# Introducción a la Computación Evolutiva

**Dr. Carlos A. Coello Coello**

**Departamento de Computación**

**CINVESTAV-IPN**

**Av. IPN No. 2508**

**Col. San Pedro Zacatenco**

**México, D.F. 07300**

**email:** ccoello@cs.cinvestav.mx

**http:** //delta.cs.cinvestav.mx/~ccoello

## Coevolución

En la naturaleza existen organismos que tienen una relación simbiótica con otros organismos. La **Simbiosis** se define como el "fenómeno mediante el cual dos organismos disimilares viven íntimamente juntos en una relación mutuamente benéfica". En la comunidad de computación evolutiva se han desarrollado algoritmos que adoptan simbiosis, aunque son relativamente escasos.

## Coevolución

Llamamos **coevolución** a un cambio en la composición genética de una especie (o grupo de especies) como respuesta a un cambio genético de otra. En un sentido más general, la coevolución se refiere a un cambio evolutivo recíproco entre especies que interactúan entre sí.

## Coevolución

El término **coevolución** suele atribuírsele a Ehrlich y Raven, quienes publicaron un artículo sobre sus estudios de las mariposas y las plantas a mediados de los 1960s [Ehrlich, 1964]. Las relaciones entre las poblaciones de dos especies distintas **A** y **B** pueden ser descritas considerando todos los tipos posibles de interacciones.

## Coevolución

Tales interacciones pueden ser positivas o negativas, dependiendo de las consecuencias que ésta produzca en la población. La tabla del acetato siguiente muestra todas las posibles interacciones entre dos especies diferentes.

# Coevolución

| | **A** | **B** | |
|---|---|---|---|
| Neutralism | 0 | 0 | Populations A and B are independent and don't interact |
| Mutualism | + | + | Both species benefit from the relationship |
| Commensalism | + | 0 | One species benefits from the relationship but the other is neither harmed nor benefited |
| Competition | - | - | Both species have a negative effect on each other since they are competing for the same resources |
| Predation | + | - | The predator (A) benefits while the prey (B) is negatively affected |
| Parasitism | + | - | The parasite (A) benefits while the host (B) is negatively affected |

## Coevolución

La comunidad de computación evolutiva ha desarrollado varios enfoques coevolutivos en los que normalmente dos o más especies se relacionan entre sí usando alguno de los esquemas indicados anteriormente. Asímismo, en la mayoría de los casos, tales especies evolucionan independientemente a través de un algoritmo evolutivo (normalmente un algoritmo genético).

## Coevolución

El aspecto clave en estos algoritmos coevolutivos es que la aptitud de un individuo en una población depende de los individuos de una población diferente. De hecho, podemos decir que un algoritmo es coevolutivo si presenta esta propiedad.

## Coevolución

Existen dos clases principales de algoritmos coevolutivos en la literatura especializada:

1. Aquellos basados en relaciones de competencia (denominada **coevolución competitiva**): En este caso, la aptitud de un individuo es el resultado de "encuentros" con otros individuos [Paredis, 1998]. Este tipo de esquema coevolutivo se ha adoptado normalmente para los juegos.

## Coevolución

2. Aquellos basados en relaciones de cooperación (denominada **coevolución cooperativa**): En este caso, la aptitud de un individuo es el resultado de una colaboración con individuos de otras especies (o poblaciones) [Potter, 1994]. Este tipo de esquema coevolutivo se ha adoptado normalmente para resolver problemas de optimización.

## Coevolución

Existen varios trabajos en la literatura especializada que usan la coevolución y la simbiosis en un algoritmo evolutivo. Paredis [1995,1998] proporciona una muy buena introducción a la coevolución, discutiendo primero la **aptitud competitiva**.

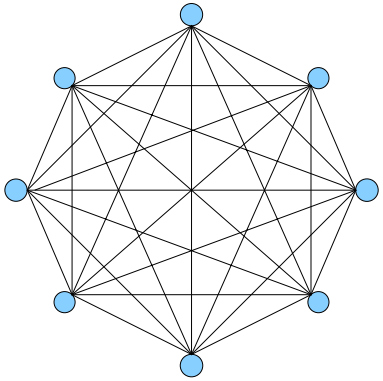
## Coevolución

Este tipo de aptitud se obtiene mediante cálculos que dependen en cierto grado de la población actual. La dependencia puede ser mínima (por ejemplo, se puede depender de un solo miembro de la población) o exhaustiva (o sea, se puede depender de toda la población y combinar sus aptitudes en un solo valor escalar).

