

Diplomado de especialización de desarrollo de aplicaciones con Inteligencia Artificial

Algoritmos Genéticos Parte I



Dra. Soledad Espezua. LI.
sespezua@pucp.edu.pe



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

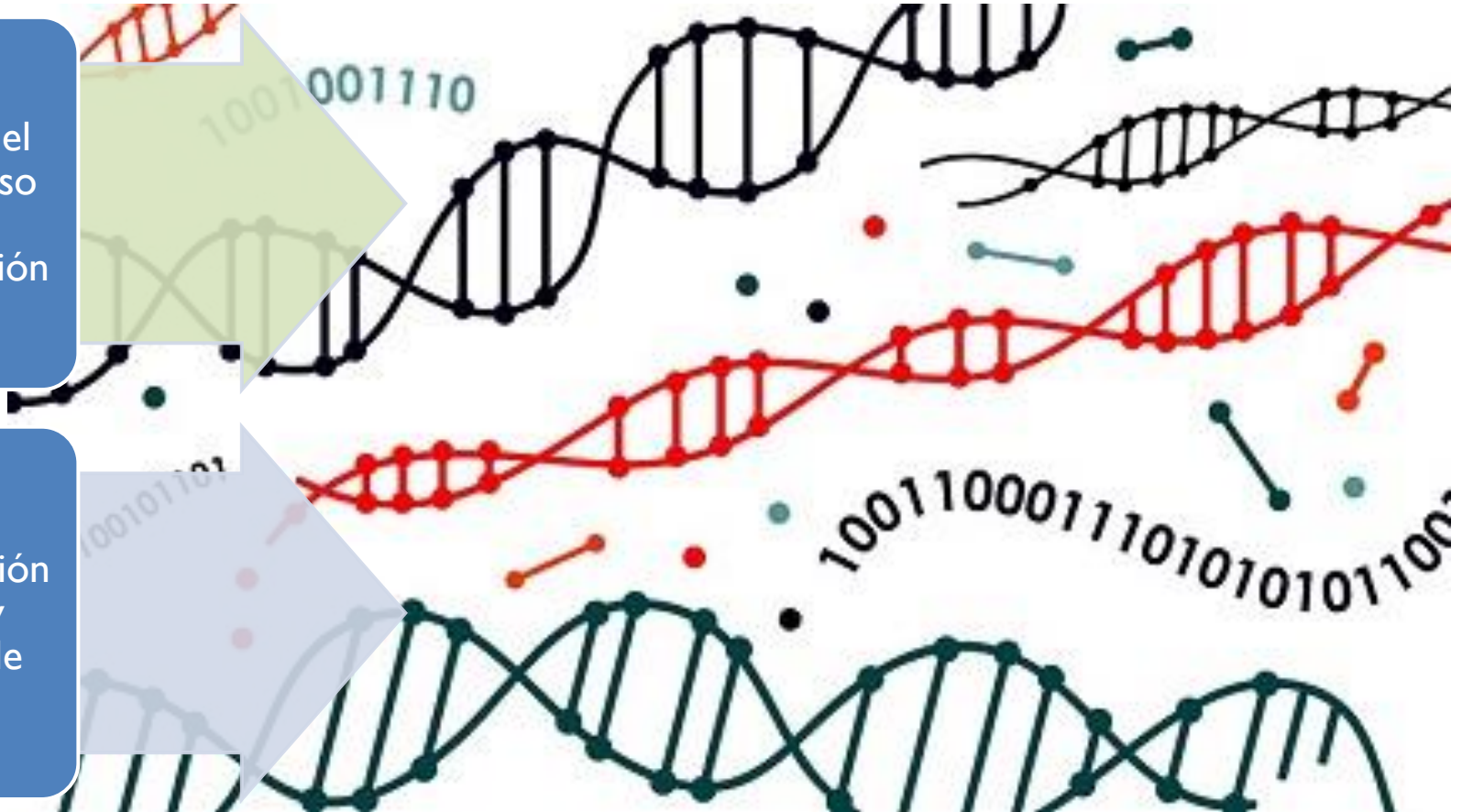


ia.inf.pucp.edu.pe

Algoritmos Genéticos

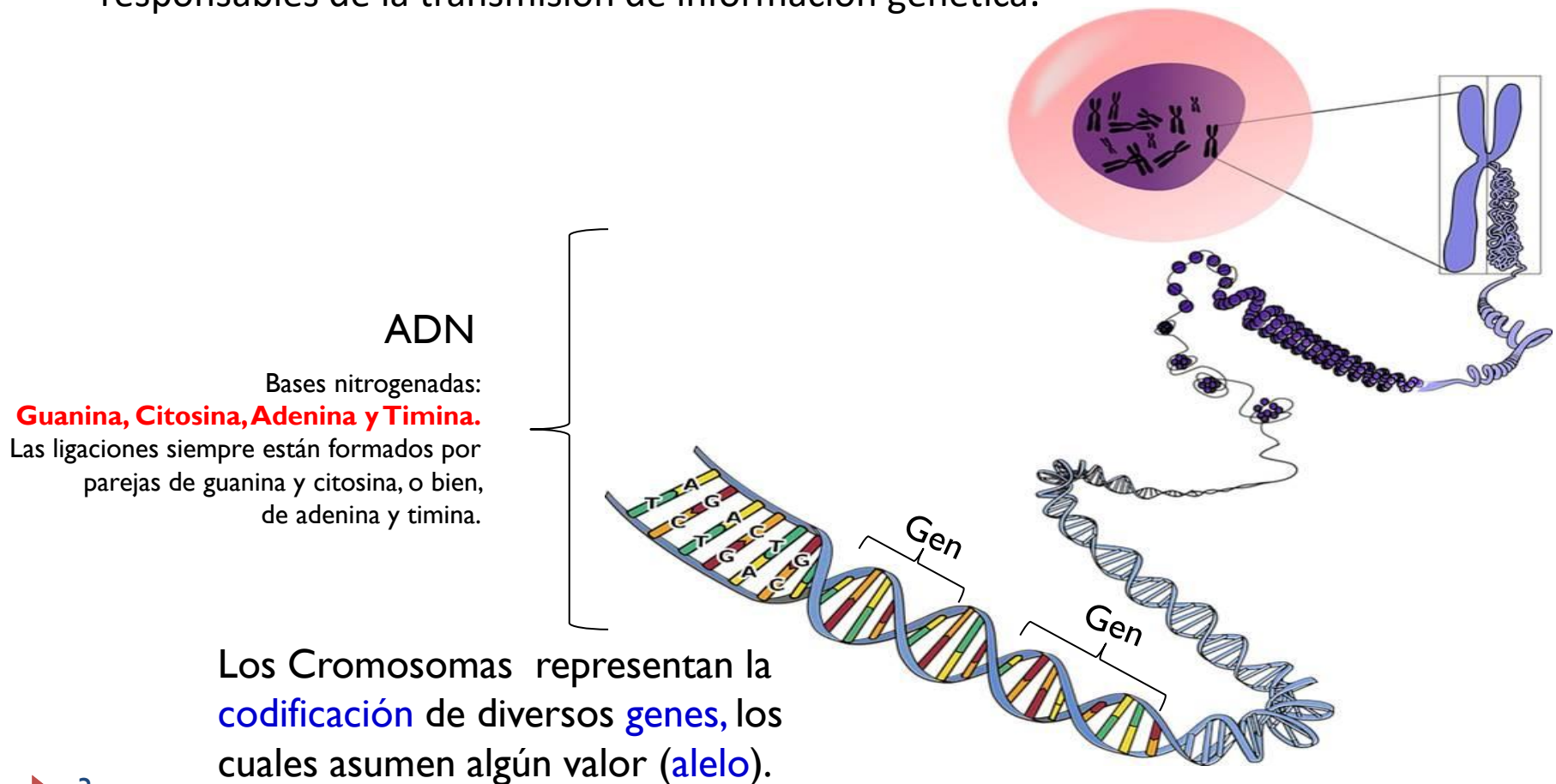
Imita el
proceso
de
evolución

Evolución
muy
simple



Conceptos Biológicos

- ▶ **Cromosomas:** estructuras químicas compuestas de proteínas y cadenas de ADN que se encuentra en el núcleo de las células. Los cromosomas son responsables de la transmisión de información genética.



