Unidad 2

Inteligencia artificial: *Algoritmos Genéticos*

Dra. Soledad Espezua
sespezua@pucp.edu.pe

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

PUCP

Contenido

- Unidad 2
  - Introducción a los Algoritmos Genéticos
    - Conceptos, características, tipos de codificación.
  - Algoritmos Genéticos
    - Operadores genéticos: Selección, Cruzamiento y Mutación
    - Selección de sobrevivientes
    - Condición de parada
    - Características avanzadas: paralelización
    - Consideraciones en los AGs
    - Resumen

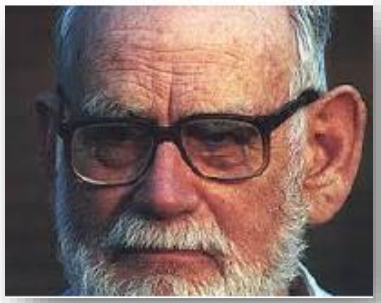
Introducción

❑ Historia

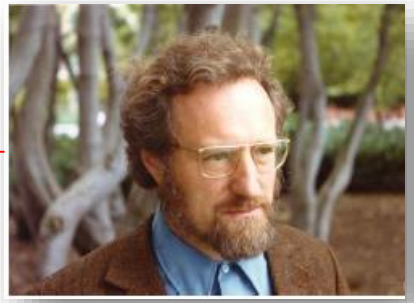
Padre del AG

- John Henry Holland [3] (1929-2015), formaliza los principios básicos de los "Algoritmos genéticos", 1975.

1957 — 1962 — 1975 — 1989



- Alex S. Fraser[1] (1923-2002) Evolucion de sist. Biologicos, 1957.



- Hans J. Bremermann[2] (1926-1996) Evolución como un proceso de optimización,1962.

- David E. Goldberg[4], popularización, 1989.

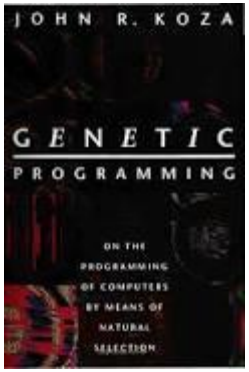
1. *Fraser, A. S. (1957). "Simulation of genetic systems by automatic digital computers. I. Introduction", Australian Journal of Biological Sciences.*
2. *Bremermann, H.J. (1962): "Optimization through evolution and recombination". In: Self-Organizing systems .*
3. *Holland, J. H. (1975). Adaptation in Natural and Artificial Systems.*
4. *Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning.*

Introducción

"Un AG es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual.

Cada uno de estos objetos matemáticos suelen ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud."

by John Koza[1]

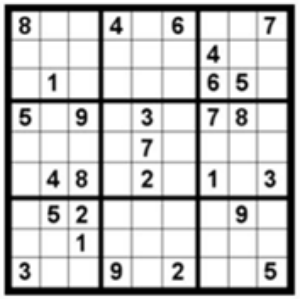


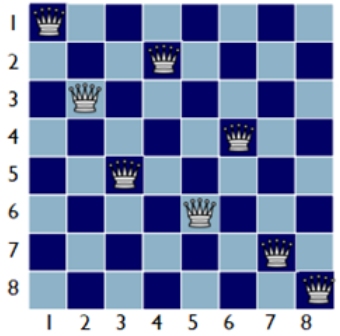
John Koza

1. Koza, J. R. "Genetic Programming. On the Programing of Computers by Means of Natural Selection",1992.

Aspectos generales

- AG encuentra soluciones a problemas que los humanos tendrían dificultades para resolver o no podrían resolver en absoluto.
  - Optimización combinatoria, optimización restringida, selección de rutas, asignación de tareas/recursos.





búsqueda de camino óptimo (TSP)

Optimizar la Tarea: duración, número de tareas, fecha límite de tareas
Optimizar Recursos: capacidad de procesamiento de nodos, potencia de procesamiento, memoria, etc.

Horarios en un aeropuerto y gestionar el tráfico aéreo

Aplicaciones en problemas de ingeniería

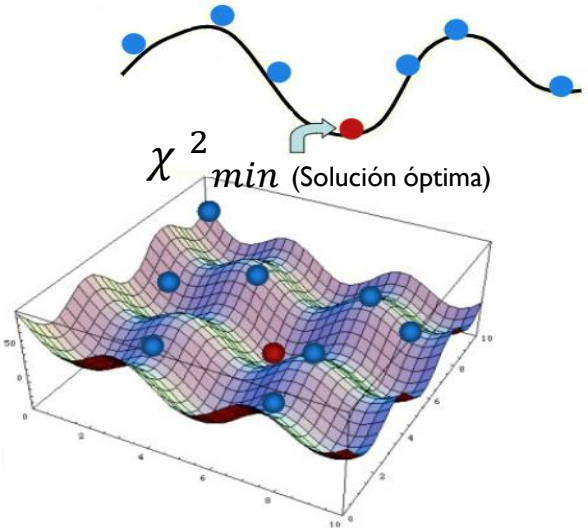
Los AGs han sido propuestos para resolver un gran número de problemas:

- En optimización:
  - Encontrar trayectorias óptimas de vehículos
  - Encontrar el peso mínimo de estructuras resistentes a terremotos y viento
  - Diseñar equipos y estructuras buscando minimizar el peso
  - Diseñar estructuras de obras civiles (puentes, torres, chimeneas, presas. . .) buscando minimizar el precio
  - Diseño de reservas de agua para un uso eficiente
  - Diseño de engranajes, y todo tipo de componentes mecánicos
- Planificación de producciones óptimas
- Planificación de estrategias para obtener el máximo beneficio en presencia de competidores
- 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 costos.

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

Características del AG

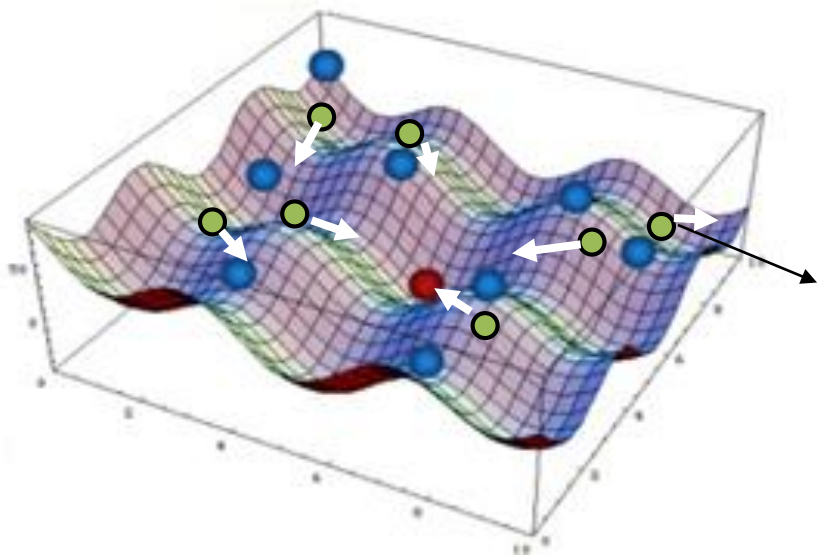
1. Constituyen el paradigma más completo de la CE.
2. No tienen inteligencia. No aprenden.
3. Son algoritmos de búsqueda estocástica que distribuyen su población uniformemente sobre toda la superficie. No garantizan optimalidad.



Analizan en paralelo todo el espacio de soluciones

Características del AG

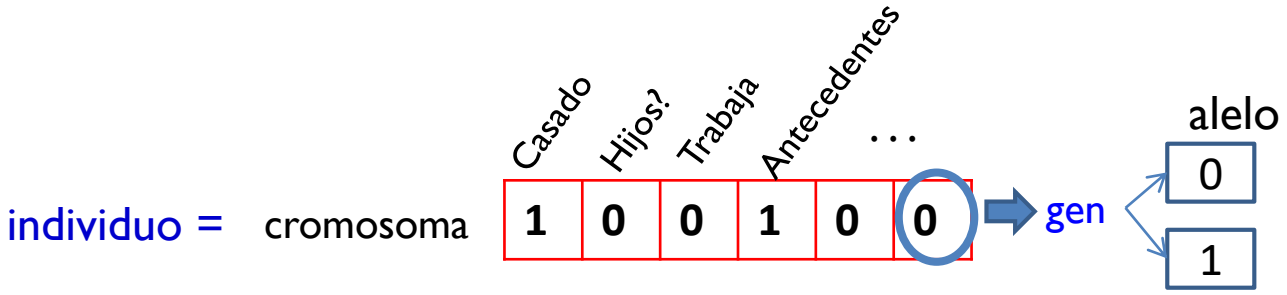
4. A pesar de ser estocásticos, el proceso de búsqueda sigue un pseudo-proceso de búsqueda guiada debido al operador de cruzamiento.



Hijos se generan cerca de los padres

Cada individuo, da pasos direccionados, explorando informaciones genéticas para encontrar nuevos puntos de búsqueda, donde se espera mejores desempeños.