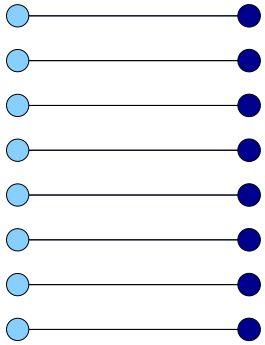
## Coevolución

Cuatro ejemplos de funciones de aptitud competitivas incluyen: competencia total (todos vs. todos), competencia bipartita (uno vs. uno o posiblemente uno vs. muchos), aptitud de torneo (torneo binario con eliminación de un solo elemento, la cual no debe confundirse con la selección de torneo), y la competencia elitista (todos vs. el mejor). Estos esquemas se ilustran gráficamente en el acetato siguiente.

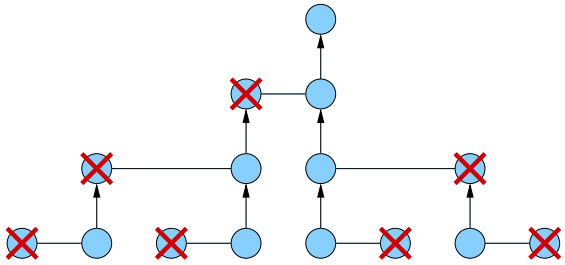
# Coevolución



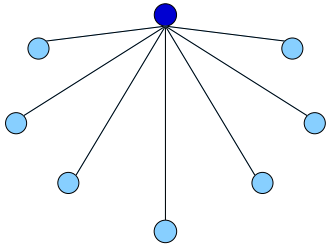
All vs. All



Bipartite



Tournament



All vs. Best

## Coevolución

La aptitud competitiva se ha aplicado extensamente en los juegos evolutivos. Algunos de los juegos en donde se han usado algoritmos evolutivos son: Gato [Angeline, 1993], Otelo [Chong, 2005], Awari [Davis, 2002], Poquer [Billings, 2001], Backgammon [Pollack, 1996] y el dilema del prisionero [Chong, 2005a].

## Coevolución

Otra forma de coevolución es el denominado **modelo depredador-presa**. Un ejemplo de este modelo lo encontramos en el trabajo que hizo Danny Hillis con las redes de ordenamiento (*sorting networks*) [Hillis, 1990]. En este caso, se usaron dos poblaciones, una conteniendo las redes de ordenamiento y la otra consistente en un conjunto de listas con 16 números para ser ordenados.

## Coevolución

Estas poblaciones estaban distribuidas geográficamente sobre un cúmulo de computadoras. En cada computadora, se aplicaba un conjunto de redes de ordenamiento a un conjunto de listas. La aptitud de una red era el porcentaje de listas que eran ordenadas, mientras que la aptitud de una lista era el porcentaje de redes que no podían ordenarlas. Este tipo de interacción de aptitud inversa es típica en un modelo depredador-presa.

## Coevolución

Hillis encontró que sus soluciones coevolucionadas eran mejores que las obtenidas con un algoritmo evolutivo tradicional. Esto lo atribuyó a dos razones. Primero, puesto que las poblaciones están cambiando constantemente, esto motiva una mayor exploración y evita la convergencia prematura. La otra razón es que el chequeo de aptitud es más eficiente porque el enfoque es en las listas que no pudo ordenar correctamente.

## Coevolución

Paredis [1995,1998] introdujo el algoritmo genético coevolutivo (CGA). Este algoritmo está basado en el modelo depredador-presa, en el que se usa una población de soluciones y otra de pruebas (o sea, soluciones de ejemplo).

## Coevolución

Primero se inicializan las dos poblaciones (soluciones y pruebas), y luego se generan valores de aptitud para cada una, viendo qué tan bien se comparan con respecto a 20 individuos aleatorios de la población opuesta (se asigna 1 o 0 por cada éxito o falla).

## Coevolución

Estos resultados se almacenan en una historia para cada individuo y la aptitud es el promedio de estos 20 resultados. En cada generación (a las que Paredis llama ciclos), se seleccionan 20 soluciones y pruebas, haciendo que el individuo más apto tenga una probabilidad 1.5 veces mayor de ser elegido sobre el de aptitud media.

