

Unidad 2:

Resolución de problemas mediante búsquedas y heurísticas

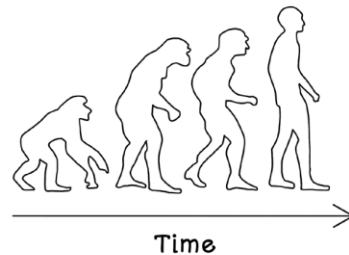
2.4 Algoritmos evolutivos

Introducción (1)

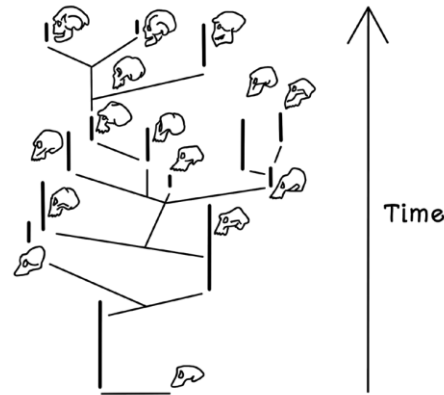
- La teoría de la evolución sugiere que los organismos vivos existentes en la actualidad no han sido siempre como ahora.
- Son el resultado de cambios sutiles a lo largo de los años dirigidos a adaptarse al entorno.
- Estos cambios se han realizado a través de la reproducción, con hijos que poseen genes de ambos padres.

Introducción (2)

- No es un proceso lineal:
 - Caótico
 - Cambios tardan años en manifestarse.



Perceived evolution

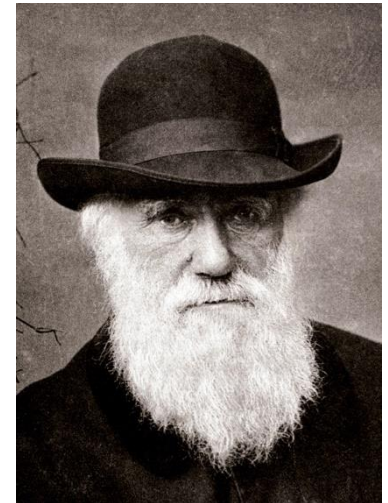


Actual evolution

Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

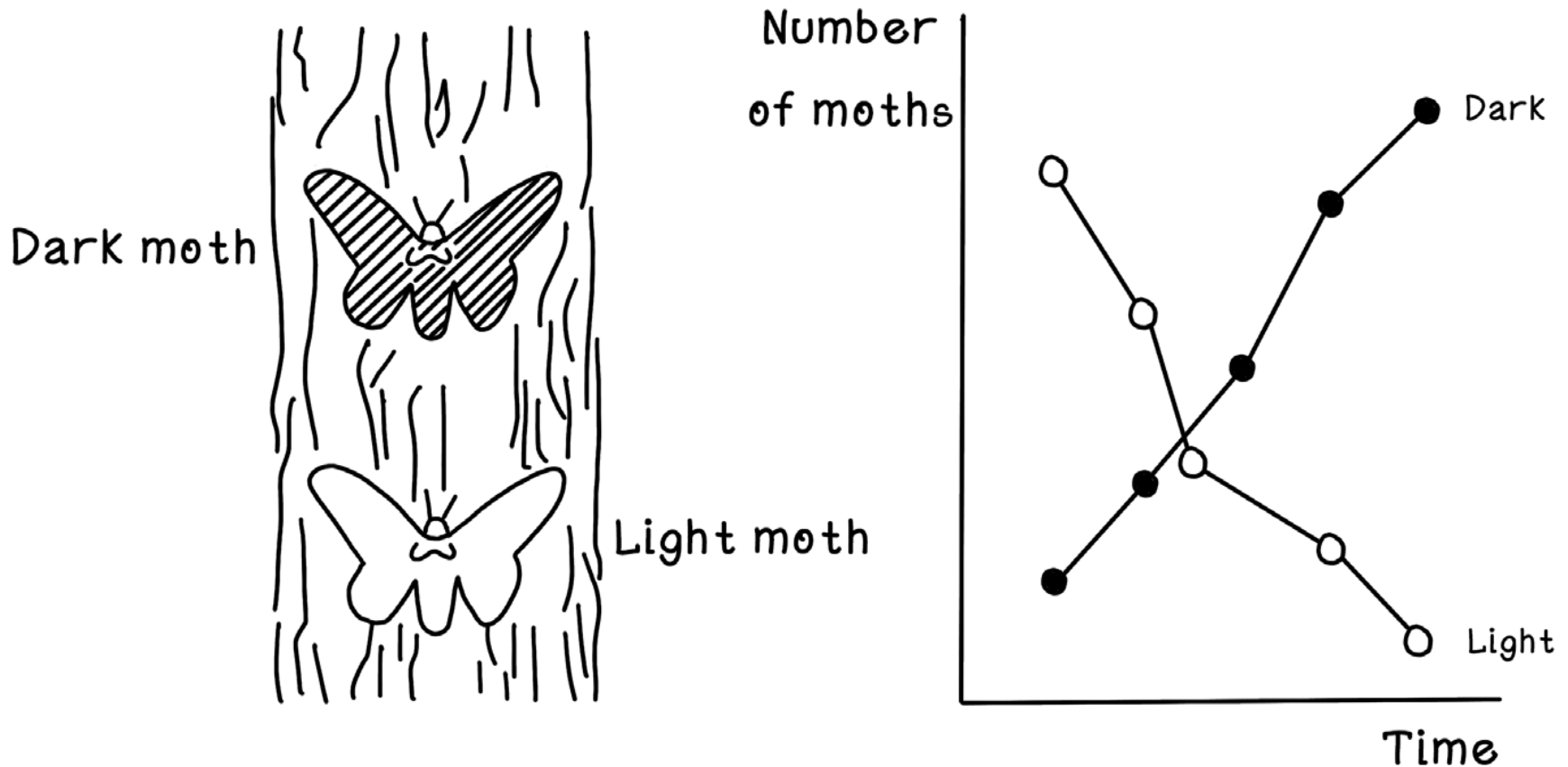
Introducción (3)

- Charles Darwin propuso:
 - Selección natural.
 - Se reproducen más y por tanto incorporan sus ventajas de supervivencia a las generaciones posteriores.
 - Podría derivar en un comportamiento mejorado respecto a sus ancestros.
 - Ej. Polilla moteada.



Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

Introducción (3)



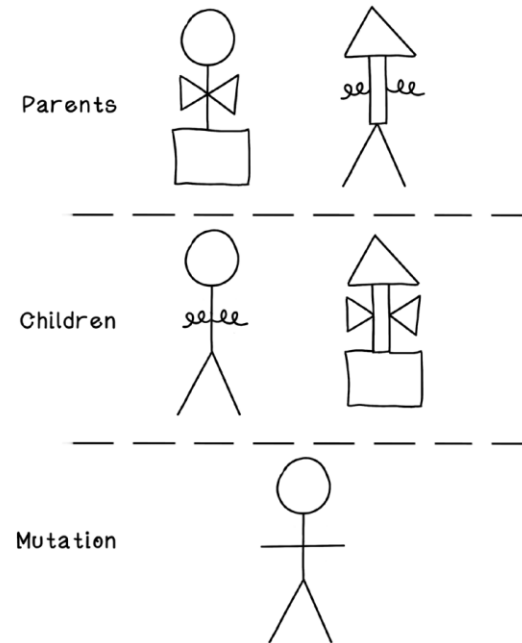
Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

Introducción (4)

- Los descendientes son una combinación de los genes de sus padres con pequeños cambios denominados mutaciones.
- Además, algunos miembros mueren por diversas causas.
- Supervivencia del más apto.

