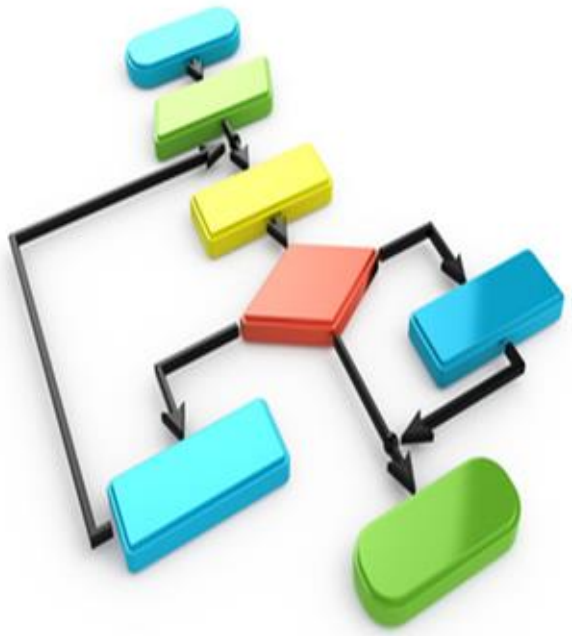
UNIVERSIDAD NACIONAL GUILLERMO BROWN

Licenciatura en
CIENCIA DE DATOS
Tecnicatura en
PROGRAMACIÓN



ALGORITMOS PARA RESOLVER PROBLEMAS P.1

Acomodar mercadería en un almacén, cortar piezas en un material, maximizar la ganancia de una empresa, reducir el tiempo de viaje, etc, etc son problemas que nos podremos encontrar intentando resolver mediante un programa.

Si bien ya existen muchas aplicaciones que nos pueden solucionar estas tareas sin tener que pensar en maneras de resolverlos, debemos conocer cómo funcionan para esas pocas ocasiones en las que haya que hacer todo desde cero.

Vamos a ver algunos algoritmos comunes que se aplican mucho en la vida real.

Y nos centraremos en dos de ellos: genéticos y de enjambre

ALGORITMOS DE FUERZA BRUTA

Son aquellos donde se revisan todas las posible combinaciones de valores posibles hasta encontrar la o las soluciones correctas.

En algunos casos se puede acelerar asumiendo condiciones iniciales del problema (algunos valores son mayores que, o similares a, o posiblemente se encuentran dentro de alguna lista o quizás responden a alguna regla)

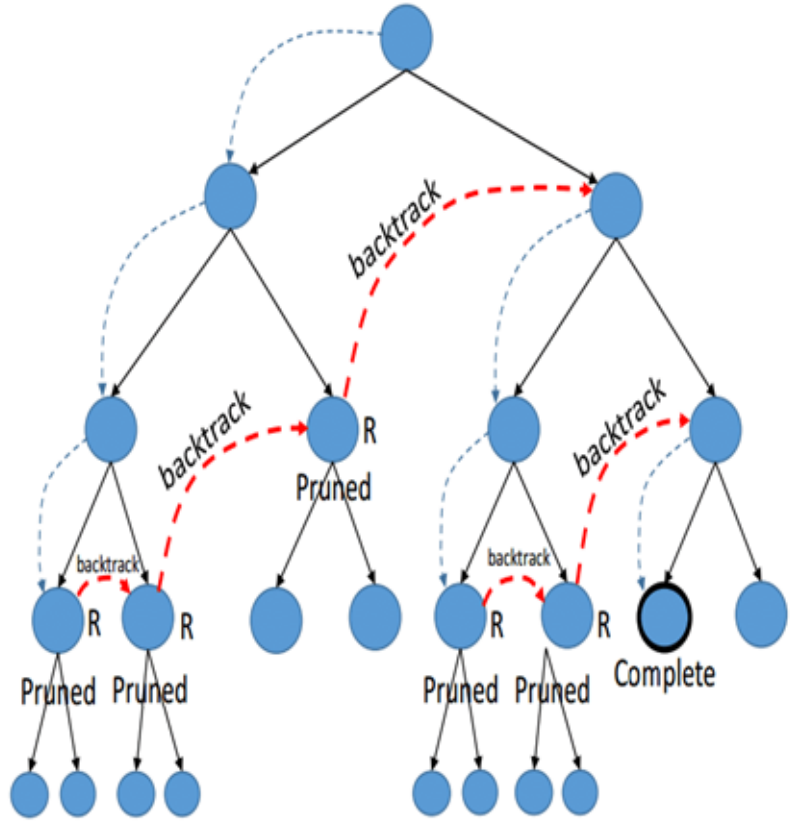
Pero generalmente son muy costosos en tiempo y recursos y en muchas ocasiones no alcanza la duración del universo para probar todas las posibles combinaciones.