## Coevolución

Si la solución es exitosa con la prueba seleccionada, recibe un 1; de lo contrario, recibe 0. La historia de ambas se actualiza. La historia lleva un registro de los 20 resultados más recientes de los "encuentros". Después de 20 encuentros, se eligen dos padres de la población de soluciones. Los padres se recombinan y el hijo es mutado.

## Coevolución

La aptitud del hijo se determina y el hijo es insertado en la población en la jerarquía apropiada, posiblemente reemplazando a uno de sus padres. Esta es una estrategia de selección $(\mu + 1)$. Los parámetros usados por el autor son, como él mismo admite, totalmente arbitrarios.

## Coevolución

Paredis no evoluciona sus pruebas a propósito, puesto que éstas fueron diseñadas específicamente para moldear el espacio de búsqueda. Sin embargo, menciona que hay ocasiones en que las pruebas podrían ser evolucionadas también.

## Coevolución

Algunas de las aplicaciones del CGA incluyen: clasificación, control de procesos, planeación de rutas de robots, satisfacción de restricciones, clasificación de densidad y simbiosis. En el caso de la clasificación, el control de procesos y la planeación de rutas de robots, se evolucionan redes neuronales.

## Coevolución

El ejemplo de clasificación usa ejemplos de entrenamiento inalterados para la población de pruebas, mientras que los otros dos ejemplos hacen evolucionar a las soluciones de prueba. El problema de satisfacción de restricciones usa como población de pruebas las restricciones y adopta arreglos de variables como las soluciones.

## Coevolución

El CGA supera los resultados de un AG en estos problemas. Para la clasificación de densidad, Paredis coevoluciona autómatas celulares con cadenas de bits, a fin de clasificar la densidad de unos en cada cadena. Luego se aleja del modelo depredador-presa y aplica el CGA de forma simbiótica, de tal forma que las dos poblaciones proporcionan retroalimentación positiva de aptitud, en vez de que ésta sea negativa.

## Coevolución

Paredis usó simbiosis para tratar de determinar la mejor representación genética para un individuo en un problema. Este problema intenta colocar genes con los enlaces más fuertes de forma consecutiva, a fin de preservar intactas las buenas soluciones después de aplicar la cruza.

## Coevolución

En este caso, las soluciones se evolucionaron junto con las representaciones. Este esquema realmente intenta ligar explícitamente los buenos bloques constructores en el cromosoma y resulta exitoso en la creación de mejores soluciones.

## Coevolución

Un subconjunto interesante de los algoritmos coevolutivos es el de los algoritmo cooperativos (CCGA). Este tipo de algoritmos fueron propuestos originalmente por Potter y de Jong [1994], y consisten de un enfoque simbiótico, pero en vez de usar una población de soluciones y otra de pruebas, se evolucionan poblaciones de especies.

## Coevolución

Para formar una solución, se selecciona un individuo de cada especie y se combina con los demás individuos seleccionados. La solución se evalúa y las especies que conformaron la solución reciben una puntuación con base en la aptitud de la solución combinada.

## Coevolución

Potter y de Jong propusieron dos algoritmos: el CCGA-1 y el CCGA-2. El primero, inicializa cada población de especies y asigna la aptitud inicial combinando cada miembro de una sobpoblación con individuos aleatorios de las otras subpoblaciones. Luego, cada una de las especies se coevoluciona en un esquema del tipo "round robin", usando un AG tradicional.

## Coevolución

La aptitud de los miembros de cada sobpoblación evolucionada se obtienen de combinarlos con el mejor individuo de las otras sobpoblaciones y de obtener la aptitud del individuo. Se hace notar que este modelo de asignación de crédito tiene varios problemas, tales como el sub-muestreo, pero fue creado únicamente como un punto de partida en el cual basar la efectividad de las versiones posteriores del algoritmo.

## Coevolución

CCGA-1 superó a un AG estándar cuando las especies representaron funciones que fueron independientes entre sí, pero tuvo un mal desempeño cuando las funciones tenían dependencias. El algoritmo del CCGA-1 se muestra en el acetato siguiente.