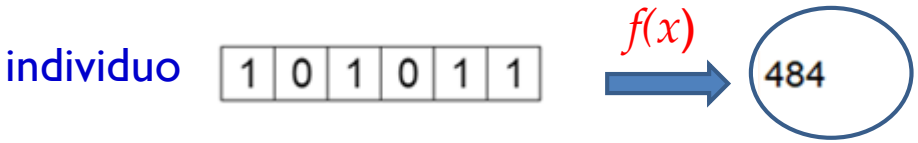
Terminología fundamental



- **Cromosoma**
  - Como sinónimo de "individuo".
  - Codifica una posible solución a partir de un conjunto de atributos.
  - Los cromosomas están formados por genes.
- **Gen**
  - Codifica un atributo/característica particular.
- **Alelo**
  - Se denomina alelo a cada valor diferente que un gen pueden adquirir.

Terminología fundamental

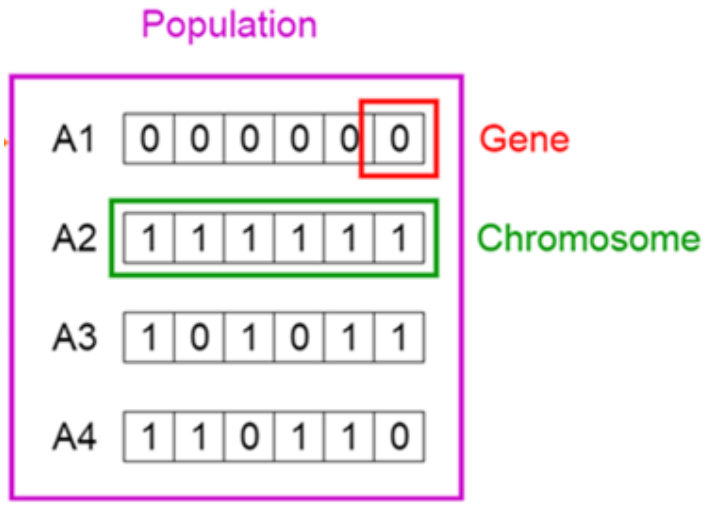
- ***Fitness* o aptitud**
  - Es un número positivo que nos dice qué tan adecuada es la solución/ individuo.



- ***Función de fitness***
  - Se usa una función de fitness $f(x)$ para evaluar los individuos generados.
  - Esta función debe dar mayores valores para mejores individuos. En optimización por ejemplo:
    - Si el problema es maximización, las soluciones de mejor fitness son la de mayor costo.
    - Si el problema es minimización, las soluciones de mejor fitness son la de menor costo.

Terminología fundamental

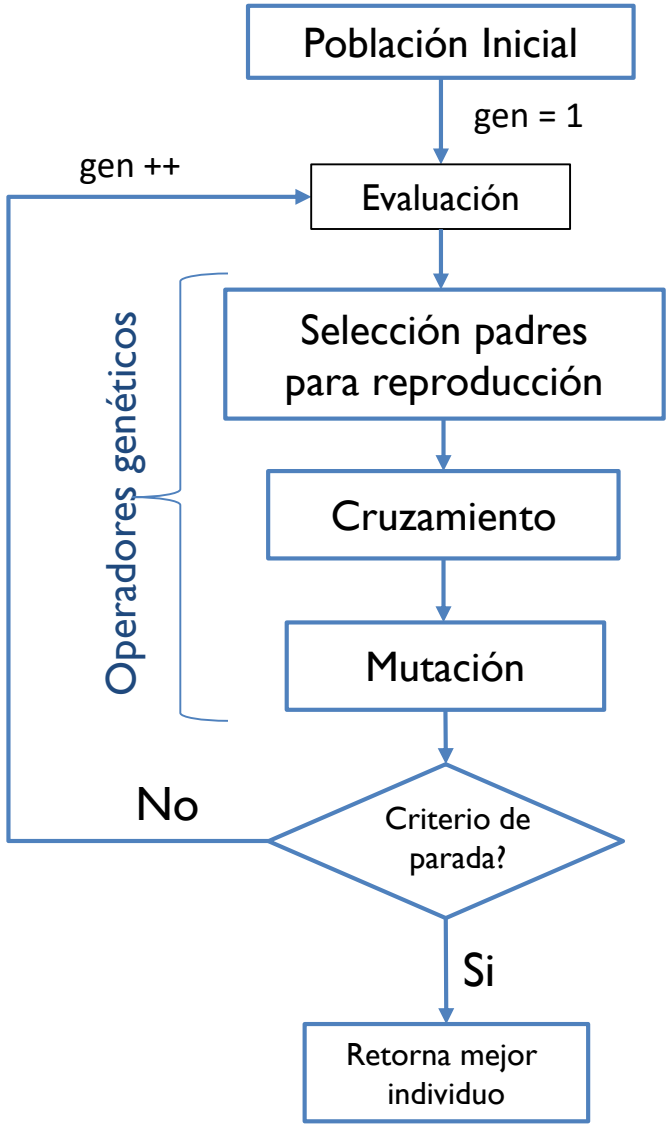
- ***Población:*** es un conjunto de individuos, por lo general de tamaño fijo. Cada individuo es considerado como una posible solución.



- Poblaciones pequeñas tienen una gran posibilidad de perder la diversidad necesaria y corren el riesgo de no explorar todo el espacio del problema. soluciones.
- Grandes poblaciones perderán gran parte de su eficiencia debido al retraso en la evaluación de la función de fitness (costo computacional).

Funcionamiento fundamental

1. Generar una población inicial (aleatoria) de ***n*** individuos
2. Evaluar el *fitness* de cada individuo
3. Seleccionar los individuos más adaptados. Descartar una parte de los individuos menos adaptados.
4. Aplicar Cruzamiento (reproducción)entre los individuos más adaptados.
5. Aplicar Mutación en algunos individuos.
6. Si Criterio de parada es satisfecho :
   - Entonces ➡ retornar mejor individuo y terminar proceso
   - No ➡ regresar al paso 2.



Inicialización de la población

La inicialización de la población generalmente es aleatoria, pero puede incluir soluciones/individuos existentes o encontradas por otro método.

Tipos:

- Aleatoria uniforme

  Cada gen del cromosoma recibirá un valor proveniente de una distribución uniforme;

- Aleatoria no uniforme

  Cada gen del cromosoma recibirá un valor proveniente de una distribución no uniforme; haciendo que algunos cromosomas sean escogidos con mayor frecuencia que el resto.

- Aleatoria con "dope"

  Algunos individuos ya optimizados son ingresados en una población generada aleatoriamente. Esta alternativa presenta el riesgo de hacer que uno o más súper individuos tiendan a dominar en el proceso de evolución y causar el problema de convergencia prematura.

Tipos de representación de individuos

1. Representación Binaria

Es la representación tradicional usada para codificar un conjunto de soluciones. En esta representación un cromosoma es una cadena de bits (ceros o unos).

Población inicial (fenotipos)	x valor genotipo
01101	13
11000	24
01000	8
10011	19

2. Números reales

Los cromosomas se representan como vectores de valores reales:

	Población				
X_1	0.125	0.5	1.128	0.25	-0.12
X_2	0.5	0.5	1.3	-0.25	0.4

- Podemos usar directamente números reales en cada gene
- Esta representación es útil en problemas con muchas variables, donde se necesita una muy buena precisión para cada una de ellas.

3. Representación de enteros

- El cromosoma puede ser representado por un vector de valores enteros.

1.345=

1	3	4	5

- Esta representación es útil en problemas donde la tarea es organizar algunos objetos manteniendo un cierto orden/secuencia, como en el caso de la representación entera tipo permutación.

3. Representación de enteros

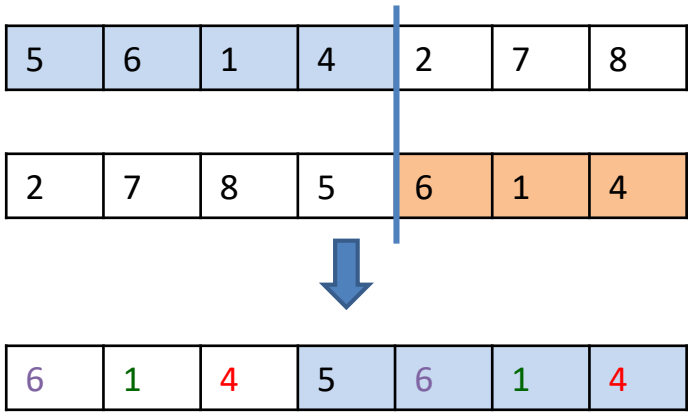
❖ Representación de Permutación

- Una permutación se representa como una lista de **n** enteros, cada uno de los valores ocurre exactamente una vez.

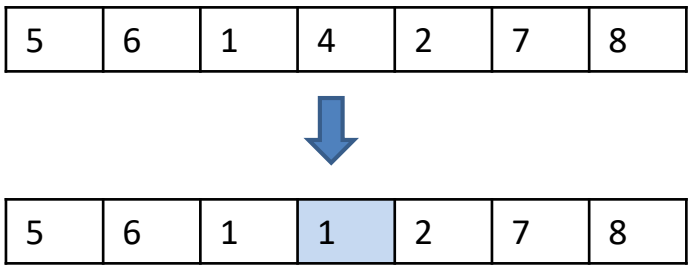
5	6	1	4	2	7	8

- Se necesita tener cuidado con los operadores de cruzamiento y mutación para garantizar que el resultado continúe siendo una permutación.

