Técnicas de Inteligencia Artificial

Tema 4

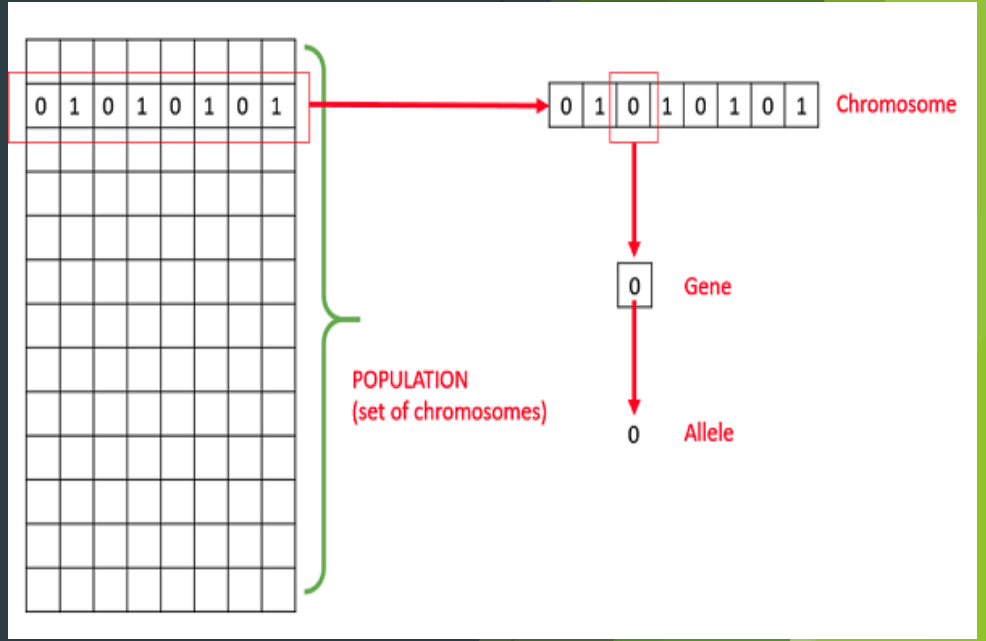
Algoritmos genéticos

Introducción a los algoritmos genéticos

- Los algoritmos genéticos son aproximaciones probabilísticas basadas en heurísticas para resolver problemas complejos.
- La premisa principal de un algoritmo genético es el proceso de selección natural.
- En la naturaleza, los individuos se deben adaptar al entorno mediante un proceso conocido como evolución, según el modelo de Darwin.
  - Aquellas características que hacen al individuo más apto para competir son preservadas cuando se reproduce.
  - Las características que le hacen más débil son eliminadas.
  - Las características son controladas por unidades llamadas genes, que forman conjuntos conocidos como cromosomas.
  - En generaciones sucesivas no solamente los individuos más aptos sobrevives, sino que sus genes son transmitidos a sus descendientes mediante el proceso de recombinación sexual conocido como cruce (crossover).

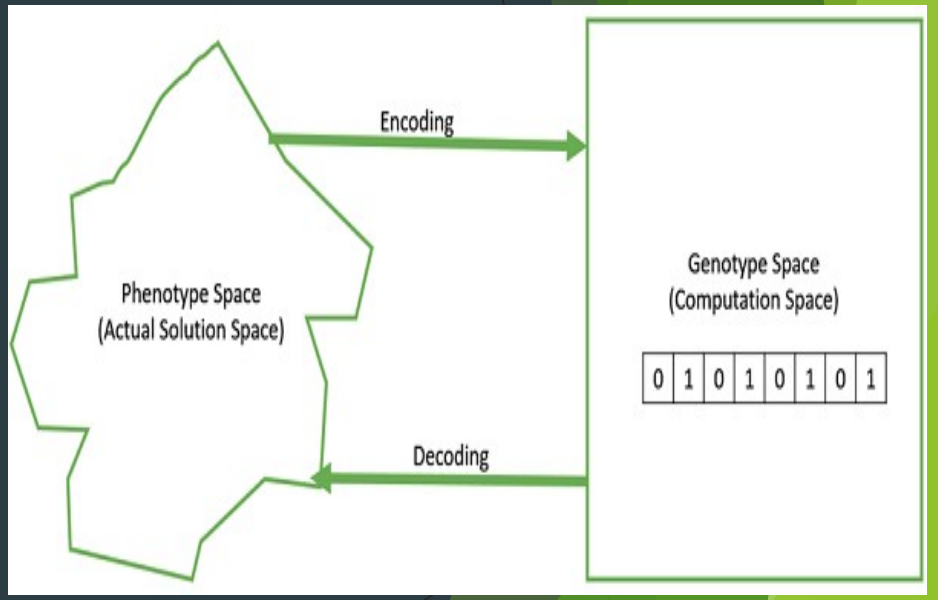
Introducción a los algoritmos genéticos

- Terminología básica:
  - **Población**. Subconjunto codificado de todas las posibles soluciones al problema.
  - **Cromosomas**. Una solución al problema.
  - **Gen**. Un elemento posicional de un cromosoma.
  - **Alelo**. Valor que toma un gen en un cromosoma particular.
  - **Genotipo**. Población en el espacio de computación. Las soluciones son representadas de forma que sean fácilmente entendidas y manipuladas por el ordenador.
  - **Fenotipo**. Población en el espacio de soluciones del mundo real, representación real.