Conceptos Biológicos

- ▶ **Genotipo:** es el conjunto de todos los genes de un individuo. Es decir, es lo que potencialmente puede llegar a ser un individuo.
 - El genotipo después del desarrollo fetal, da origen al fenotipo del organismo.



Genotipo



Fenotipo

determinan el tipo de rasgos que podrán observarse

- ▶ **Fenotipo:** representa el individuo formado a partir de las informaciones del genotipo. Un fenotipo son los rasgos (observables) específicos de un individuo.



AG - historia

Padre del AG



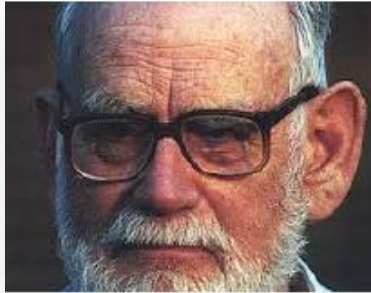
- ▶ John Henry Holland³ (1929-2015), formaliza los principios básicos de los “Algoritmos genéticos”, 1975.

1957

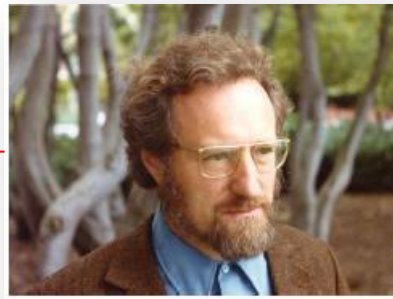
1962

1975

1989



- ▶ Alex S. Fraser¹ (1923-2002)
Evolucion de sist.
Biologicos, 1957.



- ▶ Hans J. Bremermann² (1926-1996)
Evolución como un proceso de
optimización, 1962.



- ▶ David E. Goldberg⁴,
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*.

¿Cómo funciona un Algoritmo genético?



<https://www.youtube.com/watch?v=ziMHaGQJuSI&t=60s>
<http://boxcar2d.com/index.html>

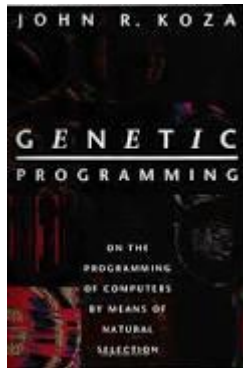


AG - definición

John Koza¹:

“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.”



John Koza

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

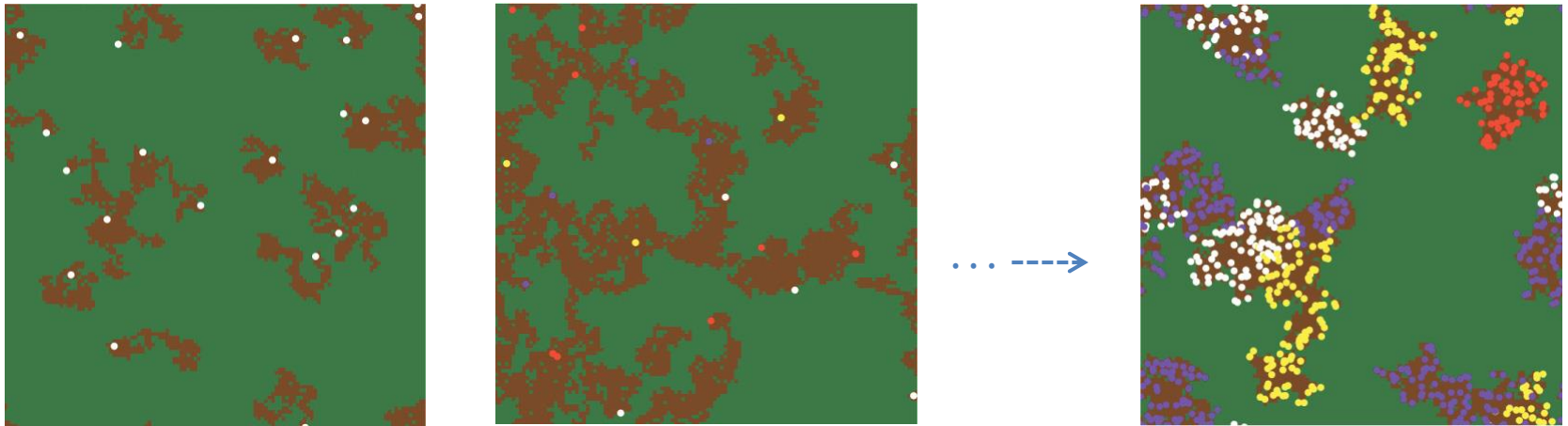
AG- Seudocódigo clásico

- ▶ El algoritmo básico es el siguiente:

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

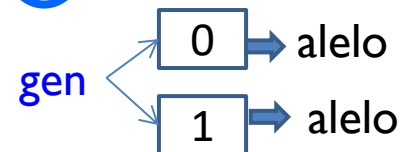
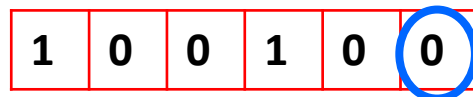

AG - Terminología

- **Generación**: creación de una nueva población por medio de operadores de reproducción en una iteración.



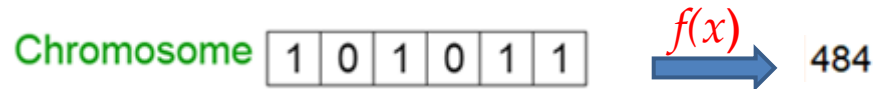
- **Cromosoma**: representa la información que determina a cada **individuo** (una posible solución). Los cromosomas están formados por **genes**, los cuales pueden tomar diferentes valores en sus **alelos**.

individuo = cromosoma



AG - Terminología

- **Fitness**: Valor que recibe cada cromosoma.



La función de aptitud asocia a un vector un valor real de evaluación

$$f = R^n \longrightarrow R$$

- **Población**: Conjunto de cromosomas (individuos) en cada generación.

