

Diplomado de especialización de desarrollo de aplicaciones con Inteligencia Artificial

Optimización industrial con Computación Evolutiva

Sesión 3 Algoritmos Genéticos

Dr. Soledad Espezua. LI.
sespezua@pucp.edu.pe

Dr. Edwin Villanueva T.
evillatal@pucp.edu.pe



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ



ia.inf.pucp.edu.pe

Agenda

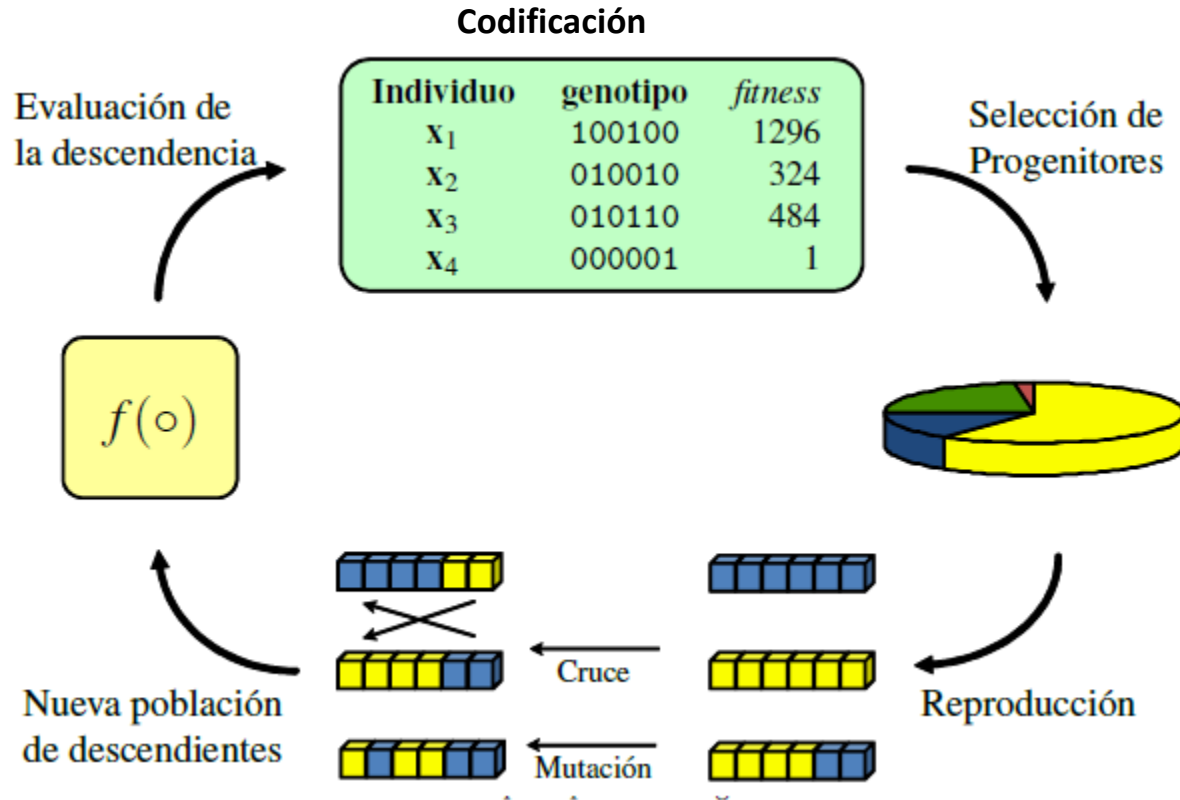
- Mecanismos de evolución
- Algoritmos Genéticos
 - Operadores genéticos
 - Selección
 - Ejemplos
 - Cruzamiento y Mutación
 - Ejemplos

Optimización

- Evaluación
 - Ejemplo
- Selección de sobrevivientes

Ejemplos completos

Ciclo evolutivo de un AG

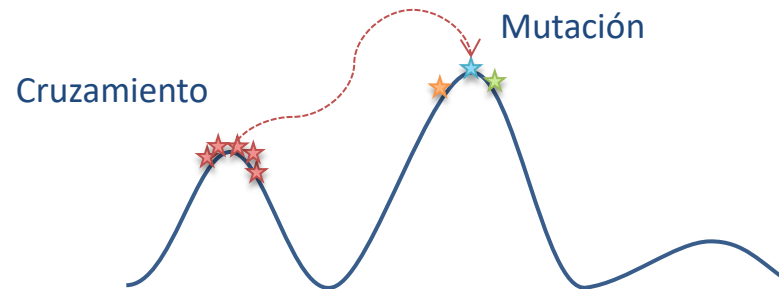


AG- resumen

Representación	Cadenas Binarias, Punto flotante/Real, Entera (Permutaciones)
Selección (Padres)	Ruleta, Torneo, Muestreo Estocástico Universal
Cruzamiento	<p>Pc (normalmente entre 0.6 y 0.99)</p> <ul style="list-style-type: none">• Binaria y Entera: [Un punto, Multi-punto, Uniforme]• Punto flotante: [Cruce aritmético: único, simple, completo]• Permutaciones: [Orden, Cruzamiento parcialmente mapeado (PMX)]
Mutación	<p>Pm (normalmente entre $1/\text{tamaño_población}$ y $1/\text{longitud_cromosoma}$)</p> <ul style="list-style-type: none">• Binario: [Bit flip, Bitwise, Inversión]• Punto flotante: [no uniforme: altera aleatoriamente un alelo]• Permutaciones:[Inserción, Swap, Inversión, Perturbación]

Efectos de Cruzamiento y Mutación

- ▶ En general, es bueno tener ambas operaciones
- ▶ Cruzamiento tiene un rol **explotativo**, hace que la población genere individuos con la mejor combinación de alelos observados en el proceso evolutivo.
- ▶ Mutación tiene un rol **explorativo**, es una fuente de generación de nuevos alelos en la población con la esperanza de encontrar nuevas regiones prometedoras o escapar de la convergencia actual.



- ▶ Existen AGs con solo mutación, pero es raro encontrar AG con solo cruzamiento. En un AG, cuando una población no tiene **variedad**, el cruzamiento no será útil, porque tendrá propensión a simplemente regenerar a los padres.