ALGORITMOS DE RETROCESO O PODA (BACKTRACKING)

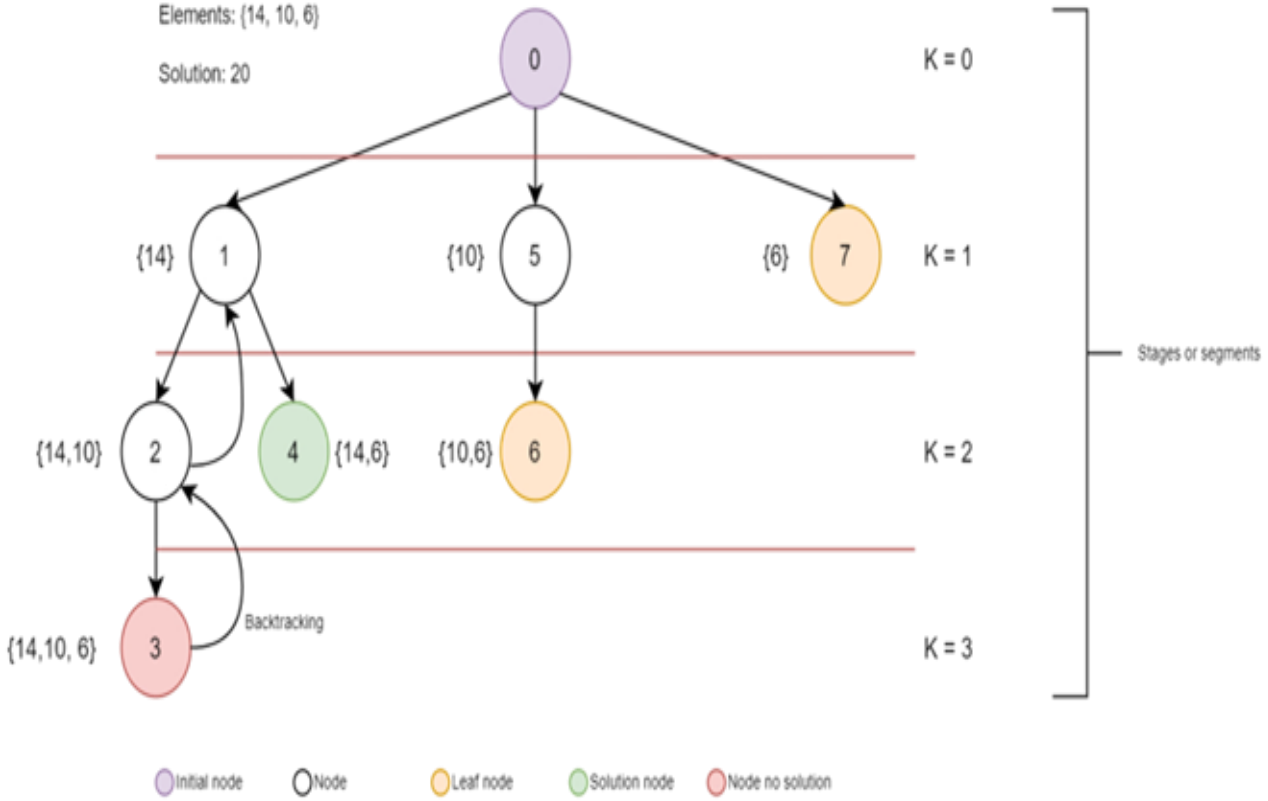
Son una mejora a los de fuerza bruta pura. Y sirven para situaciones en las que debemos ir tomando decisiones dicotómicas (o una cosa o las otras) El de recorrer varios caminos es un ejemplo.

Se toma una decisión y se avanza hacia la siguiente toma de decisiones. Cuando se llega a un punto en el que ya no se puede avanzar, se retrocede hasta el punto donde se tomo la decision previa.

Si existe un límite al número de decisiones, la rama que supera esa cantidad se elimina directamente (poda)



ALGORITMOS DE RETROCESO (BACKTRACKING)



Este es un ejemplo donde se busca encontrar dentro de una lista de valores (14, 10, 6) si la suma de algunos de ellos nos da 20.