Population

A1	0	0	0	0	0	0
A2	1	1	1	1	1	1
A3	1	0	1	0	1	1
A4	1	1	0	1	1	0

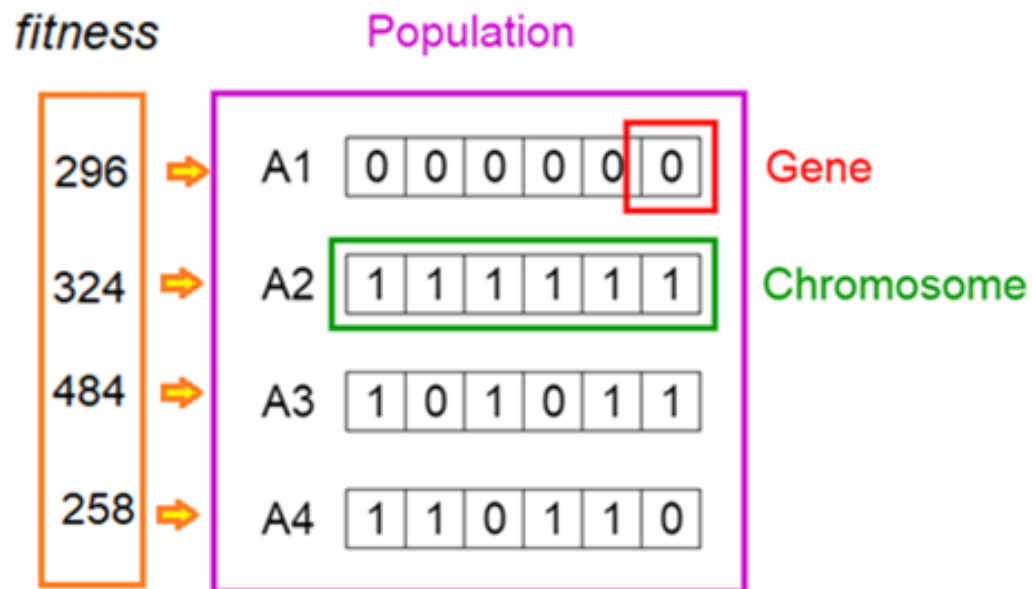
AG - Terminología

- **Función de aptitud (evaluación):** Este es el paso más costoso para una aplicación real.
 - Puede ser una subrutina, un simulador, o algún proceso externo (ej. experimentos,)
 - Se pueden utilizar funciones aproximadas para reducir el costo de evaluación.
 - Cuando hay restricciones, éstas se pueden introducir en el costo como penalización.

Eje. Los valores fitness de la siguiente función de aptitud: $f(x) = x^2$, para la siguiente población seria:

Index	Población inicial	x Value	Fitness $f(x) = x^2$
X_1	0 1 1 0 1	13	169
X_2	1 1 0 0 0	24	567
X_3	0 1 0 0 0	8	64
X_4	1 0 0 1 1	19	361

AG - Terminología



AG - Terminología

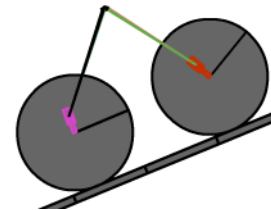
Genotipo Num. real: $\langle 0, 2 \rangle$

Car 1	0.05	0.10	0.76	...	2.00
Car 2	0.41	0.77	1.00	...	2.00
Car 3	0.81	0.35	0.28	...	2.00
	:				:
	:				:
Car n	0.32	0.33	0.8	...	2.00



Fenotipo

Car 1



Car 2

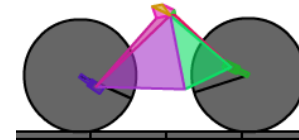


.

.

.

Car n



AG - Componentes

1. Codificar las estructuras que se replicaran
 - Una representación de las soluciones potenciales del problema.
 - Una forma de crear una población inicial de posibles soluciones (normalmente un proceso aleatorio).

2. Usar operadores que afecten a los “individuos”
 - Operadores genéticos que alteren la composición de los hijos que se producirán para las siguientes generaciones.



AG - Componentes

3. Crear una función de aptitud

- Una función de aptitud (evaluación) que realice el papel del ambiente, clasificando las soluciones en términos de su “aptitud”.

4. Generar un mecanismo de selección.

- Con valores para los diferentes parámetros que utiliza el AG (tamaño de la población, probabilidad de cruce, probabilidad de mutación, número máximo de generaciones, etc.)



Características del AG

Características del AG

1. Constituyen el **paradigma más completo de la CE** ya que combinan de modo natural todas las ideas fundamentales de dicho enfoque.
2. No tienen inteligencia. No aprenden.
3. Buscan soluciones (casi optimas) intentando obtener un costo computacional razonable, aunque **sin garantizar optimalidad** o factibilidad de las mismas.
 - En algunos casos, ni siquiera puede determinar qué tan cerca del optimo se encuentra una solución factible en particular.

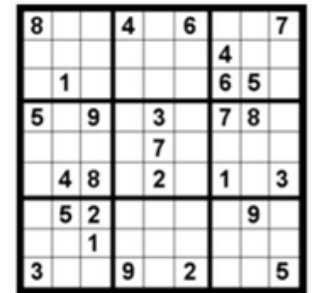
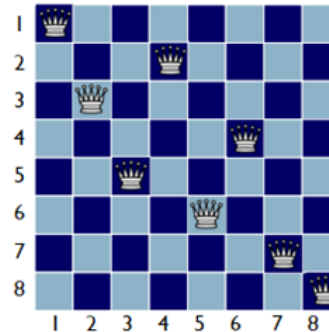
Características del AG

- Encuentran soluciones a problemas que los humanos tendrían dificultades para resolver o no podrían resolver en absoluto.

► Optimización combinatoria



búsqueda de camino óptimo (TSP)



- asignación de horarios
- planificación de tareas

Tarea: duración de la tarea, número de tareas y fecha límite de tareas

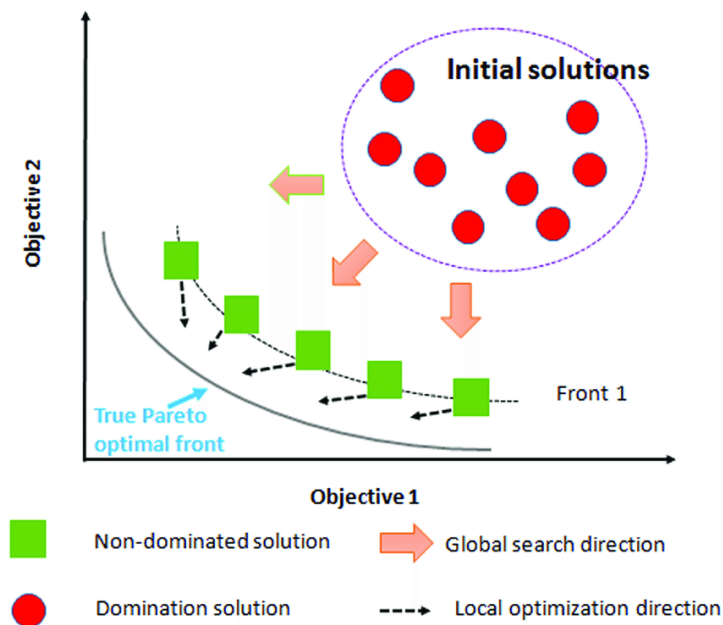
Recursos: capacidad de procesamiento de nodos, potencia de procesamiento, memoria, etc.



tráfico aéreo y ordenación secuencial

Características del AG

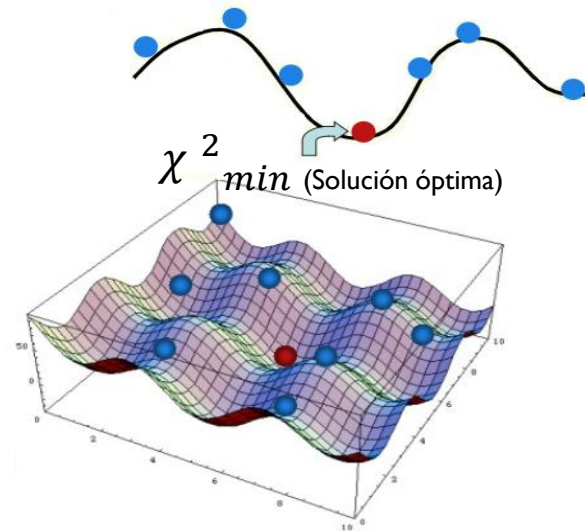
4. Los AG destacan por conseguir manipular muchas restricciones simultáneamente y alcanzar **múltiples objetivos**.