Introducción (5)

- De acuerdo con Darwin una población se caracteriza:
 - Variedad
 - Herencia
 - Selección
- Estas propiedades implican que durante la evolución aparezcan:
 - Reproducción
 - Recombinaciones y mutaciones



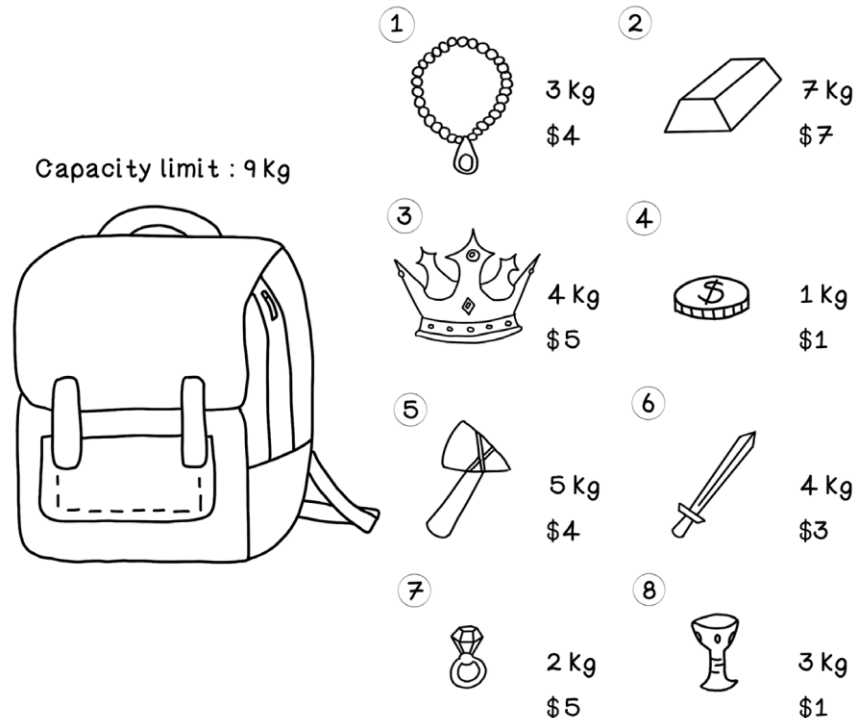
Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

Introducción (6) - Conclusiones

- Proceso caótico que produce distintas variantes de forma de vida.
- Unas variantes son mejores que otras para unos entornos determinados.
- Esta teoría sirve de base para los algoritmos evolutivos, que aprenden de la evolución biológica para encontrar soluciones óptimas a problemas prácticos.
- Generando soluciones que convergen entre sí para dar mejores resultados en próximas generaciones.
- Los algoritmos evolutivos pueden usarse solos o en compañía de otras técnicas como las redes neuronales.

Problemas aplicables a los algoritmos evolutivos (1)

- No todos los problemas son aptos.
- Problemas cuya solución está formada por un gran número de permutaciones.
- Hay varias soluciones posibles, pero unas más óptimas que otras.
- Ej. Problema de la mochila



Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

Observación reflexiva #1

Establece 3 soluciones para el problema de la mochila propuesto.

ID	Nombre	Peso (Kg)	Valor (\$)
1	Perlas	3	4
2	Oro	7	7
3	Corona	4	5
4	Moneda	1	1
5	Hacha	5	4
6	Espada	4	3
7	Anillo	2	5
8	Copa	3	1

- Solución 1:
 - 1, 4, 6
 - 8kg y 8\$
- Solución 2:
 - 1, 3, 7
 - 9kg y 14\$
- ...
- Solución 256:
 - 2, 3, 6
 - 15kg

Problemas aplicables a los algoritmos evolutivos (2)

Item ID	Item name	Weight (kg)	Value (\$)
1	Axe	32,252	68,674
2	Bronze coin	225,790	471,010
3	Crown	468,164	944,620
4	Diamond statue	489,494	962,094
5	Emerald belt	35,384	78,344
6	Fossil	265,590	579,152
7	Gold coin	497,911	902,698
8	Helmet	800,493	1,686,515
9	Ink	823,576	1,688,691
10	Jewel box	552,202	1,056,157
11	Knife	323,618	677,562
12	Long sword	382,846	833,132
13	Mask	44,676	99,192

14	Necklace	169,738	376,418
15	Opal badge	610,876	1,253,986
16	Pearls	854,190	1,853,562
17	Quiver	671,123	1,320,297
18	Ruby ring	698,180	1,301,637
19	Silver bracelet	446,517	859,835
20	Timepiece	909,620	1,677,534
21	Uniform	904,818	1,910,501
22	Venom potion	730,061	1,528,646
23	Wool scarf	931,932	1,827,477
24	Crossbow	952,360	2,068,204
25	Yesteryear book	926,023	1,746,556
26	Zinc cup	978,724	2,100,851

¿Fuerza bruta?

Combinations
Iterations
Accuracy
Compute time

$2^{26} = 67,108,864$
 $2^{26} = 67,108,864$
 100%
 ~7 minutes

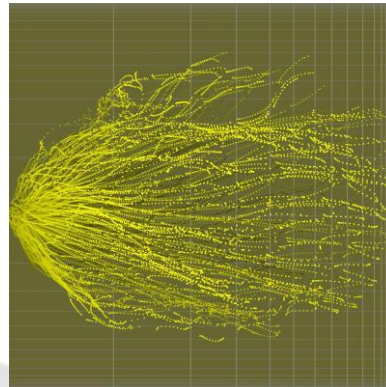
Problemas aplicables a los algoritmos evolutivos (3)

Otros ejemplos de problemas aplicables son:

- Compañía que intenta optimizar los paquetes de cada camión de acuerdo a sus destinos.
- La misma compañía que intenta encontrar la ruta más corta entre varios destinos.
- Si una fábrica refina los artículos en materia prima a través de un sistema de cinta transportadora y el orden de los artículos influye en la productividad.

Problemas aplicables a los algoritmos evolutivos (4)

- Los algoritmos evolutivos son estocásticos:
 - La salida del algoritmo no tiene por qué ser igual cada vez que se ejecuta.
- Proporciona una buena solución, que no tiene por qué ser la mejor solución.
- Su empleo depende del contexto.



Algoritmos genéticos (1)

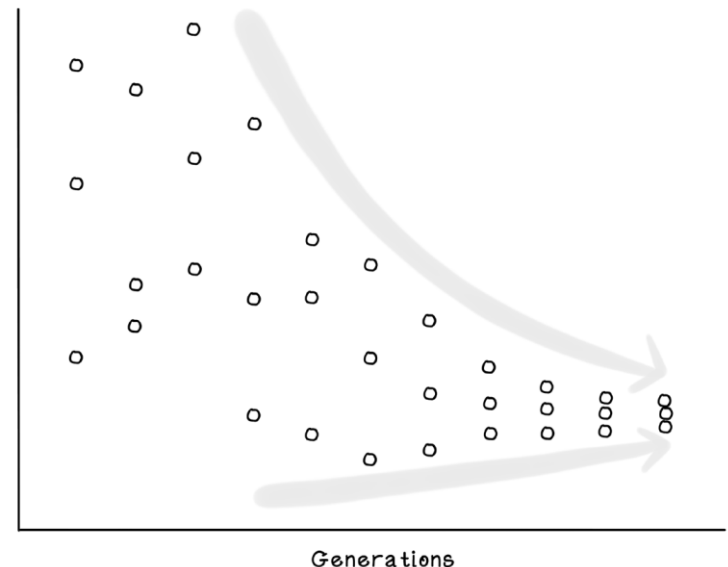
- Pertenecientes a la familia de los algoritmos evolutivos.
- Evalúan grandes espacios de búsqueda para encontrar una buena solución.
- Intenta buscar **la mejor solución global** mientras evita las mejores **soluciones locales**.
- La idea es intentar incrementar la búsqueda de soluciones locales para encontrar la mejor solución global.



Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

Algoritmos genéticos (2)

- Intenta buscar una diversidad de soluciones para, gradualmente, alcanzar una mejor solución en cada generación.
- Al principio, las soluciones varían de acuerdo con sus atributos genéticos.
- Sin una variedad inicial, se corre el riesgo de estancarse en las generaciones posteriores.