Cruzamiento



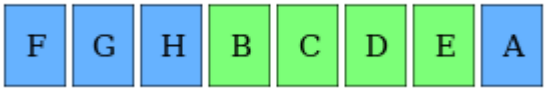
Mutación



Alterar un alelo puede implicar repetir un valor.

4. Representación tipo *string*

- Es un subtipo de la codificación binaria, donde la población resulta de cadenas del alfabeto.
- Un cromosoma representa un vector de valores tipo string.



- El nro de combinaciones posibles, depende de la cantidad de letras que se use para generar la población.

  Ejm. Un cromosoma de 8 posiciones con un alfabeto de 20 caracteres, puede tener hasta 20^8 posibles combinaciones.

Diferentes formas de representar un mismo problema

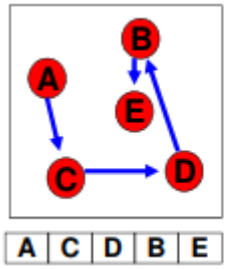
- Por ejemplo:
  - TSP de 6 ciudades, representado por sub-cadenas de 3 bits.

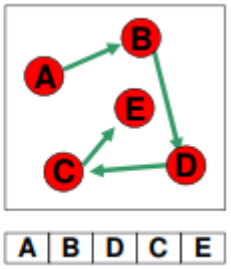
i	ciudad i	i	ciudad i
1	000	4	011
2	001	5	100
3	010	6	101

Representación binaria para un TSP con 6 ciudades

El camino 1 – 2 – 3 – 4 – 5 – 6 se representaría por medio de (000 001 010 011 100 101).

  - TSP de 4 ciudades, las ciudades representado por string.





  - TSP de 6 ciudades, las ciudades representado por enteros

ciudad	ciudades conectadas
1	2, 6, 3, 5
2	1, 3, 4, 6
3	2, 4, 1
4	3, 5, 2
5	4, 6, 1
6	1, 5, 2

Conexión de arcos, para las giras $(1\,2\,3\,4\,5\,6)$ y $(2\,4\,3\,1\,5\,6)$

Diferentes formas de representar un mismo problema

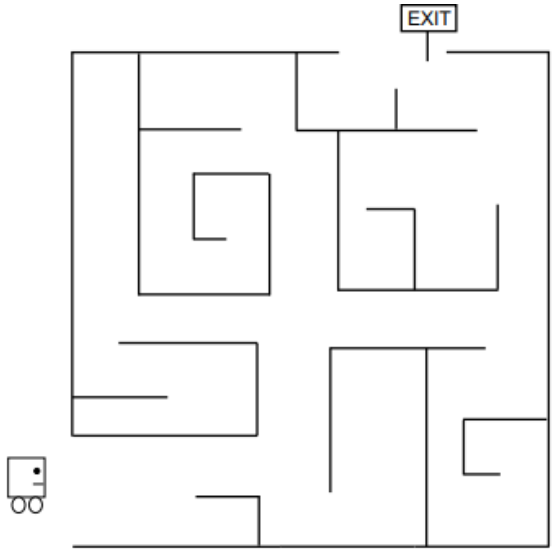
- Robot en el laberinto

Codificación binaria:

00 = Retrocede;
01= Izquierda;
10= Avanza;
11=Derecha

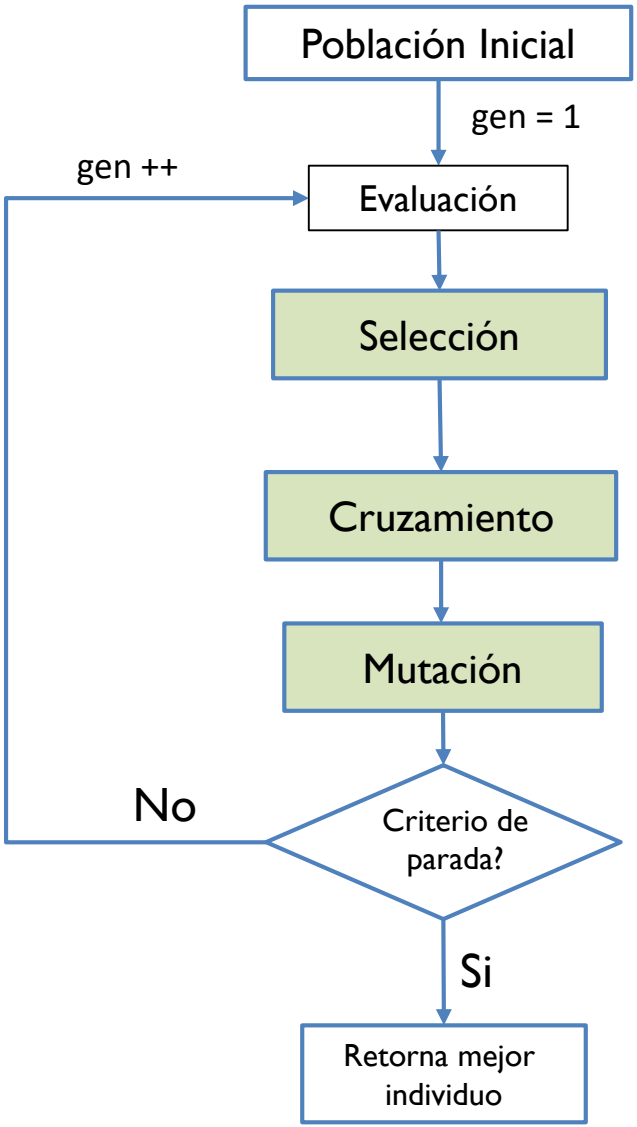
Codificación string:

R = Retrocede;
I = Izquierda;
A = Avanza;
D = Derecha



10 10 01 10 11 00 10 ....
A A I A D R A

Operadores genéticos fundamentales



1. Selección

- Es el operador que guía la dirección de evolución de una población.
  - Se selecciona un conjunto de individuos de la población (progenitores), basado en sus fitness, para generar nuevos individuos
  - Es usualmente probabilístico
    - Individuos con mejor *fitness* (más adaptados) tienen mayores posibilidades de trasmitir sus genes, aunque esto no es garantizado
    - Individuos con poco fitness pueden también ser seleccionados (con poca probabilidad), lo que ayuda a escapar de óptimos locales
  - Métodos comunes de selección de progenitores:
    a) Método de la ruleta
    b) Muestreo Estocástico Universal
    c) Selección por torneo

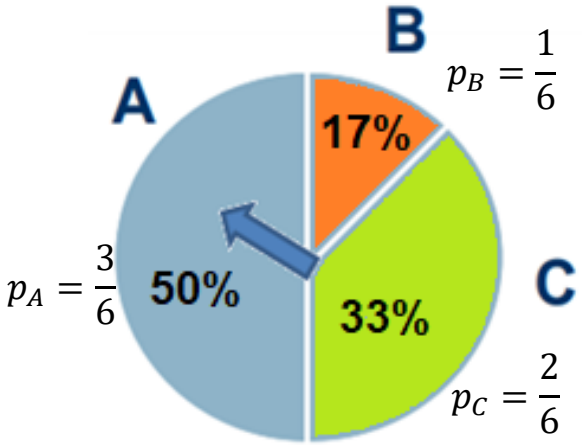
a) Método de la Ruleta

- Procedimiento:
  - Se coloca a todos los individuos en una ruleta, asignando a cada uno una "porción" de la ruleta cuyo ancho es proporcional a su fitness

f(A) = 3 → 50%

f(B) = 1 → 17%

f(C) = 2 → 33%



Cada individuo es representado en la ruleta proporcionalmente a su *fitness.*

  - Se gira la ruleta y se escoge el individuo donde la aguja se detiene para pasar a la próxima generación.
  - El sorteo se repite ***k*** veces para escoger ***k*** progenitores.

Ejemplo del método de la ruleta

Seleccionar 2 individuos de una población $\{x_1, x_2, x_3, x_4, x_5\}$ con valores *fitness*: $f(x_{1:5})=\{2,7,1,4,6\}$

- Calcular el $f_{ac}(N) = \sum_{i=1}^{N} f(x_i)$ =20 y generar un numero aleatorios entre [0 y 20], por eje.: r=[5]

Seleccionar el 1er individuo cuyo $f_{ac}(x_i) > r$

$f_{ac}(x_1) = 2$

$f_{ac}(x_2) = 7 + 2 = 9$ 9 > [5], seleccionar x_2

- Reiniciar la suma acumulada y retirar el individuo ya seleccionado y su fitness. Generar un numero aleatorios entre [0 y 13], por eje.: r=[11].

