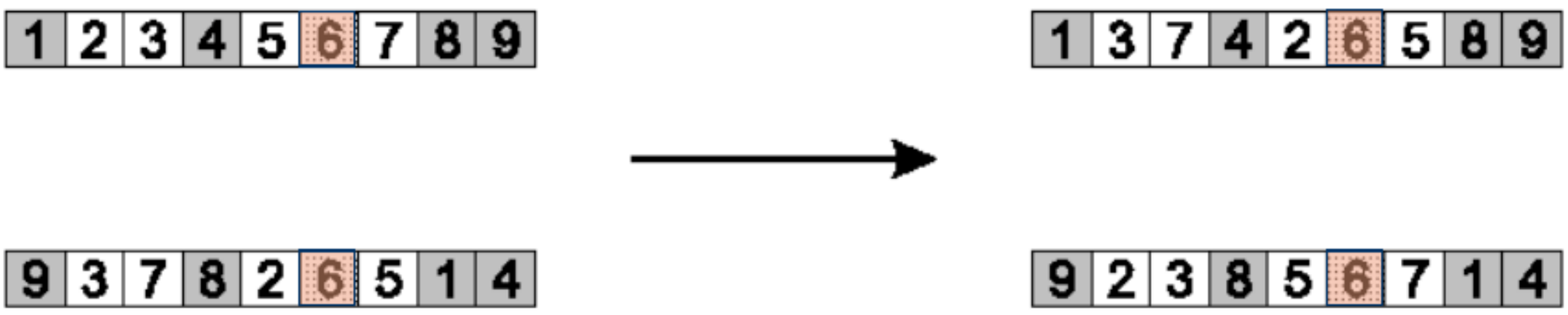
1. Construir un ciclo de alelos de P1 de la siguiente forma:
    A. Comenzar con el primer alelo de P1
    B. Buscar el alelo en la misma posición en P2
    C. Ir a la posición con el mismo alelo en P1
    D. Agregar este alelo al ciclo
    E. Repetir los paso B hasta D hasta que se llegue al primer alelo en P1
2. Poner los alelos del ciclo en el primer hijo en las posiciones que tienen en el primer padre (P1)
3. Tomar el siguiente ciclo del segundo padre

Cruce circular

- Paso 1: Identificar ciclos



- Paso 2: copiar los ciclos alternados en la descendencia



Otros operadores de cruce

- Recombinación de arcos (edge recombination)

- Cruce multipariente

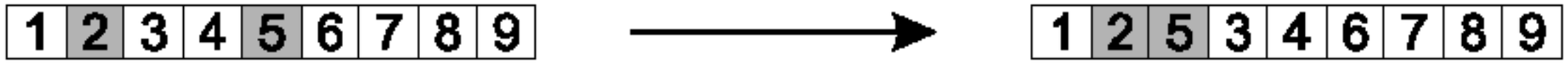
(revisar Michalewicz)

Operadores de mutación para permutaciones

- Los operadores tradicionales de mutación también generan soluciones inválidas
  - Supongamos que el gen de la posición i tiene valor j
    - Cambiar a cualquier valor k significaría que k aparezca dos veces y j no aparezca
- Por lo tanto, hay que cambiar al menos dos valores
- La probabilidad de mutación es aplicada una vez a toda la permutación, en lugar de hacerlo para cada posición

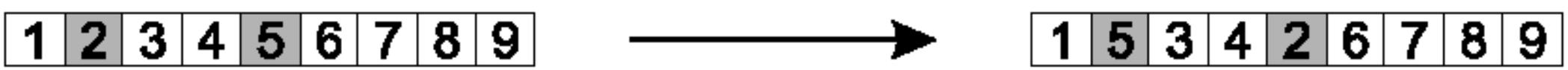
Mutación por inserción

- Seleccionar dos alelos aleatoriamente
- Hacer que el segundo valor seleccionado siga al primero en la secuencia, reacomodando el resto
- Esta operación preserva mucha de la información del orden y adyacencia



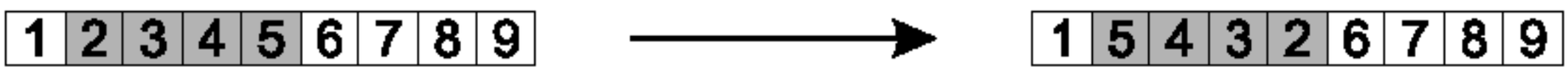
Mutación por intercambio

- Seleccionar dos alelos aleatoriamente e intercambiar sus posiciones
- Preserva mucha de la información de la adyacencia (sólo rompe 4 arcos/enlaces) e interrumpe más el orden



Mutación por inversión

- Seleccionar dos alelos aleatoriamente e invertir la subsecuencia entre ellos
- Preserva mucha información de la adyacencia (solo rompe dos enlaces) pero afecta significativamente el orden