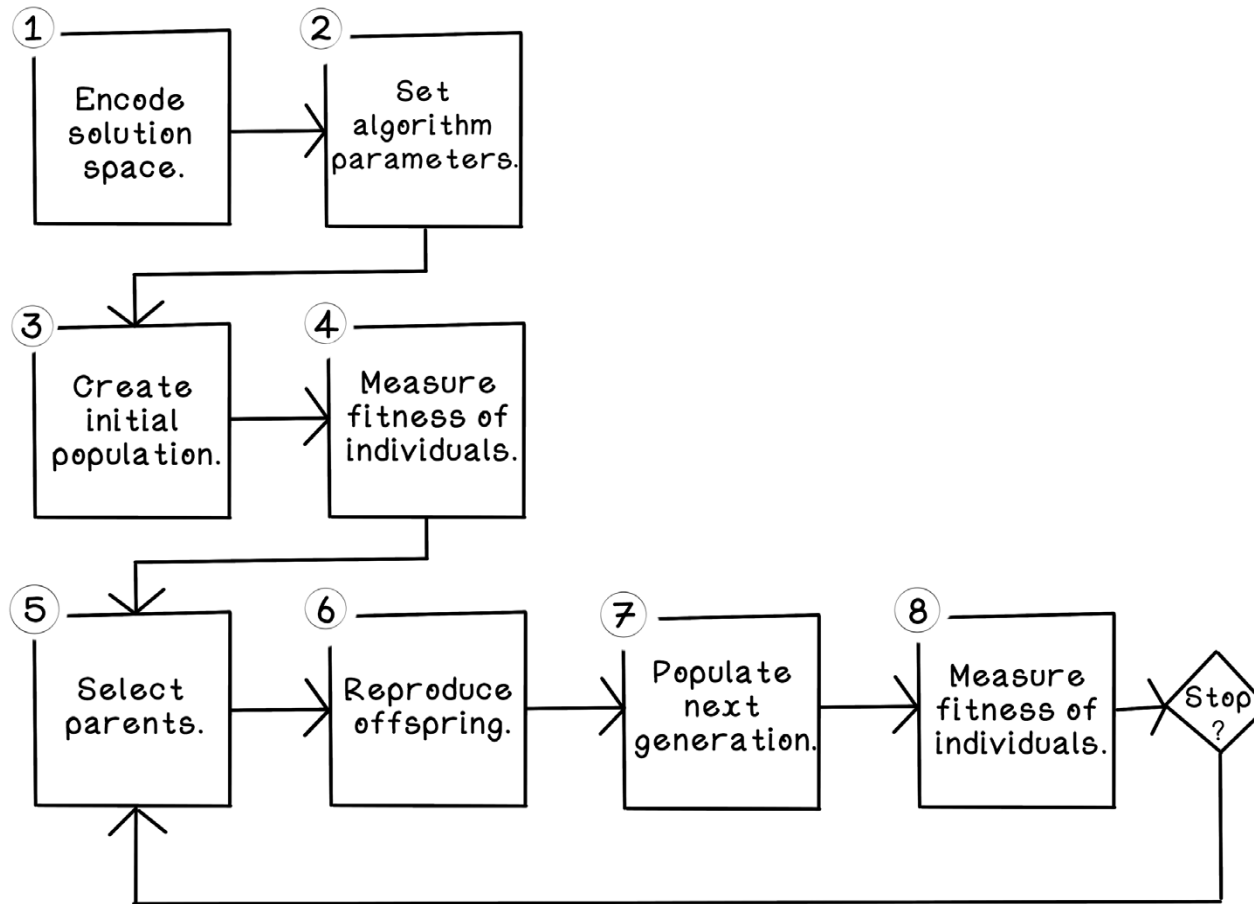


Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

Algoritmos genéticos (3) – Ciclo de vida

- Cada problema tiene un contexto único y un dominio diferente en el cual se representan los datos, por lo que las soluciones son evaluadas de forma distinta.
- El ciclo de vida de un algoritmo genético es el siguiente:
 - Crear una población.
 - Medir la aptitud de los individuos de la población.
 - Seleccionar padres basados en su aptitud.
 - Reproducir individuos usando estos padres.
 - Poblar la siguiente generación.

Algoritmos genéticos (4) – Algoritmo

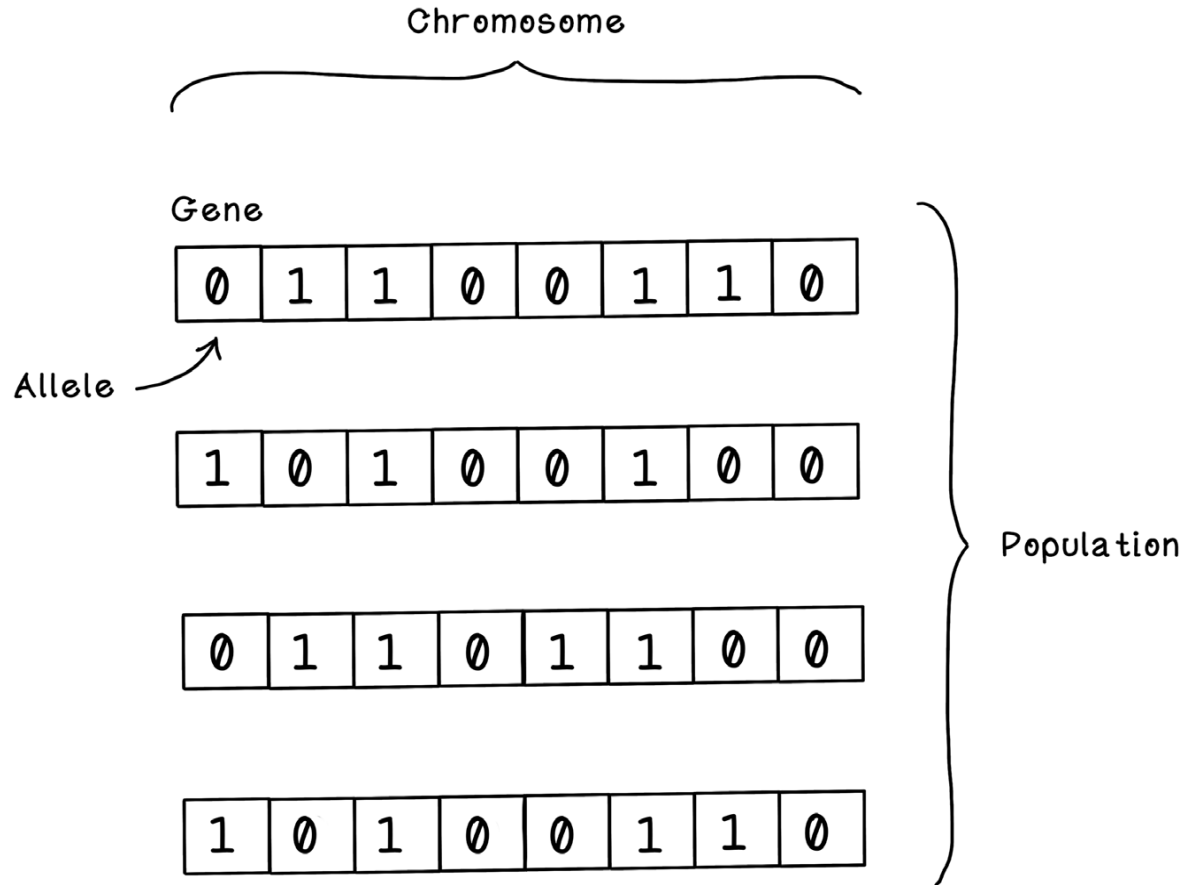


Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

1 – Codificar los espacios de la solución (1)

- Es necesario representar adecuadamente los posibles estados.
- Terminología:
 - Cromosoma: un candidato a solución. Los cromosomas están formados por distintos genes. Cada cromosoma tiene el mismo nº de genes.
 - Gen: Es la unidad lógica de cada unidad.
 - Alelo: Es el valor almacenado en cada gen.
 - Genotipo: La representación artificial de una solución potencial
 - Fenotipo: La representación en el mundo real de una solución potencial.
 - Población: Una colección de cromosomas.

1 – Codificar los espacios de la solución (2)



Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

Observación reflexiva #2

¿Cómo representarías los elementos colocados en la mochila?

- Una posible solución es emplear una codificación binaria.

Capacity limit: 9 Kg




Diagram illustrating a knapsack problem with 8 items and a binary solution vector.

Item	Weight (Kg)	Value (\$)
1. Necklace	3	4
2. Gold bar	7	7
3. Crown	4	5
4. Coin	1	1
5. Pickaxe	5	4
6. Sword	4	3
7. Ring	2	5
8. Chalice	3	1

Binary solution vector (1 = included, 0 = excluded):

Item	1	2	3	4	5	6	7	8
Value	1	0	1	0	0	0	1	0

Codificación binaria

- 0 y 1.
- Datos primitivos:
 - Menos demanda de memoria.
 - Normalmente las operaciones binarias son más rápidas.
- No todos los problemas pueden adaptarse a esta codificación.

Entregable 2.5

Representa una posible representación binaria del siguiente problema.
¿Cuál es el número de cromosomas?

THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Incorrect phrases

THE BROWN JUMPS OVER
QUICK FOX OVER THE
THE FOX THE LAZY

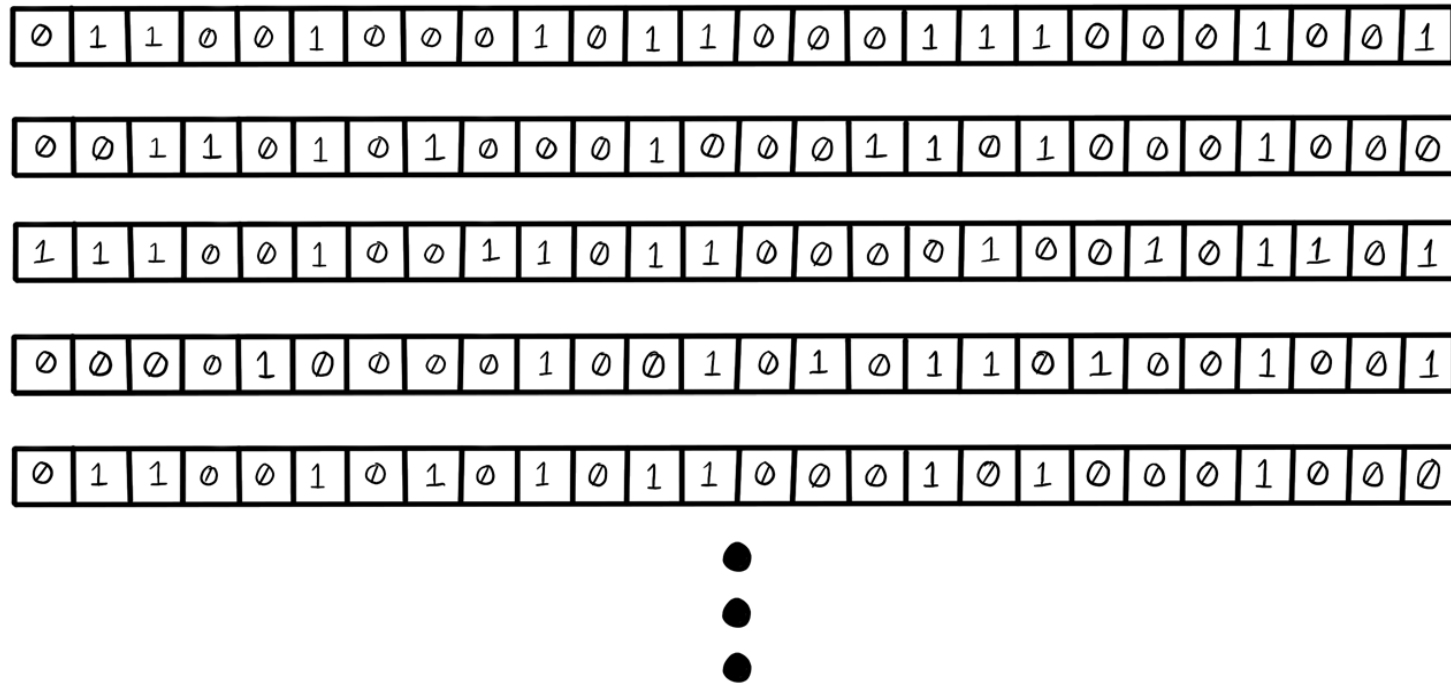
Correct phrases

THE QUICK FOX
QUICK FOX JUMPS
THE BROWN FOX DOG
THE BROWN LAZY DOG
THE QUICK DOG
QUICK OVER THE DOG
THE QUICK LAZY DOG

3 – Crear una población de soluciones (1)

- Hay que crear un conjunto **aleatorio** de posibles soluciones.
- Aunque los cromosomas son creados aleatoriamente, hay que tener en cuenta las restricciones del problema.
- Si una solución viola las restricciones, podemos **no incluirla** o asignarle un valor de aptitud muy malo.
 - Asignarles valores de aptitud posibilita que haya asignaciones imparciales.

3 – Crear una población de soluciones (2)



Population size

Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

3 – Crear una población de soluciones (3)

```
generate_initial_population (population_size, individual_size)
  let population be an empty array
  for individual in range 0 to population_size
    let current_individual be an empty array
    for gene in range 0 to individual_size
      let random_gene be 0 or 1 randomly
      append random_gene to current_individual
    append current_individual to population
  return population
```

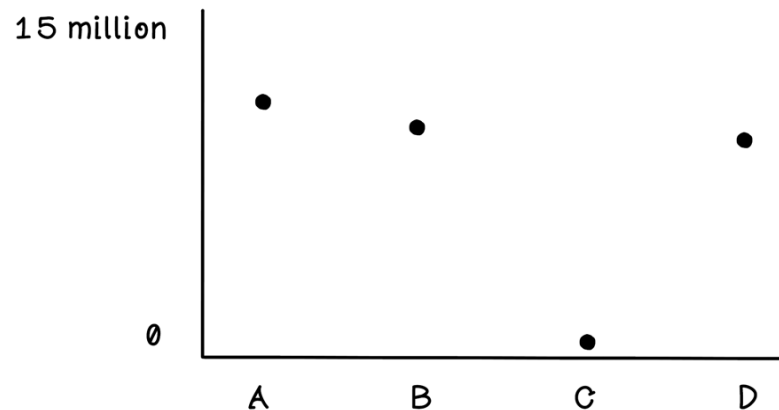
Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

4 – Medir la aptitud de los individuos de una población (1)

- Hay que evaluar la aptitud de cada individuo que forma parte de la población.
- Proceso crítico, si no se realiza adecuadamente, es decir, con el objetivo de buscar la solución óptima:
 - La selección de padres para la creación de nuevas generaciones se verá influenciada.
 - El algoritmo será defectuoso y no alcanzará la mejor solución.
- Similar al concepto de heurística, son solamente guías, no verdades absolutas.
- Dependiendo del problema esta función puede intentar minimizar o maximizar.

4 – Medir la aptitud de los individuos de una población (2)

A	<table><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	0	1	1	0	0	1	0	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	1	0	0	1	11,393,360
0	1	1	0	0	1	0	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	1	0	0	1			
B	<table><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	1	1	0	1	0	1	0	0	0	1	0	0	0	1	1	0	1	0	0	0	1	0	0	0	10,866,684
0	0	1	1	0	1	0	1	0	0	0	1	0	0	0	1	1	0	1	0	0	0	1	0	0	0			
C	<table><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr></table>	1	1	1	0	0	1	0	0	1	1	0	1	1	0	0	0	0	1	0	0	1	0	1	1	0	1	0 (Overweight)
1	1	1	0	0	1	0	0	1	1	0	1	1	0	0	0	0	1	0	0	1	0	1	1	0	1			
D	<table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	0	0	0	0	1	0	0	0	0	1	0	0	1	0	1	0	1	1	0	1	0	0	1	0	0	1	10,715,475
0	0	0	0	1	0	0	0	0	1	0	0	1	0	1	0	1	1	0	1	0	0	1	0	0	1			



4 – Medir la aptitud de los individuos de una población (3)

```
calculate_individual_fitness (individual,  
                             knapsack_items,  
                             knapsack_max_weight)  
  
let total_weight equal 0  
let total_value equal 0  
for gene_index in range 0 to length of individual  
    let current_bit equal individual[gene_index]  
    if current_bit equals 1  
        add weight of knapsack_items[gene_index] to total_weight  
        add value of knapsack_items[gene_index] to total_value  
if total_weight is greater than knapsack_max_weight  
    return value as 0 since it exceeds the weight constraint  
return total_value as individual fitness
```

5 – Seleccionar padres según su aptitud (1)

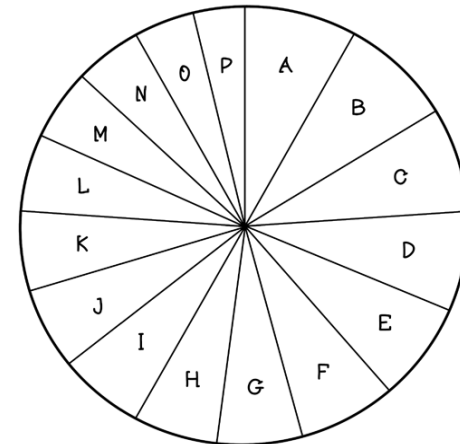
- Basándose en las teorías de Darwin ...
- No obstante, algunos individuos también se pueden reproducir, aunque en global su aptitud no es muy buena, puede que algunos cromosomas sí lo sean.
- Cada individuo tiene su aptitud calculada y en base a ella se calcula la **probabilidad** de ser seleccionado para ser padre.
 - Aquí reside la naturaleza estocástica del algoritmo.
- Existen varias técnicas para esta selección.