En los nodos del primer nivel dejamos los valores que tenemos y en los siguientes niveles volvemos a poner los otros elementos de la lista (exceptuando los que ya sumamos)

Existe una variante (llamada de poda) En la que si una suma ya nos da más de 20, no seguimos ampliando esa rama.

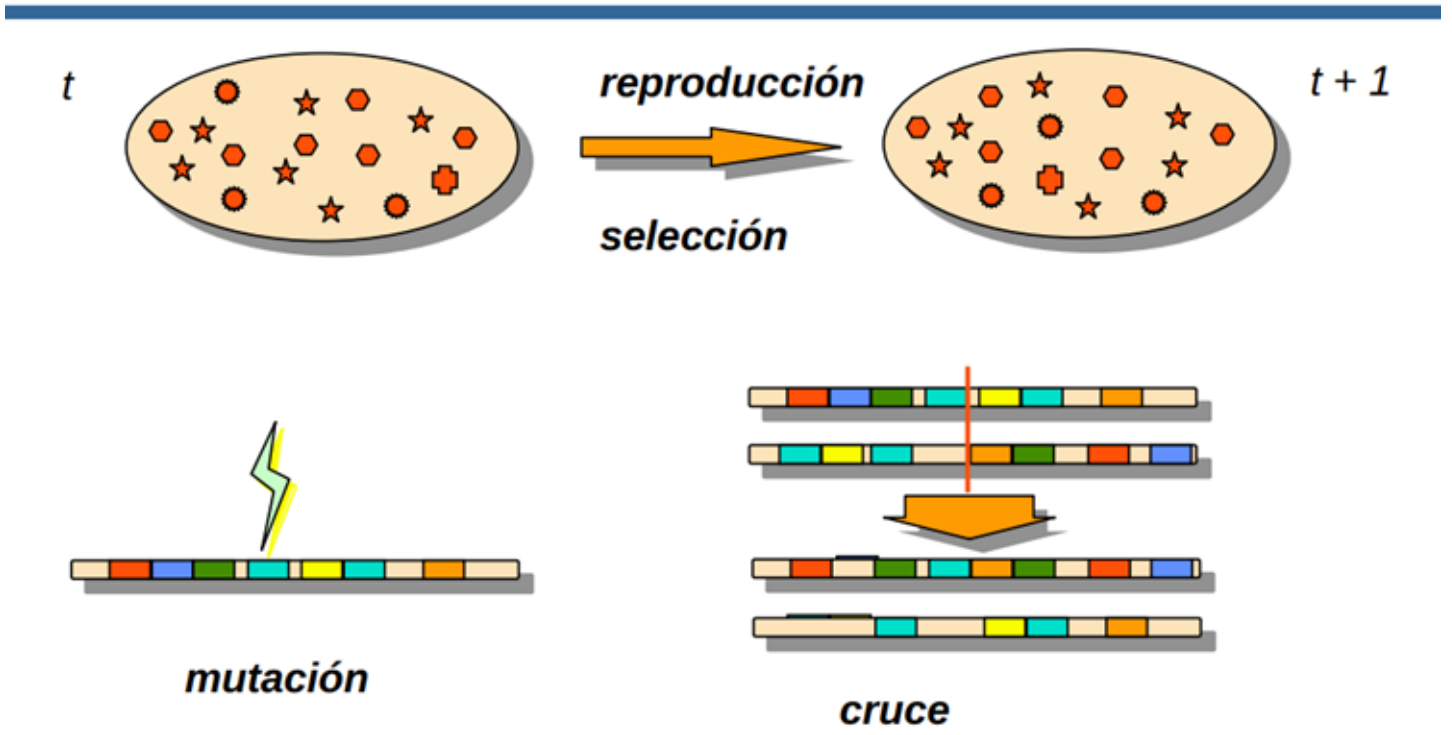
ALGORITMOS GENÉTICOS

Estos algoritmos basados en la naturaleza, hacen evolucionar una población de individuos sometiéndola a acciones aleatorias, semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una selección.

De acuerdo con algún criterio, se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles son los menos aptos, que son descartados.

El proceso se repite hasta encontrar un individuo que representa a la solución buscada.