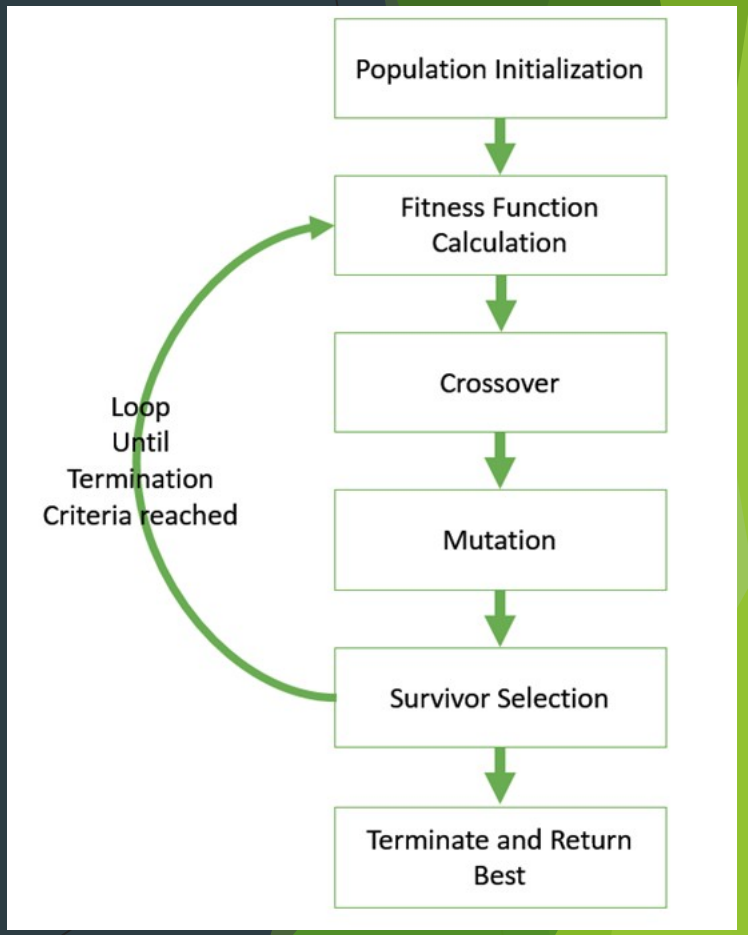
Introducción a los algoritmos genéticos

- Terminología básica:
  - **Decodificar y codificar**. En problemas simples, los espacios del fenotipo y el genotipo coinciden. Sin embargo, en la mayoría de problemas son distintos.
    - Decodificar es un proceso para transformar una solución desde el espacio del genotipo al del fenotipo.
      - La decodificación debe ser rápida ya que se realiza repetidamente en el proceso de evaluación de la función de aptitud.
    - Codificar es el proceso de transformar desde el espacio del fenotipo al genotipo.
  - **Función de aptitud**. Función que toma la solución como entrada y produce una medida de la validez de la solución como salida.
  - **Operadores genéticos**. Alteraciones de la composición genética de la descendencia. Incluye los cruces, mutaciones, selecciones...



Introducción a los algoritmos genéticos

- Los algoritmos genéticos tratan de similar el proceso de evolución en un ordenador.
- Habitualmente se utilizan para encontrar la combinación de elementos que maximiza una función de aptitud.
- Consta de varias etapas principales, a su vez divididas en subetapas:
  - **Inicialización**. Creación de la población inicial.
  - **Selección de padres**. Evaluación de aptitud y selección de los mejores individuos para reproducirse.
  - **Operadores genéticos**. Combinación y mutación para generar nuevos individuos.
  - **Selección de supervivientes**. Selección de los miembros de la siguiente generación.
  - **Terminación**.



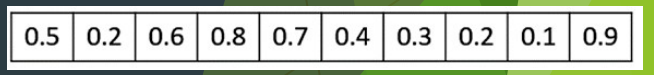
Introducción a los algoritmos genéticos

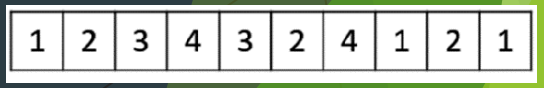
```
function GENETIC-ALGORITHM(population, FITNESS-FN) returns an individual
  inputs: population, a set of individuals
          FITNESS-FN, a function that measures the fitness of an individual

  repeat
    new_population ← empty set
    for i = 1 to SIZE(population) do
      x ← RANDOM-SELECTION(population, FITNESS-FN)
      y ← RANDOM-SELECTION(population, FITNESS-FN)
      child ← REPRODUCE(x, y)
      if (small random probability) then child ← MUTATE(child)
      add child to new_population
    population ← new_population
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to FITNESS-FN

function REPRODUCE(x, y) returns an individual
  inputs: x, y, parent individuals

  n ← LENGTH(x); c ← random number from 1 to n
  return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c + 1, n))
```

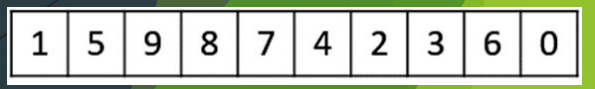
Workflow del algoritmo genético

0) **Representación de la solución**.

- Una de las decisiones más importantes es decidir la representación a utilizar en la solución.
- Malas representaciones pueden llevar a malos resultados en los algoritmos genéticos.
- La representación es muy dependiente del problema a solucionar.
- Tipos de representación:
  - Binaria. Una de las más simples y utilizadas. El genotipo consiste en un array de bits. Utilizado sobre todo en problemas de decisión booleana
  - Valores reales. Problemas en los que los genes se definen con variables continuas.
  - Entera. Problemas con genes con valores discretos.
  - Permutación. Problemas en los que la solución es representada por el orden de los elementos.



