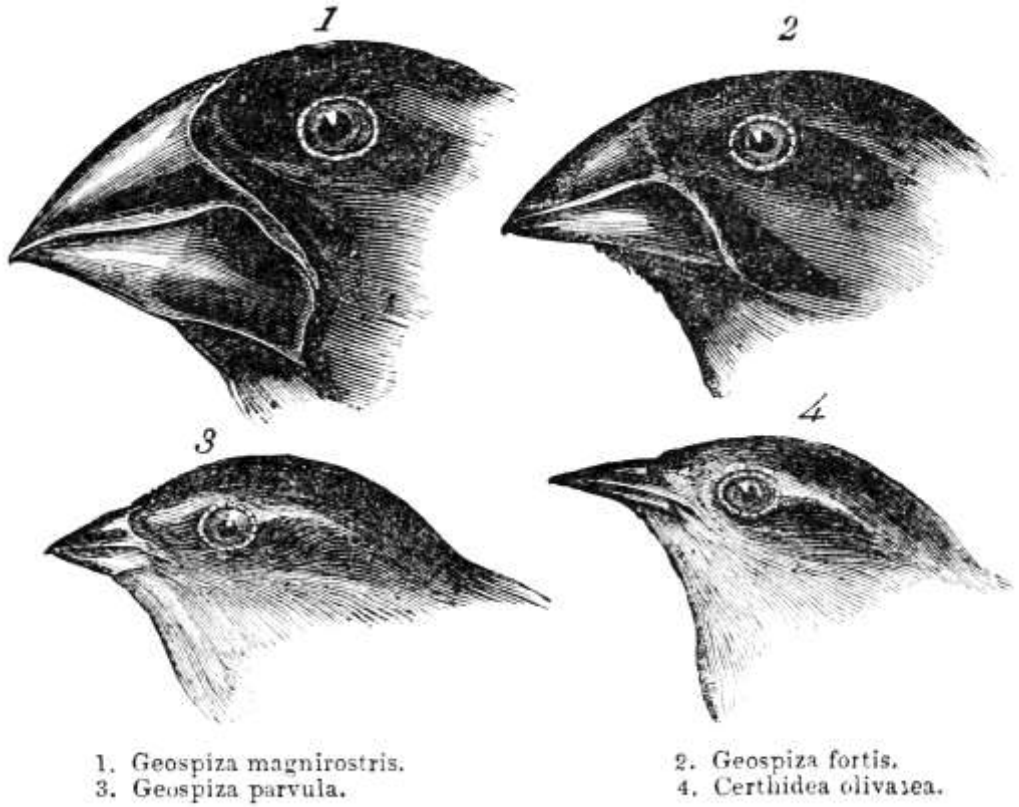
Algoritmos inspirados en la naturaleza

La evolución

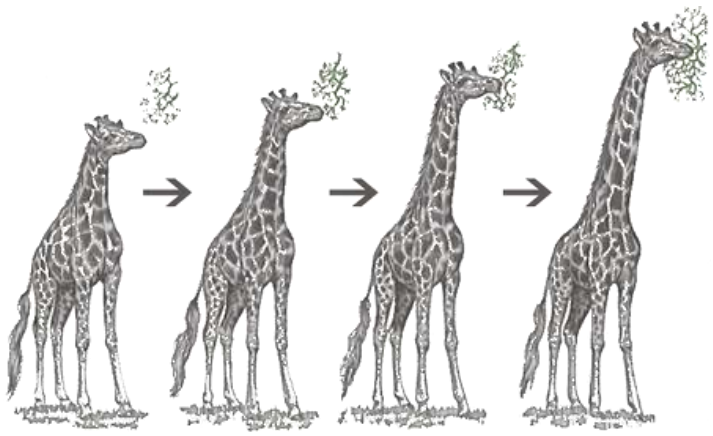


1. Geospiza magnirostris.
2. Geospiza fortis.
3. Geospiza parvula.
4. Certhidea olivacea.

La evolución

- Cambia las **características heredadas** de una población sobre generaciones sucesivas.
  - Características heredables
    - ADN
  - Cambio o variación genética
    - Mutaciones: pequeños cambios aleatorios
    - Recombinación: mezcla de genes mediante la reproducción