ALGORITMOS GENÉTICOS
Los Ingredientes



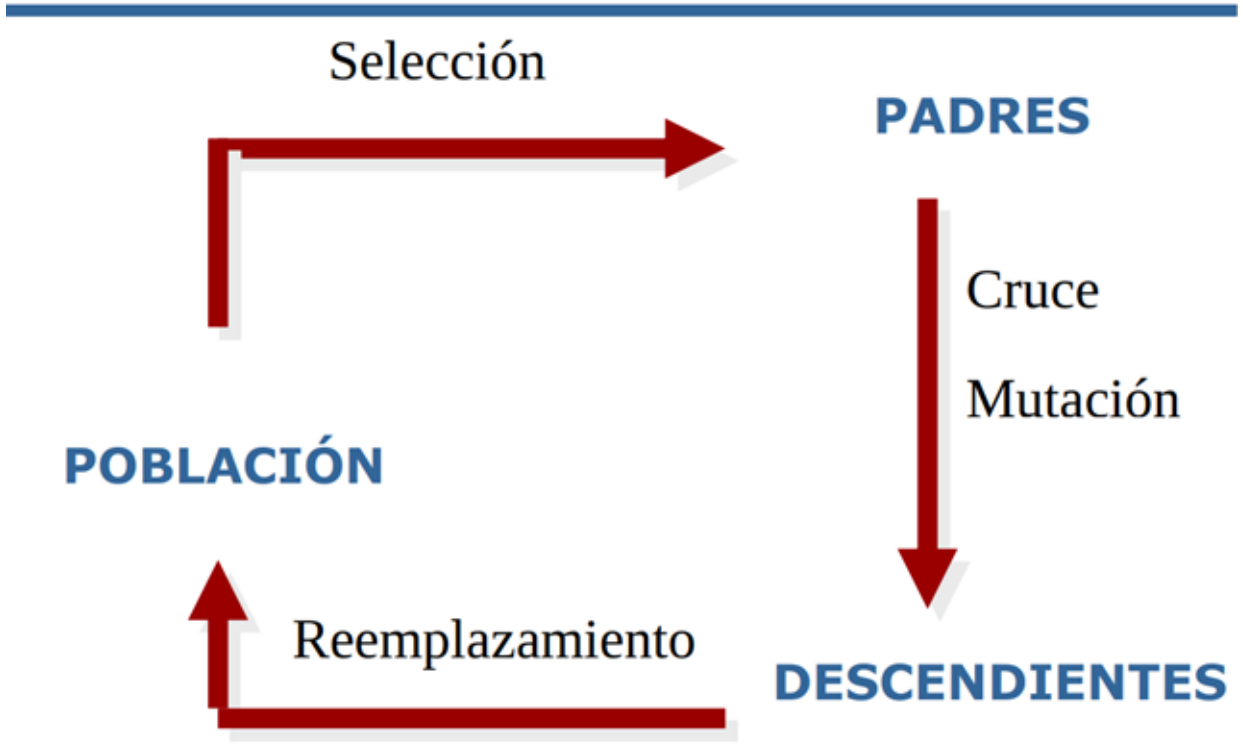
Cada nueva generación está mejor adaptada al problema.

Los mas adaptados (con menos errores) sobreviven.

Sufren mutaciones y se cruzan entre ellos.

Para saber cuales son los mejor adaptados se emplea una función de **Fitness** (ajuste) Cuando mayor el valor de fitness, mejor la solución.

ALGORITMOS GENÉTICOS
El ciclo de la Evolución



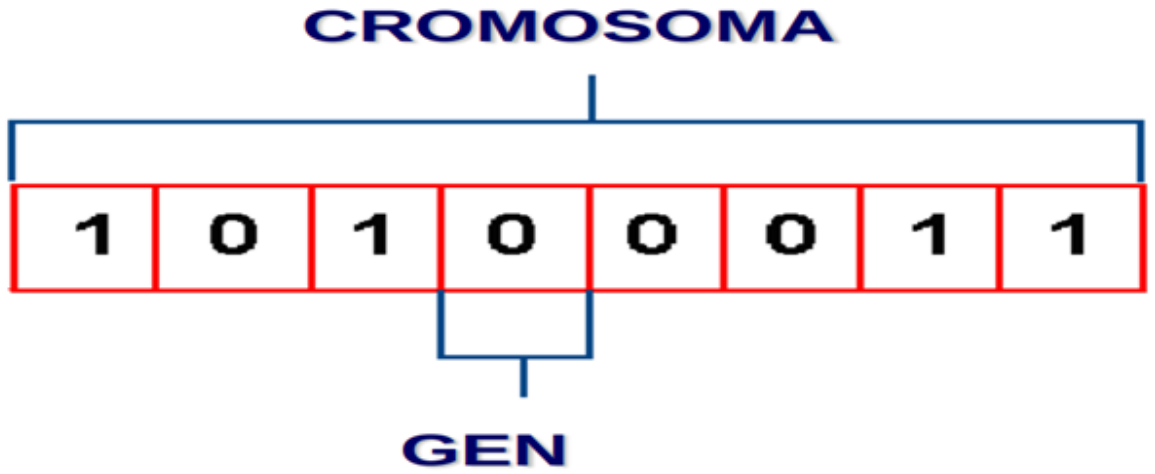
Este ciclo se repite hasta encontrar una solución óptima o se supera un número de ciclos.