Optimización

Optimización

- ▶ Un problema de optimización consiste en buscar y encontrar una solución o soluciones óptimas de acuerdo a una función en un determinado problema, considerando determinadas restricciones.

Por ejemplo:

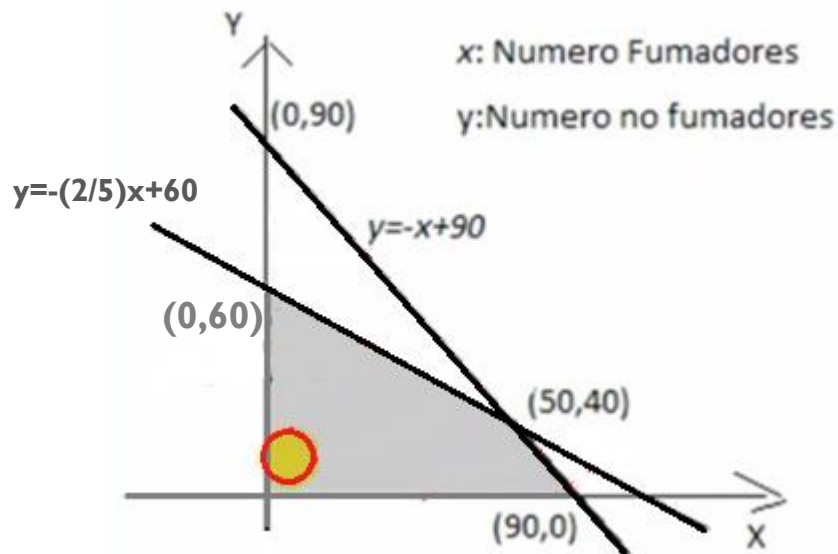
- Aumentar la rentabilidad de un negocio
- Aumentar la eficacia de un proceso,
- Reducir gastos,
- Reducir multas o pérdidas.



Optimización

- ▶ En la optimización clásica se busca encontrar el valor de las variables x^* que maximice o minimice una determinada función $f(x)$

Venta de asientos



Función Objetivo

maximizar la oferta de venta de pasajes

Variables de decisión

Cantidad de asientos reservados a fumadores

Cantidad de asientos reservados a no fumadores

Restricciones

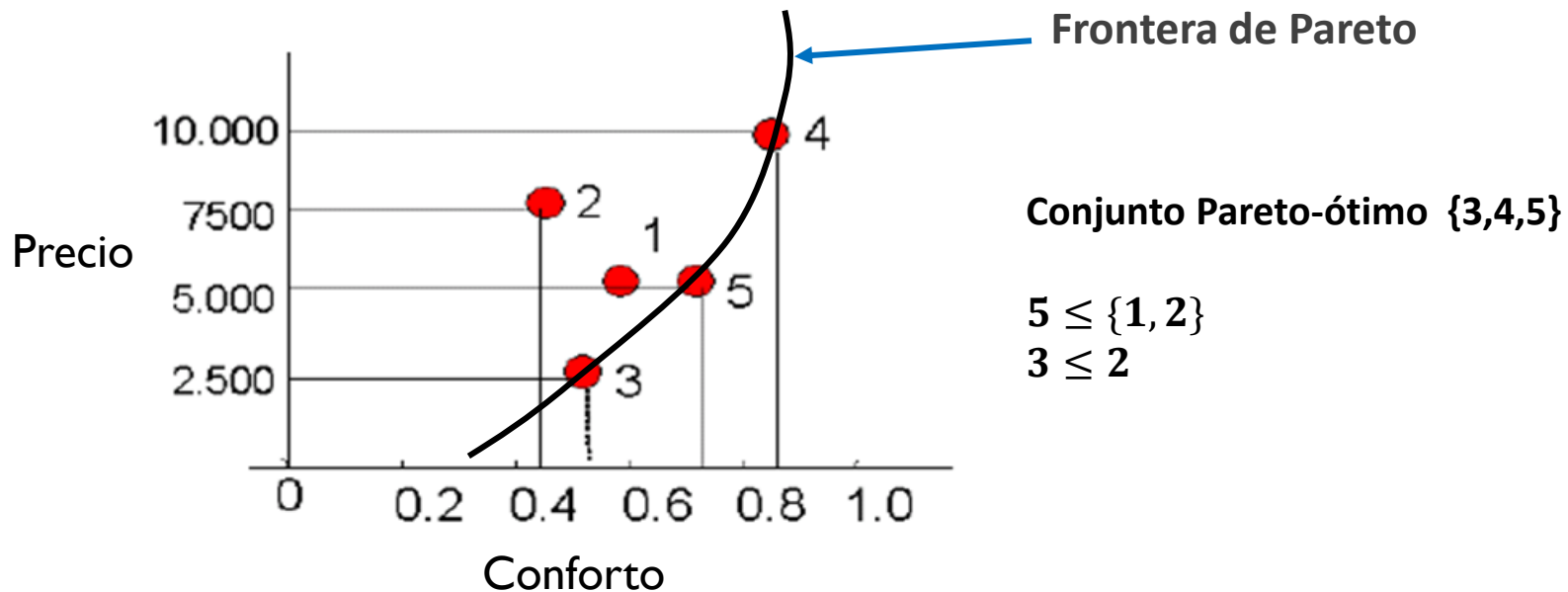
Cantidad total de asientos utilizados

Peso total del equipaje de los pasajeros

Cantidad de asientos reservados a no fumadores

Optimización multiobjetivo

- ▶ Un problema multiobjetivo no tiene una única solución eficiente, más bien, tiene un conjunto de soluciones eficientes que no pueden ser consideradas diferentes entre sí. A este conjunto de soluciones se le denomina **Frontera de Pareto**.



- ▶ Una solución es eficiente cuando no es dominada. Esto significa que es al menos tan buena como las otras en todos sus objetivos y es mejor en al menos uno de ellos.

Técnicas clásicas de Búsqueda y Optimización

- Existen muchas técnicas clásicas para resolver problemas con ciertas características específicas.