Seleccionar el 1er individuo cuyo $f_{ac}(x_i) > r$

$f_{ac}(x_1) = 2$

$f_{ac}(x_3) = 1+ 2 = 3$

$f_{ac}(x_4) = 4 + 1 + 2 = 7$

$f_{ac}(x_5) = 6 + 4 + 1 + 2 = 13$ 13 > [11], seleccionar x_5

individuos seleccionados:$\{x_2, x_5\}$

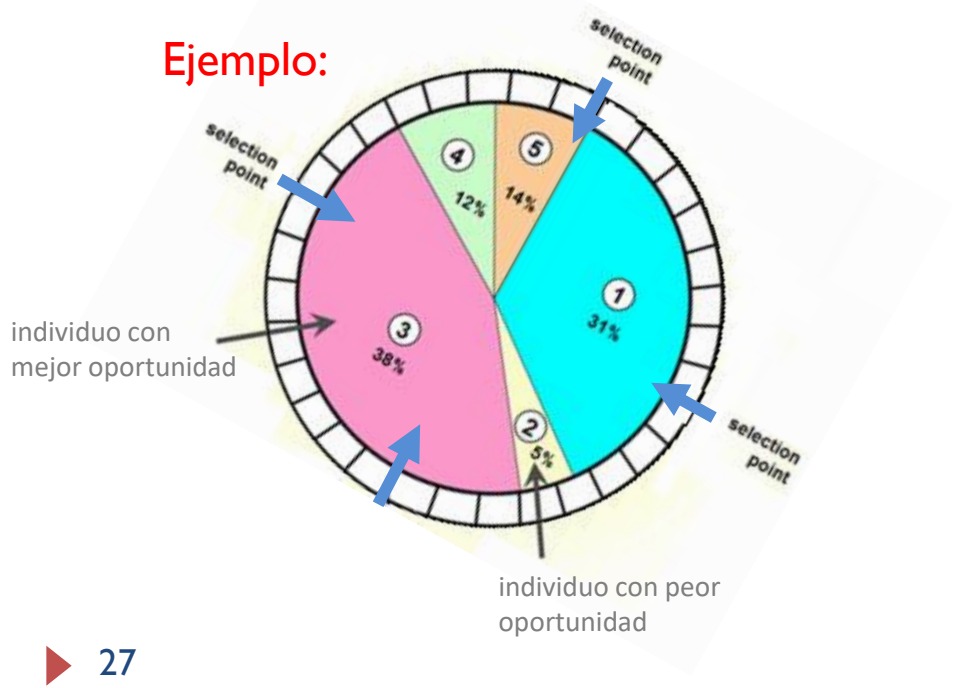
b) Muestreo Estocástico Universal

- SUS - *Stochastic Universal Sampling*.
- Objetivo: Minimizar la mala distribución de los individuos en la población en función de sus valores esperados.
- Procedimiento:

  Se coloca a todos los individuos en una ruleta, asignando a cada uno una "porción" de la ruleta proporcional a su fitness

  Para seleccionar ***k*** individuos, dividir la ruleta en ***k*** espacios igualmente espaciados (cada espacio es para una aguja). Luego girar la ruleta en conjunto una sola vez y seleccionar un individuo por cada ***k*** espacio.

Ejemplo:



$f(x_1) \rightarrow 31\%$
$f(x_2) \rightarrow 5\%$
$f(x_3) \rightarrow 38\%$
$f(x_4) \rightarrow 12\%$
$f(x_5) \rightarrow 14\%$

Para k =4 seleccionar:$\{x_1, x_3, x_3, x_5\}$

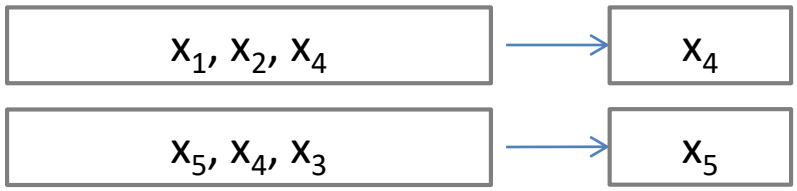
c) Selección por Torneo

- Objetivo:
  - Utilizar sucesivas disputas (comparaciones directas de los individuos) para elegir a los padres.
- Procedimiento:
  - Realizar ***k*** disputas para seleccionar ***k*** individuos. Cada disputa implica ***n*** individuos contrincantes elegidos al azar.
  - Generalmente se utiliza ***n* = 3** como número de contrincantes. El individuo con mejor *fitness* en la disputa es el ganador del "torneo".

Ejemplo:

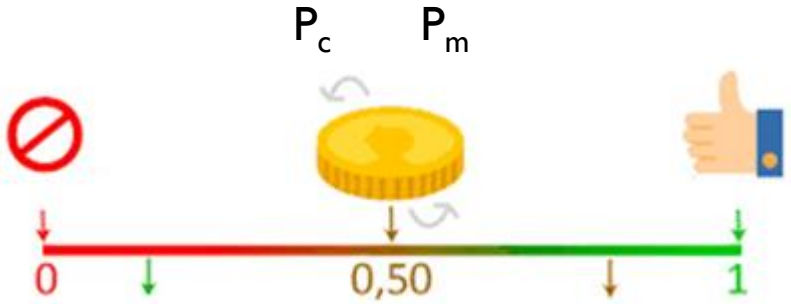
$f(x_1) \rightarrow 5$
$f(x_2) \rightarrow 11$
$f(x_3) \rightarrow 13$
$f(x_4) \rightarrow 24$
$f(x_5) \rightarrow 47$

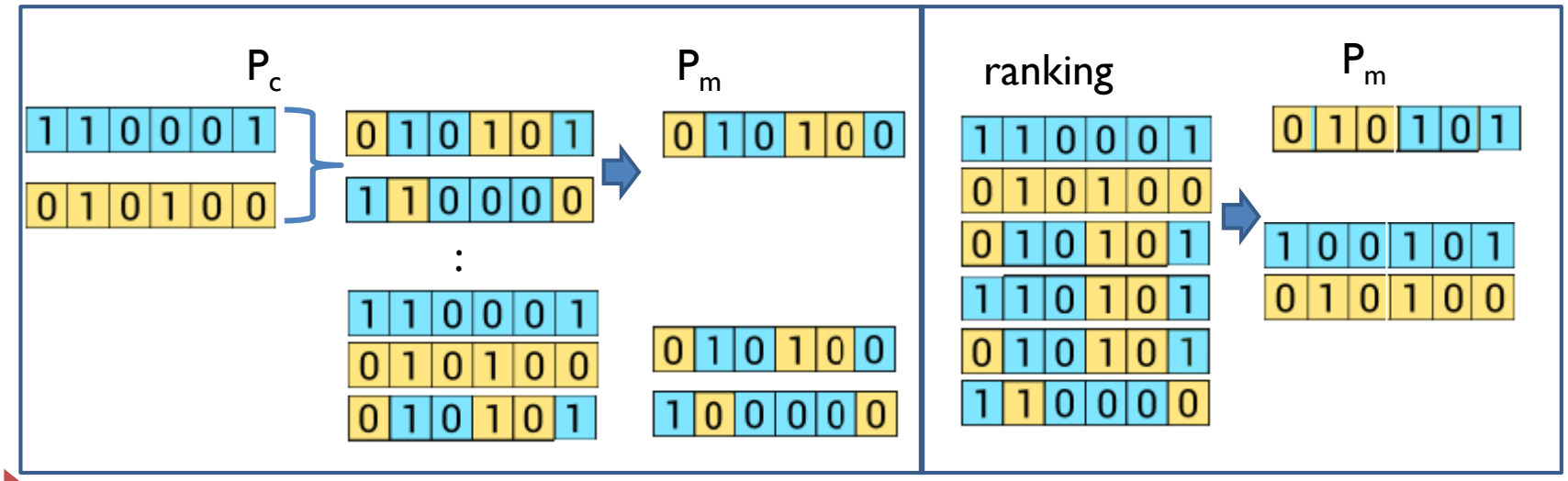
***k*=2; *n*=3**



Cruzamiento y Mutación

Se utilizan reglas de transición probabilísticas





2. Cruzamiento

- También llamado de recombinación.
- Responsable por generar nuevos cromosomas (sean mejores o peores) a partir de cromosomas prometedores.
  - Combina las informaciones genéticas de dos cromosomas (padres) para generar nuevos descendientes (hijos).
- Las versiones más comunes siempre crean dos hijos por cada operación.
- La probabilidad de cruzamiento (P_C) por cada par de cromosomas normalmente es entre [0.6 y 0.99].