Los nuevos descendientes pueden incorporarse a la población en un cierto porcentaje o reemplazarla totalmente.

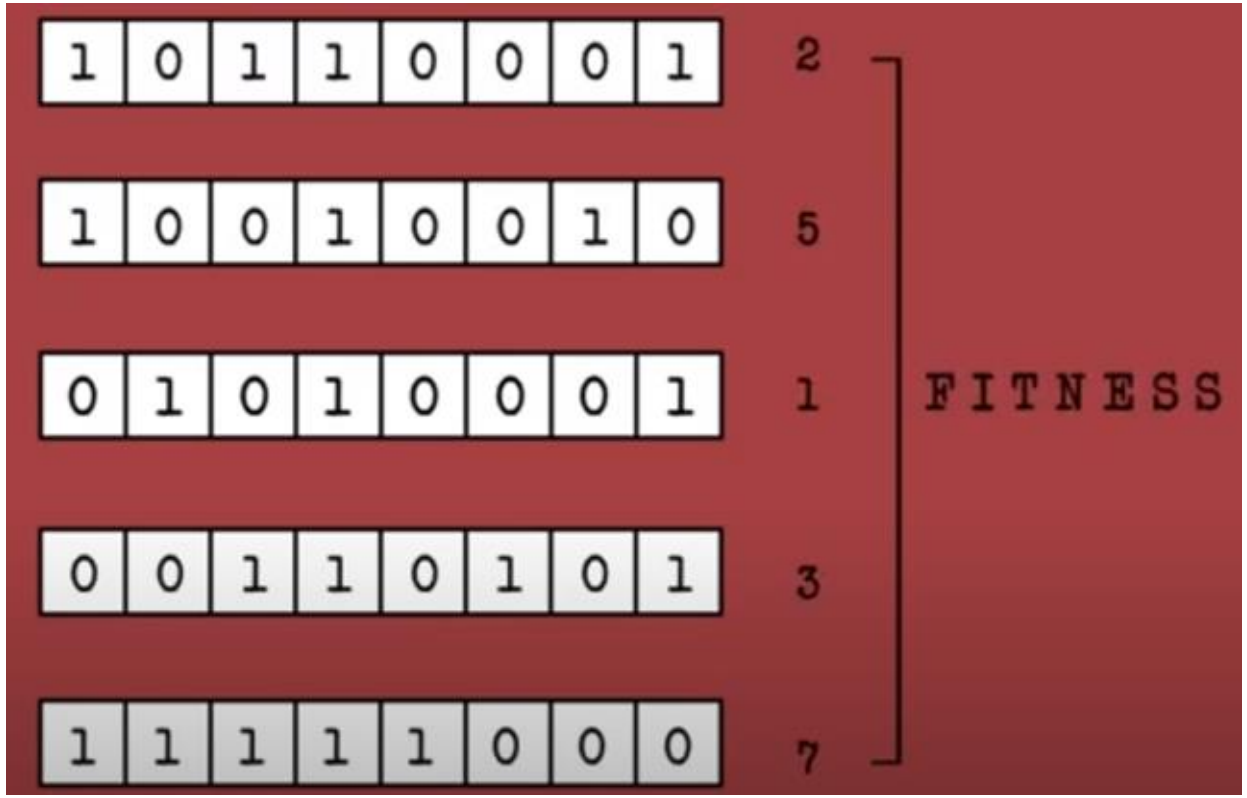
Los cruces y mutaciones pueden ser aleatorios (o sea se pueden dar o no)

ALGORITMOS GENÉTICOS

Cada individuo está representado por su cromosoma.
Y a su vez cada cromosoma está formado por genes. Estos genes pueden ser datos de cualquier clase (binarios, enteros, numeros flotantes o incluso caracteres)
Una población está formada por un cierto número de individuos