- Para optimización **lineal**, el método Simplex sigue siendo la opción más viable.

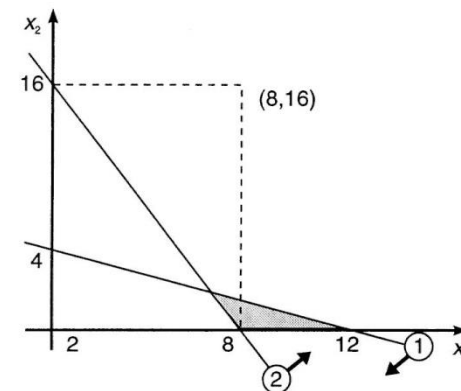
Tipo de problema (min o max) Función objetivo Beneficios o constantes

max $5x_1 - 6x_2 + 2x_3$

Restricciones $\rightarrow 3x_1 + 2x_2 - x_3 = 12$

$\rightarrow 2x_1 - 4x_2 + 4x_3 = 8$

Variables ($x_i > 0$)



Simplex

- Para optimización **no lineal**, hay métodos directos (p.ej. la búsqueda aleatoria) y métodos no directos (p. ej. el método del gradiente conjugado).

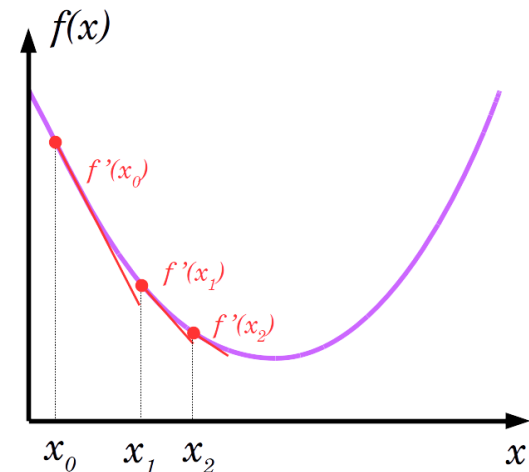
Función objetivo a ser maximizada

max $f(\mathbf{x}) = x_1x_2 + x_2x_3$

donde $\mathbf{x} = (x_1, x_2, x_3)$.

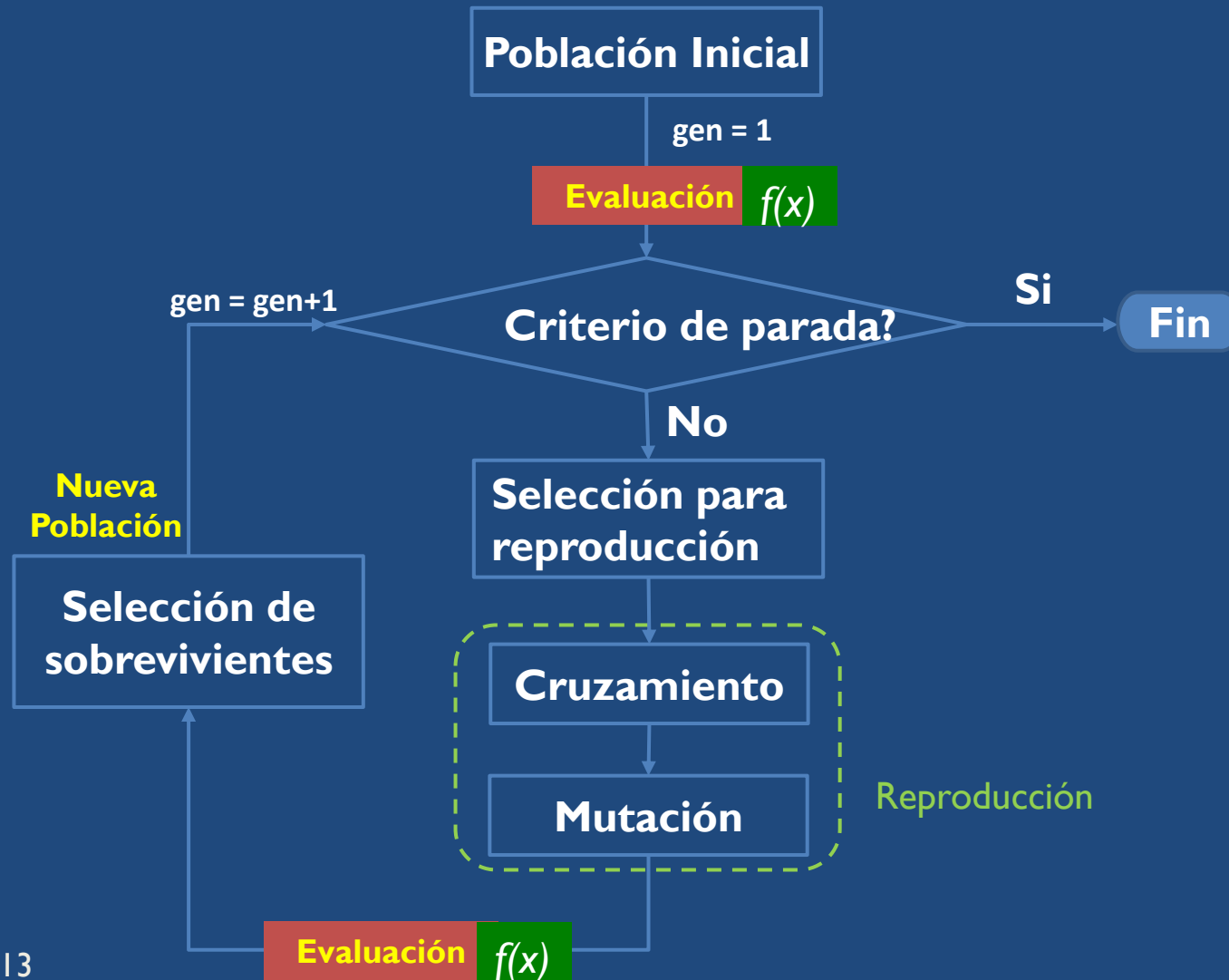
$x_1^2 - x_2^2 + x_3^2 \leq 2$

$x_1^2 + x_2^2 + x_3^2 \leq 10$



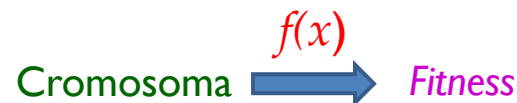
Gradiente descendiente


Evaluación de la aptitud



Evaluación del *fitness* en AG

- ▶ Objetivo: evaluar la conveniencia de las soluciones codificadas.
- ▶ La función de aptitud $f(x)$, también llamada función objetivo o de evaluación, asocia a un cromosoma un valor de aptitud denominado *fitness*.