5 – Seleccionar padres según su aptitud (2)

- Selección de la ruleta.
- Estado estable.
- Generacional.

5 – Seleccionar padres según su aptitud (3)

A	1 0 1 1 1 0 0 1 1 0 1 1 0 1 0 0 0 1 1 0 0 1 0 0 0 1	13,107,019
B	1 1 0 0 0 1 0 0 1 1 1 1 1 1 1 1 0 1 0 0 0 1 0 0 0	12,965,145
C	0 0 1 1 0 1 1 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1 1 0 0 0	12,344,873
D	0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 0 0 0 0	11,739,363
E	1 1 0 0 0 1 0 0 1 1 1 1 1 1 1 0 1 1 0 1 0 0 0 1 0 0 0	11,711,159
F	1 1 0 0 0 1 0 0 1 1 1 1 0 1 0 1 1 0 1 0 0 0 1 0 0 0	11,611,967
G	1 0 1 0 0 1 1 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0	10,042,441
H	1 1 0 0 0 1 0 0 1 1 1 1 1 1 1 0 1 1 0 1 0 0 0 0 0 0	9,883,682
I	1 1 0 0 0 1 0 0 1 1 1 1 1 1 1 0 0 1 0 1 0 0 0 1 0 0 0	9,857,597
J	0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 1	9,670,184
K	0 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 0	9,277,580
L	1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 1 0 1 0 0	8,931,719
M	0 1 0 0 0 0 0 1 1 1 0 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0	8,324,936
N	1 1 1 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 0	8,018,760
O	0 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1	6,900,314
P	0 0 0 1 1 0 0 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 1 0 0 0	6,056,664



5 – Seleccionar padres según su aptitud (4)

```
set_probabilities_of_population (population)
    let total_fitness equal the sum of fitness of the population
    for individual in population
        let the probability_of_selection of individual...
            ...equal it's fitness/total_fitness

roulette_wheel_selection(population, number_of_selections):
    let possible_probabilities equal
        set_probabilities_of_population (population)
    let slices equal empty array
    let total equal 0
    for i in range(0, number_of_selections):
        append [i, total, total + possible_probabilities[i]]
            to slices
        total += possible_probabilities[i]
    let spin equal random(0, 1)
    let result equal [slice for slice in slices if slice[1] < spin <= slice[2]]
    return result
```

6 – Reproducir individuos a partir de los padres (1)

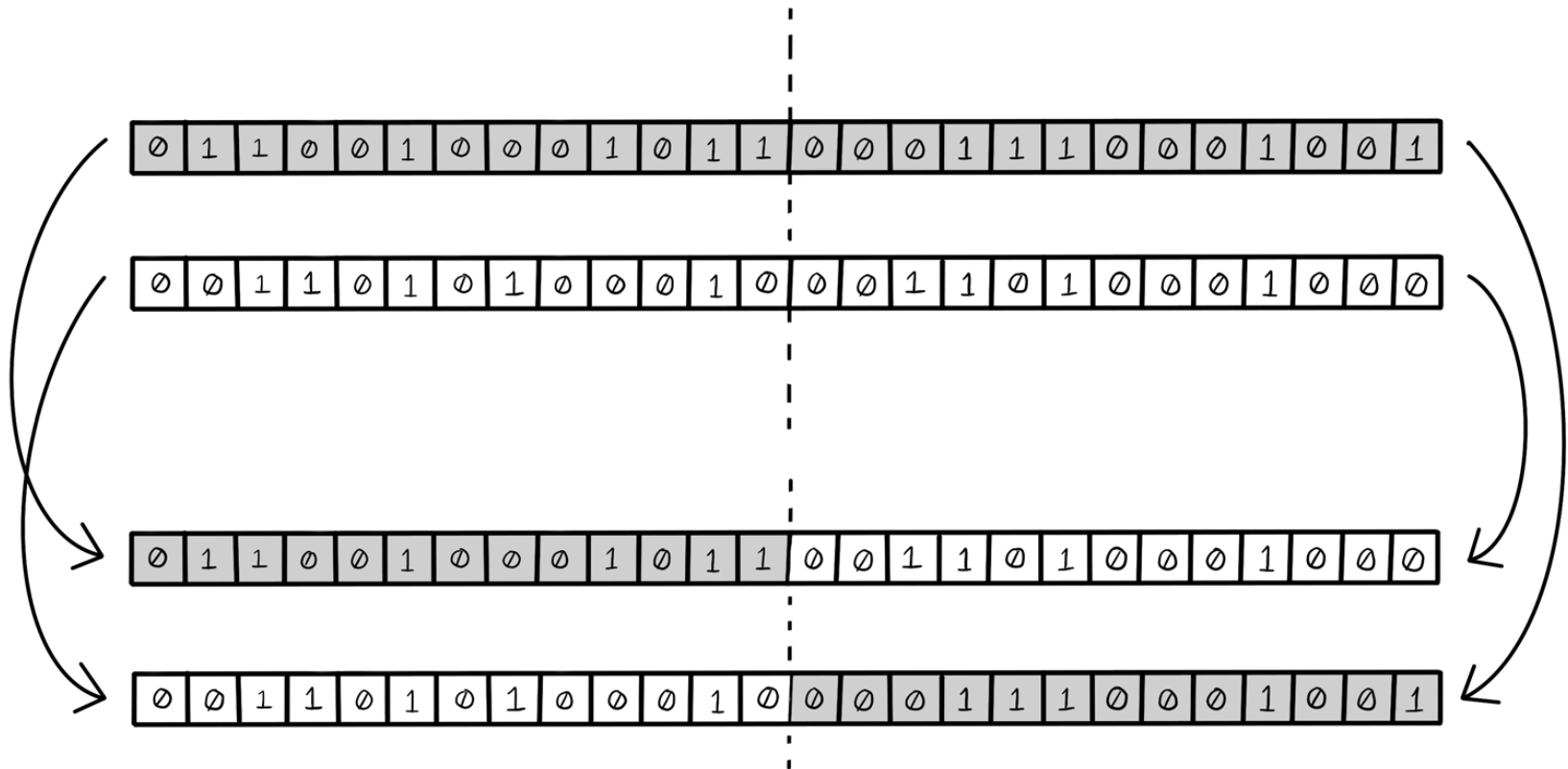
- Crear descendientes.
- Recombinaciones y mutaciones.
 - Recombinaciones: Partes de un cromosoma del primer padre y partes de un cromosoma del segundo padre.
 - Muy dependiente de la codificación empleada.
 - Mutaciones: Cambiar el descendiente de forma aleatoria para que haya variedad en la población.
- Los descendientes son el resultado de estos procesos.

6 – Reproducir individuos a partir de los padres (2)