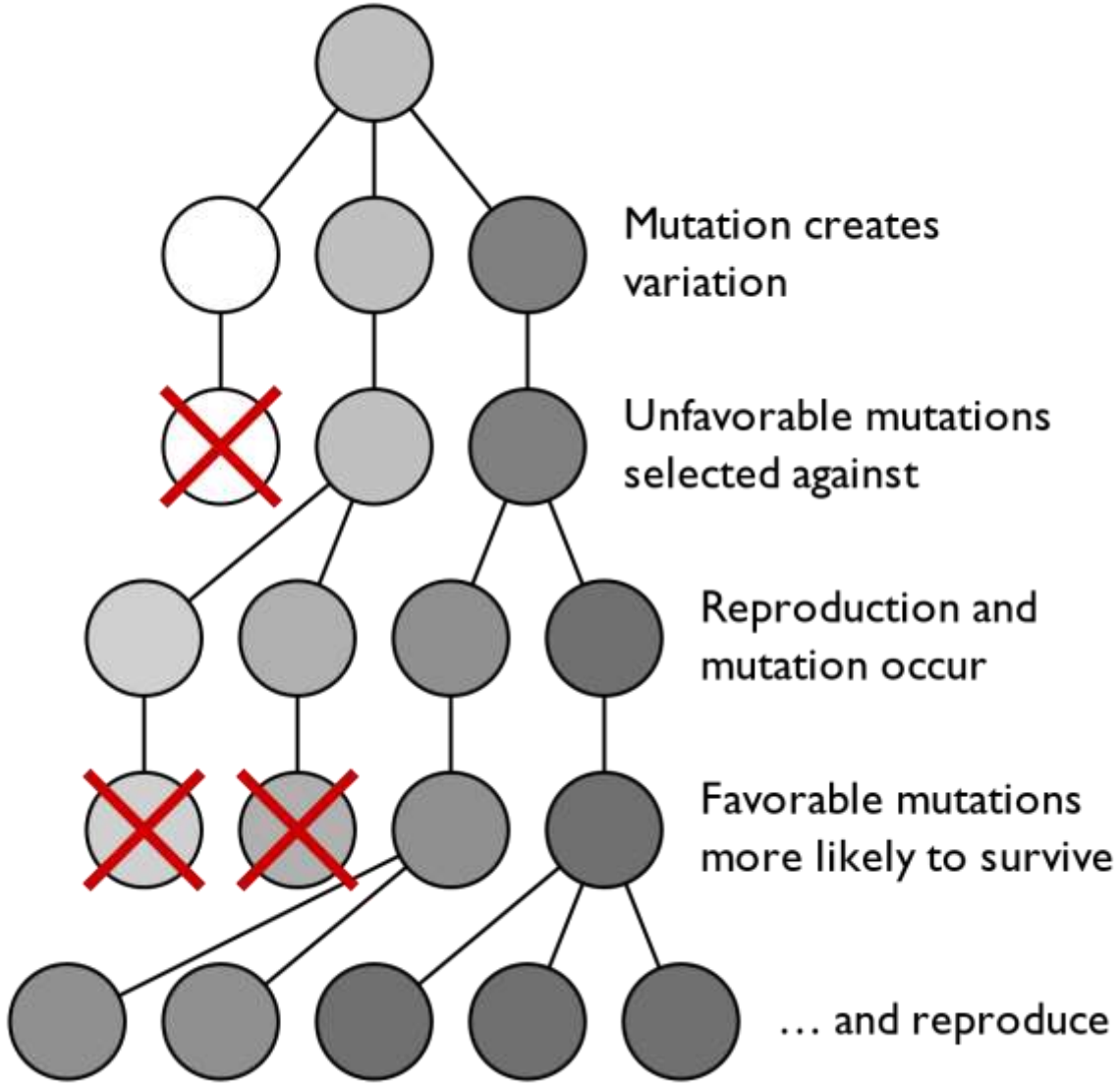
La fuerza de la evolución

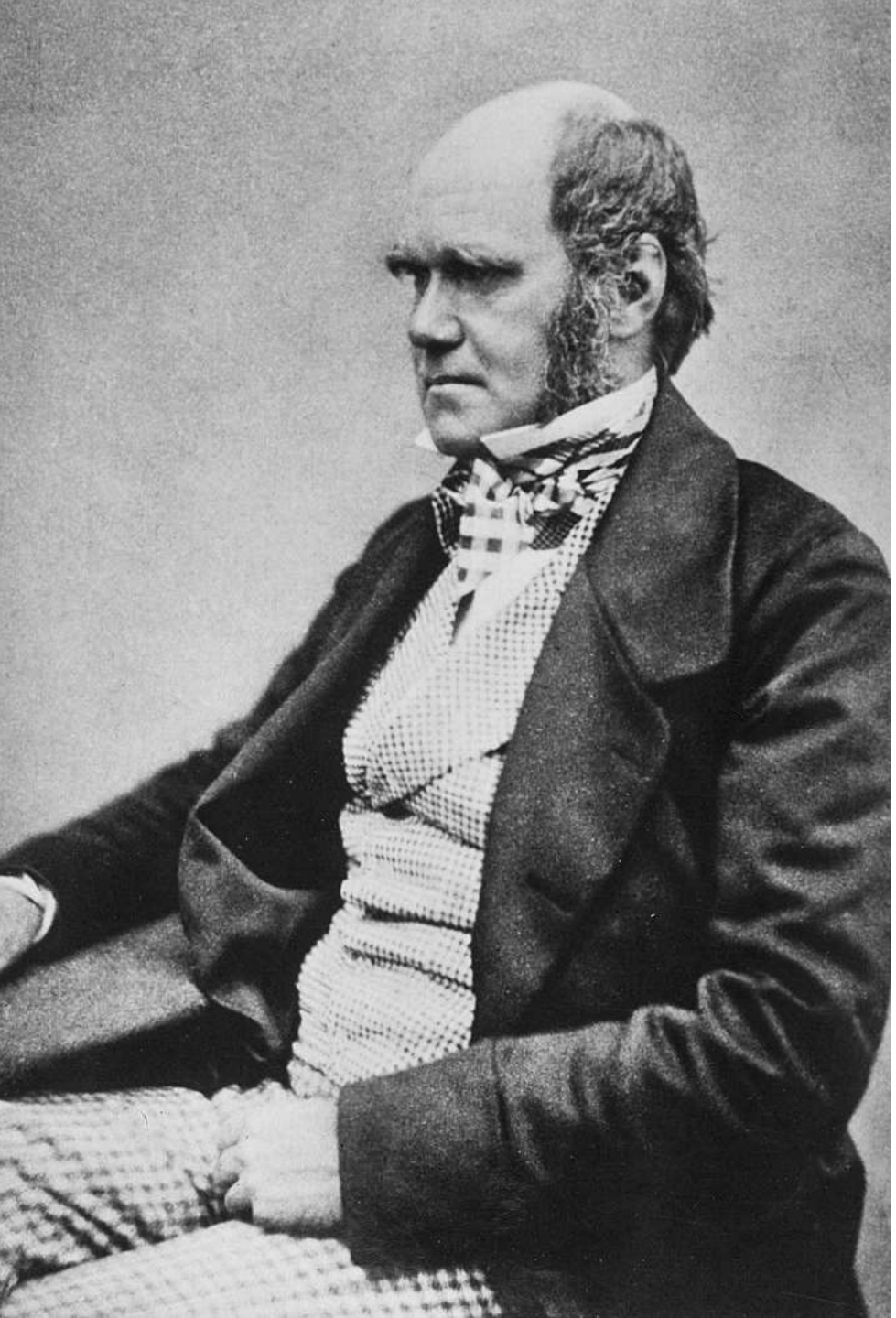


Selección natural, sobrevivencia del más apto

Las variaciones genéticas que mejoran la probabilidad de sobrevivencia y reproducción se vuelven más comunes en generaciones sucesivas

Selección Natural





¿Qué es lo que nos enseñó la evolución?

- Las especies actuales son versiones con variación genética de las especies anteriores
- La evolución está impulsada por la selección natural. La supervivencia del más apto (Survival of the fittest)

¿Qué tiene que ver con ML?

Aptitud (fitness) →

Restricciones en optimización

Individuos en una especie →

Parámetros (o hiperparámetros) en optimización

Algoritmos Evolutivos

- Se inspiran en la evolución biológica
- Se basan en una población: generan, mantienen y optimizan una población de soluciones