Cromosoma	Función de aptitud $f(x)$	<i>Fitness</i>
0 1 1 0 1  13	$f(x) = x^2$	169
[0.1 , 0.8, 0.9]	$f(x) = x_1^2 + x_2^3 + x_3$	1.422

- El *fitness* con el mejor valor de aptitud de un cromosoma nos indica que tan cerca de la solución óptima (máximo/ mínimo) está un cromosoma.
- ▶ La función de aptitud es la única información que los AGs usan mientras buscan posibles soluciones.
- ▶ La función de aptitud es proporcionada por el usuario.

Ejemplo de evaluación del *fitness*



Representación Binaria

- ▶ Representación: binaria
- ▶ Función de aptitud : $f(x) = x^2$
- ▶ Selección: torneo (size_torneo=3)
- Cruzamiento: 1 punto
- Mutación: bit flip

Población de Cromosomas	<i>Valor decodificado del cromosoma (x)</i>	<i>Fitness</i>
0 1 1 0 1	13	169
1 1 0 0 0	24	576
0 1 0 0 0	8	64
1 0 0 1 1	19	361

Selección de sobrevivientes



Selección de sobrevivientes

- ▶ Objetivo: escoger una población de tamaño \mathbf{N} , para la siguiente generación (X_{t+1}) a partir de la población actual (X_t), existen diversas propuestas :

1. Reemplazo total

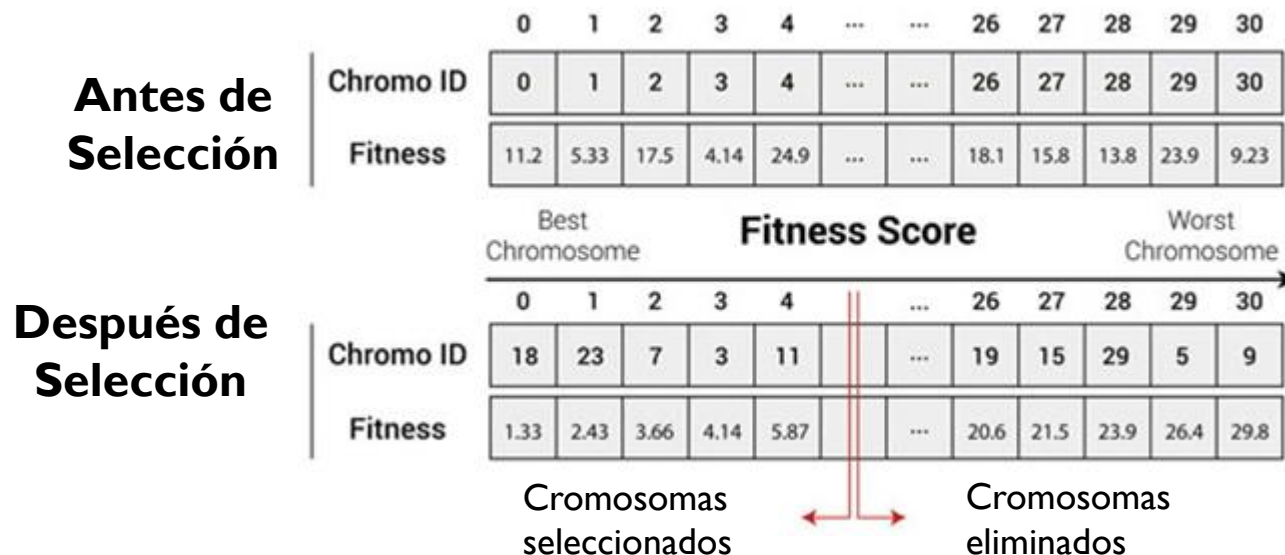
Este tipo de reemplazo es el más simple, ya que la población para la siguiente generación estará compuesta por todos los hijos (descendientes), generados en la generación actual.

Cuando todos los hijos reemplazan a los padres, se pueden perder buenos individuos ya encontrados.

Selección de sobrevivientes

2. Selección por ranking

Esta selección consiste en unir la población de padres con la de hijos y seleccionar los **N** mejores individuos ordenados por *fitness* (la mejor mitad de cada población para no alterar el tamaño **N**).



Ejemplo de selección donde la población se ordena según la puntuación del *fitness*

Selección de sobrevivientes

3. Selección Elitista

Tiene como objetivo asegurar que los cromosomas de los miembros más aptos de una población (la élite x_{ibest}), pasen a la siguiente generación (X_{t+1}) sin perderse por alteraciones de los operadores genéticos o depender de la selección o reproducción.

$$x_{ibest} \in X_t \rightarrow X_{t+1}$$

El elitismo garantiza que la mejor aptitud de una población nunca será peor de una generación t a la siguiente $t + 1$.

