

# Tema-4.-Sistemas-Bio-Inspirados.pdf



**Anónimo**



**Sistemas Inteligentes**



**4º Grado en Ingeniería de Computadores**



**Facultad de Informática  
Universidad Complutense de Madrid**

Formamos  
**talento** para un futuro  
**Sostenible**



MÁSTER EN

**Big Data &  
Business Analytics**

**EOI** Escuela de  
organización  
industrial

[saber más](#)

Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato  
→ Planes pro: más coins

#### TEMA 4: SISTEMAS BIO-INSPIRADOS.

Tradicionalmente, los humanos hemos intentado imitar el comportamiento de ciertos individuos o procesos naturales para el avance tecnológico.

##### Computación evolutiva.

La computación evolutiva es una rama de la inteligencia artificial que busca resolver problemas de optimización, inspirándose en los mecanismos de evolución biológica.

Existen diferentes métodos y algoritmos que se pueden englobar dentro de este campo como los algoritmos genéticos, las estrategias evolutivas, etc.

##### Definiciones.

- Gen: la unidad mínima de transferencia genética.
- El genotipo: conjunto de todos los genes de un individuo.
- El fenotipo es la expresión del genotipo en un individuo concreto (la implementación de ese genotipo).

Ejemplo: hay un gen que codifica el color de los ojos de un individuo. Su genotipo sería la secuencia de bases del gen en concreto y su fenotipo el color que genera al final.

##### Algoritmos genéticos.

Son algoritmos evolutivos que se basan en los principios explicados de la selección natural. Estos algoritmos pretenden resolver problemas de optimización y búsqueda representando la solución en forma de genes. Estos se cruzarán y mutarán para conseguir individuos (soluciones) cada vez mejores.

En los algoritmos genéticos es muy importante la codificación del problema en individuos. Los algoritmos genéticos deben codificar las diferentes soluciones como un vector o matriz de genes. A cada vector de genes se le denomina individuo.

Una población es el conjunto de soluciones (individuos) que disponemos en un momento dado. Inicialmente esta población se puede generar de forma aleatoria o con algo más de conocimiento del dominio. A cada uno de estos individuos se le aplica una serie de operadores genéticos.

##### Operadores genéticos.

- Selección: se selecciona a los mejores candidatos para la reproducción. Se genera una población intermedia del mismo tamaño que la original normalmente.
- Cruce: se seleccionan de 2 en 2 los individuos de esa población intermedia y se cruzan de alguna forma.
- Mutación: se modifica de forma aleatoria mediante una probabilidad de mutación los genes de los individuos recién cruzados.
- Inversión: se invierten alguno de los genes.

##### La función de Fitness.

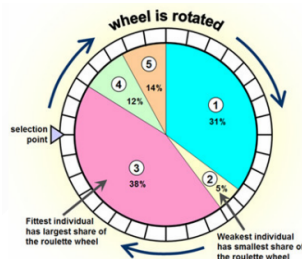
Esta función evalúa las soluciones para determinar cuál es el más apto, es decir, los que mejor resuelven el problema.

El cruce funciona como un algoritmo voraz y hace que muchas veces se realice una convergencia prematura en un mínimo local. Esto es debido a que ciertos rasgos genéticos se pierden (Se pierde variabilidad genética). La mutación permite explorar más allá del mínimo local y mantener esa variabilidad genética.

Aparte de la mutación hay multitud de técnicas que permiten modular la variabilidad genética.

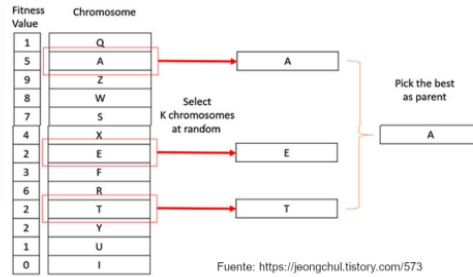
##### Tipos de selección.

- Ruleta: individuos con mejor fitness tendrán más probabilidad de que sean elegidos



- Selección jerárquica: se ordena la población de mayor a menor según su fitness y se selecciona el individuo en proporción al rango que ocupa calculando una probabilidad.