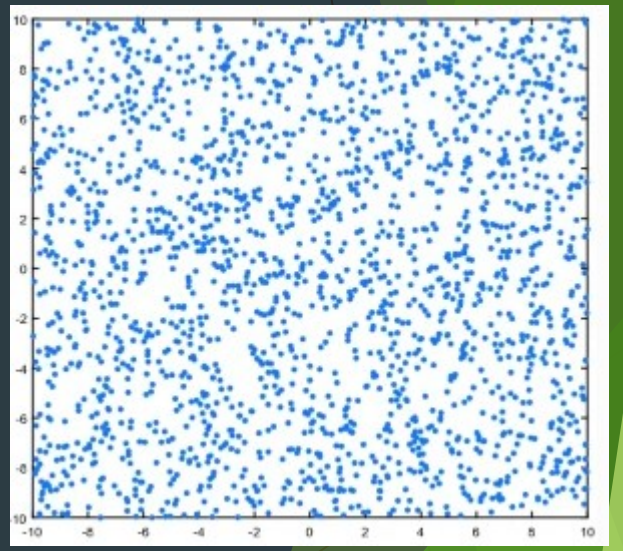




Workflow del algoritmo genético

1) **Inicialización**.

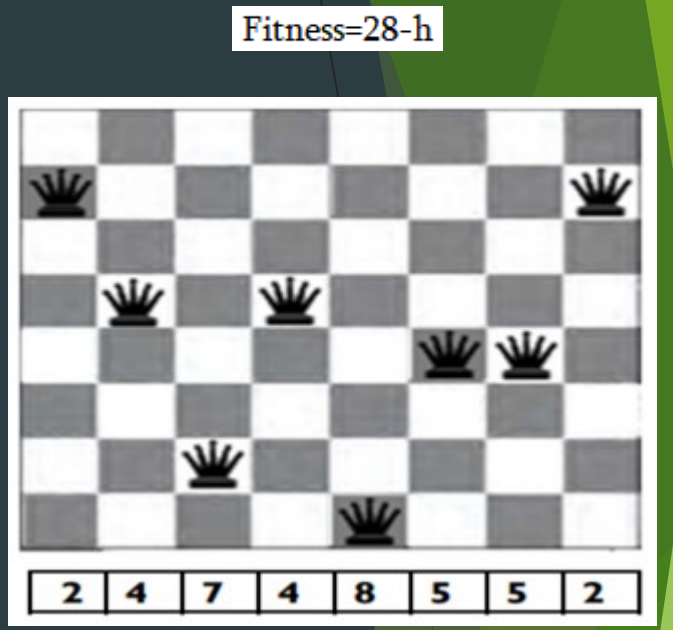
- La población es un subconjunto de soluciones en la generación actual, es un conjunto de cromosomas.
  - La diversidad de la población se debe mantener para evitar convergencias prematuras.
  - La población no debe ser demasiado grande para evitar una disminución de la velocidad, no demasiado pequeña que no pueda encontrar la solución.
  - La población es usualmente definida como un array bidimensional de tamaño de población x tamaño de cromosoma.
- Se genera una población inicial de manera aleatoria o usando conocimiento específico del dominio.
- Se ha observado que la inicialización aleatoria permite alcanzar la optimalidad al provocar mayor diversidad que utilizando heurísticas específicas del dominio.



Workflow del algoritmo genético

2) **Evaluación**.

- La función de aptitud toma una solución candidata al problema como entrada y produce como salida cómo de buena es.
- Evalúa cómo de óptimo es cada individuo de la población.
- El cálculo de esta función se realiza repetidamente, por lo que debe ser lo suficientemente rápida.
- En algunos casos, el cálculo directo de la función no es posible debido a la complejidad del problema, por lo que se debe calcular alguna función aproximada.
- En ocasiones requiere realizar algún proceso, como la simulación de la tarea a realizar utilizando los pesos del individuo, y medir su función de aptitud según el desempeño del agente.

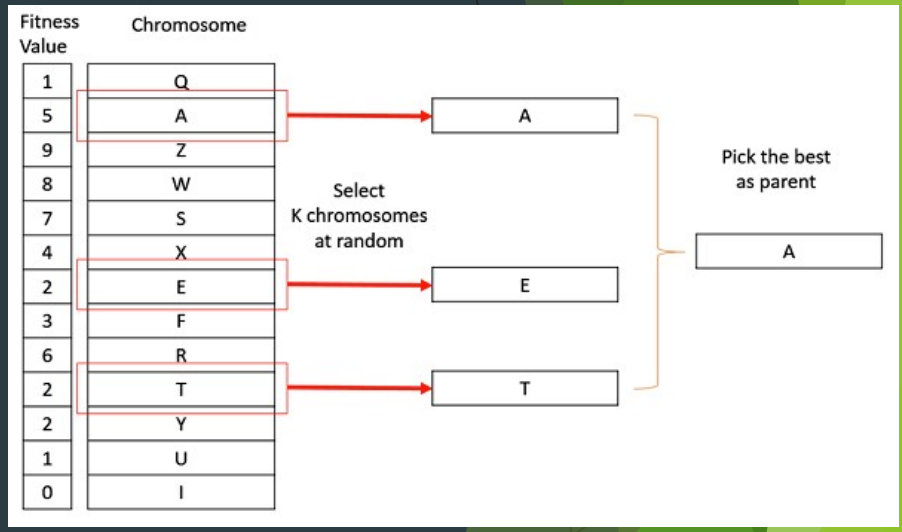


$v_1 = (24748552)$	$eval(v_1) = 24$
$v_2 = (32752411)$	$eval(v_2) = 23$
$v_3 = (24415124)$	$eval(v_3) = 20$
$v_4 = (32543213)$	$eval(v_4) = 11$

Workflow del algoritmo genético

3) **Selección de padres**.