# Coevolución

1: **procedure** CCGA-1($\mathcal{N}, g, f_k(\vec{x})$)
2: **for** Each species $s$ **do**
3: $\mathbb{P}'_s(g)$ = Randomly initialize population
4: Evaluate fitness of each individual in $\mathbb{P}'_s(g)$
5: **end for**
6: **while** Termination condition = false **do**
7: $g = g + 1$
8: **for** Each species $s$ **do**
9: Select $\mathbb{P}'_s(g)$ from $\mathbb{P}'_s(g-1)$ based on fitness
10: Apply genetic operators to $\mathbb{P}'_s(g)$
11: Evaluate fitness of each individual in $\mathbb{P}'_s(g)$
12: **end for**
13: **end while**
14: **end procedure**

## Coevolución

A fin de superar estas deficiencias, se creó un nuevo esquema de asignación de crédito. Además de crear un individuo, tal y como se describió antes, se crea un segundo individuo, seleccionando un miembro al azar de cada subpoblación para combinarlo con dicho individuo. La mejor aptitud de entre los dos individuos es la que se asigna al hijo. Esto dio pie a CCGA-2, cuyo algoritmo se muestra en el acetato siguiente.

```
1: procedure CCGA-2(𝒩, g, f_k(x⃗))
2:     for Each species s do
3:         ℙ′_s(g) = Randomly initialize population
4:         Evaluate fitness of each individual in ℙ′_s(g)
5:     end for
6:     while Termination condition = false do
7:         g = g + 1
8:         for Each species s do
9:             Select ℙ′_s1(g) from ℙ′_s(g − 1) based on fitness
10:            Select ℙ′_s2(g) from ℙ′_s(g − 1) at random
11:            if ℙ′_s1(g) is most fit then
12:                Let ℙ′_s(g) = ℙ′_s1(g)
13:            else
14:                Let ℙ′_s(g) = ℙ′_s2(g)
15:            end if
16:            Apply genetic operators to ℙ′_s(g)
17:            Evaluate fitness of each individual in ℙ′_s(g)
18:        end for
19:    end while
20: end procedure
```

## Coevolución

Este algoritmo tuvo buen desempeño tanto en funciones con dependencias como en las que no tenían dependencias. En general, el diseño del CCGA proporciona un mapeo natural para arquitecturas paralelas de grano grueso.

## Coevolución

Este tipo de esquema puede ser particularmente útil en algunas aplicaciones tales como, por ejemplo, la programación de tareas de mantenimiento de motores de aviones, en el cual cada miembro de la población es una solución total.

## Coevolución

Wallin et al. [2005] introdujeron un algoritmo coevolutivo cooperativo que se basa en el concepto de endosimbiosis. La endosimbiosis se refiere a la relación simbiótica entre un organimso y otro que vive adentro del cuerpo del huésped. En los algoritmos evolutivos, el uso de coevolución competitiva puede evitar que las poblaciones se estanquen, usando co-especies para mantener la presión evolutiva.

## Coevolución

Wallin et al. [2005] propusieron el **Symbiotic Coevolutionary Algorithm** (SCA), el cual usa dos especies: los huéspedes y los parásitos. Los huéspedes son una solución completa, mientras que el parásito es una solución parcial.

## Coevolución

Para que se pueda evaluar un parásito, debe estar ligado a un huésped. El parásito está conformado por dos cadenas. La primera cadena es un número binario con codificación de Gray, cuyo fin es dirigir los valores parasíticos a una posición dentro del huésped.

## Coevolución

La segunda cadena es el valor parasítico que se usa para reemplazar una porción de la cadena del huésped. Otras implementaciones pueden incluir el uso de operadores Booleanos (p.ej., AND, OR, XOR) para combinar el huésped y el parásito.

## Coevolución

Wallin et al. [2005] generan un número igual de parásitos y huéspedes, y efectúan un chequeo todos contra todos, en el cual cada huésped se asocia con cada parásito de forma individual y se evalúa. Se usa un esquema de selección $(\mu + \lambda)$, de forma que los $|\mu|$ mejores de las combinaciones evaluadas constituyen el conjunto de candidatos al apareamiento para la siguiente generación de huéspedes.