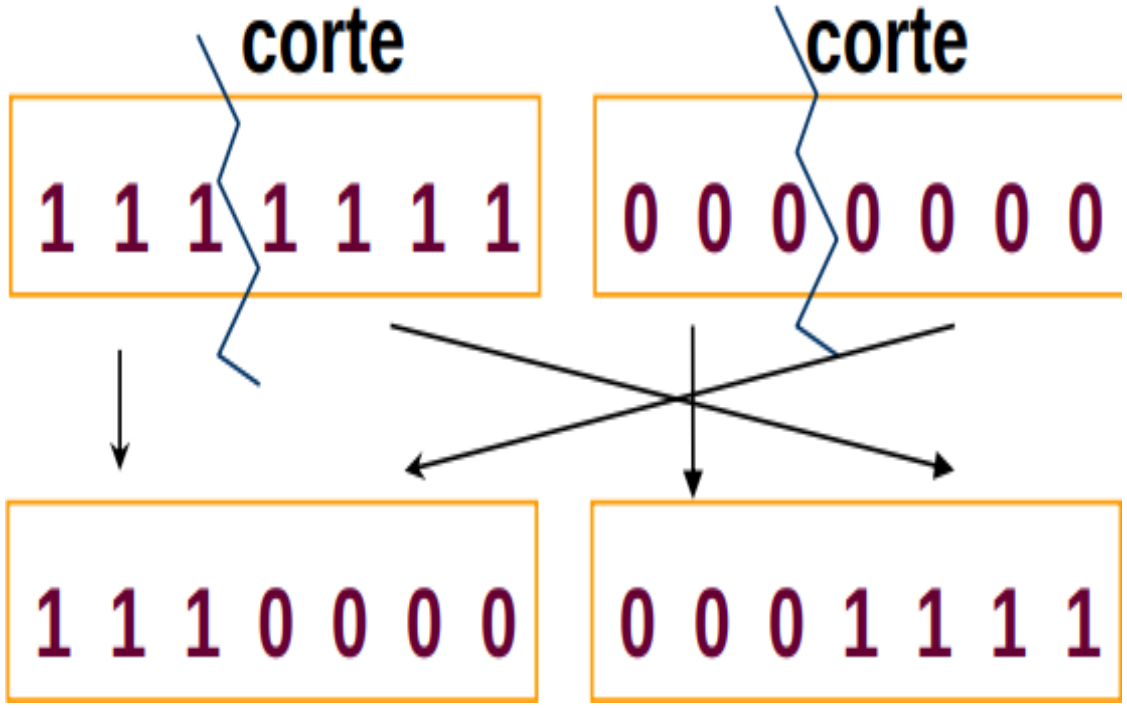
ALGORITMOS GENÉTICOS



Cada individuo tiene asociado un valor de fitness o de ajuste que nos informa de cuán cercano está a la ser la solución óptima al problema que estamos analizando.

Y esto nos permite seleccionar una proporción de individuos (los que tienen mejor fitness) para cruzarlos entre ellos y producir la siguiente generación.