Problemas que no pueden definirse sólo en términos de minimizar o maximizar un valor, sino que deben expresarse en términos de **múltiples objetivos** (problema de la mochila, asesor de inversiones), a menudo involucrando contrapartidas, es decir: **uno sólo puede mejorar a expensas de otro** (dilema del prisionero).

Características del AG

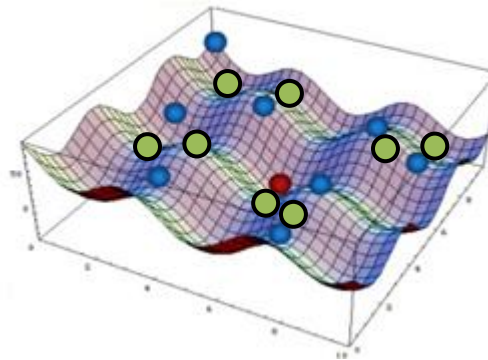
5. Son algoritmos de **búsqueda estocástica** que distribuyen su población uniformemente sobre toda la superficie.



- Analizan en paralelo todo el espacio de soluciones.
- La aplicación más común de los AG ha sido la solución de **problemas de optimización (max, min)**, en donde han mostrado ser muy eficientes y confiables.

Características del AG

6. A pesar de estocásticos (aleatorios), no funcionan únicamente con este concepto. La búsqueda en AG tiene un pseudo-proceso de **búsqueda guiada** (operador de cruzamiento: hijos nacen cerca de los padres)



- Se intenta dar pasos direccionados, explorando informaciones genéticas para encontrar nuevos puntos de búsqueda, donde se espera mejores desempeños.

Características del AG

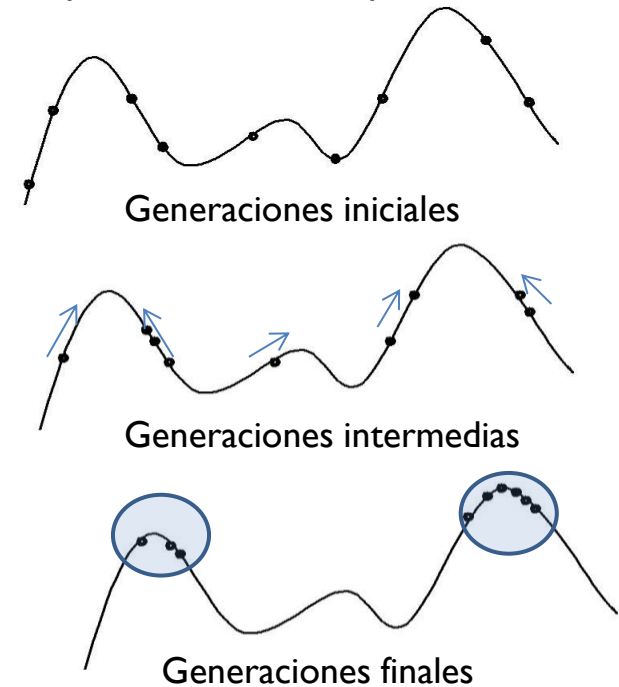
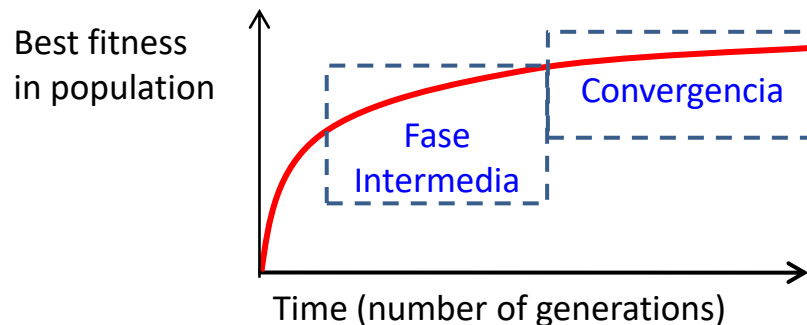
7. Se utilizan reglas de transición probabilísticas y no reglas determinísticas.



Características del AG

8. Generan registros en diversas regiones del espacio de búsqueda 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



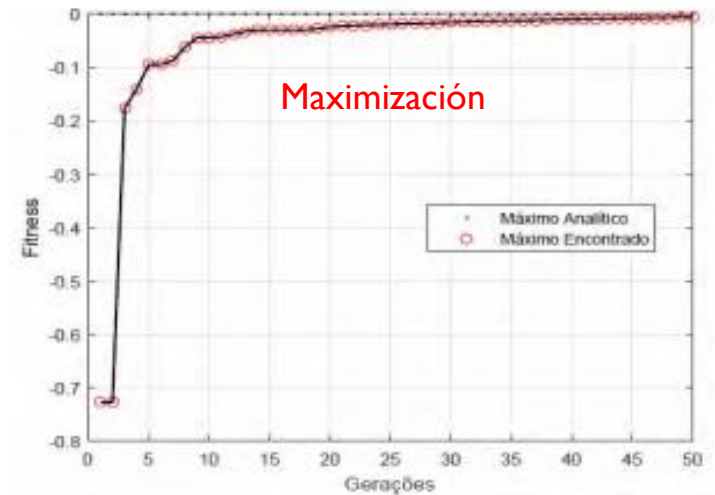
Características del AG

- Registros de las diversas evolutivas

```
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]
```

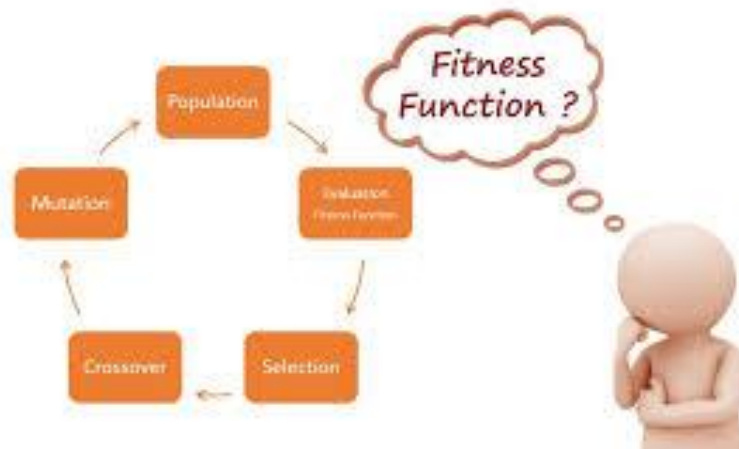
Desempeño

Mejor fitness
en la población



Características del AG

- 9. Se pueden **hibridar** fácilmente con otros paradigmas (RN, Log. difusa).
- 10. La principal característica de los AG es su **Flexibilidad** para adaptarse a problemas generales o específicos.