Mutación por revuelta (*scramble*)

- Seleccionar un subconjunto de genes de manera aleatoria
- Reordenar aleatoriamente los alelos en esas posiciones
- Los subconjuntos no necesariamente son contiguos

Otras representaciones para el TSP

Representación de adyacencias

- La ciudad j es listada en la posición i sí y solo sí el camino incluye un arco de i a j

- Por ejemplo, el vector (2, 4 , 8, 3, 9, 7, 1, 5, 6) representa el camino: 1-2-4-3-8-5-9-6-7

- Cada camino tiene sólo una representación basada en adyacencias, aunque algunas listas de adyacencias pueden denotar tours inválidos. Por ejemplo, el vector (2, 4, 8, 1, 9, 3, 5, 7, 6) denota el camino 1-2-4-1 que contiene un ciclo prematuro

- Los operadores genéticos para este tipo de reparación requieren “reparación”

Otras representaciones para el TSP

Representación ordinal

- Un camino es una lista donde el i-ésimo elemento es un número en el rango 1 a n-i+1

- Se tiene una lista ordenada de ciudades C = (1, 2, 3, 4, 5, 6, 7, 8, 9), que sirve como punto de referencia para los caminos en representación ordinal

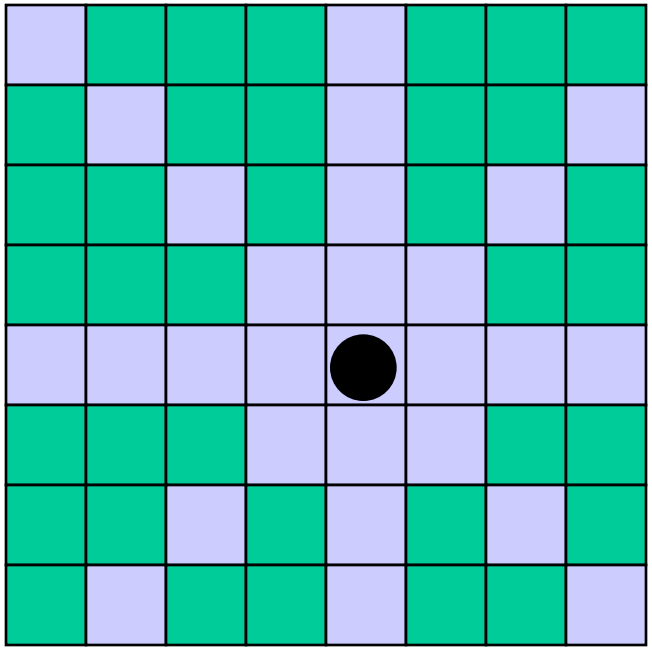
- Por ejemplo, el camino 1-2-4-3-8-5-9-6-7 es representado por la lista de referencias I = (1, 1, 2, 1, 4, 1, 3, 1, 1)

- Esta representación soporta el operador de cruce de un punto clásico

Ejemplo permutaciones (no TSP)

- Problema de las 8 reinas (ver láminas aparte)

Example: the 8 queens problem



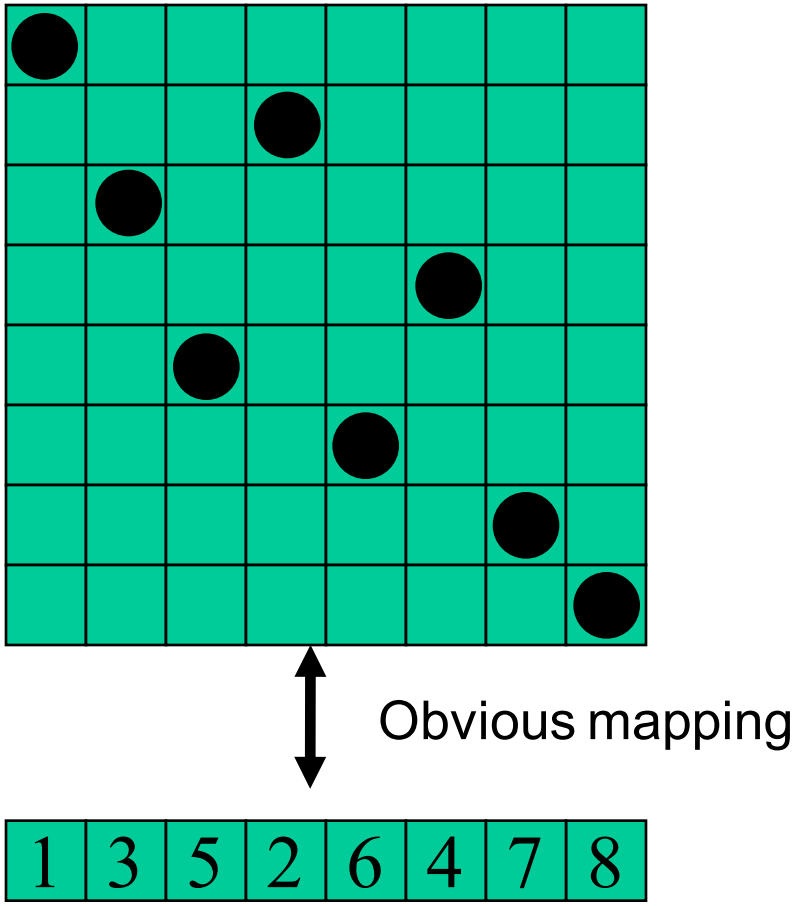
Place 8 queens on an 8x8 chessboard in such a way that they cannot check each other