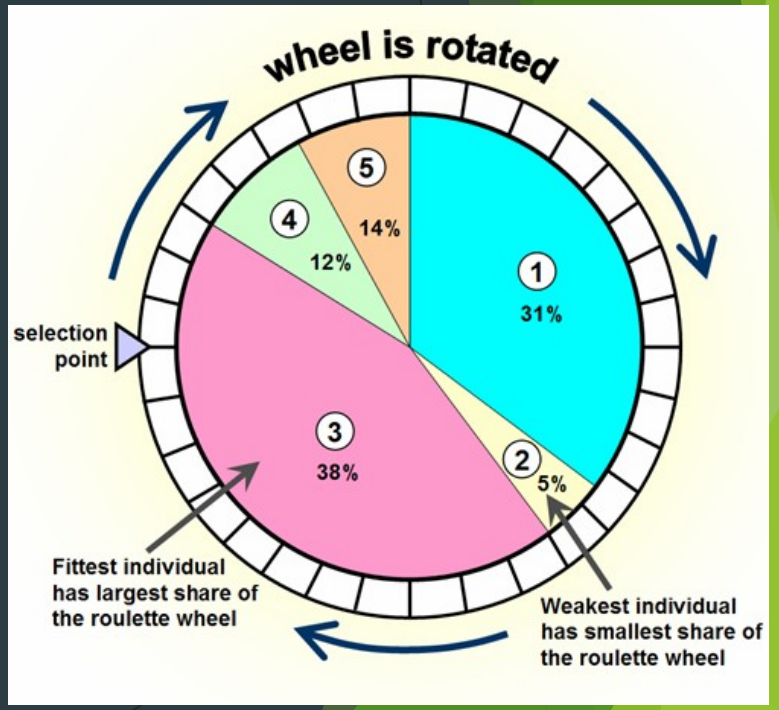
- Selecciona los mejores individuos de la población para servir como padres para la siguiente generación.
- Se utilizan mecanismos de selección.
- **Selección determinista**.
  - Se ordenan los individuos según su función de aptitud.
  - Seleccionar los k mejores para sobrevivir.
  - Reemplazar al resto con la descendencia.
  - Puede conducir a una convergencia rápida a óptimos locales.



Workflow del algoritmo genético

3) **Selección de padres**.

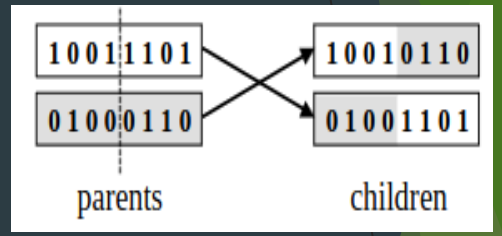
- **Selección estocástica o de ruleta**
  - En vez de seleccionar los k mejores, seleccionar cada individuo con una probabilidad directamente proporcional a su función de aptitud relativa a la población.
  - Menos propensa a óptimos locales, pero se pueden perder buenas soluciones.
- **Selección de torneo**
  1) Para cada individuo i, crear un subconjunto q de la población seleccionada aleatoriamente.
  2) Asignar un punto a i por cada individuo en q que tenga menor puntuación que él.
  3) Reconstruir la población en base a los puntos acumulados en el torneo.
  - A medida que aumenta el tamaño de q, se parece más a una selección determinista.

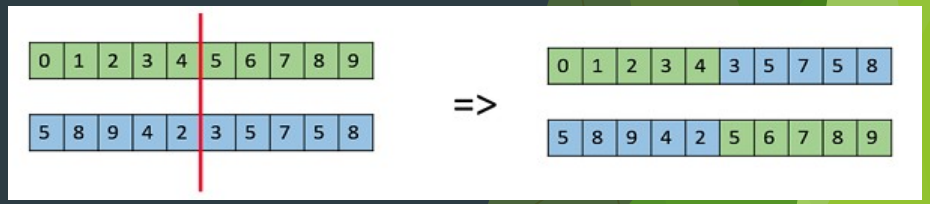


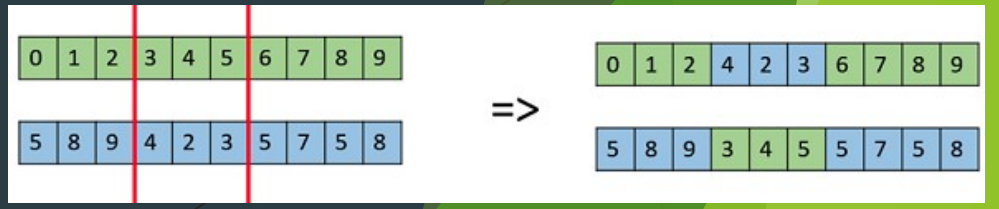
Workflow del algoritmo genético

4) **Cruce (crossover)**.