Características del AG

Original layout



The original elementary school. Found somewhere in Maine.

“Optimized” floor plan with genetic algorithms by Joel Simon

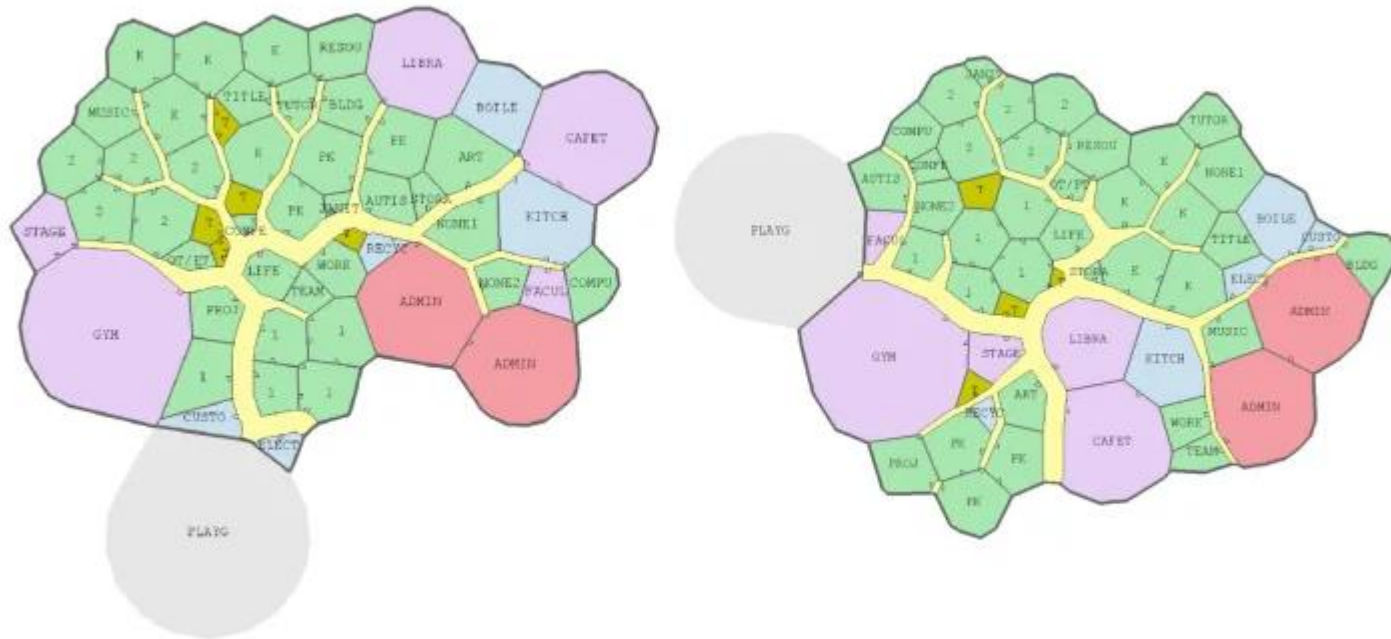
http://www.joelsimon.net/evo_floorplans.html

https://github.com/joel-simon/evo_floorplans



Características del AG

'Optimized'



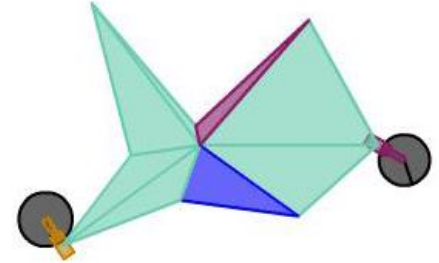
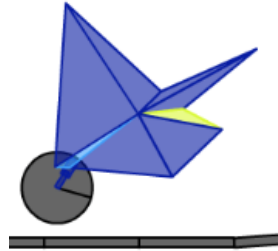
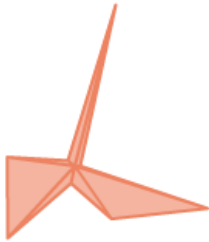
Left: Optimized for minimizing traffic flow between classes and material usage. Right: Also optimized for minimizing fire escape paths.

“Optimized” floor plan with genetic algorithms by Joel Simon

http://www.joelsimon.net/evo_floorplans.html

https://github.com/joel-simon/evo_floorplans





Un AG, solo será tan bueno para resolver un problema como la persona que escribe el código.



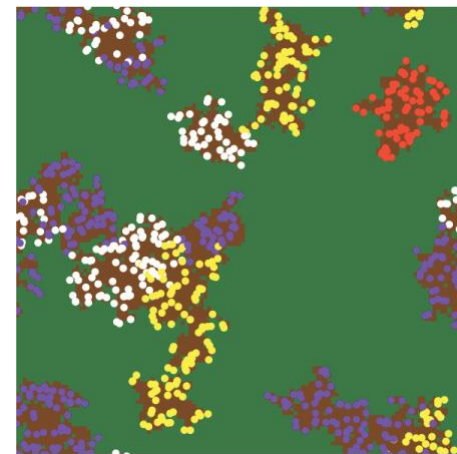
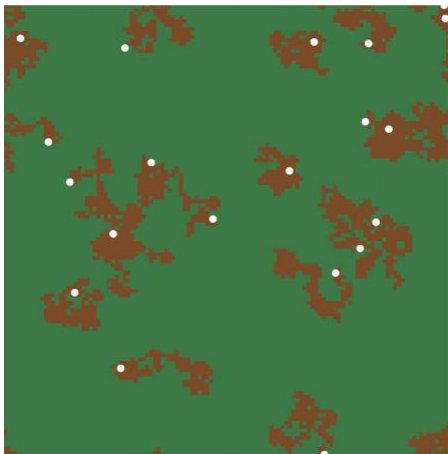
Etapas del AG

Población



Población

- ▶ La población es un conjunto de cromosomas, por lo general de tamaño fijo.
- ▶ Poblaciones pequeñas corren el riesgo de **no cubrir todo el espacio** de búsqueda.
- ▶ Poblaciones muy grandes acarrear mucho **costo computacional**.
 - ❖ Todo ambiente soporta un número limitado de individuos.



Inicialización

La **inicialización** generalmente es aleatoria, pero puede incluir soluciones existentes o encontradas por otro método.

Tipos:

- ▶ **Inicialización randomica uniforme:**

Cada gen del cromosoma recibirá como valor un elemento del conjunto de alelos, sorteado de forma aleatoriamente uniforme;