2. Cruzamiento (Cx)

Tipos de cruzamiento por su representación:

2.1) Representación binaria y entera

- *Onepoint, multipoint, uniform*

2.2) En permutaciones:

- Cruzamiento en orden (*Order Crossover*)
- Cruzamiento parcialmente mapeado (PMX)

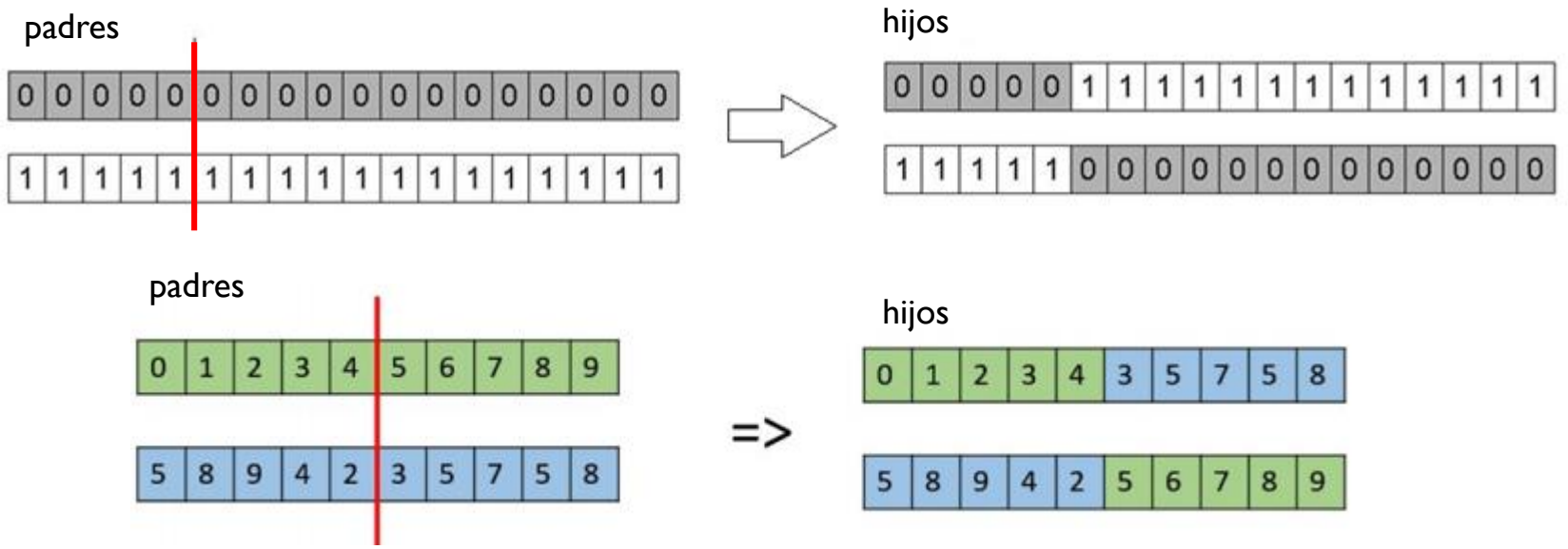
2.3) Representación en punto flotante

- Cruce aritmético único, simple, completo

2.1) Cx: representación binaria y entera

❑ *Onepoint*

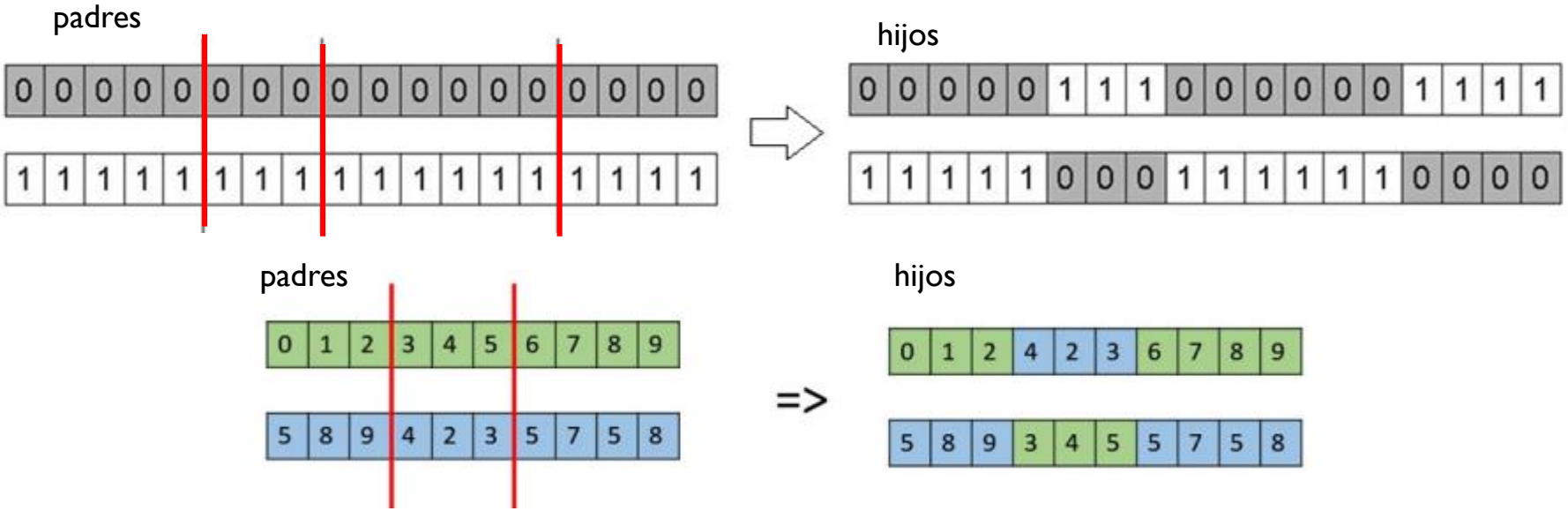
- Seleccionar 2 padres y elegir un punto aleatorio de corte en el tamaño del cromosoma.
- Crear a los hijos intercambiando las secciones a la derecha del punto de corte.



2.1) Cx: representación binaria y entera

❑ *Multipoint*

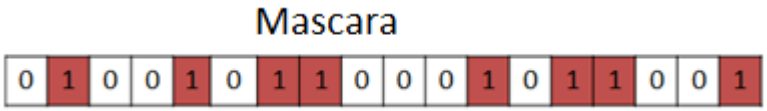
- Seleccionar 2 padres y elegir **n** puntos de corte aleatorios en los padres
- Partir a los padres en esos puntos
- Unir esas partes, alternando entre los padres



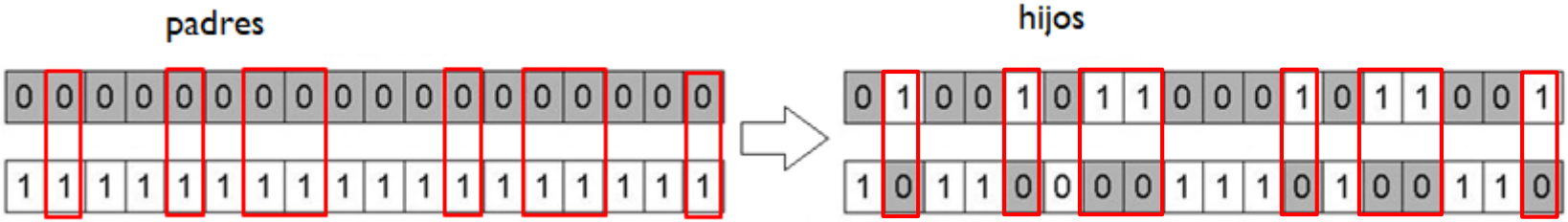
2.1) Cx: representación binaria y entera

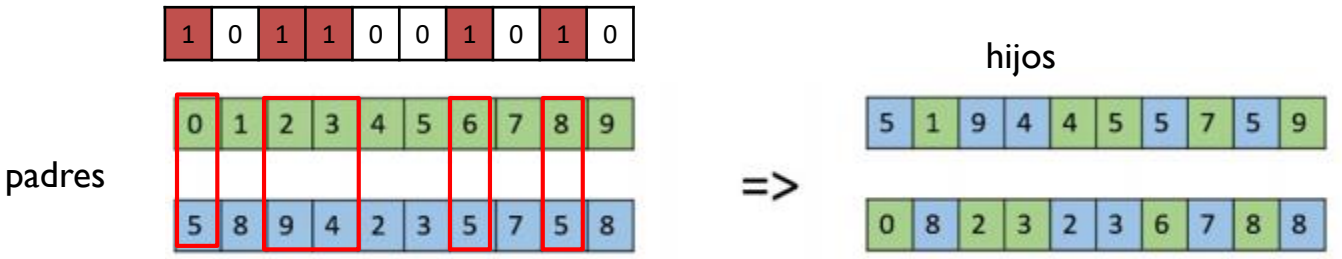
❑ *Uniform*

- Crear una mascara binaria aleatoria



- Los hijos son generados intercambiando las posiciones en los padres donde hay un **1** en la máscara.





2.2) Cx: en permutaciones

- En el caso de permutaciones, usar los operadores de cruce *onepoint, multipoint o uniform*, llevarán a soluciones inadmisibles.

Ejemplo:

- Se han creado operadores especializados que combinan el orden o la información de adyacencia de los padres.