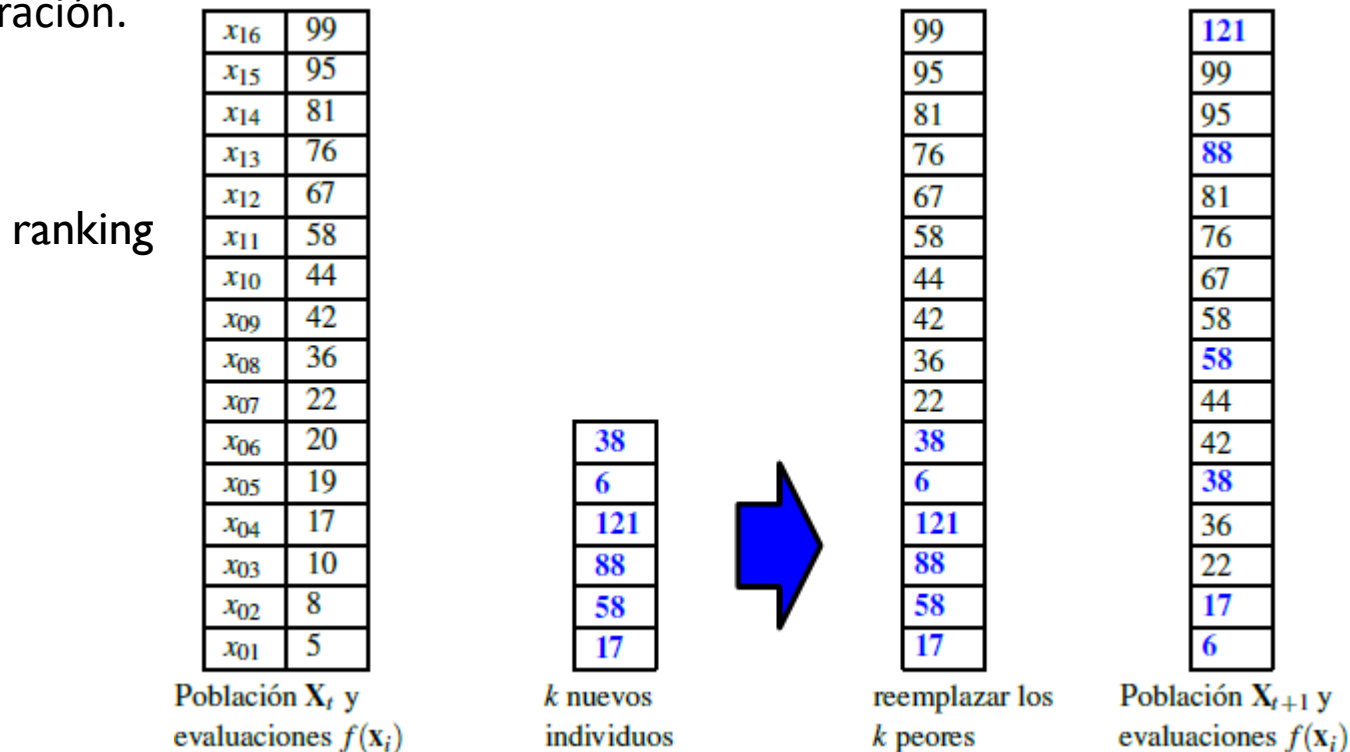
No es una condición suficiente para encontrar el óptimo global pero sí está demostrado que es una condición necesaria para garantizar la convergencia al óptimo de un algoritmo genético.



Selección de sobrevivientes

4. Selección Uniforme (Steady-State)

En cada generación son generados unos pocos cromosomas y ellos reemplazan a un número igual de padres con menor o igual *fitness* para formar la siguiente generación.



Esta estrategia es útil cuando la representación de una solución se distribuye en varios cromosomas, posiblemente en toda la población.

Selección de sobrevivientes

5. Ruleta

Se crea un pool formado por cromosomas de la generación actual, en una cantidad proporcional a su *fitness*. Si la proporción hace que un individuo domine la población, se le aplica alguna operación de escalado. Dentro de este pool, se escogen parejas aleatorias de cromosomas y se emparejan.

6. Torneo

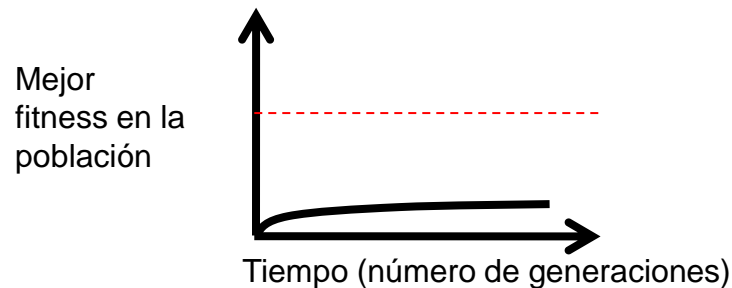
Se escogen aleatoriamente un número M de individuos de la población, y el que tiene puntuación mayor se reproduce, sustituyendo en su descendencia al que tiene menor puntuación. Este proceso se repite todas las veces necesarias hasta formar la nueva población.

Criterio de parada



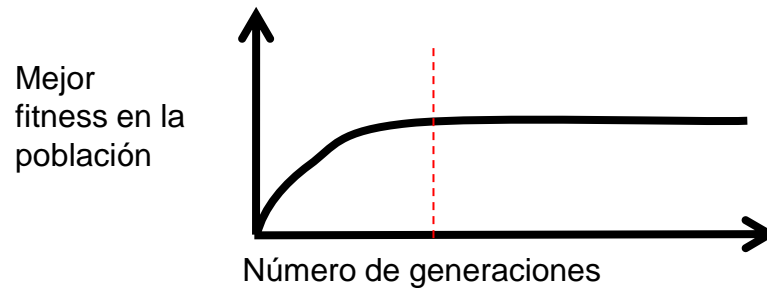
Criterio de parada

- Tiempo de ejecución
- Número máximo de generaciones alcanzado.
- El valor del *fitness* permanece por debajo de un valor umbral durante un tiempo determinado (varias generaciones).