- Algoritmos genéticos
- Programación evolutiva
- Estrategias de evolución
- Evolución diferenciada

Piezas de los Algoritmos Evolutivos

Representación: cada solución potencial al problema

Función objetivo: para evaluar la solución (fitness)

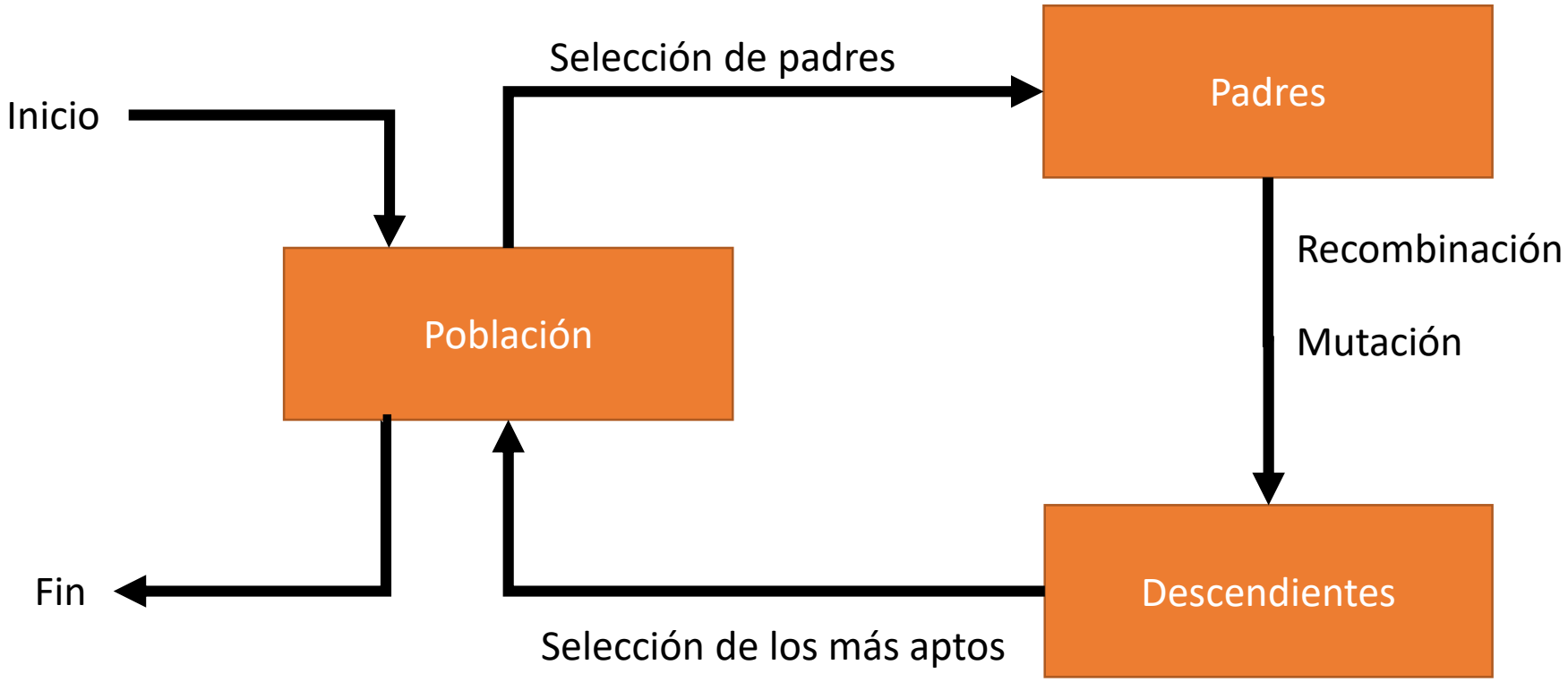
Operadores de variación: Mutación y crossover

Selección y reproducción: sobrevivencia del más apto

Pseudocódigo

- Generar una población inicial de soluciones
- Evaluar el fitness de cada individuo
- Repetir hasta condición
  - Seleccionar: Seleccionar padres en base al fitness
  - Recombinar: Aplicar operaciones de crossover
  - Mutar: Aplicar variaciones aleatorias a los individuos
  - Evaluar: evaluar el fitness de cada individuo
  - Reproducir: Pasar a la siguiente generación los individuos más aptos
- Seleccionar la mejor solución de la población