- En la selección por torneos, se eligen K elementos de la población y se selecciona aquel que tenga el



mejor fitness. K modula la presión selectiva.

### Algoritmo.



### Elitismo.

Los algoritmos genéticos con elitismo tienen algún mecanismo que permite proporcionar a la siguiente generación a las N mejores individuos sin hacer una selección de los mismos.

Suele funcionar bien, pero el número N de individuos a promocionar normalmente debe ser bajo para mantener la variabilidad genética. También es un parámetro que podemos modificar de forma online durante la ejecución del algoritmo.

### Ejemplo

Vamos a ver un ejemplo de cómo codificar el problema del viajero usando algoritmos genéticos.

Queremos codificar 4 ciudades, podemos utilizar 2 bits.

Codificación	id
00	1
01	2
10	3
11	4

Si usamos ruleta decimos que la probabilidad de ser elegido será mas alta para el individuo 2 y 4. Supongamos que se eligen ambos para el cruce. Se cruzan con cruce simple.

$11011001 \times 01010101 \Rightarrow 11010101$

Se muta con una probabilidad  $11010101 \Rightarrow 11010111$   
 Se repite hasta tener 4 nuevos individuos.  
 Se repite el proceso hasta que no se mejore más.

Pensemos en un individuo que va de la 1 a la 3 de esta a la 2 y de esta al 4. Se podría codificar así:

$1 \Rightarrow 3 \Rightarrow 2 \Rightarrow 4 = 00100111$

#### Generamos una población aleatoria

$p = \{ 10101010, 11011001, 00010101, 01010101 \}$

La función de fitness será la distancia recorrida.

$10101010 \Rightarrow 3$   
 $11011001 \Rightarrow 2$   
 $00010101 \Rightarrow 4$   
 $01010101 \Rightarrow 2$

#### Problema de codificación

Esta codificación tiene un problema y es que permite visitar la misma ciudad dos veces, lo que haría individuos no válidos.

















$01011000$  equivale A:  $2 \Rightarrow 2 \Rightarrow 3 \Rightarrow 0$

#### ¿Cómo lo solucionamos?

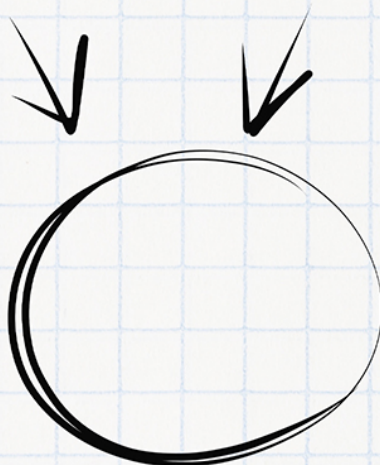


# Imagínate aprobando el examen

## Necesitas tiempo y concentración

Planes	 PLAN TURBO	 PLAN PRO	 PLAN PRO+
 Descargas sin publi al mes	10 	40 	80 
 Elimina el video entre descargas			
 Descarga carpetas			
 Descarga archivos grandes			
 Visualiza apuntes online sin publi			
 Elimina toda la publi web			
 Precios <span>Anual <input type="checkbox"/></span>	0,99 € / mes	3,99 € / mes	7,99 € / mes

Ahora que puedes conseguirlo,  
¿Qué nota vas a sacar?



# WUOLAH

# Sistemas Inteligentes



**Comparte estos flyers en tu clase y consigue más dinero y recompensas**



**Banco de apuntes de la**

**WUOLAH**

**1**

Imprime esta hoja

**2**

Recorta por la mitad

**3**

Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes

**4**

Llévate dinero por cada descarga de los documentos descargados a través de tu QR



### Formas de controlar individuos no válidos

Ordenamos las ciudades y las metemos en una lista. Los elementos se sacan de la lista al ser usados. Se codifica la posición de la ciudad

A	B	C	D
0	1	2	3

¿Cuántos bits necesito? Peor caso  $A \Rightarrow D$  0 - 3 pero como el primero se elimina sería de 0 - 2  $\Rightarrow$  N - 1 ciudades : 3 (2 bits)  
¿Como codificaría  $A \Rightarrow B \Rightarrow D \Rightarrow C$ ?  
00|00|01|00  
¿Y este individuo? 00|03|03|02 (No sería válido, hay que controlarlo)