ALGORITMOS GENÉTICOS

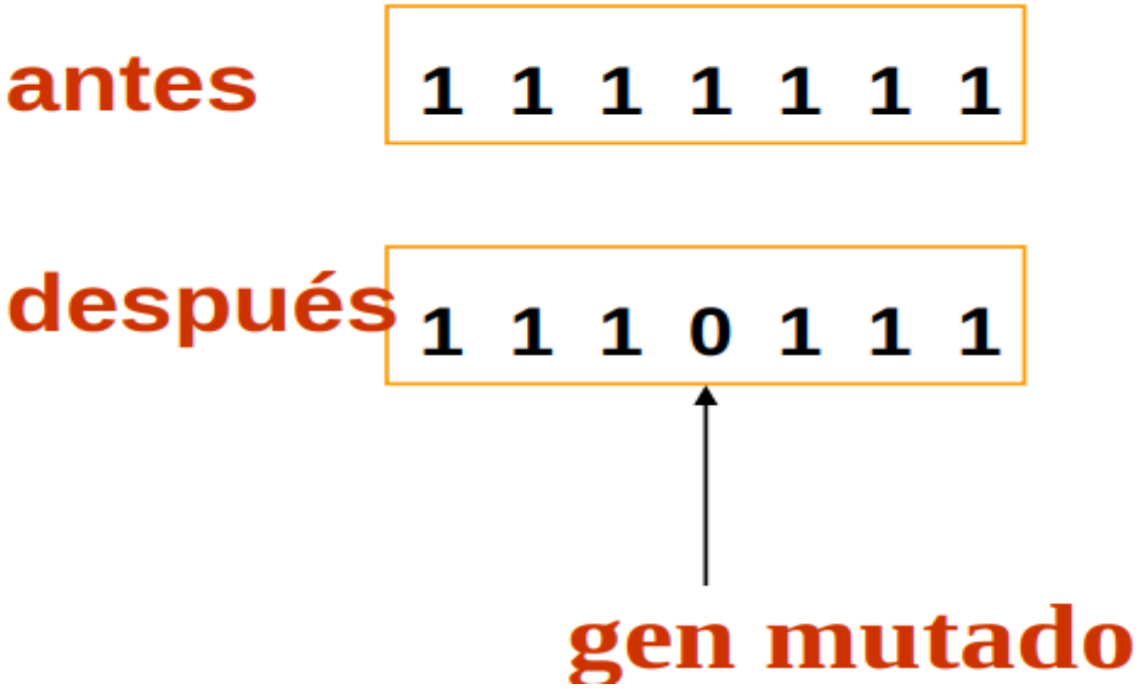


Entre los mejores individuos los vamos separando de a pares y los combinamos entre ellos.
El cruce se produce cortando los cromosomas en cierta parte y con los segmentos que se forman (dos del "padre" y dos de la "madre") se forman dos individuos nuevos para la siguiente generación