Gráficamente

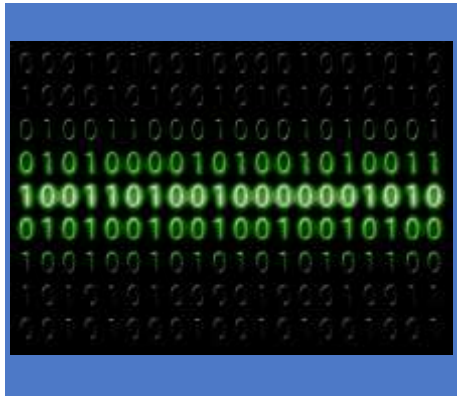


Basado en: Introduction to evolutionary computing, A.E. Eiben, J.E. Smith

Representación

- Una forma de codificar las soluciones
- Función objetivo (fitness) $f(x)$, donde x es la solución
- Cada solución x es llamada **fenotipo**.
- La representación de la solución es llamada **genotipo.**
- Las operaciones de variación se realizan en el genotipo
- Los fenotipos se evalúan usando la función objetivo
- La representación mapea el fenotipo y el genotipo

Representación



- El método de representación depende del problema
- Se puede usar:
  - Representación Binaria
  - Representación de números reales
  - Representación de llaves aleatorias
  - Otros específicos al problema

Representación Binaria

- Tradicionalmente la forma más popular de representación
- Cada solución es una cadena de bits con una longitud L
- Se usa una función de codificación y decodificación para mapear los fenotipos a los genotipos.

Función de codificación

- Convertir números a representación binaria es trivial, pero no usamos la forma “tradicional”

Función de codificación

$\vec{a} \in \{0,1\}^L$ Cadena de bits de longitud L

La solución x está compuesto de x_D variables

$x_i \in [u_i, v_i]$ Cada variable x_i puede estar en un intervalo $[u_i, v_i]$

Para codificar cada variable:

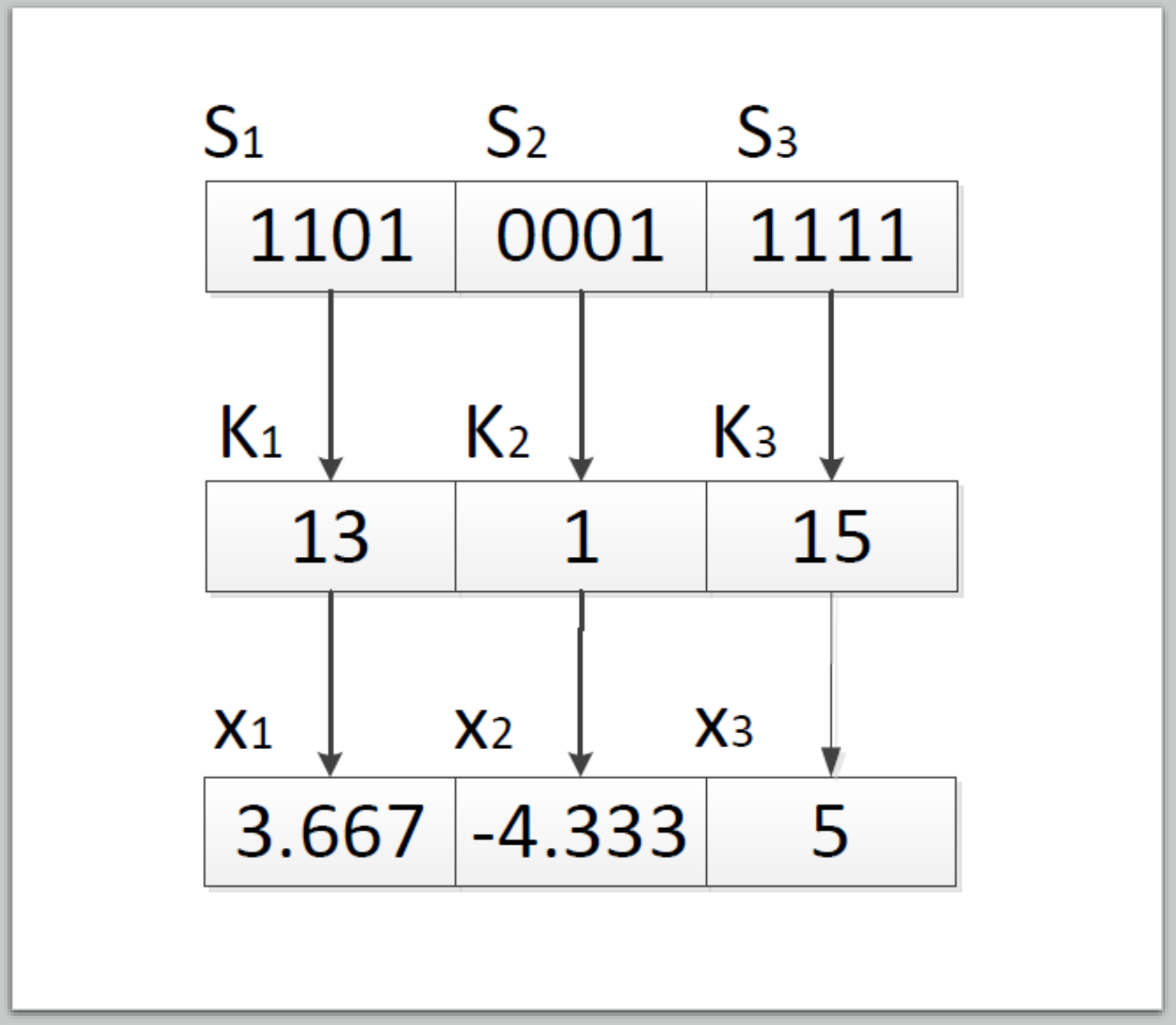
1. Dividir $\vec{a} \in \{0,1\}^L$ en n segmentos de igual longitud $\vec{s}_i \in \{0,1\}^{\frac{L}{n}}, i = 1, \dots, n$