- ▶ **Inicialización randomica no uniforme:**

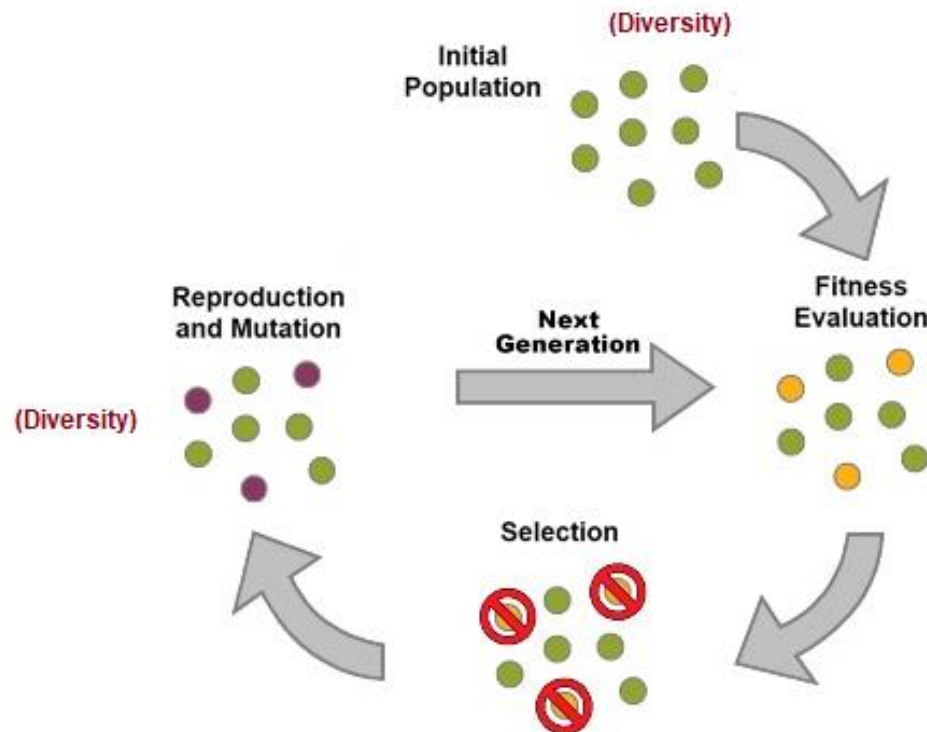
Determinados valores que serán almacenados en el gen tenderán a ser escogidos con una frecuencia mayor que el resto. Por ejemplo, cuando se usa un operador (cruzamiento o mutación) con una distribución de probabilidad diferente de 0.5

- ▶ **Inicialización randomica con “dope”:**

Algunos individuos optimizados son ingresados en la 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.

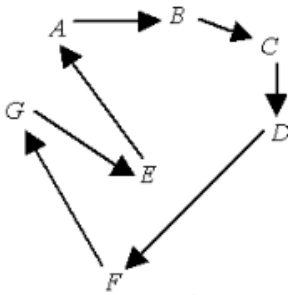
Diversidad

- ▶ La **diversidad** de una población es el número de diferentes cromosomas en una población.
 - Diversidad es fundamental para la búsqueda en todo el espacio; poca diversidad está asociada al fenómeno de convergencia prematura.



Representación

- Existen diversas formas de **representar** un cromosoma, se ha de elegir la más relevante para el problema en cuestión:
 - 1) Cadenas de Bits (0101... 1100)
 - 2) Números Reales (43.2 -33.1... 0.0 89.2)
 - 3) Permutaciones de caracteres($E_{11} E_3 E_7 \dots E_1 E_{15}$), etc

Genotipo	Fenotipo	Problema
0010101001110101	10869	optimización numérica
CGDEHABF	 <p>comience por la ciudad C, después pase por las ciudades G, D, E, H, A, B y termine en F</p>	cajero viajante (TSP)
$C_1 R_4 C_2 R_6 C_4 R_1$	si condición 1 (C_1) ejecute regla 4 (R_4), si (C_2) ejecute (R_6), si (C_4) ejecute (R_1)	reglas de aprendizaje para agentes



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).

Cromosoma

	Población inicial (fenotipos)	x valor genotipo	$f(x)$ valor (función adaptación)
1	01101	13	169
2	11000	24	576
3	01000	8	64
4	10011	19	361

$$f(x) = x^2$$



Ej. de Representación Binaria

Problema de un
robot en un
laberinto

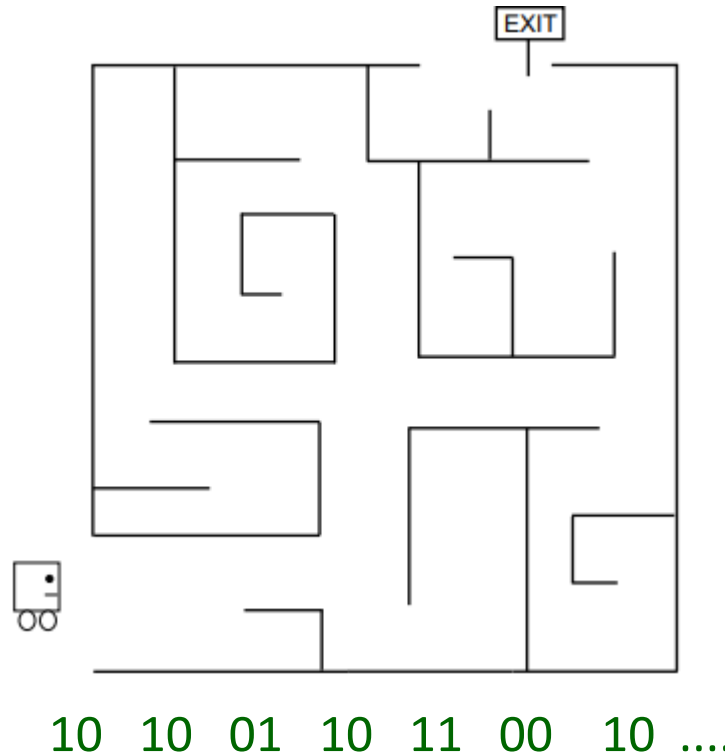
Codificación:

00 = Retrocede;

01= Izquierda;

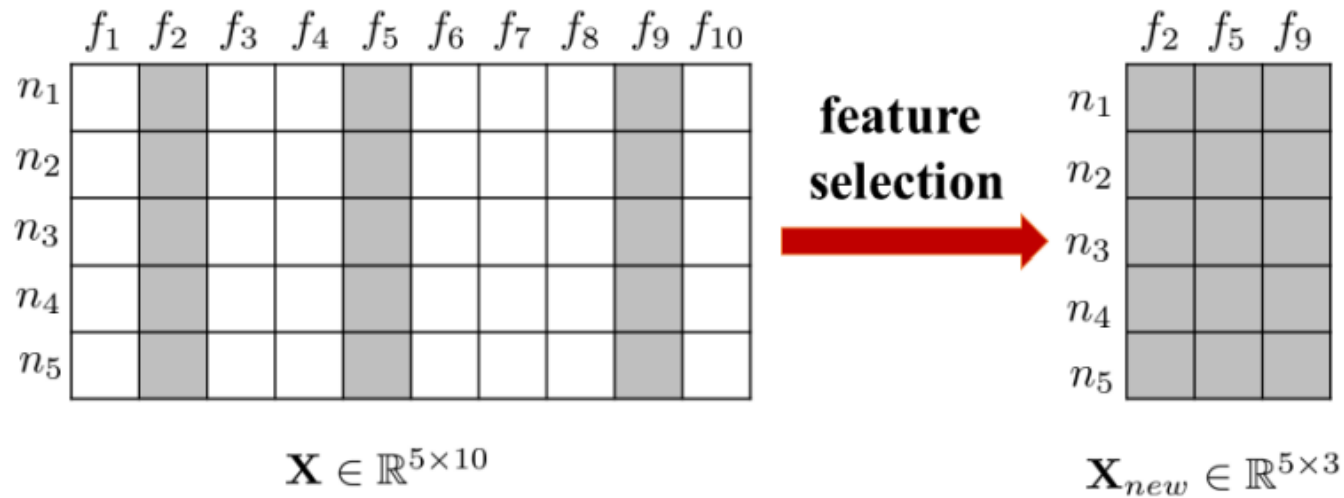
10= Avanza;