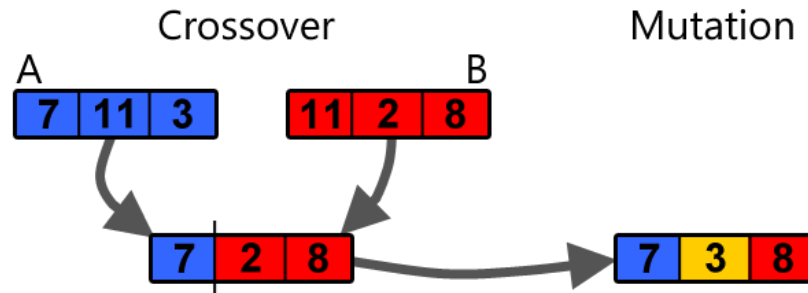


Criterio de parada

- Convergencia: En las ultimas k - iteraciones no hubo mejora en el *fitness*



- La diversidad de la población cae por debajo de un umbral dado



Parámetros y reporte de un AG

Parámetros de un AG

- ▶ Un AG requiere la configuración de una serie de parámetros de entrada:
 - tipo de representación,
 - longitud de cromosomas,
 - tamaño de la población,
 - cantidad de alelos (dependiente de la representación: binario, string, entero),
 - tipo de cruce,
 - tasas de mutación ($n=3$, $n = \% \text{ del tamaño de la población}$),
 - Selección de sobrevivientes (elitismo, ranking, etc)
 - Condición de parada del AG (cantidad de experimentos alcanzado, tiempo max, nro de generaciones max, etc),

Cada parámetro tiene un valor predeterminado.



Reporte del AG

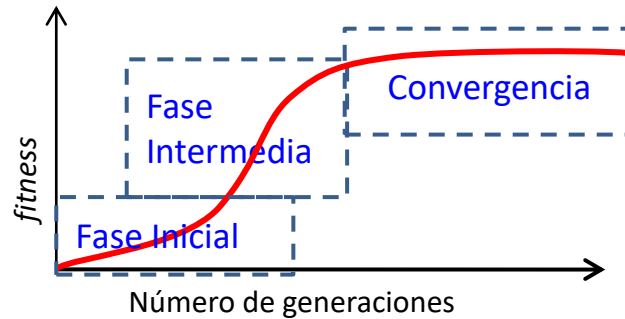
- ▶ El programa del AG debe generar un reporte con una descripción del rendimiento del algoritmo, conteniendo por ejemplo:
 - Media y varianza de los mejores cromosomas, rango de varias mediciones entre fitness, rendimiento promedio de la población actual, mejor valor fitness actual, tiempo de procesamiento, entre otras.

```
Poblacion inicial, best_fitness = 29.221853968951123
generacion 0, best_fitness = 8.5, best_chromosoma = [1, -1.168]
generacion 1, best_fitness = 2.0, best_chromosoma = [1, 1]
generacion 2, best_fitness = 1.0, best_chromosoma = [1, 0]
generacion 3, best_fitness = 1.0, best_chromosoma = [1, 0]
.
.
.
generacion 19, best_fitness = 0.0, best_chromosoma = [0, 0]
```

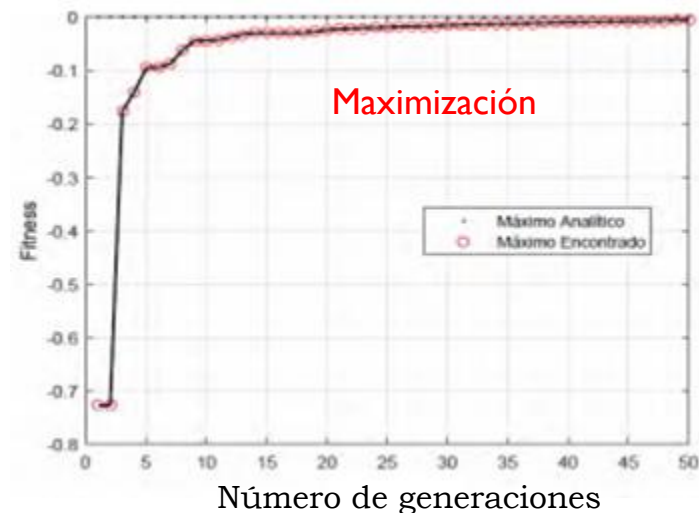
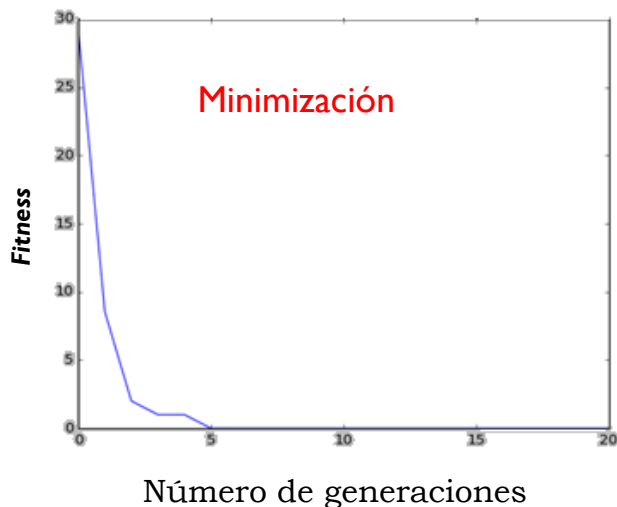
Registro de las diversas fases evolutivas

Reporte del AG

- Registros de las diversas fases evolutivas



Desempeño del mejor *fitness* por generación



Un AG completo



Problema de búsqueda de frases

- Problema: Buscar una frase (*target string*) a partir de una población de cadenas aleatorias.
- Representación: string, cada cromosoma puede generar n^p posibles frases, donde n representa el número de caracteres y p el tamaño de la frase

La búsqueda exhaustiva que una frase de 8 caracteres puede generar es de 53^8 . Donde cada carácter puede tomar una de 53 caracteres posibles (26 minúsculas, 26 mayúsculas y un espacio en blanco).

- Fitness: suma 1 a cada coincidencia entre los caracteres del cromosoma con el *target string*

```
fitness = 0
for i in range(len(chromosome)):
    if chromosome[i] == target_string[i]:
        fitness += 1
```

- Selección: Ruleta
- Cruzamiento: 1 punto
- Mutación: bit flip
- Selección de sobrevivientes: por ranking

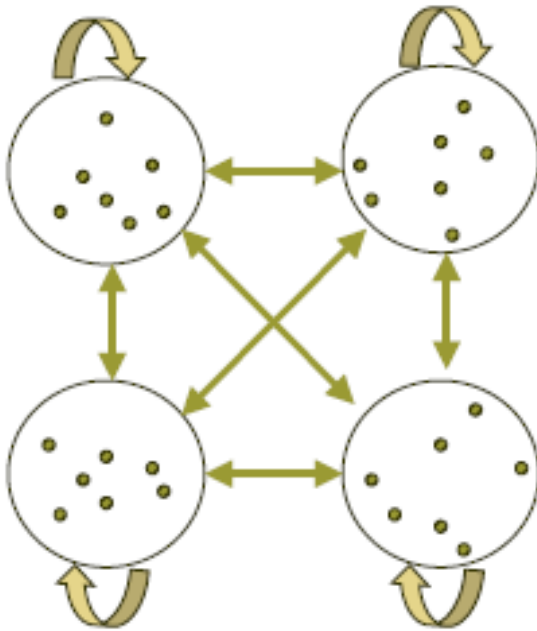
Otras características de los AGs

AG - Características

Los AGs tienen naturaleza paralela: En cada iteración podemos tener un número x de poblaciones que se inician y evolucionan paralelamente.