Recombinaciones

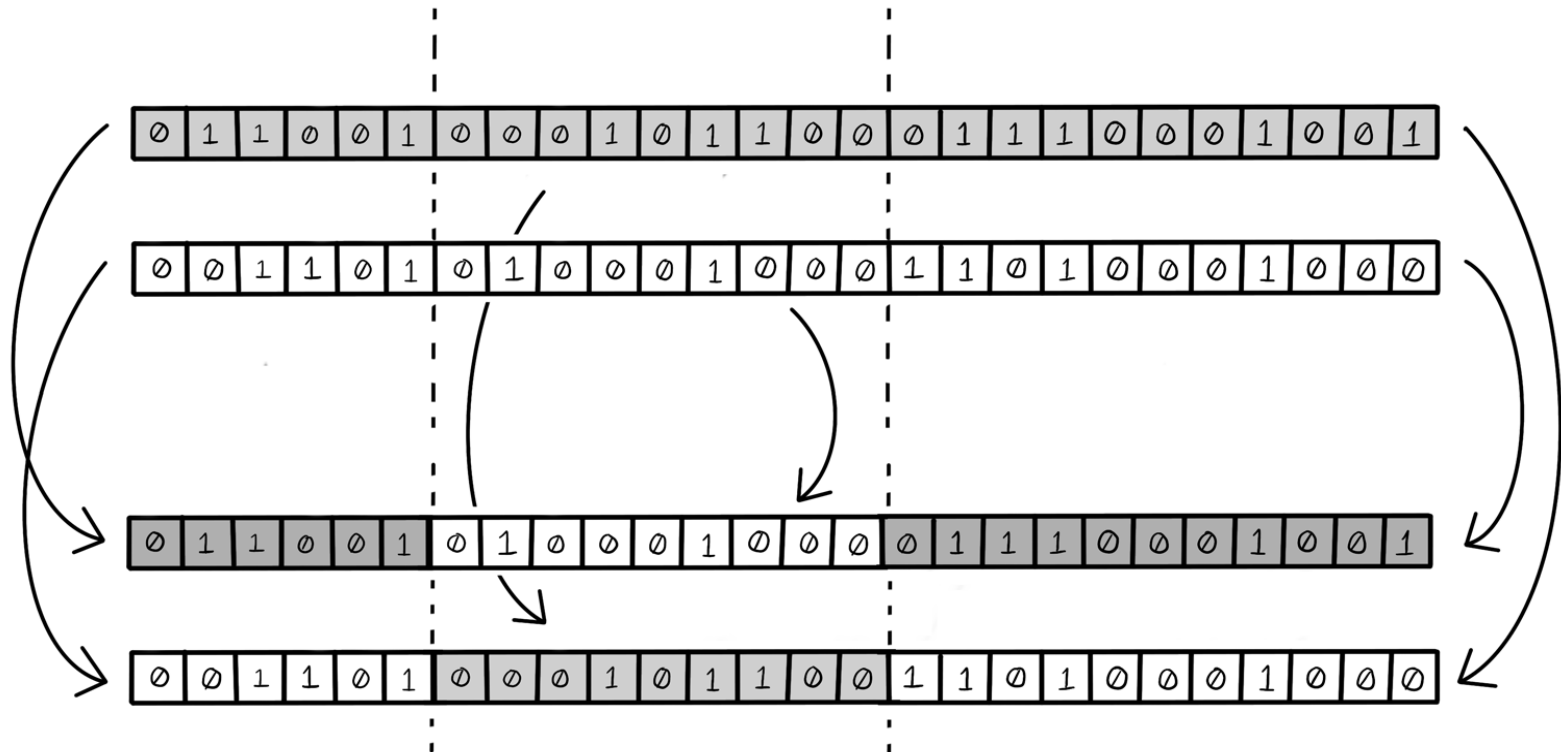
- Recombinaciones en un punto.
 - Aplicable a codificación binaria, codificación de permutación y codificación de valores reales.
- Recombinación en dos puntos.
 - Aplicable a codificación binaria y codificación de valores reales.
- Recombinación uniforme.
 - Aplicable a codificación binaria y codificación de valores reales.

Recombinación en un punto



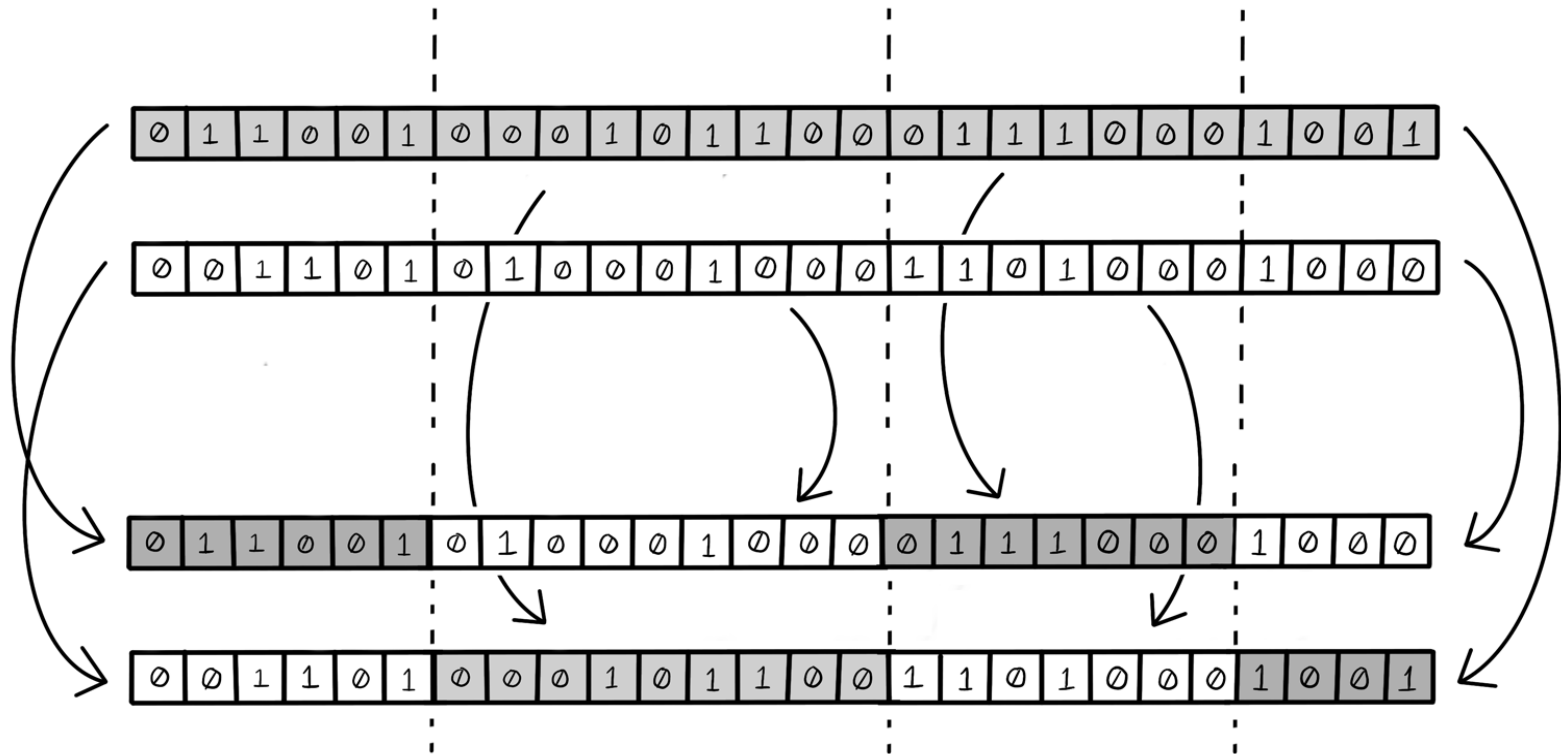
Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

Recombinación en dos puntos



Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

Recombinación uniforme



Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

6 – Reproducir individuos a partir de los padres (3)

```
one_point_crossover (parent_a, parent_b, xover_point)
  let children equal empty array

  let child_1 equal genes 0 to xover_point from parent_a plus...
  ...genes xover_point to parent_b length from parent_b
  append child_1 to children

  let child_2 equal genes 0 to xover_point from parent_b plus...
  ...genes xover_point to parent_a length from parent_a
  append child_2 to children

  return children
```

Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

6 – Reproducir individuos a partir de los padres (4)

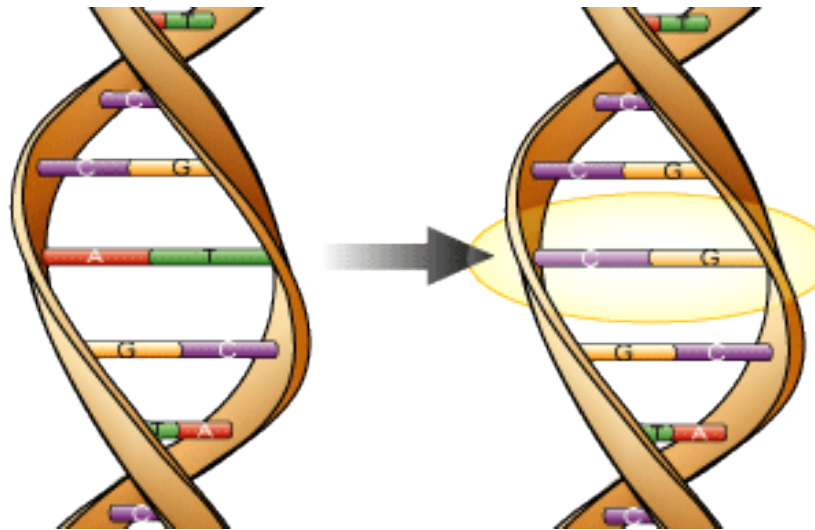
Mutaciones

- Ratio de mutación.
 - Similar a los seres vivos.
 - No es exactamente una combinación de los genes de sus progenitores.
 - Contiene ligeras diferencias.
 - Promueve la diversidad y previene que el algoritmo se quede atascado.
 - Un alto ratio de mutación puede provocar demasiada diversidad y que el algoritmo no sea bueno.