- Realizar el cruce genético entre pares de individuos seleccionados para crear descendencia.
- Esta operación combina los parámetros de dos padres para generar un nuevo individuo.
- La forma más básica de realizar el cruce consiste en dividir en partes el estado de cada padre y que el hijo herede una parte de cada uno de los padres.
- Diferentes técnicas para elegir el punto de cruce:
  - **Punto único**. Elegir el centro o un punto óptimo de la descripción, tomar la primera parte de un padre y la otra del segundo.
  - **Multipuntos**. Elegir n diferentes puntos de corte.
  - **Aleatorio**. Elegir el punto de corte de forma aleatoria o proporcional a la función de aptitud de los padres.





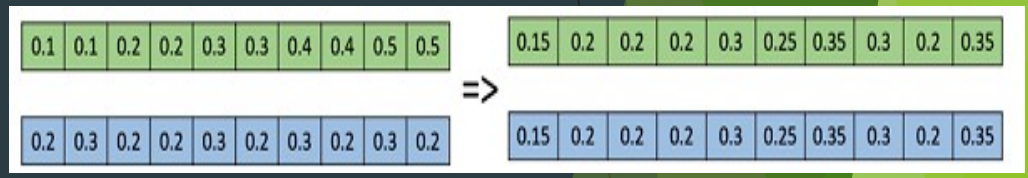


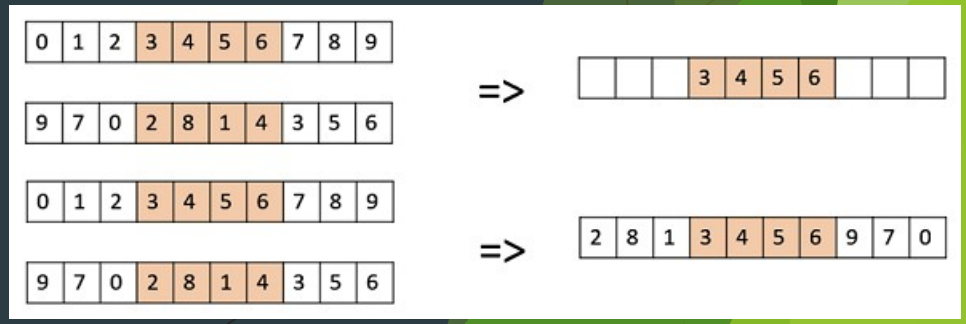
Workflow del algoritmo genético

4) **Cruce (crossover)**.

- Diferentes técnicas para elegir el punto de cruce:
  - **Uniforme**. Elegir cada elemento de forma independiente, de manera aleatoria o proporcional a la función de aptitud.
  - **Recombinación aritmética**. Usado usualmente en representaciones de enteros o números reales, funciona tomando la suma ponderada de los dos padres.

    $Hijo_1 = a*x + (1-a)*y$. $Hijo_2 = a*y + (1-a)*x$.
  - **Cruce de orden de Davis (OX1)**. Utilizado en cruces con permutación con intención de transmitir información sobre el orden relativo a la descendencia.
    - Crear dos puntos de cruce aleatorios en los padres y copiar el segmento entre ellos del primer padre en el primer hijo.
    - Empezando por el segundo punto de corte del segundo padre, copiar los números restantes no utilizados al primer hijo, envolviendo la lista.
    - Repetir en el segundo hijo con los roles de los padres invertidos.

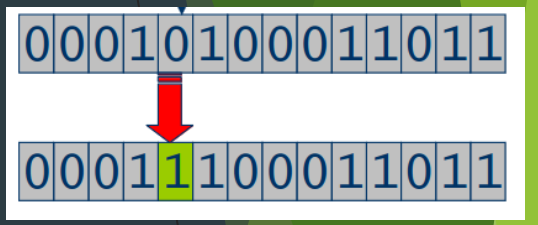




Workflow del algoritmo genético

5) **Mutación**.

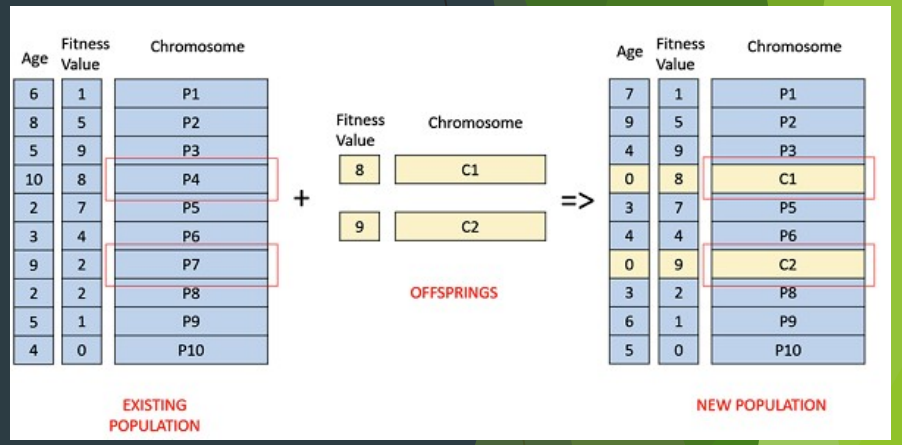
- Introduce cambios aleatorios en los parámetros de la descendencia para explorar nuevas regiones del espacio de soluciones.
- La mutación previene una convergencia prematura subóptima y permite una mayor diversidad en la población.
- Variantes:
  - **Mutación por inversión de bits**. Seleccionar uno o más bits aleatorios y cambiar su valor.
  - **Reseteo aleatorio**. Extensión del anterior para enteros o reales. Sustituir un gen por un valor aleatorio permitido.
  - **Mutación de cambio**. Seleccionar dos posiciones aleatoria e intercambiarlas.
  - **Mutación revuelta**. Usado con permutaciones. Se selecciona un subconjunto de genes y sus valores de barajan aleatoriamente.
  - **Mutación de inversión**. Usado con permutaciones. Se selecciona un subconjunto de genes y se invierte la cadena completa.