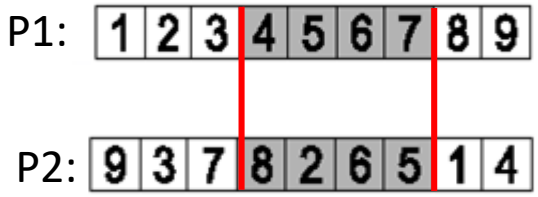
2.2) Cx: en permutaciones

- Cruzamiento en orden

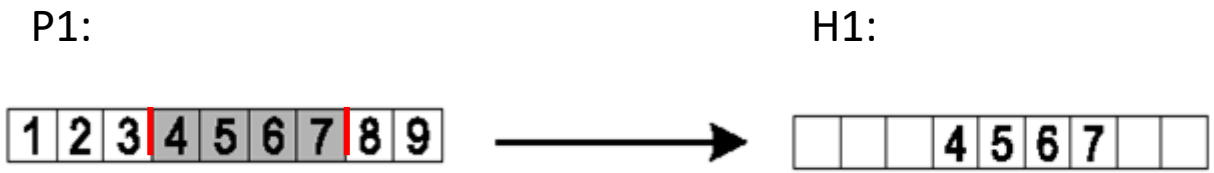
Ejemplo:

Procedimiento para encontrar el primer hijo (H1)

- Elegir aleatoriamente 2 puntos de corte en los padres



- Copiar el segmento generado entre los puntos de corte del primer padre (P1) al primer hijo (H1)



2.2) Cx: en permutaciones

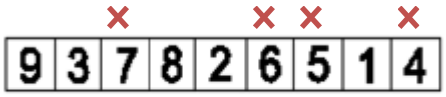
- Cruzamiento en orden

P1:

H1:

- Desconsiderar en el segundo padre los elementos que ya existen en H1

P2:



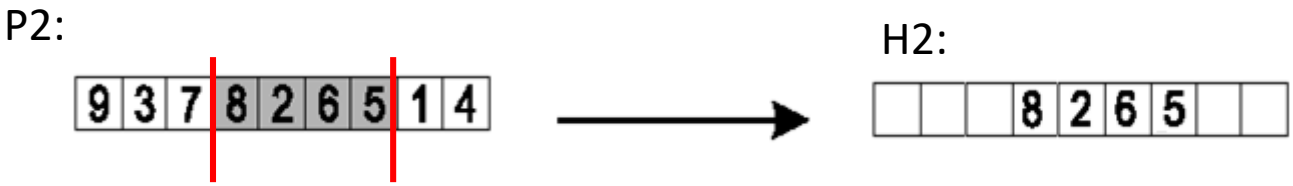
- Completar los valores en H1, copiando de P2 a partir de la derecha del 2do punto de corte, los valores que no fueron desconsiderados. Respetar el orden en que aparecen en el P2.

P2:

H1:

2.2) Cx: en permutaciones

Repetir el procedimiento para encontrar el segundo hijo (H2)



9,1,3,4,7

Cx - Formas de cruzamiento

- Sexuada: Los genes se intercambian entre dos padres (cruzamiento) y están sujetos a cambios (mutación).
  - *Random Mating* (aleatorio)

    Se eligen los individuos aleatoriamente, con la misma probabilidad.
  - *Inbreeding* (entre parientes)

    Se recombinan individuos similares
  - *Line Breeding* (semental)

    Un solo super-individuo (aptitud alta) se recombina con una población base y sus hijos se seleccionan como padres.
  - *Outbreeding (entre desconocidos)*

    Sólo se recombinan individuos muy diferentes.
  - *Self-fertilization (auto-fertilización)*

    Un individuo se recombina con sí mismo.
- Asexuada: Los hijos son duplicaciones de los genes de los padres.
  - *Cloning (clonación)*

    Un individuo se copia sin modificaciones

3. Mutación

- Altera aleatoriamente los alelos de los cromosomas con una probabilidad de mutación P_m. Esta probabilidad debe ser muy baja.
  - P_m = <1/tamaño_población, 1/longitud_cromosoma>
- Crea nuevas características, introduciendo o manteniendo la diversidad genética de la población.
- Genera la posibilidad de alcanzar cualquier punto del espacio de búsqueda (escapar y rodear mínimos/ máximos locales).
- Si su uso es exagerado, reduce la evolución a una búsqueda totalmente aleatoria.
- Depende fuertemente de la representación escogida

3. Mutación

3.1) Representación binaria

- *Bit flip, Bitwise*, Inversión

3.2) Representación punto flotante

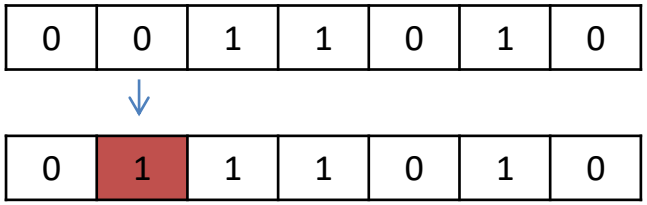
- Mutación no uniforme

3.3) En permutaciones

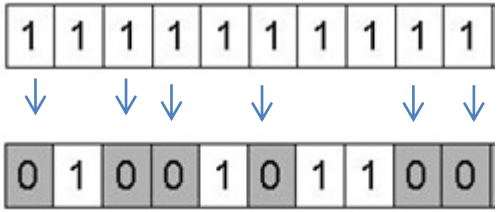
- Inserción, *Swap*, Inversión , Perturbación

3.1) Mutación: en representación binaria

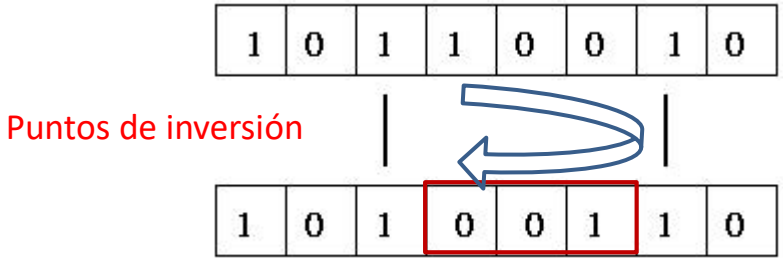
- Bit flip:



- Bitwise:



- Inversión: Se invierte el orden de todos los genes comprendidos entre 2 puntos seleccionados al azar en el cromosoma

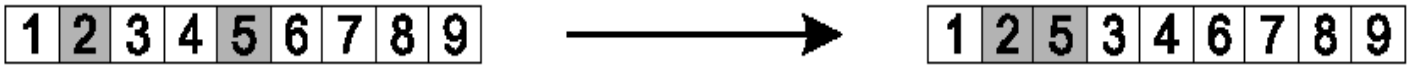


3.3) Mutación: en permutaciones

Mutación como tal, no seria una buena opción. Una buena alternativa seria intercambiar al menos dos valores.

- Inserción (conserva el orden y la información de adyacencia)

- Elegir 2 valores de alelos al azar
- Mover el segundo alelo al lado del primero, luego desplazar el resto de alelos

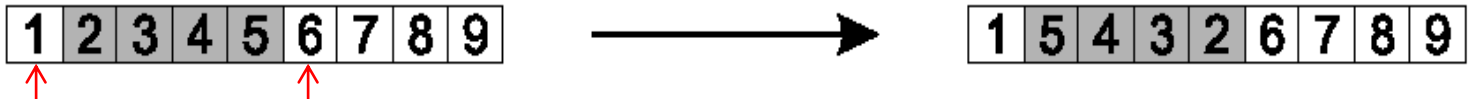


- Swap(conserva la información de adyacencia pero irrumpe el orden)

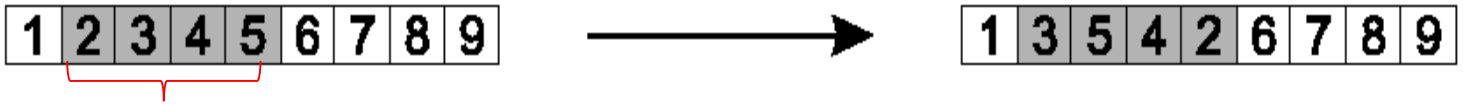
- Elegir 2 alelos al azar e intercambiar sus posiciones

3.3) Mutación: en permutaciones

- Inversión (conserva la información de adyacencia, solo rompe dos enlaces, pero altera el orden)
  - Elegir 2 puntos al azar y luego invertir la subcadena entre ellos.

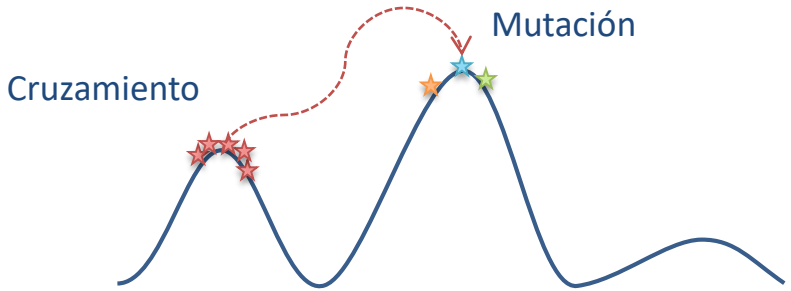


- Perturbación (reorganiza aleatoriamente los alelos en esas posiciones)
  - Elegir una subcadena de genes al azar y luego reorganizarla aleatoriamente.



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.