11=Derecha



Ej. de Representación Binaria

Feature selection



Ej. de Representación Binaria

Feature selection

Gen: 1 (feature incluido) o
0 (feature excluido).

feature
included



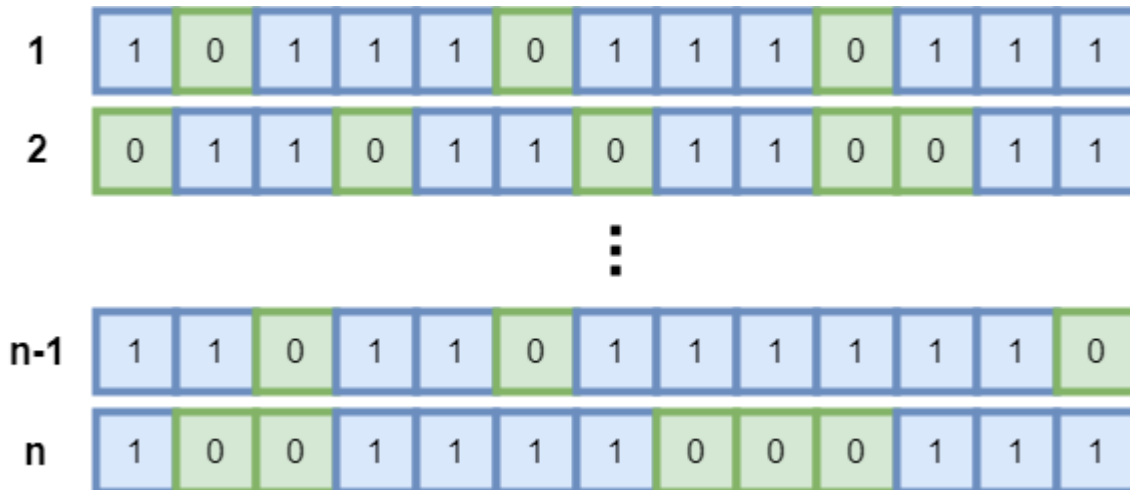
feature
excluded



Cromosoma



Población



Ejemplo: representación binaria



2. Punto flotante

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
- Realmente estaríamos operando a nivel fenotípico.
- Esta representación es útil en problemas con muchas variables, donde se necesita una muy buena precisión para cada una de ellas.

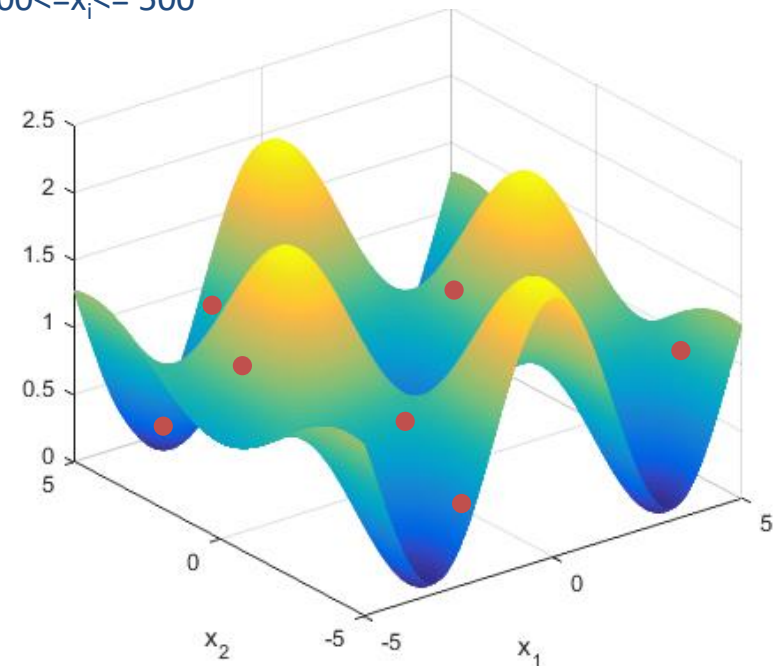


Ej. de Representación en punto flotante

Problemas de optimización no lineal (valores continuos).

Minimizar la siguiente función: $f(\mathbf{x}) = f(x_1, \dots, x_n) = 1 + \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right)$
Sujeto a: $-500 \leq x_i \leq 500$

	x_1	x_2	x_3
Cromosoma 1	0,1	0,8	0,9
Cromosoma 2	0,3	0,2	0,3
⋮			
Cromosoma n-1	0,2	0,5	0,6
Cromosoma n	0,2	0,5	0,6



Griewangk's function in 3D

Ejemplo: representación en punto flotante



3. Representación de enteros

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

$$1.345 = \begin{array}{|c|c|c|c|} \hline 1 & 3 & 4 & 5 \\ \hline \end{array}$$

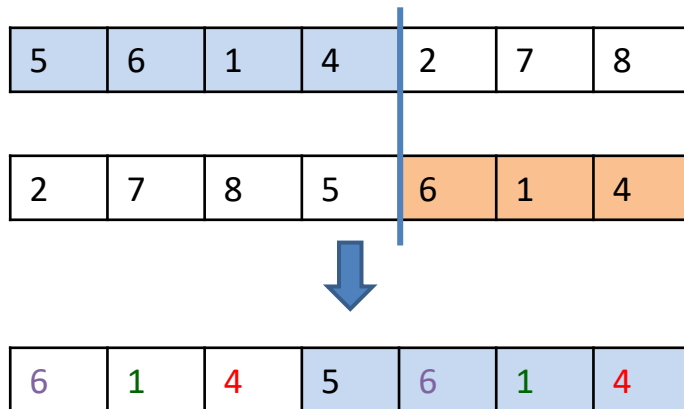
- ▶ Esta representación es útil en problemas donde la tarea es organizar algunos objetos manteniendo un cierto orden/secuencia, como en el caso de la representación entera tipo permutación.



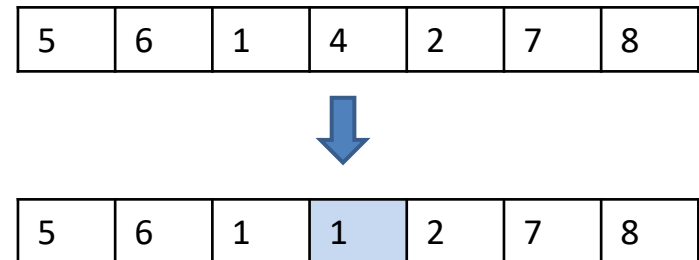
3. Representación de enteros

❖ Representación de Permutación

- Una permutación se representa como una lista de n enteros, cada uno de los valores ocurre exactamente una vez.
- Se necesita tener cuidado con los operadores para garantizar que el resultado continúe siendo una permutación.
 - ▶ Alterar un alelo puede implicar repetir un valor.
 - ▶ Operaciones de cruzamiento y mutación requieren un tratamiento especial.



Cruzamiento



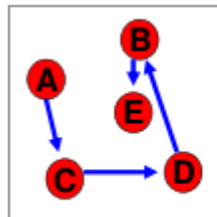
Mutación

Ej. De permutaciones

Problema del viajero

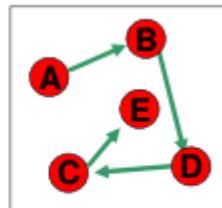
- ▶ Objetivo: Encontrar un recorrido completo que visite n ciudades en una distancia mínima.
- ▶ Representación:
 - Un vector de n enteros para representar un recorrido completo (permutación con orden).
 - Cada posición en el vector corresponde a una ciudad a visitar.