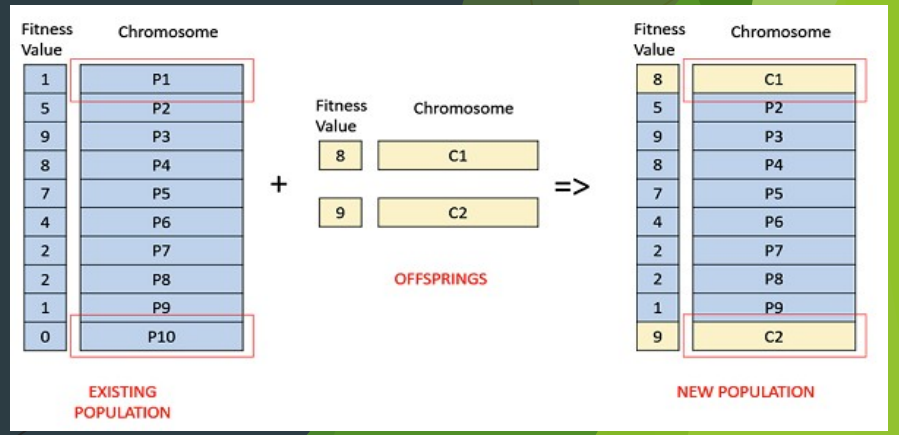


Workflow del algoritmo genético

6) **Reemplazo**.

- Se sustituye una parte de la población existente con la descendencia, manteniendo constante el tamaño de la población.
- La política de selección de supervivientes determina qué individuos se mantienen en la siguiente generación.
- Es crucial asegurar que los individuos con mejor puntuación se mantienen mientras se mantiene la diversidad de la población para explorar nuevas soluciones.
- Criterios de selección:
  - **Selección por edad**. Cada individuo se permite que esté en la población por una cantidad finita de generaciones para reproducirse. Los miembros más antiguos de la población son descartados.
  - **Selección por aptitud**. Los hijos reemplazan a los individuos menos aptos. Se utilizan variaciones de las mismas técnicas que en la selección de padres.

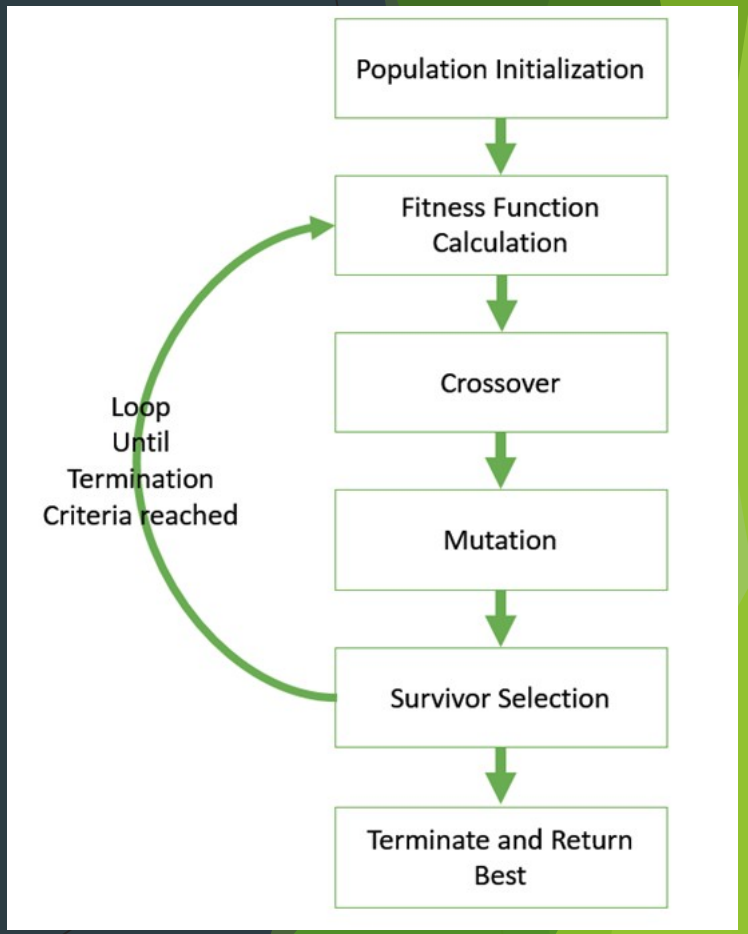




Workflow del algoritmo genético

7) **Repetición**.

- Se repiten iterativamente los pasos del 2 al 6 para un número específico de generaciones o hasta que se alcance un criterio de terminación.
- La condición de terminación es importante para gestionar el final del algoritmo con una buena solución.
- Usualmente, se suelen utilizar estos criterios:
  - Cuando no ha habido mejora en la población en X iteraciones.
  - Cuando se ha alcanzado un número absoluto de iteraciones.
  - Cuando la función objetivo ha alcanzado un valor predefinido.
- Como ocurre con otros parámetros, el criterio de terminación es específico para el problema a resolver.