Selección de sobrevivientes

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

1. Reemplazo total

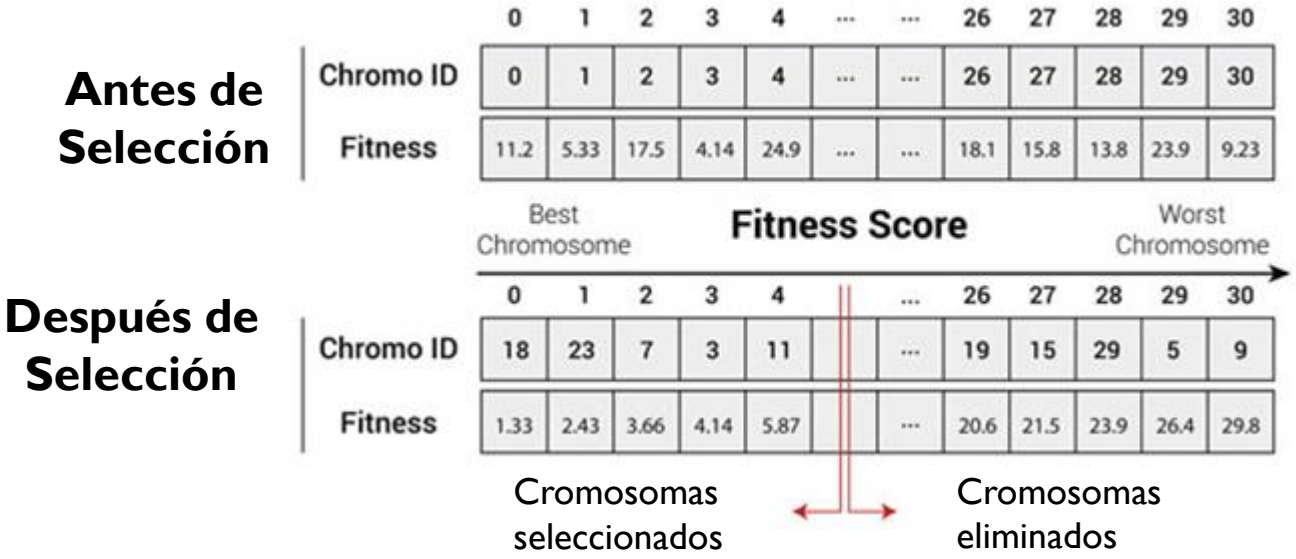
   Todos los hijos reemplazan a los padres. Este tipo de reemplazo es el más simple, ya que la población para la siguiente generación estará compuesta por todos los hijos (descendientes), generados en la generación actual.

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

Selección de sobrevivientes

2. Selección por ranking

Se une 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**

Asegura 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.

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

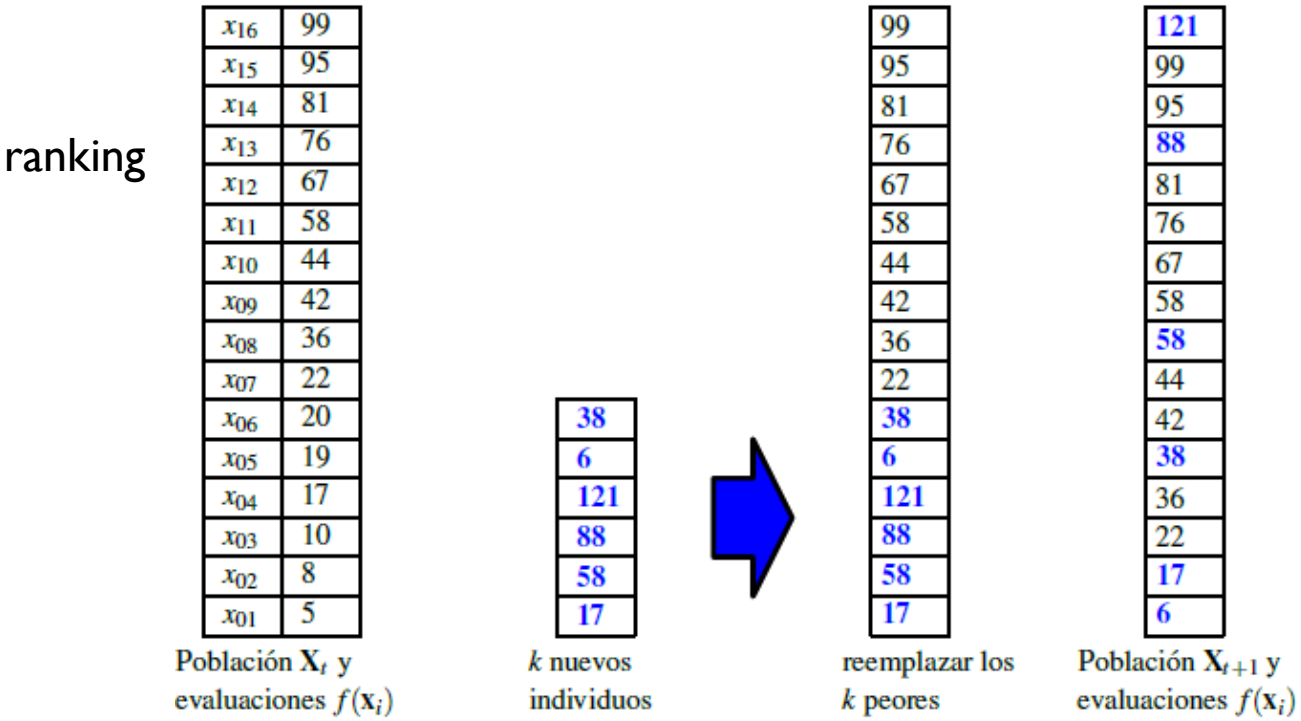
El elitismo garantiza que la mejor aptitud de una población nunca será peor de una generación t a la siguiente $t + 1$.

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)

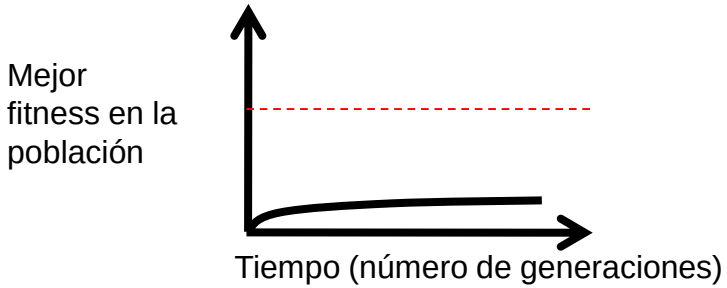
En cada generación se crean unos pocos individuos y ellos reemplazan a un número igual de padres con menor o igual *fitness* para formar la siguiente generación.



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

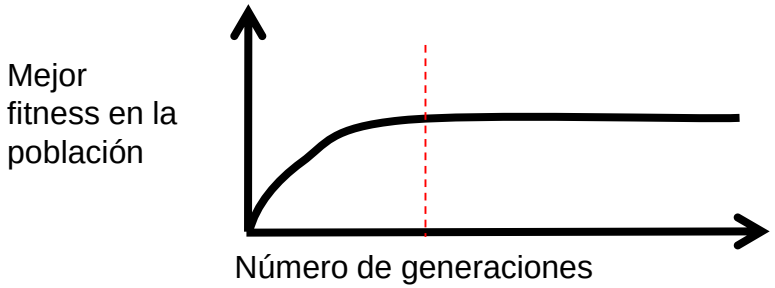
Condición de parada

- 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).

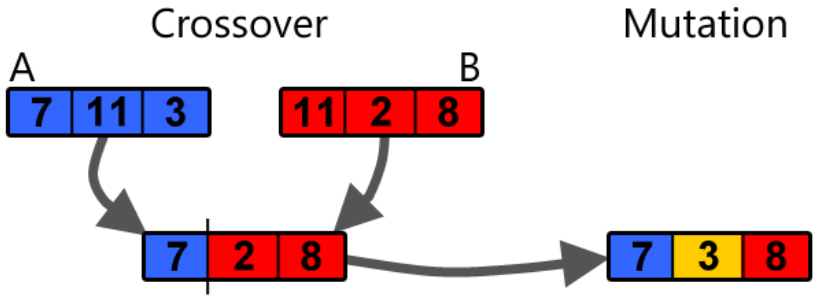


Condición de parada

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



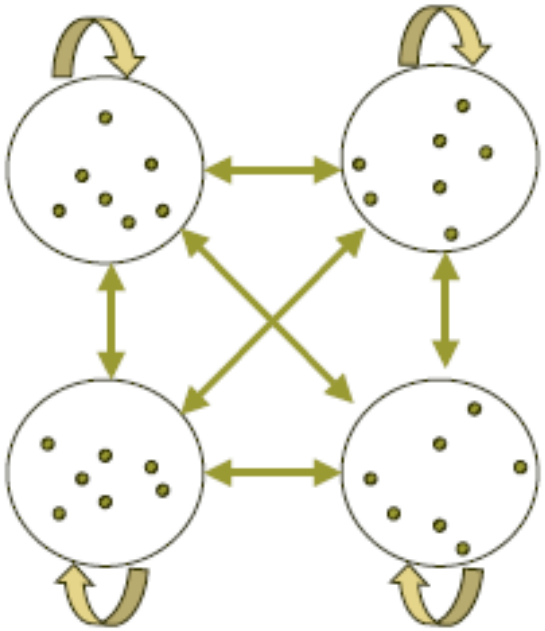
- La diversidad de la población cae por debajo de un umbral dado

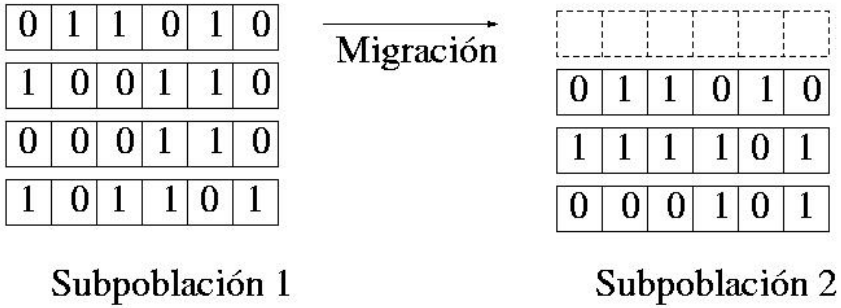


Características avanzadas

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 .