1. Decodificar cada segmento en un entero $K_i, i = 1, \dots, n$ y $K_i = \sum_{j=0}^{\frac{L}{n}} \vec{s}_{i\,j} \times 2^j$
   - Pasar cada segmento binario al número que representa, ej.. 1111 a 15

2. Aplicar la función de codificación: $h(K_i) = u_i + K_i \times \frac{v_i - u_i}{2^{\frac{L}{n}} - 1}$
   - Mapear linealmente el número entero del paso 2 al intervalo de la variable x_i

Ejemplo

- $x = \{x_1, x_2, x_3\}$ y $x \in [-5,5]$
- Usando una cadena de bits de longitud $L = 12$
- Tenemos 3 segmentos de tamaño $\frac{12}{3} = 4$ bits



Hasta ahora

Tenemos una forma de representar el problema

Una forma de evaluar el problema: función objetivo (fitness)

Falta una forma de explorar soluciones: **operadores de variación**

Falta una forma de guiar el algoritmo a mejores soluciones (explotación): **selección y reproducción**

Mutación

- Voltear cada bit con una probabilidad p_m
- Normalmente $p_m = \frac{1}{L}$, pero puede ser hasta ½

- Original: 00101011
- Mutado: 0**1**1010**0**1

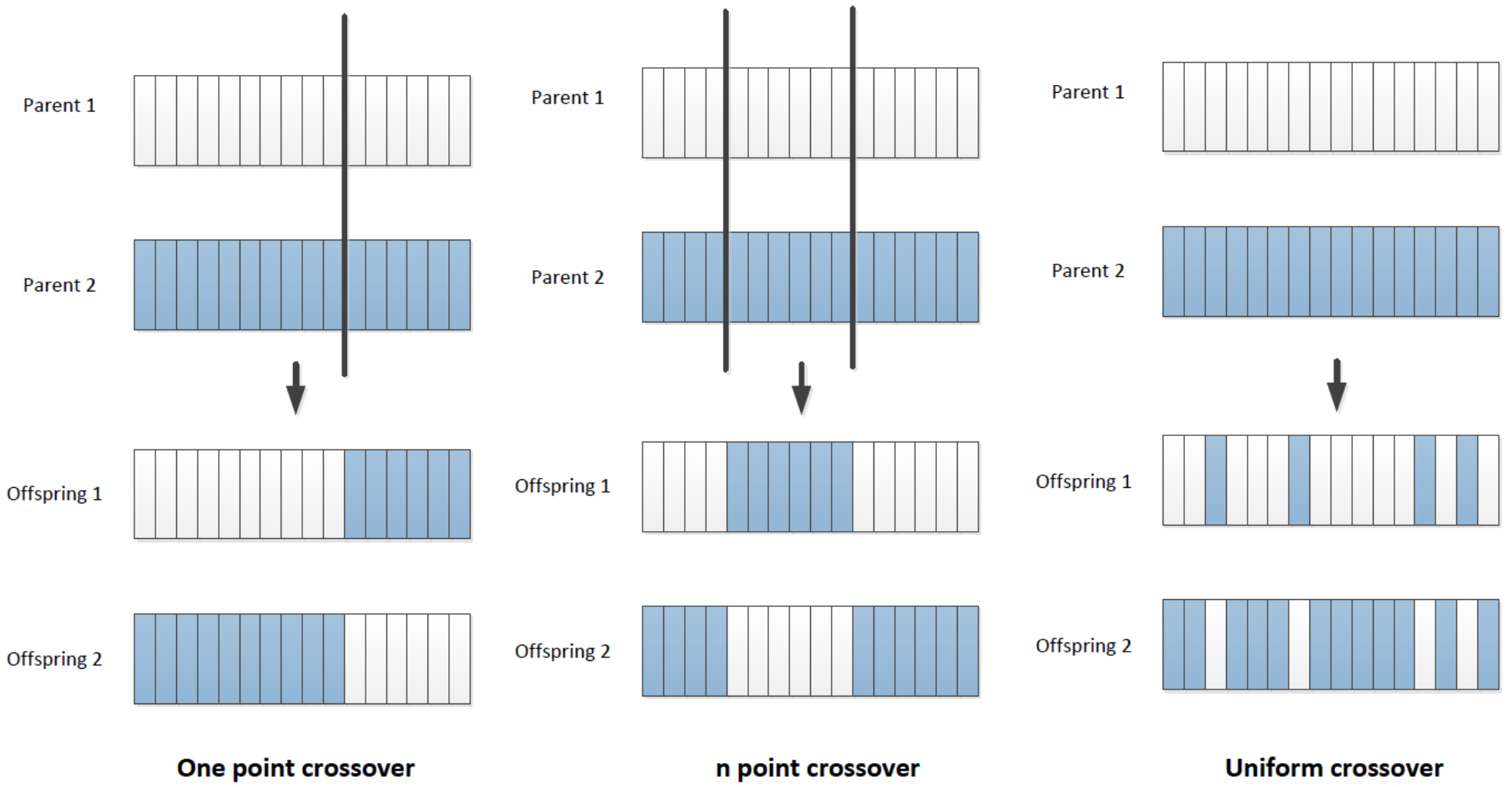
Mutación

- Si la probabilidad de mutación es baja, puede verse como pequeñas perturbaciones al genotipo de los padres.
- Los descendientes mutados serán muy similares a los padres, lo cual implica que se encuentren cercanos en el espacio de búsqueda
- Si combinamos esto con selección, lo que hace es explorar el espacio cercano aleatoriamente.