Algoritmos genéticos

- Los algoritmos genéticos tienen una serie de ventajas:
  - **Optimización automática**. Automatizan el proceso de encontrar unos parámetros óptimos mediante el refinamiento automático de la población. Elimina la necesidad de realizar fine-tuning y de tener conocimiento experto sobre el dominio.
  - **Exploración del espacio de soluciones**. Los operadores de cruce y mutación permite explorar una gran cantidad de parámetros. Esta estrategia de exploración aumenta las posibilidades de encontrar nuevas soluciones efectivas.
  - **Búsqueda paralela**. Permiten evaluar varios conjuntos de parámetros de forma simultánea, lo que permite un gran proceso de optimización y una búsqueda mayor sobre el espacio de estados de manera eficiente.
  - **Adaptabilidad**. Permiten adaptarse a cambios en el entorno de juego o la estrategia. A medida que el juego evoluciona, el algoritmo puede optimizar continuamente la función de evaluación para mantener una alta precisión.

Algoritmos genéticos

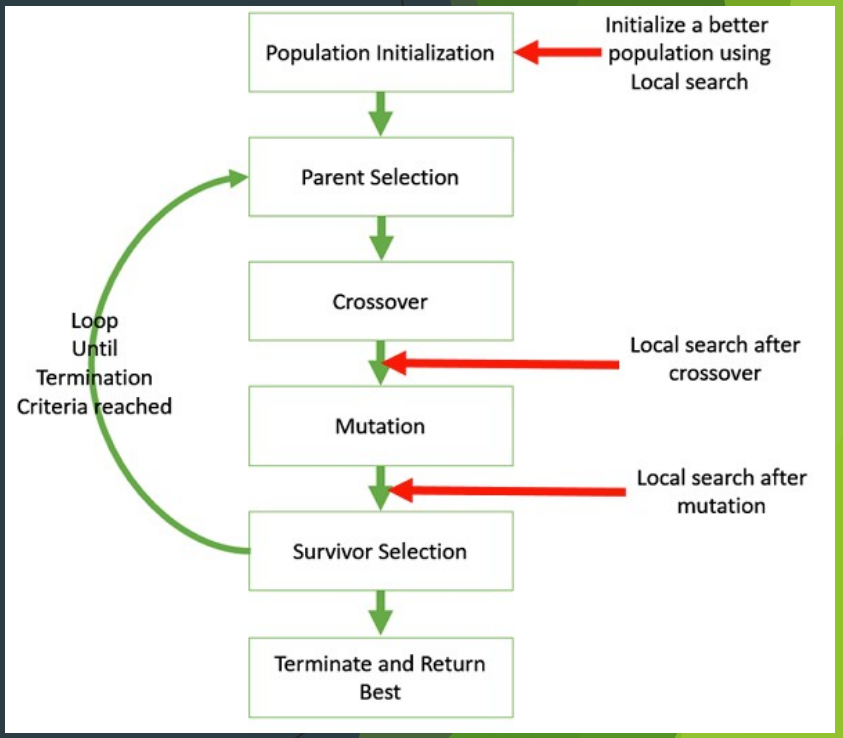
- Los algoritmos genéticos también tienen una serie de retos y consideraciones:
  - **Complejidad computacional**. La optimización de los parámetros puede ser muy intensiva computacionalmente, especialmente en juegos con un gran espacio de estados. Se requiere una gran labor de optimización para mitigar este problema.
  - **Representación y codificación**. Seleccionar un representación adecuada de cromosomas y un esquema de codificación para los parámetros es crucial. La elección tiene influencia sobre el espacio de estados y la habilidad del algoritmo para descubrir soluciones significativas.
  - **Función de optimalidad**. Evaluar la optimalidad de un conjunto de parámetros requiere, en muchas ocasiones, una simulación del juego durante varias veces. Este proceso es muy costoso, por lo que se requieren técnicas eficientes para la simulación.
  - **Hiperparámetros**. Requieren el ajuste adecuado de varios parámetros, como el tamaño de población, el ratio de mutación y los mecanismos de selección.

Implementación efectiva

- Los algoritmos genéticos son muy generales, por lo se requiere aplicar optimaciones para mejorar los resultados:
  - **Introducir conocimiento específico sobre el dominio**.
    - A medida que se incluye mayor conocimiento específico sobre el dominio del problema, se obtienen valores más objetivos. Este conocimiento se pone de manifiesto en las representaciones de los cromosomas, las operaciones específicas de cruce o mutación…
  - **Reducir el hacinamiento**.
    - El hacinamiento sucede cuando un cromosoma con una gran puntuación se reproduce demasiado, provocando que en pocas generaciones toda la población esté llena de soluciones similares con puntuaciones parecidas, reduciendo la diversidad. Se deben aplicar técnicas para limitarlo:
      - Mutación para introducir diversidad.
      - Cambiar a selección por ranking o selección por torneo que tienen más presión de selección que la selección proporcional para individuos con puntuación similar.
      - Aptitud compartida. La aptitud de un individuo es reducida si la población ya tiene individuos similares.

Implementación efectiva

- Los algoritmos genéticos son muy generales, por lo se requiere aplicar optimaciones para mejorar los resultados:
  - **Introducir aleatoriedad.**
    - Se ha observado experimentalmente que las mejores soluciones se obtienen con cromosomas aleatorios al añadir diversidad en la población.
  - **Combinar con búsquedas locales.**
    - La búsqueda local consiste en comprobar las soluciones en la vecindad de una solución dada para buscar mejores valores objetivos.
    - Existen varias etapas del workflow donde introducir las búsquedas locales y mejorar los resultados.
  - **Variación de parámetros y técnicas.**
    - No existe una fórmula mágica genérica para todos los problemas. Se debe dedicar tiempo para ajustar parámetros como el tamaño de la población, probabilidad de mutación y cruce...