Cuando mutamos o cruzamos puede generarse un individuo no válido como en el caso anterior. En ese momento tendremos que hacer un chequeo de integridad del individuo. Si no se supera hay diferentes formas de abordar el problema:

- Penalizar con un fitness muy malo al individuo para que sea improbable que este sea elegido
- Descartar el individuo y volver a seleccionar otros dos para reproducirse (ojo si probabilidad alta)
- Modificar el cruce y la mutación para impedir que suceda o adaptar la solución al individuo válido más cercano

### Estrategias evolutivas (EE).

Las estrategias evolutivas son similares a los algoritmos genéticos. La principal diferencia es que opera con una codificación de valores continuos. La codificación discreta de los algoritmos genéticos canónicos aquí no sirve ya que el espacio de búsqueda sería inmenso y si se discretiza se pierde precisión.

Hay que adaptar los operadores genéticos. Principalmente la mutación. Esta va a seguir una distribución Normal de probabilidad. Esto hace que la heurística de búsqueda se establezca precisamente en el operador de mutación y no en el cruce como en los algoritmos genéticos.

#### Codificación de individuos en EE.

Un vector de codificación  $x_i$  y un vector de varianzas  $\sigma_i$  que indica lo alejado que se encuentra la solución del problema.

Pero no sabemos cuál es la solución óptima, por lo tanto, en realidad son predicciones de la distancia al óptimo que también evoluciona. Los diferentes algoritmos de estrategias evolutivas siguen la notación  $(\mu/p,\alpha)$  donde:

- $\mu$ : tamaño de la población.
- $p$ : número de padres que se seleccionan para recombinarse.
- $\alpha$ : número de individuos en la descendencia.

#### El algoritmo más simple (1 + 1) - EE

Este es el algoritmo más básico y es un método de escalada puro. Un individuo se denota de la siguiente forma  $P=x,\sigma$  donde  $x$  y  $\sigma$  son el vector de codificación y el de varianza.

- Se genera una población intermedia mutando al individuo y después se evalúa.
- Si es mejor se reemplaza y si no se mantiene el actual.
- Como sólo hay un individuo no hay cruce.

La mutación del vector de soluciones se realiza usando una distribución normal de media 0 y varianza  $\sigma$ .

$$x'_i = x_i + \sigma'_i \cdot N(0, 1)$$

Esto se denota como  $\gamma_s^t$  donde  $t$  es el número de veces que se ha realizado reemplazo y  $s$  el número de generaciones anteriores que tenemos en cuenta. Si la ratio entre ellas varía de 1/5 se incrementa o se decrementa el valor de la varianza siguiendo la siguiente regla:

$$\sigma' = \sigma \cdot \text{const}_d \leftrightarrow \gamma_s^t < 1/5$$

$$\sigma' = \sigma \cdot \text{const}_t \leftrightarrow \gamma_s^t > 1/5$$

Donde  $\text{const}_d$  y  $\text{const}_t$  son dos valores. El primero menor que 1 y el segundo mayor que 1 que son elegibles por el diseñador.

Vamos evolucionando el algoritmo hasta que no cambie sustancialmente en una iteración o hasta que el error sea admisible

La varianza al principio será muy grande (la distancia a la solución) y los saltos que se produzcan serán más grandes. A medida que nos vayamos acercándonos a la solución, los saltos más pequeños porque estamos más cerca (Exploración vs Explotación)

#### Modelos con múltiples individuos.

La idea es la misma, pero sí que tienen el operador de cruce.

El cruce de dos individuos se realiza haciendo la media aritmética de cada uno de ellos. También hay que cruzar el vector de varianzas, pero este se debe hacer calculando la raíz cuadrada de la suma de las varianzas de los dos individuos:

$$x' = 1/2 \cdot (x_1 + x_2)$$

$$\sigma' = \sqrt{\sigma_1 + \sigma_2}$$

#### Computación evolutiva como Agente.



Importante

Puedo eliminar la publi de este documento con 1 coin

¿Cómo consigo coins? → Plan Turbo: barato  
→ Planes pro: más coins

Desde un punto de vista de Agentes estamos ante agentes que resuelven un problema de optimización, por tanto estamos al igual que en la búsqueda, en una clase de agente basado en objetivos llamado agente resolve-problemas, específicamente de optimización, donde el camino hasta obtener la solución no es importante si no la configuración final del sistema.