Recombinación (Crossover)

- Aleatoriamente seleccionar dos padres para el crossover

- 1-point crossover: seleccionar un único punto de crossover e intercambiar la data mas allá del punto
- N-point crossover:
  - seleccionar múltiples puntos de crossover
  - Alternar el intercambio entre los puntos
- Uniform crossover: por cada punto tiene con probabilidad 50%, de ser intercambiado.


Parent 1
Parent 2
Offspring 1
Offspring 2
One point crossover
Parent 1
Parent 2
Offspring 1
Offspring 2
n point crossover
Parent 1
Parent 2
Offspring 1
Offspring 2
Uniform crossover

Selección de padres

La selección se realiza previo a la mayoría de operadores

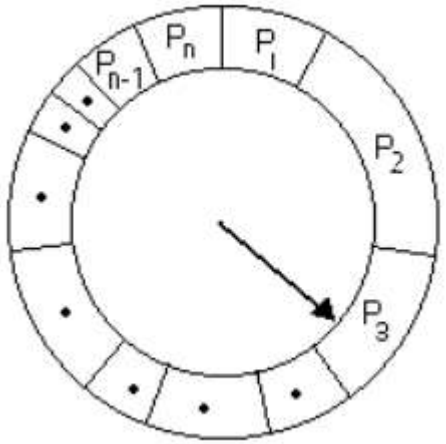
Énfasis en explotar las mejores soluciones

- Seleccionar uno o más soluciones buenas
- Soluciones inferiores también serán seleccionadas, pero con menor probabilidad. ¿Por qué?

Formas de seleccionar

- Selección Proporcional
- Selección por Ranking
- Selección Truncada
- Selección por Torneo

Selección Proporcional



- Similar a una rueda de la fortuna
- La probabilidad de seleccionar al individuo i es

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

Donde f_i es el valor del fitness del individuo, y M es el número de individuos

- No se permiten valores negativos del fitness
- Individuos con fitness alto tienen mayor probabilidad de sobrevivir, pero igualmente pueden ser eliminados
- Individuos con fitness bajo, pueden ser seleccionados (ayuda con escapar óptimos local)

Problemas

- En generaciones iniciales, podemos tener “super individuos” con valores muy altos en el fitness
  - Problema: convergencia prematura en un óptimo local
- En generaciones posteriores, puede no haber mucha separación entre individuos
  - Problema: convergencia lenta

¿Cómo mantener la presión de selección?

- En lugar de usar el valor f_i directamente, se va a escalar proporcionalmente f'_i.