Una población puede subdividirse en grupos a los que se denomina subpoblaciones.

- A cada n -ésima generación, las subpoblaciones intercambian individuos .



0	1	1	0	1	0
1	0	0	1	1	0
0	0	0	1	1	0
1	0	1	1	0	1

Subpoblación 1

→
Migración

0	1	1	0	1	0
1	1	1	1	0	1
0	0	0	1	0	1

Subpoblación 2

Migración: permite la transferencia de los genes de una subpoblación a otra.



AG – Tipos de Paralelización

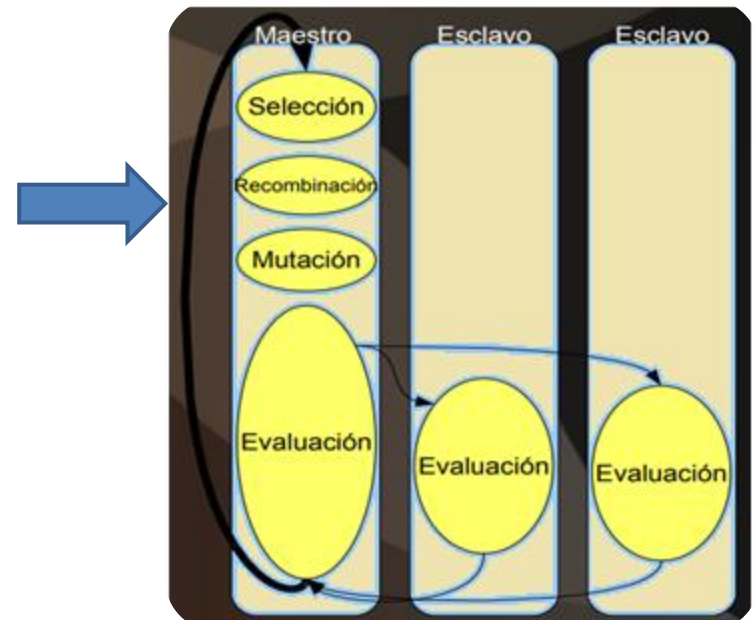
► Principales métodos:

1. Grano fino: se realiza a nivel de instrucción.

- Sobre una única población, diversos procesadores realizan una parte de cada paso del algoritmo (selección, cruce y mutación) . Cambia la estructura del algoritmo.

2. Grano medio: esta se realiza habitualmente de forma automática en los compiladores. Existen dos formas:

- Se mantiene una única población y se paraleliza la evaluación de los individuos por cada procesador (el programa se paraleliza a nivel de bucle).
- Se divide la población y se realizan ejecuciones de distintos AGs simultáneamente en cada procesador (no cambia la estructura del algoritmo)



AG – Tipos de Paralelización

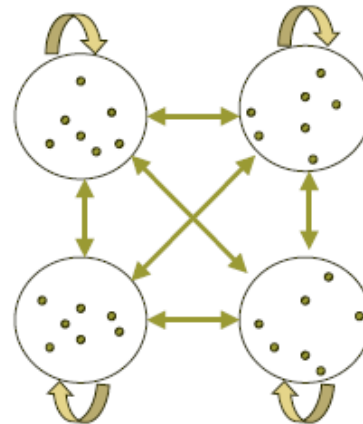
3. Grano grueso: si el número de individuos en la población es muy... muy grande, esta se divide en múltiples subpoblaciones entre diferentes nodos (islas), siendo cada uno de ellos el responsable de realizar los cálculos sobre sus datos locales.

Incluye un nuevo operador denominado migración, este operador se usa sobre cada subpoblación:

- A cada n-ésima generación, las poblaciones intercambian individuos.

Parámetros a considerar

- ❖ Tamaño de la población
- ❖ Topología de islas
- ❖ Frecuencia de migración
- ❖ Tasa de migración



Nota: Debido a que cada subpoblación evoluciona independientemente, el porcentaje de migración será muy importante para obtener resultados satisfactorios.

AG - Aplicaciones

- ▶ Algunas aplicaciones generales de los AGs son las siguientes:
 - ❖ Optimización (estructural, de topologías, numérica, combinatoria, etc.)
 - ❖ Aprendizaje de máquina (sistemas clasificadores, predicción)
 - ❖ Bases de datos (optimización de consultas)
 - ❖ Reconocimiento de patrones (por ejemplo, imágenes)
 - ❖ Planeación de movimientos de robots

AG en problemas de Optimización

La optimización se aplica a un innumerable número de problemas:

- ▶ Diseño de equipos y estructuras para minimizar peso
- ▶ Encontrar trayectorias óptimas de vehículos
- ▶ Diseño de estructuras de obras civiles (puentes, torres, chimeneas, presas. . .) a menor precio
- ▶ Peso mínimo de estructuras resistentes a terremotos y viento
- ▶ Diseño de reservas de agua para un uso eficiente
- ▶ Diseño óptimo de estructuras de plástico
- ▶ Diseño óptimo de engranajes, levas, y todo tipo de componentes mecánicos
- ▶ Camino más corto pasando por una serie de puntos
- ▶ Planificación de una producción óptima
- ▶ Análisis de datos estadísticos y construcción de modelos a partir de datos experimentales para obtener la representación más realista de un fenómeno físico
- ▶ Control de los tiempos de espera en una línea de producción para minimizar costes
- ▶ Planificación de la mejor estrategia para obtener el máximo beneficio en presencia de competidores
- ▶ Diseño óptimo de sistemas de control ...