The 8 queens problem

Representation

Phenotype:
a board configuration

Genotype:
a permutation of
the numbers 1 - 8



The 8 queens problem

Fitness evaluation

Penalty of one queen:
the number of queens she can check.

Penalty of a configuration:
the sum of the penalties of all queens.

Note: penalty is to be minimized

Fitness of a configuration:
inverse penalty to be maximized

The 8 queens problem

Mutation

Small variation in one permutation, e.g.:

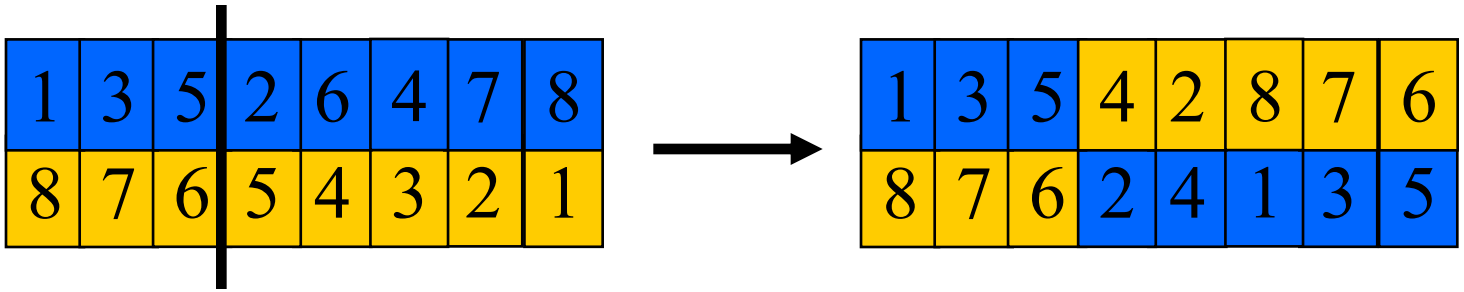
- swapping values of two randomly chosen positions, or
- inverting a randomly chosen segment

The 8 queens problem

Recombination

Combining two permutations into two new permutations:

- choose random crossover point
- copy first parts into children
- create second part by inserting values from other parent:
  - in the order they appear there
  - beginning after crossover point
  - skipping values already in child



The 8 queens problem

Selection

Parent selection:

Roulette wheel selection, for instance

Survivor selection (replacement)

When inserting a new child into the population, choose an existing member to replace by:

- sorting the whole population by decreasing fitness
- enumerating this list from high to low
- replacing the first with a fitness lower than the given child

Note: selection works on fitness values, no need to adjust it to representation

Frameworks y bibliotecas (Python)

- **DEAP (Distributed Evolutionary Algorithms in Python)**
  https://github.com/DEAP/deap
  https://deap.readthedocs.io/

- **Pyvolution**
  https://pypi.org/project/Pyvolution/
  https://pyvolution.readthedocs.io/

- **Pyevolve**
  https://pypi.org/project/Pyevolve/
  http://pyevolve.sourceforge.net/

Frameworks y bibliotecas (otros lenguajes)

- Watchmaker Framework for Evolutionary Computation (Java)
  https://watchmaker.uncommons.org/

- Jenetics.io (modern Java)
  http://jenetics.io/

- openGA (C++)
  https://github.com/Arash-codedev/openGA

- GALib: A C++ Library for Genetic Algorithm Component
  http://lancet.mit.edu/ga/

- GA (R)
  https://cran.r-project.org/web/packages/GA/

- JGAL (Java)
  https://github.com/chango1611/JGAL
  https://github.com/chango1611/JGAL_GUI

Referencias

- Genetic Algorithms + Data Structures = Evolution Programs, Third Edition. Michalewicz, Zbigniew. Springer, 1996. **(Capítulos 1, 2, 3, 10)**

- Introduction to Evolutionary Computing, Second Edition. A.E. Eiben, J.E. Smith. Springer, 2015.

- Adaptation in Natural and Artificial Systems. Holland, J. H. University of Michigan Press. 1975.

- Inteligencia Artificial (6325). Material de docencia, Licenciatura en Computación, Facultad de Ciencias, Escuela de Computación, Universidad Central de Venezuela.