$$f'_i = a + b * f_i$$

Donde a y b son constantes y se definen generalmente como

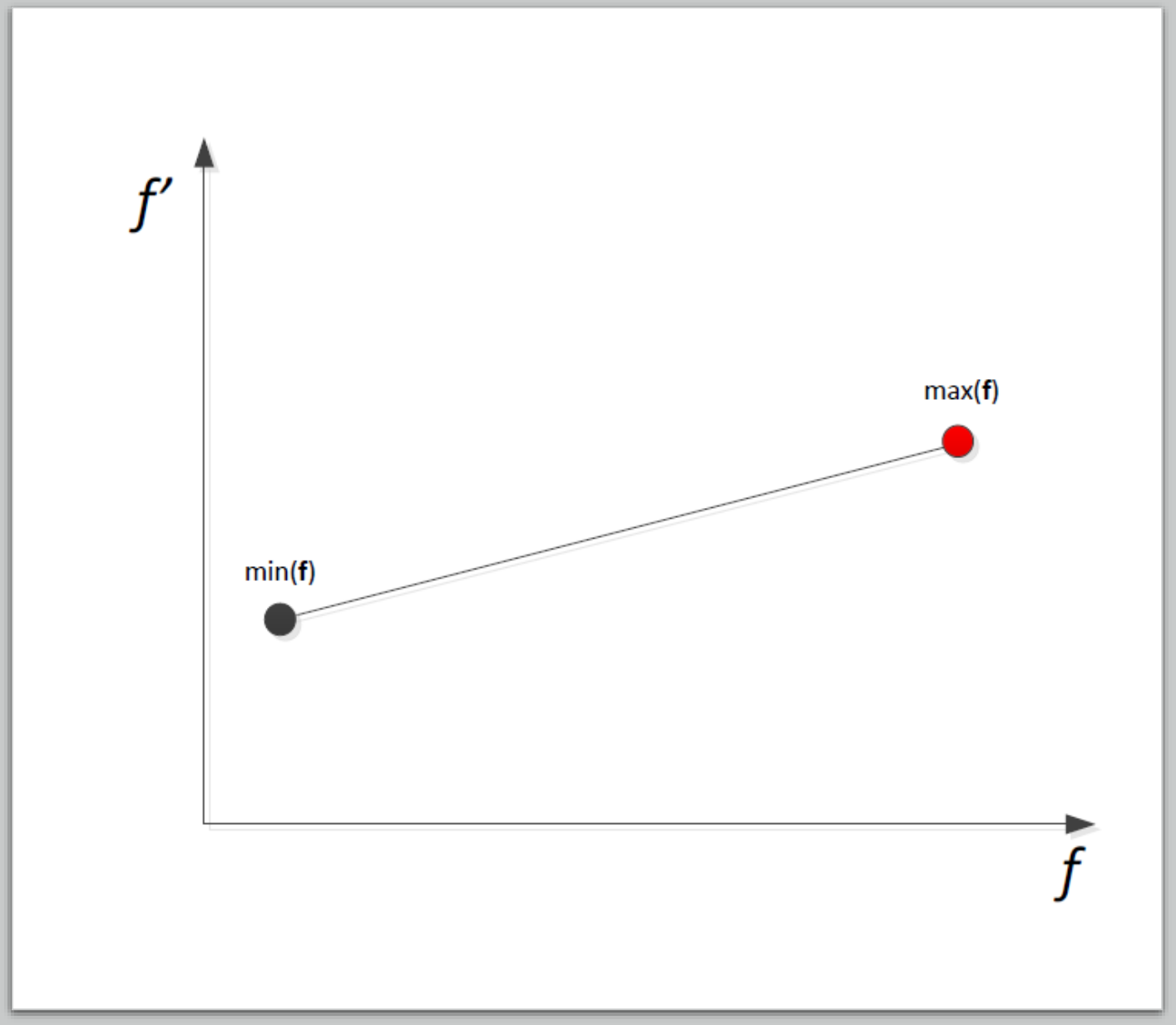
$$a = \max(f)$$

$$b = \frac{\min(f)}{M}$$

Donde M es el número de individuos

Selección Proporcional

$$p_i = \frac{f'_i}{\sum_{j=1}^{M} f'_j}$$

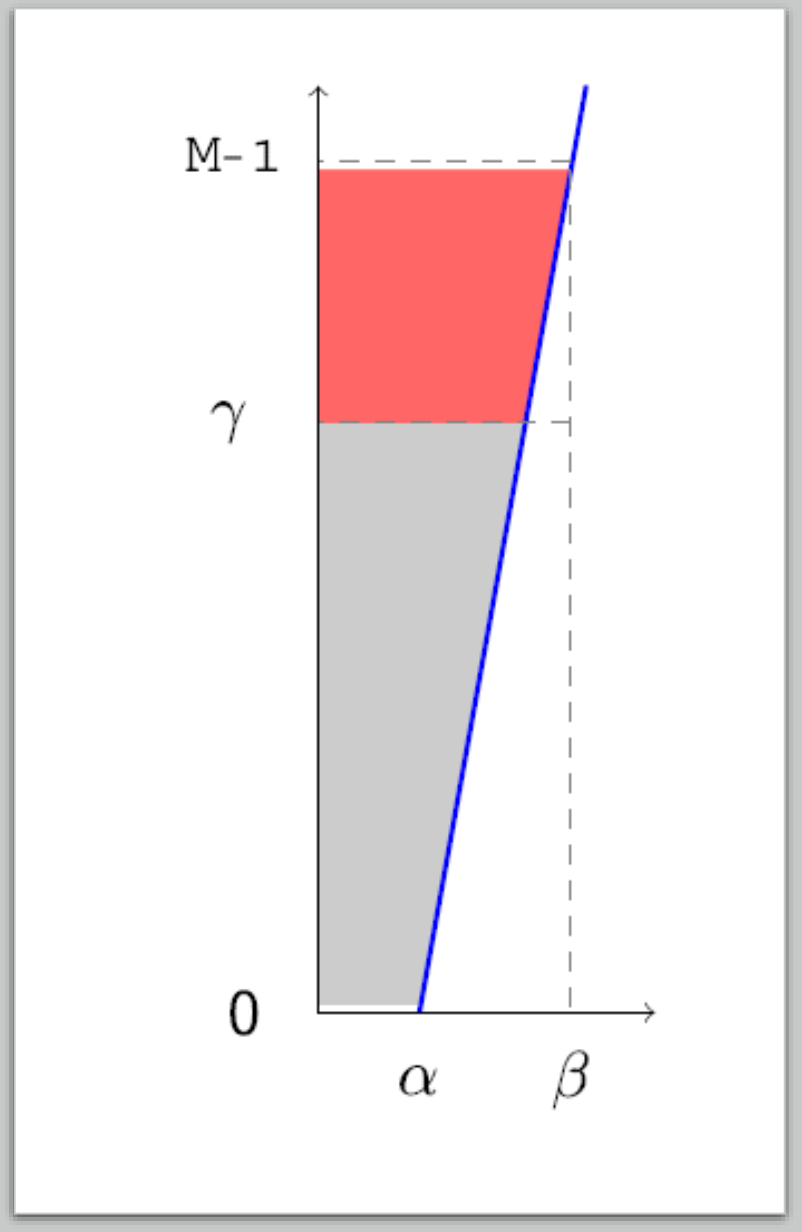


Selección por Ranking

- Ordenar la población M del mejor al peor según el fitness.
- Solo se considera el ranking.

$$x_{M-1}, x_{M-2}, \dots, x_0$$

- Seleccionar los top γ individuos con probabilidad $p(\gamma)$, donde $p(\gamma)$ es una función de ranking
  - Lineal
  - Exponencial
  - Power
  - Geométrica



Ranking Lineal

$$p(\gamma) = \frac{\alpha + (\beta - \alpha) * \frac{\gamma}{M-1}}{M}$$

Donde la suma de todas las probabilidades es 1.

La expectativa es:

- El individuo más apto, va a ser seleccionado β veces
- El peor individuo es seleccionado α veces

Selección Truncada

- Ordenar los individuos de mayor a menor según el valor de fitness
- Seleccionar una proporción de los top.