1. Veslin, Elkin. (2014). Aplicación de algoritmos genéticos en problemas de Ingeniería.

Ampliaciones y mejoras

- ▶ Existen numerosas variantes y mejoras de los algoritmos genéticos. Una de las más habituales es la del elitismo.
- ▶ Otras propuestas para mejorar los algoritmos genéticos incluyen el cruce entre más de dos individuos y los algoritmos **meméticos**, en los que los individuos tienen la capacidad de aprender y transmitir lo aprendido a sus descendientes.

Algunas aplicaciones con AGs



Problema de las N reinas

- Problema: Colocar N reinas en un tablero de ajedrez de NxN sin que se amenacen entre ellas. Una reina amenaza a otra si está en la misma fila, columna o diagonal.
- Representación: **entera** , cada dígito indica la fila dentro de la columna i-ésima donde está situada la reina i
- Fitness: pares de reinas que no se atacan
$$\text{Máximo fitness} = \frac{N}{2} * (N - 1)$$
- Selección: Torneo
- Cruzamiento: 1 punto
- Mutación: bit flip
- Selección de sobrevivientes: por ranking

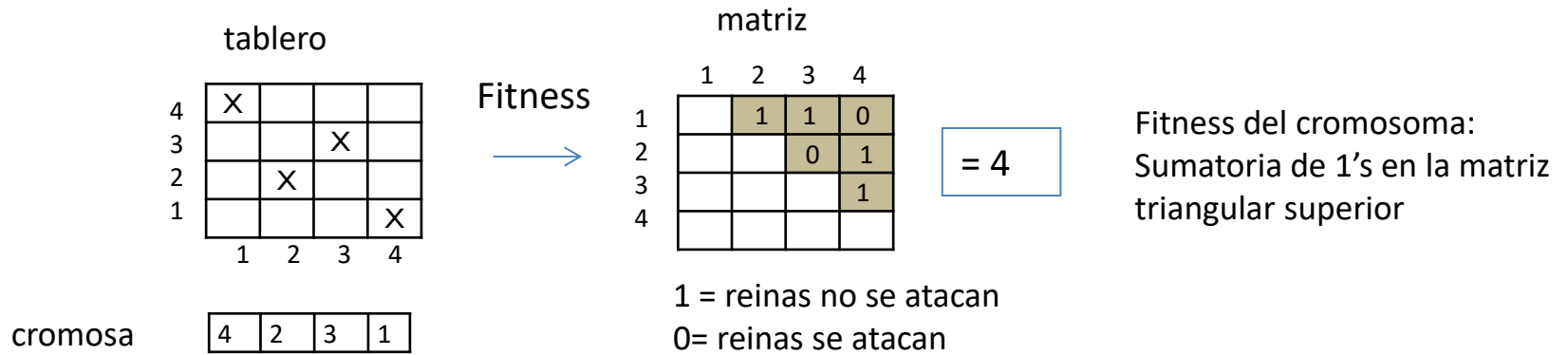
❖ Para implementación con **permutación**:

- Cruzamiento: Implementar (Cx PMX, Cx en orden)
- Mutación: Implementar (Inserción, Swap, Inversión Perturbación)



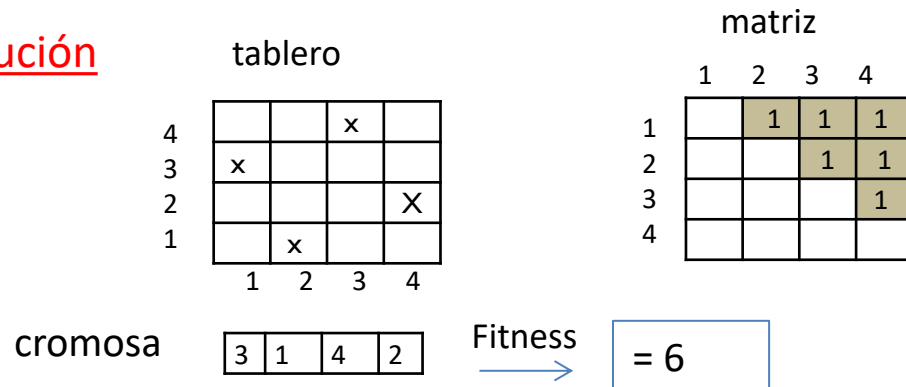
Problema de las N reinas

Ejemplo: Evaluación del *fitness* para $n=4$:



$$\text{Máximo fitness} = \frac{n}{2} * (n - 1) = 6$$

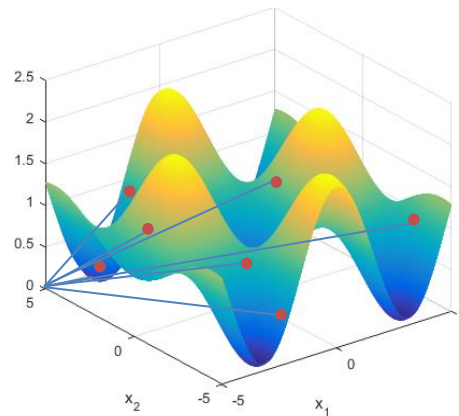
Solución



Problema de optimización continua

- Problema: para minimizar funciones continuas no convexas
- Representación: punto flotante, cada cromosoma representa un vector

	x_1	x_2	x_3
Cromosoma 1	0,1	0,8	0,9
Cromosoma 2	0,3	0,2	0,3
...			
Cromosoma n-1	0,2	0,5	0,6
Cromosoma n	0,2	0,5	0,6



De Jong's function:

$$\text{Minimizar: } f(x) = \sum_{i=1}^n x_i^2$$

Área de búsqueda se restringe a un hipercubo:

$$-5.12 \leq x_i \leq 5.12$$

- Fitness: maximizar/ minimizar una determinada función de aptitud
- Selección: Ruleta
- Cruzamiento: 1 punto
- Mutación: bit flip
- Selección de sobrevivientes: ranking