Por ej. para $n=4$, dos posibles recorridos serian:



Cromosomas:

A	C	D	B	E
---	---	---	---	---



Cromosomas:

A	B	D	C	E
---	---	---	---	---

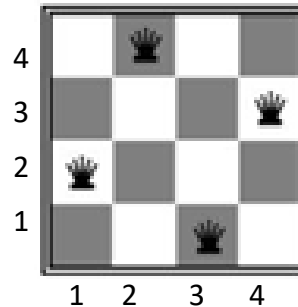


- ▶ Espacio de búsqueda es muy grande:
para 30 ciudades hay $30! \approx 10^{32}$ posibles tours

Ej. De permutaciones

Problema de las N reinas:

- ▶ Objetivo: Situar 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:

- Usar un vector de N enteros, donde las posiciones y valores del vector representan respectivamente las columnas y filas en la que se encuentran las reinas.

1	2	3	4
2	4	1	3

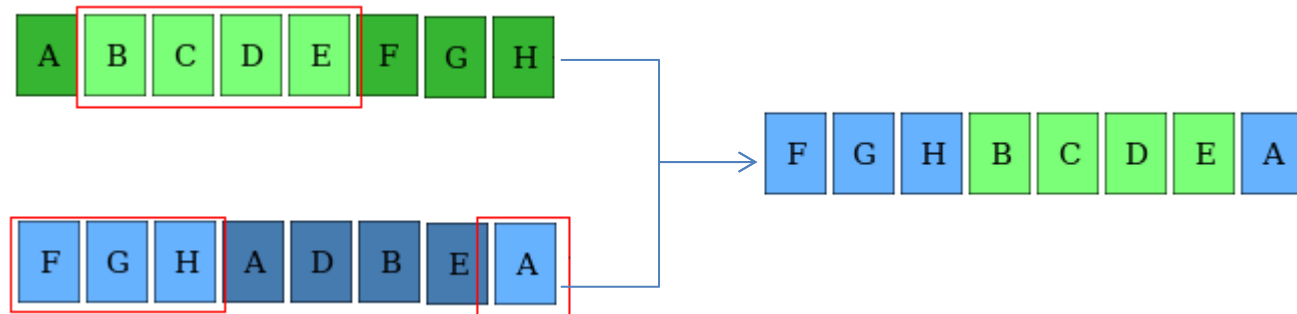
Fitness:

Numero de pares de reinas que no se atacan \therefore Máximo $fitness = \frac{N}{2} * (N - 1)$

Ejemplo: representación en permutaciones

4. Representación tipo *string*

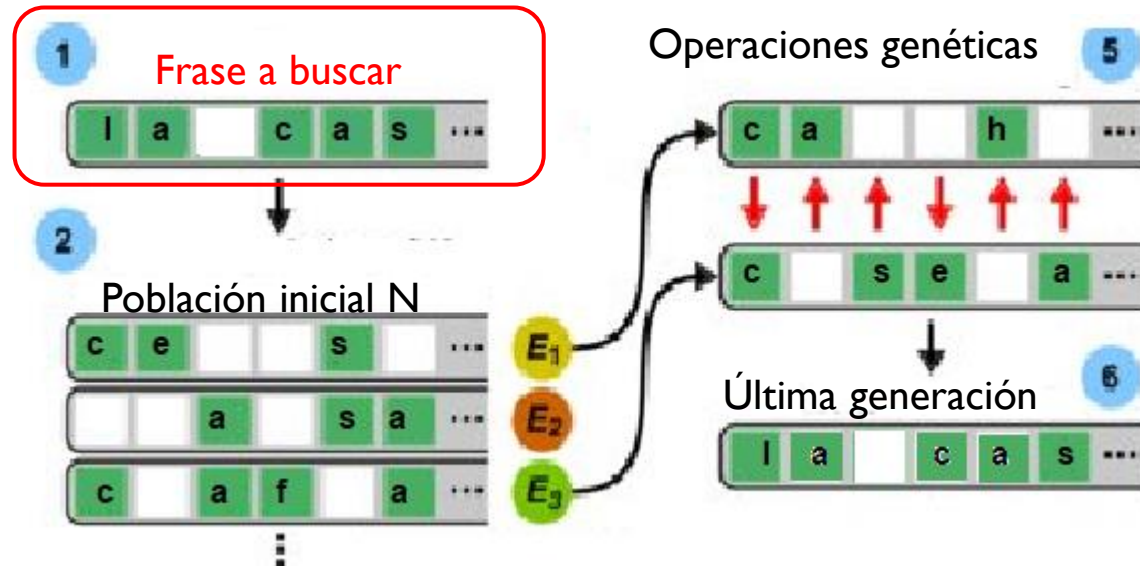
- Es un subtipo de la codificación binaria, donde la población resulta de cadenas del alfabeto.
- El nro de combinaciones posibles de una cadena, depende de la cantidad de letras del alfabeto que se use para generar la población.
- Ejm. Cadena de 8 caracteres en un alfabeto de 20 caracteres ,necesitaría 20^8 posibles combinaciones.



4. Representación tipo *string*

Problema de Generación de frases

- Objetivo: Buscar generar una frase a partir de una población de cadenas aleatorias.
 - Si se quisiera generar una frase de 8 caracteres de longitud. Cada carácter puede tomar 1 de 53 caracteres posibles (26 minúsculas, 26 mayúsculas y un espacio en blanco).
 - La cantidad de frases posibles de 53^8 caracteres, lo que haría inviable la búsqueda por fuerza bruta.
- Representación:



Ejemplo: representación en string



Diferentes formas de representar un mismo problema

► Por ejemplo:

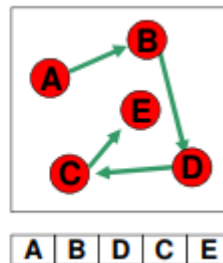
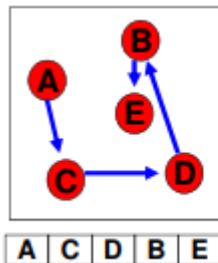
- TSP de 6 ciudades, representado por sub-cadenas de 3 bits.

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 (123456) y (243156)

Diferentes formas de representar un mismo problema

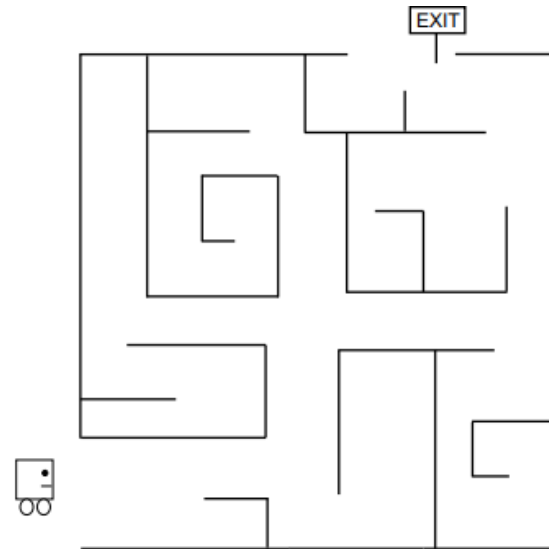
► Robot en el laberinto

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

Preguntas