- Top 50%, o top 30% usualmente

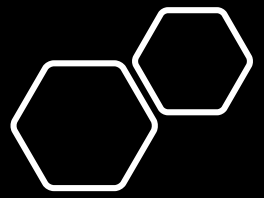
Selección por torneo

- Selección por torneo de tamaño k
  1) Seleccionar aleatoriamente una muestra P de k individuos de la población total
  2) Seleccionar el individuo en P con el fitness más alto
  3) Repetir 1 y 2 hasta generar suficientes descendientes

Torneo binario k=2 es el más popular

Selección de la nueva generación

- Controla como los algoritmos generan la nueva generación
- Generacional vs Steady State
  - Generacional: es el “Estándar”, usa todos los nuevos individuos para reemplazar los peores individuos en la generación para crear la nueva población.
  - Steady State: emplea pocos o incluso un individuo para reemplazar a la población.
- n-Elitista: siempre copia los n mejores individuos a la siguiente generación



Representación con números reales

- Un vector con números reales representa cada variable de la solución
- No hay diferencias entre fenotipo y genotipo
- Mejor precisión
- Debemos redefinir mutación y crossover

Mutación con números reales

Mutación uniforme

- Reemplaza el cromosoma con un valor seleccionado de forma uniforme de un rango de valores

Mutación gaussiana

- Reemplaza el cromosoma con un valor seleccionado de forma gaussiana de un rango de valores

Recombinación (crossover) real

Aleatoriamente seleccionar dos padres y aplicar operaciones de crossover

$$\boldsymbol{x}^{(1)} = \left\{x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)}\right\}$$
$$\boldsymbol{x}^{(2)} = \left\{x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)}\right\}$$

- Flat
- Simple
- Aritmetico completo
- Aritmetico local
- Aritmetico único

Flat: Un descendiente $\boldsymbol{h} = \{h_1, h_2, \dots, h_n\}$ es generado donde cada cromosoma h_i es generado uniformemente del intervalo dado por $x_i^{(1)}$ y $x_i^{(2)}$

Simple: Elegir un punto de crossover, los cromosomas posteriores al punto se intercambian.

Aritmetico completo: Dos descendientes

$$\boldsymbol{h}^{(1)} = \left\{h_1^{(1)}, h_2^{(1)}, \dots, h_n^{(1)}\right\}$$
$$\boldsymbol{h}^{(2)} = \left\{h_1^{(2)}, h_2^{(2)}, \dots, h_n^{(2)}\right\}$$

Donde

$$h_i^{(1)} = a x_i^{(1)} + (1-a) x_i^{(2)}$$

$$h_i^{(2)} = a x_i^{(2)} + (1-a) x_i^{(1)}$$

Y a es un numero aleatorio entre 0 y 1. Todos los genes usan el mismo a

Aritmetico local: Lo mismo que aritmético completo, pero cada gen utiliza un α diferente.

Aritmetico único: Se selecciona un gen aleatorio. Ese gen es reemplazado por el promedio de los padres. El resto es copiado tal cual

Y hay muchos más

http://bib.irb.hr/datoteka/640222.CEC_2013.pdf

- https://phys.org/news/2013-04-virtual-squishy-creatures-evolve-evolutionary.html

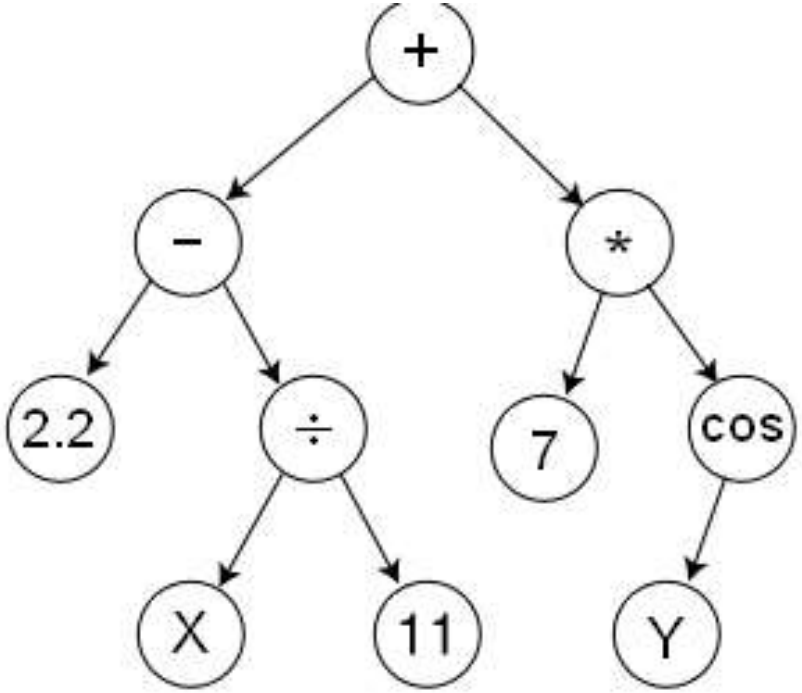
Programación Evolutiva

- Cada gen es una operación
- Operadores matemáticos, +,-,*,/, log, sum, pow
- Booleanos, And, Or, Not, Xor
- Funciones de memoria, escribir, leer,
- Estructuras de control, if then else, for
- Otros, mover, girar, abrir

Las funciones deben ser suficientemente complejas para la tarea, pero no tan complidadas.

La función debe aceptar todas las entradas válidas.

(división entre 0?)



$$\left(2.2 - \left(\frac{X}{11}\right)\right) + \left(7 * \cos(Y)\right)$$