6 – Reproducir individuos a partir de los padres (5)

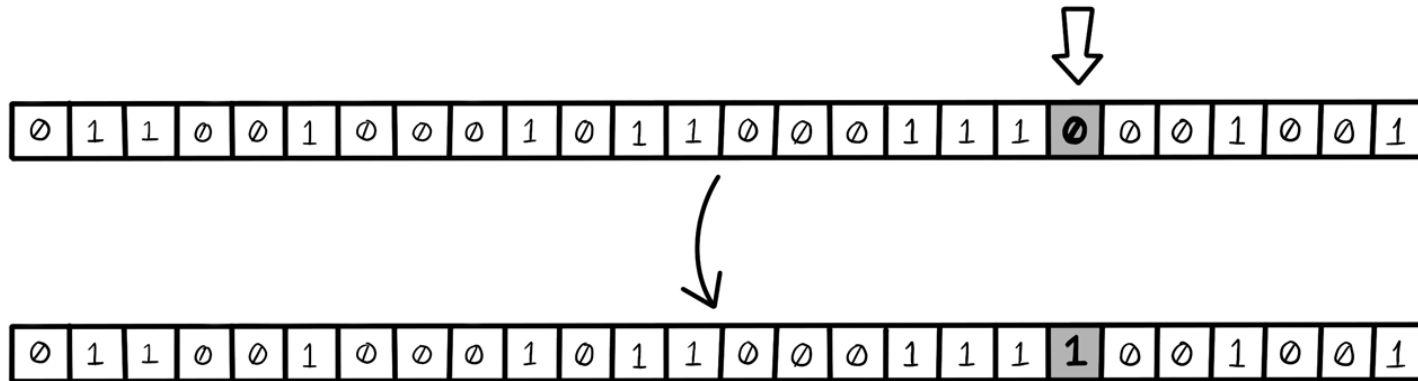
Mutaciones

- Mutación bit-string para codificación binaria.
- Mutación flip-bit para codificación binaria.



Mutación bit-string para codificación binaria

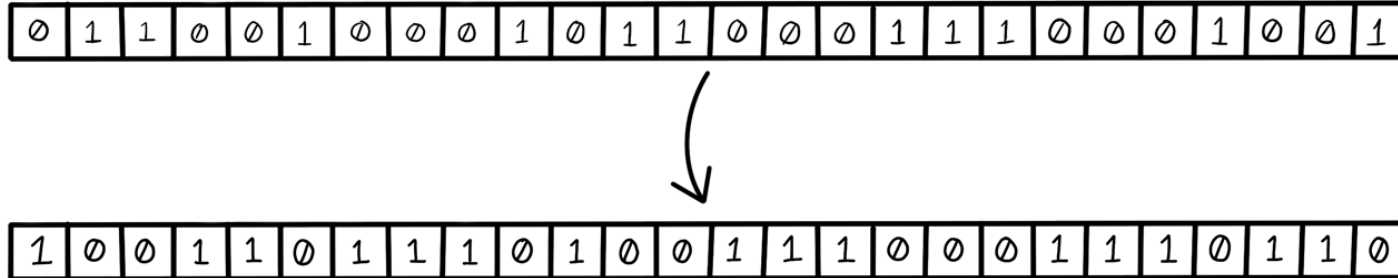
- Un cromosoma es cambiado por otro valor válido.



Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

Mutación flip-bit para codificación binaria

- Todos los cromosomas son invertidos.



Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

6 – Reproducir individuos a partir de los padres (6)

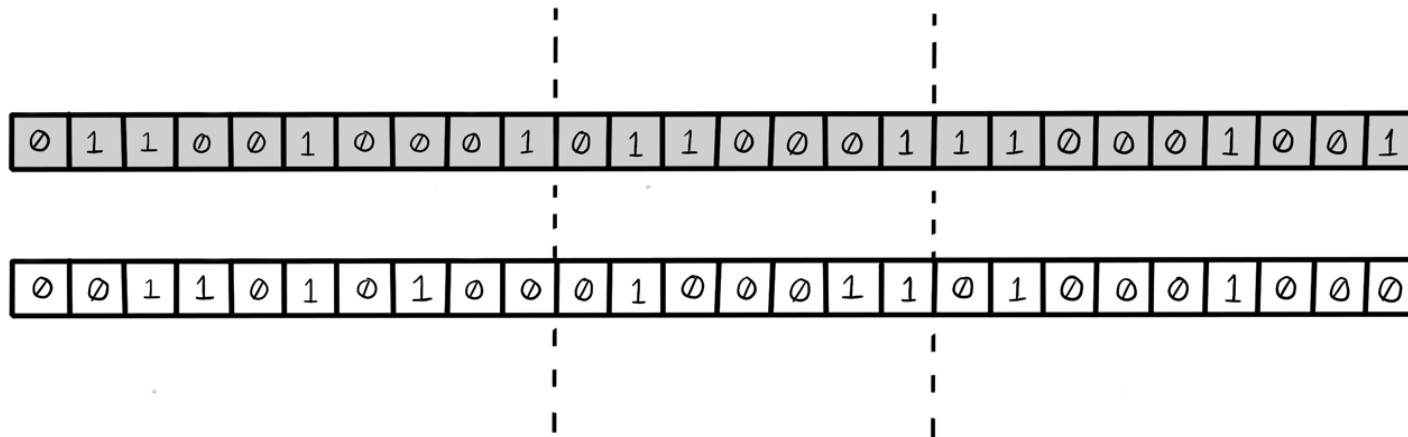
Mutaciones

```
mutate_individual (individual, chromosome_length)
    let random_index equal a random number between 0 and chromosome_length
    if gene at index random_index of individual is equal to 1:
        let gene at index random_index of individual equal 0
    else:
        let gene at index random_index of individual equal 1
    return individual
```

Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

Entregable 2.6

Realiza la recombinación uniforme de los siguientes cromosomas.



[Fecha límite: **08/11/2020 12:00**]

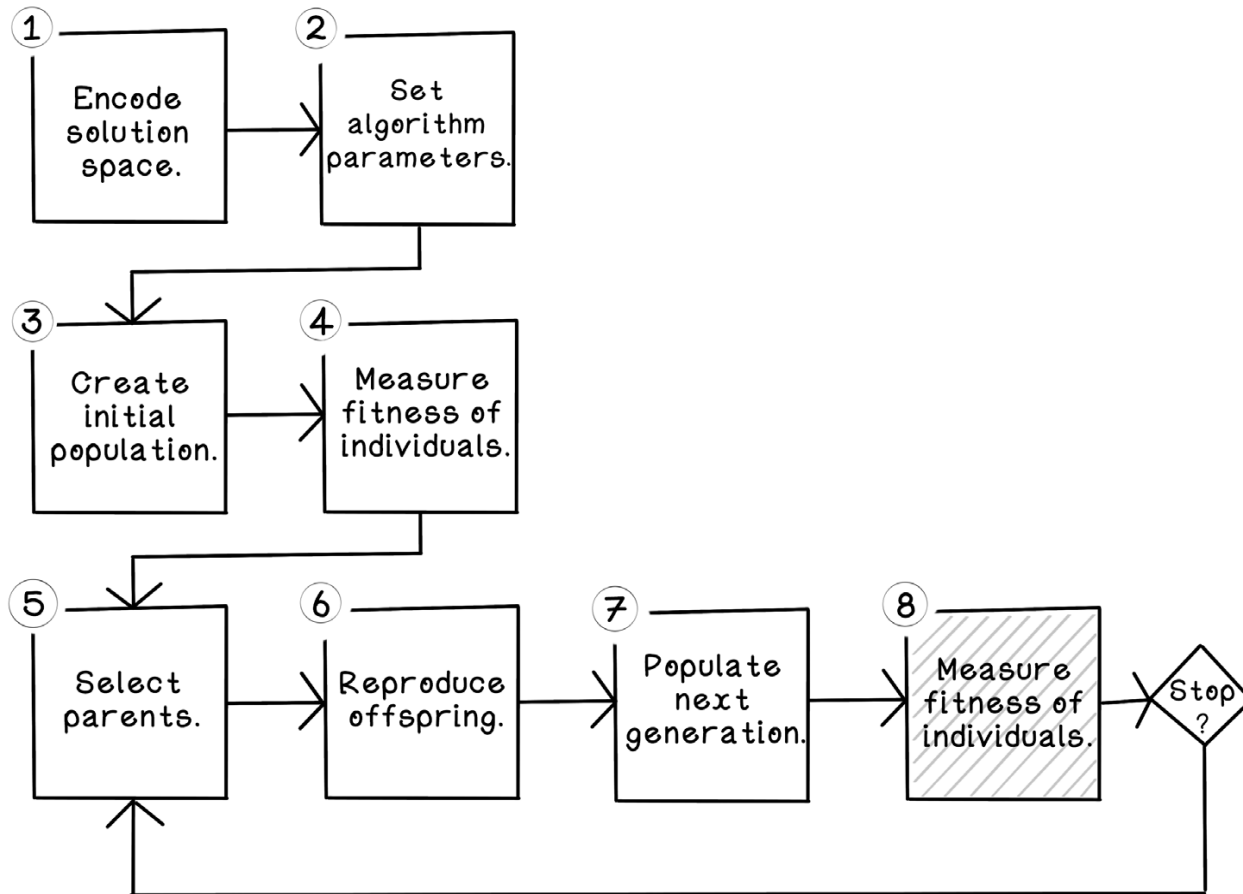
7 – Poblar la siguiente generación (1)