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

Tipos de Paralelización

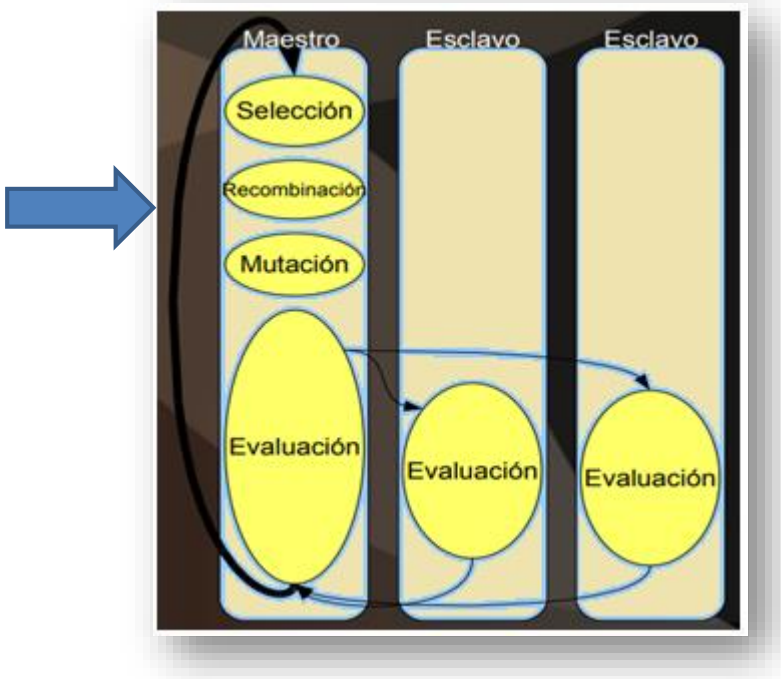
- Principales métodos:

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)



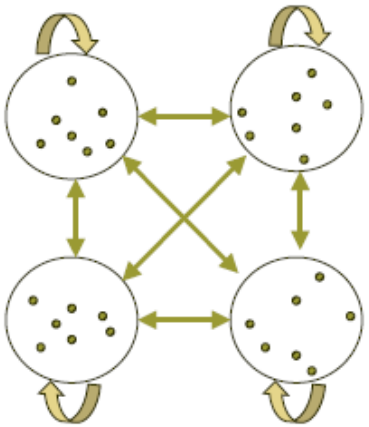
Tipos de Paralelización

3. Grano grueso: si el número de individuos en la población es muy… muy grande, lo mejor es dividir 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



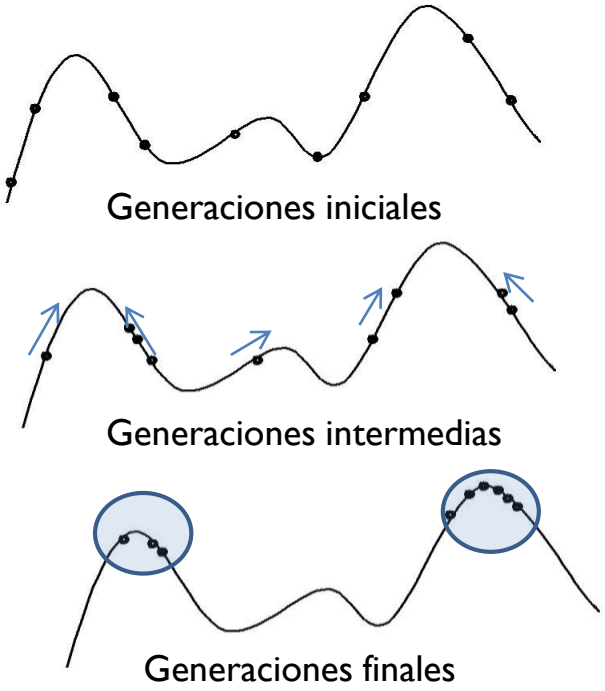
   Nota: Debido a que cada subpoblación evoluciona independientemente, el porcentaje de migración será muy importante para obtener resultados satisfactorios.

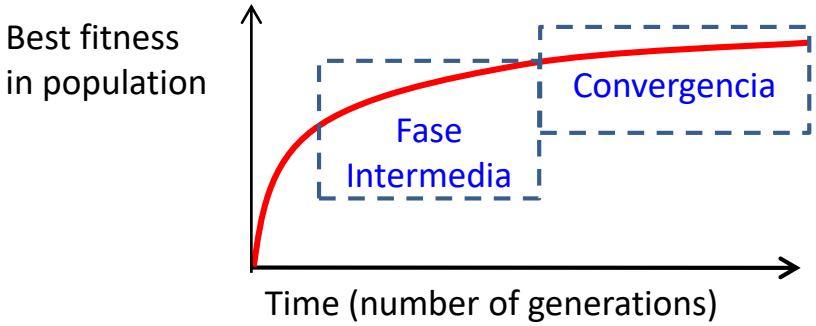
Consideraciones en los AGs

Reporte del AG

Para generar un reporte se deben guardar registros históricos de las fases del AG en cada generación.

- Fase Inicial: Distribución de la población es aleatoria.
- Fase Intermedia: Población empieza a subir a los picos
- Fase de convergencia: Población llega a la cima de los picos





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

Resumen

Representación	Cadenas Binarias, Punto flotante/Real, Entera (Permutaciones)
Selección (Padres)	Ruleta, Torneo, Muestreo Estocástico Universal
Cruzamiento	Pc (normalmente entre 0.6 y 0.99) • Binaria y Entera: [Un punto, Multi-punto, Uniforme] • Punto flotante: [Cruce aritmético: único, simple, completo] • Permutaciones: [Orden, Cruzamiento parcialmente mapeado (PMX)]
Mutación	Pm (normalmente entre 1/tamaño_población y 1/longitud_cromosoma) • Binario: [Bit flip, Bitwise, Inversión] • Punto flotante: [no uniforme: altera aleatoriamente un alelo] • Permutaciones:[Inserción, Swap, Inversión, Perturbación]

Parámetros de un AG

- Un AG requiere la configuración de una serie de parámetros de entrada:
  - tipo de representación,
  - cantidad de alelos (dependiente de la representación: binario, string, entero),
  - longitud de individuos (cromosomas),
  - tamaño de la población,
  - tipo de cruce (dependiente de la representación),
  - tasas de mutación (t_m), puede ser un valor fijo o % 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),

Criticas a los AG

- La principal crítica, es que dependen de un gran número de parámetros y operaciones (número de individuos, tasa de mutación, estrategia de cruzamiento) y no hay reglas que indiquen claramente qué valores elegir para cada problema. Por tanto, hay que ajustar esos parámetros siguiendo recomendaciones muy generales.

- Otra crítica es que, al proponer un gran número de soluciones al problema, cuando se requiere calcular una función de fitness; a menudo esa función es muy costosa en tiempo de computación, lo que hace que utilizar un algoritmo genético requiera mucho tiempo de ejecución.

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 $\mathbf{n^p}$ 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

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

$$\text{Máximo } \textit{fitness} = \frac{N}{2} * (N - 1)$$

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

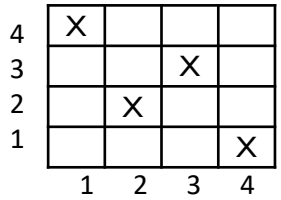
❖ Para implementación con **permutación**:

- 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:*

tablero



Fitness →

matriz

	1	2	3	4
1		1	1	0
2			0	1
3				1
4				

= 4

Fitness del cromosoma: Sumatoria de 1's en la matriz triangular superior

cromosa

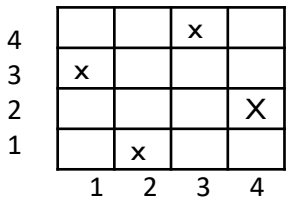
4	2	3	1

1 = reinas no se atacan
0= reinas se atacan

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

Solución

tablero



matriz

	1	2	3	4
1		1	1	1
2			1	1
3				1
4				

cromosa

3	1	4	2

Fitness →

= 6