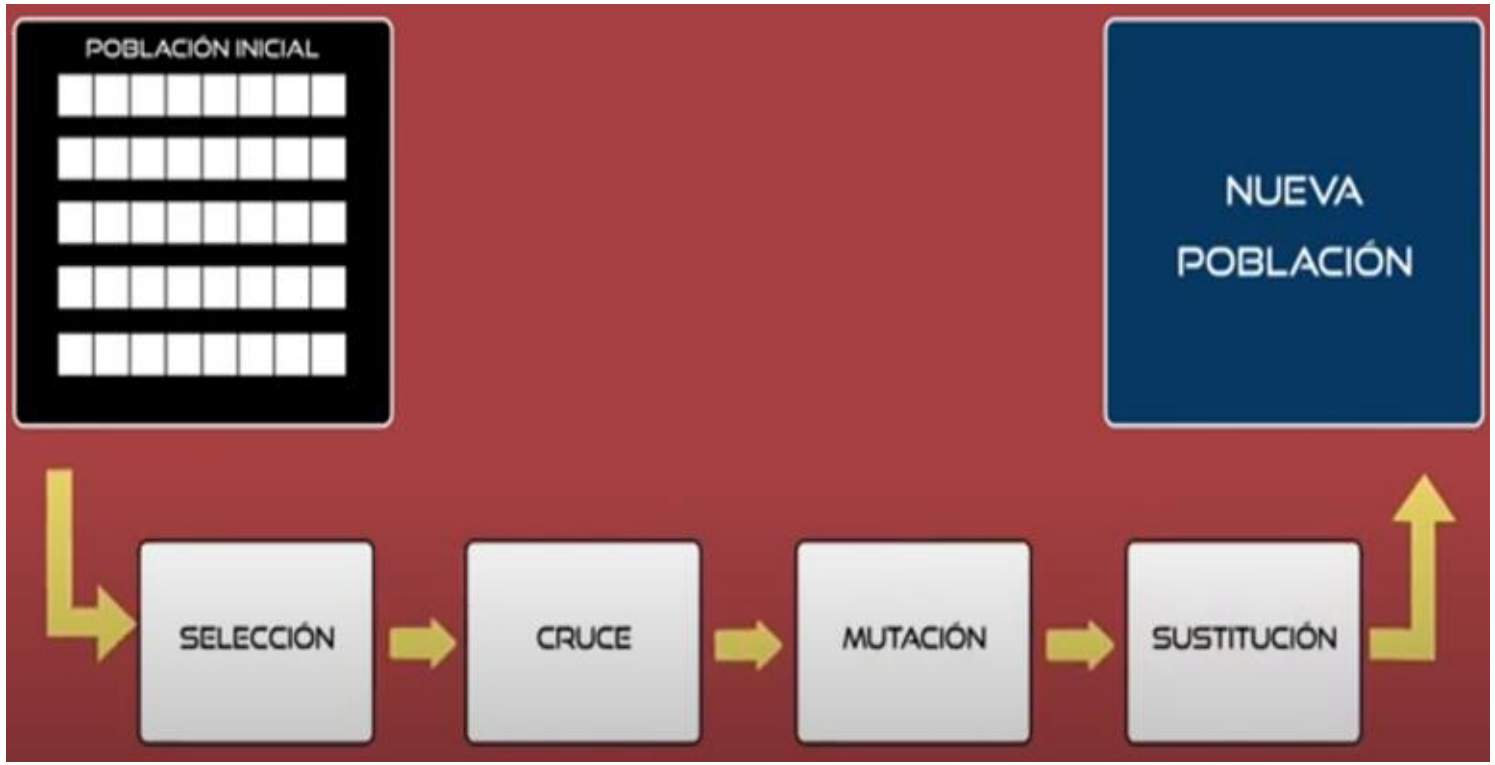
ALGORITMOS GENÉTICOS

La mutación consiste en alterar aleatoriamente alguno de los genes del cromosoma de los nuevos individuos.

En el caso de una representación binaria, consiste en invertir un valor. En otros casos puede consistir en sumar o restar un valor aleatorio.



ALGORITMOS GENÉTICOS



RESUMIENDO:
Partiendo de unos resultados iniciales, elegimos los más ajustados, los combinamos, los mutamos y sustituimos con ellos a la población inicial.
Este proceso se repite hasta obtener un resultado optimo o hasta que se supere un limite al numero de ciclos

ALGORITMOS GENÉTICOS

Desventajas:

- Complejidad de implementación: Requiere la definición de operadores genéticos específicos para cada problema, lo que puede ser complejo.
- Convergencia lenta: Pueden tardar muchas generaciones en converger a una solución óptima, especialmente en problemas de alta dimensionalidad.
- Ajuste de parámetros: El rendimiento depende en gran medida del ajuste de parámetros como la tasa de mutación y el tamaño de la población, lo que puede requerir experimentación.
- No garantizan la optimalidad: Aunque buscan la mejor solución, no hay garantía de encontrarla, especialmente en problemas complejos.
- Costosos computacionalmente: Pueden ser costosos en términos de tiempo de ejecución y recursos, especialmente para problemas grandes.

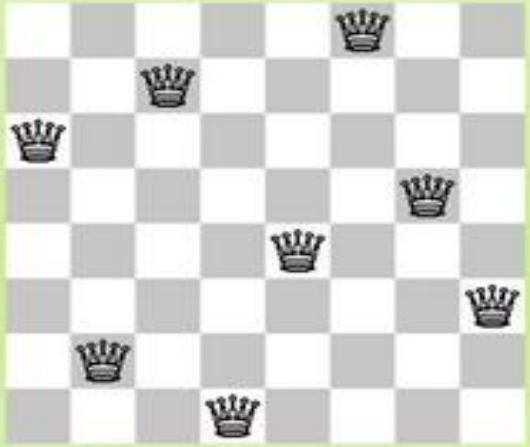
Ventajas:

- Flexibilidad: Se adaptan a una amplia variedad de problemas, incluso aquellos con funciones objetivo complejas o con múltiples soluciones óptimas.
- Robustez: Son tolerantes a ruido en los datos y pueden manejar problemas con espacios de búsqueda discontinuos.
- No requieren conocimiento específico del problema: No necesitan información sobre la estructura del problema, solo la función objetivo a optimizar.

Algoritmos genéticos - Ejemplo

Poblemas de las N reinas ("N queens"):

- Problema de optimización combinatorio muy popular.
- Consiste en colocar N reinas en un tablero de ajedrez N x N, sin que ninguna reina ataque a otra reina. → Conforme N se hace más grande es más complejo!



ALGORITMOS GENÉTICOS

Para demostrar la dificultad de representar en cromosomas algo que a primera vista no lo parece, veamos cómo podemos describir al problema de las n-Reinas.

Supongamos un tablero de 8x8. Representemos al tablero en un vector (lista) cuyo índice sea el número de columna y su valor la fila donde está la reina.
De esta forma nos aseguramos que ninguna columna va a tener más de una reina.

Para verificar que no haya más de una reina en una misma fila, solo debemos revisar que todos los valores del vector sean diferentes. Podemos usar **len**(**set**(lista)) = **len**(lista)

Para ver que no haya más de una reina en cada una de las diagonales, tenemos que verificar que los puntos de dos elementos del vector no tengan una pendiente de +-45 grados (interpretar a la lista como una función *y=f(x)*)

Esto se resuelve si $|x_j - x_i| != |j - i|$