El cambio de las metas en algoritmos genéticos es más complicado que en los algoritmos de búsqueda clásicos, donde simplemente cambiaba la condición de parada. Aquí no hay una condición de parada ya que cualquier individuo es una solución al problema.

Así que cambiar la meta implica mínimo cambiar la función de fitness y quizás la codificación del problema. En cambio, tenemos un sistema que nos permite aprender.

El sistema es capaz de aprender a resolver un problema cada vez de forma más óptima, esto sitúa a los agentes basados en computación evolutiva dentro de los agentes que aprenden. Y además lo hacen de forma no supervisada una vez definida la función de fitness.

También se adaptan muy bien a entornos cambiantes si mantenemos la diversidad genética. También es aplicable a un sistema multiagente independiente o de colaboración simple (si entendemos que la habilidad para resolver el problema se adquiere con el tiempo)

#### **Sistemas Emergentes**

Los sistemas emergentes son sistemas complejos compuestos normalmente por elementos con un comportamiento muy simple, pero que realizan en conjunto un comportamiento poco esperado.

Se caracterizan por resolver problemas, al menos en apariencia, espontáneamente (sin recurrir a una inteligencia de tipo centralizado o jerarquizado sino de forma ascendente) desde la base.

#### Colonias de hormigas.

Los algoritmos basados en colonias de hormigas son un algoritmo bio-inspirado que son capaces de encontrar el camino más corto entre dos puntos de forma automática.

Las hormigas artificiales son unas unidades o pequeños agentes que se mueven de forma aleatoria por el espacio de soluciones. Al moverse van dejando una feromona a su paso.

La sustancia emitida puede ser detectada por otras hormigas:

- La hormiga elige el camino proporcional a la feromona que esté en dicho camino.
- La feromona del camino se disipa con el tiempo, y si no se vuelve a generar desaparece.
- Los caminos poco frecuentados casi no tienen posibilidad de ser utilizados, a menos que se encuentre a menos que se encuentre comida y se repita el proceso.

Encontrar el camino más corto entre dos puntos es una propiedad emergente de la propia interacción con el entorno.

Computacionalmente, una hormiga es un pequeño agente que se mueve de forma semi-aleatoria. Tiene mayor probabilidad de moverse hacia una posición donde antes ha pasado una hormiga que haya dejado su feromona depositada. La memoria está en el medio y no en los agentes.

#### Enjambre de partículas.

Es una metaheurística que evoca el comportamiento de las partículas en la naturaleza. En un principio fueron concebidos para elaborar modelos de conducta social.

Permiten optimizar un problema a partir de una población de soluciones candidatas, que denominamos partículas. Estas partículas se mueven dentro del espacio de búsqueda según una serie de reglas matemáticas sencillas en base a su posición y su velocidad.

#### Otros.

- Programación genérica: misma idea que genéticos, pero aplicado a programas. Se necesita un lenguaje de programación muy simple.
- Redes de neuronas.
- Sistema inmunitario artificial: máquinas de aprendizaje basado en reglas computacionalmente inteligentes, inspiradas en los principios y procesos del sistema inmunitario de los seres vivos.
- Optimización multiobjetivo: permite optimizar objetivos antagónicos. Se basa en el concepto de frente de pareto que genera un conjunto de pares de soluciones válidas.

#### Otros algoritmos de búsqueda local.

pierdo espacio



Necesito concentración

ali ali oohh  
esto con 1 coin me  
lo quito yo...

WUOLAH

- El recorrido simulado: fases de exploración cambiantes. Se pueden usar en genéticos.
- La búsqueda tabú: similar a la anterior, pero genera múltiples mutaciones de un individuo. Una lista de soluciones prohibidas impide que se repitan soluciones ya generadas.
- La optimización extrema: desarrolla una única solución y realiza modificaciones locales a los peores componentes.
- Búsqueda de ascensión de colinas: un bucle que continuamente se mueve en dirección del valor creciente (o decreciente)