Algoritmos genéticos en CSP

- Los problemas de satisfacción de restricciones (CSP) son aquellos donde se debe maximizar o minimizar una función sujeta a ciertas restricciones.
  - No todos las resultados del espacio de soluciones son correctos.
- En estas circunstancias, los operadores de mutación y cruce pueden dar resultados incorrectos, que deben ser corregidos usando alguna de las siguientes técnicas:
  - **Utilizar funciones de penalización**.
    - Reducir el valor de la función de manera proporcional al número de restricciones violadas.
  - **Usar funciones de reparación**.
    - Modificar la solución para que las restricciones sean satisfechas.
  - **No permitir soluciones incorrectas**.
  - **Usar una representación o función de decodificación especial** que asegure la corrección de las soluciones.

Algoritmos genéticos en búsqueda de juegos

- Como hemos visto en otros temas, los algoritmos de búsqueda requieren de una función de evaluación estática para evaluar posiciones no terminales.
- La mayoría del conocimiento sobre el juego del programa reside en su función de evaluación, así como gran parte de su fuerza de juego.
- Aun con la aparición de nuevas técnicas que aprenden las mejores características de manera automática a partir de la posición, muchos de los módulos de juegos con mayor fuerza todavía utilizan funciones de evaluación que incluyen características diseñadas a mano.
- No obstante, resulta difícil ponderar estas características de manera manual, por lo que el aprendizaje automático utilizando algoritmos genéticos puede constituir una solución para obtener los mejores parámetros para evaluar las posiciones del juego.

Algoritmos genéticos en búsqueda de juegos

- Cada parámetro de la función de evaluación se puede codificar como un array de valores, haciendo que cada cromosoma represente una función de evaluación.
- Inicialmente, se iniciaría aleatoriamente la población de cromosomas, y sería necesaria una función de aptitud para evaluar lo bueno que es el conjunto de parámetros.
- Uno de los principales problemas es encontrar esa función de aptitud que nos permita comparar entre cromosomas.
  - Una posible solución sería permitir que los cromosomas de cada generación compitieran entre ellos en una serie de partidas y almacenar la puntuación de cada individuo como su función de aptitud.
  - El mayor problema de la simulación de partidas es el tiempo necesario en cada generación. Para ello, lo habitual es poner restricciones de duración a las partidas, de forma que puedan terminar en un tiempo finito.
  - Otro posible enfoque consiste en la elaboración de una batería de posiciones de partidas de maestros (mentorización), en las que las funciones de evaluación deben calcular la respuesta que realizó el jugador experto, y contar los aciertos como la función de aptitud.

Algoritmos genéticos en búsqueda de juegos

- El aprendizaje mediante mentorización consiste en replicar la forma de evaluar las posiciones de un experto comparando su rendimiento con el de los cromosomas:

  1) Usar una base de datos de partidas de expertos, seleccionar una lista de posiciones y los movimientos realizados en esa posición.

  2) Para cada posición, permitir que el organismo realice una búsqueda a profundidad 1 y almacenar el movimiento seleccionado.

  3) Comparar el movimiento sugerido por el organismo con el movimiento real realizado por el jugador. La función de aptitud será la cantidad de movimientos acertados.

Algoritmos genéticos en búsqueda de juegos

- El aprendizaje mediante mentorización consiste en replicar la forma de evaluar las posiciones de un experto comparando su rendimiento con el de los cromosomas:
  - Las búsquedas a profundidad 1 y la evaluación de la posición son operaciones muy rápidas que permiten evaluar miles de cromosomas por segundo.
  - Este proceso completo se repite n veces y cada vez se obtiene el mejor organismo de cada tanda completa de simulaciones, obteniendo n cromosomas distintos que representan las mejores funciones de evaluación estáticas de la posición obtenidas.
  - El último paso consiste en hacer competir estos organismos entre sí y hacerlos evolucionar con el algoritmo genético estudiado, obteniendo una buena función de evaluación final tras completarse los criterios de terminación.

Algoritmos genéticos en búsqueda de juegos

- De la misma forma que se puede aprender la función de evaluación, también se pueden optimizar distintos parámetros de la búsqueda, como las estrategias de poda avanzadas y heurísticas especiales.
  - Fijar la misma función de evaluación estática para todos los cromosomas.
  - Codificar las distintas opciones en forma de bits para representarse en un cromosoma.
  - Aplicar técnicas similares a las vistas para la optimización de la función de evaluación, aunque en este caso contando el número de nodos visitados antes de encontrar la solución.
  - La función de aptitud priorizará aquellas configuraciones de búsqueda que encuentren la solución más rápidamente.

Parameter	Value range	Bits
Null-move use	0–1	1
Null-move reduction	0–7	3
Null-move use adaptivity	0–1	1
Null-move adaptivity depth	0–7	3
Futility depth	0–3	2
Futility threshold depth-1	0–1023	10
Futility threshold depth-2	0–1023	10
Futility threshold depth-3	0–1023	10
Multi-cut use	0–1	1
Multi-cut reduction	0–7	3
Multi-cut depth	0–7	3
Multi-cut move num	0–31	5
Multi-cut cut num	0–7	3
Check extension	0–4	3
One-reply extension	0–4	3
Recapture extension	0–4	3
Passed pawn extension	0–4	3
Mate threat extension	0–4	3
Total chromosome length		70