- ¿Aptitud de individuos?
- ¿Descendientes?
- ¿Quién permanece en la siguiente generación?
 - N^o fijo
 - ¿Solo los más idóneos?

7 – Poblar la siguiente generación (2)

- Exploración vs explotación.
 - Equilibrio.
 - Situación ideal:
 - Diversidad.
 - Exprimir las soluciones posibles.
 - De este modo el algoritmo explora el espacio de búsqueda tanto como es posible y además permite la evolución.

8 – ...



Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

¿Cuándo parar? (1)

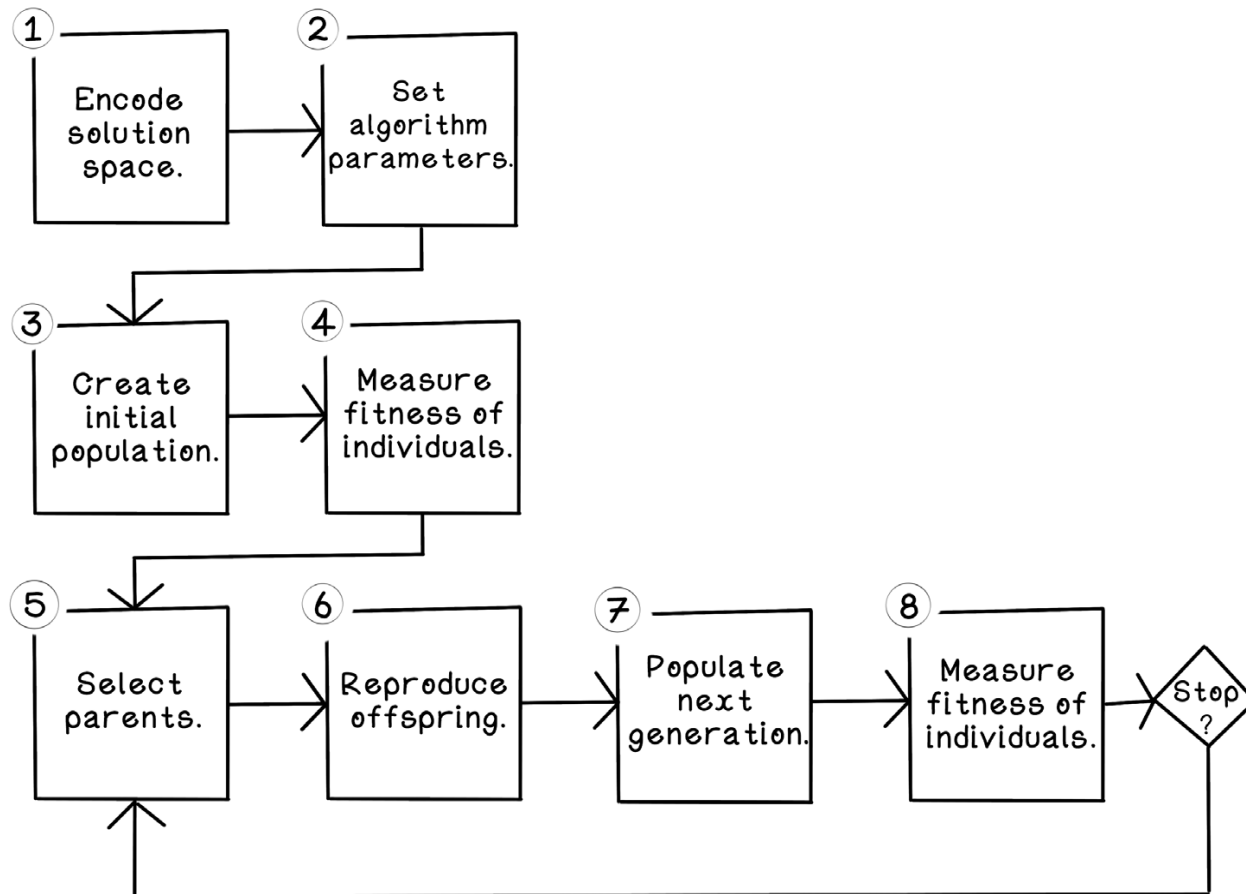
- Proceso iterativo buscando mejores soluciones cada generación.
- Establecer condición de parada o será un bucle sin fin.
 - Constante.
 - Alcanzar cierta aptitud.
 - Estancación.

¿Cuándo parar? (2)

```
run_ga (population_size, number_of_generations, knapsack_capacity):  
    let best_global_fitness equal 0  
    let global_population equal...  
    ...generate_initial_population(population_size)  
    for generation in range(number_of_generations):  
        let current_best_fitness equal...  
        ...calculate_population_fitness(global_population, knapsack_capacity)  
        if current_best_fitness is greater than best_global_fitness:  
            let best_global_fitness equal current_best_fitness  
        let the_chosen equal...  
        ...roulette_wheel_selection(global_population, population_size)  
        let the_children equal...  
        ...reproduce_children(the_chosen)  
        let the_children equal...  
        ...mutate_children(the_children)  
        let global_population equal...  
        ...merge_population_and_children(global_population, the_children)
```

Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

2 – Configurar parámetros del algoritmo (1)



2 – Configurar parámetros del algoritmo (1)

- Influyen en el rendimiento del algoritmo.
- Rendimiento:
 - Buenas soluciones.
 - Computacionalmente eficiente.
- Factores clave en el rendimiento:
 - Codificación.
 - Función de aptitud.
 - Parámetros del algoritmo.

2 – Configurar parámetros del algoritmo (2)

1. Codificación del cromosoma.
2. Tamaño de la población.
3. Inicialización de la población.
4. Número de descendientes.
5. Método de selección de padres.
6. Método de reproducción.
7. Método de mutación.
8. Método de selección de generación.
9. Condición de parada.

Fuerza bruta VS Algoritmo genético

	Brute force	Genetic algorithm
Iterations	$2^{26} = 67,108,864$	10,000 - 100,000
Accuracy	100%	100%
Compute time	~7 minutes	~3 seconds
Best value	13,692,887	13,692,887

Hurbans, R. (2020). Grokking Artificial Intelligence Algorithms. Manning Publications.

Casos de uso de los algoritmos genéticos

- Predecir comportamiento del mercado financiero.
- Selección de características en Aprendizaje Automático.
- Cifrado.

Bibliografía

Esta presentación se basa principalmente en información recogida en las siguientes fuentes:

- Hurbans, R. (2020). *Grokking Artificial Intelligence Algorithms*. Manning Publications.
- Russell, S. & Norvig, P. (2010). *Artificial Intelligence: A modern approach*. 3ª Ed. Prentice-Hall.