## Coevolución

Las $k$ mejores soluciones se almacenan en un archivo externo. Después de que se forma la siguiente generación de huéspedes, los parásitos deben ser evolucionados. Los parásitos se reproducen primero asexualmente. El algoritmo aplica entonces mutación. Finalmente, se usa selección de ruleta para elegir la siguiente generación de parásitos.

## Coevolución

Los autores comparan su SCA con respecto a un AG generacional en problemas de 64 y 128 bits que pueden descomponerse, pero son deceptivos. Se usan parásitos de 4, 7, 12 y 17 bits en los experimentos. Los autores reportan que a menor tamaño de los parásitos (o sea, 4 y 7), los resultados son mejores.

## Coevolución

Se hace notar que a pesar del vínculo simbiótico entre los parásitos y sus huéspedes, estrictamente hablando no hay coevolución en este caso, puesto que sólo se evolucionan los huéspedes, puesto que los parásitos simplemente son mutados.

## Paisajes de Aptitud

El concepto de "paisaje de aptitud" (*fitness landscape*) fue introducido por Sewall Wright en 1932, como una metáfora para describir múltiples dominios de atracción en la dinámica evolutiva. La idea de Wright fue ver el espacio de búsqueda como si tuviera múltiples picos hacia los cuales la población evolucionaría, escalándolos.

## Paisajes de Aptitud

Wright estaba principalmente interesado en analizar la forma en que las poblaciones podían escapar de los óptimos locales a través de fluctuaciones estocásticas. Por tanto, esta es una de las primeras propuestas para usar procesos estocásticos para optimizar funciones multimodales.

## Paisajes de Aptitud

El concepto de Wright fue concebido como una ayuda para visualizar el comportamiento de los operadores de selección y variación durante el proceso evolutivo. Por tanto, los paisajes de aptitud normalmente se usan para estudiar la eficacia de los algoritmos evolutivos.

## Paisajes de Aptitud

Particularmente, los investigadores han adoptado los denominados paisajes de aptitud NK propuestos por Stuart Kauffman [1987] para explorar la forma en la cual la epístasis controla la rugosidad de un paisaje adaptativo.

## Paisajes de Aptitud

La idea de Kauffman fue especificar una familia de funciones de aptitud que tuvieran una rugosidad que pudiera ser ajustable a través de la manipulación de un solo parámetro.

## Sistemas de Clasificadores

Desde los orígenes de los algoritmos genéticos, hubo investigadores interesados en aplicar este tipo de técnicas a problemas de aprendizaje de máquina. La motivación principal es que los cromosomas, los cuales representan conocimiento, son tratados como datos a ser manipulados mediante operadores genéticos y, al mismo tiempo, como código ejecutable a ser utilizado para realizar alguna tarea. Esto dio pie al desarrollo de los denominados *sistemas de clasificadores*, en los cuales se usan algoritmos evolutivos para resolver problemas de aprendizaje.

## Sistemas de Clasificadores

Desde los viejos días de los algoritmos genéticos, se consideraron dos tipos de enfoques para diseñar sistemas de clasificadores:

1. El primer enfoque consiste en representar todo un conjunto de reglas usando un individuo (o sea, una cadena cromosómica), mantener una población de conjuntos de reglas y usar el mecanismo de selección y los operadores de un algoritmo genético para producir nuevos conjuntos de reglas. A este tipo de técnica se le denomina "enfoque Pitt", dado que fue desarrollado por Kenneth De Jong y sus estudiantes, cuando éste se encontraba en la Universidad de Pittsburgh.

## Sistemas de Clasificadores

2. John Holland desarrolló, al mismo tiempo que De Jong, otro modelo en el cual los miembros de la población son reglas individuales y un conjunto de reglas es realmente toda la población. A éste se le denomina el "enfoque Michigan".

A continuación, discutiremos en un poco más de detalle estos dos enfoques.

## Sistemas de Clasificadores: El Enfoque Michigan

Los sistemas de clasificadores son un tipo de sistema basado en reglas con mecanismos generales para procesar reglas en paralelo, a fin de generar, de manera adaptativa, nuevas reglas, así como para probar la efectividad de las reglas existentes. Los sistemas de clasificadores proporcionan un marco de trabajo en el cual una población de reglas codificadas como cadenas de bits evolucionan con base en recibir, intermitentemente, estímulos y refuerzo de su ambiente. El sistema "aprende" qué respuestas son apropiadas cuando se presenta un cierto estímulo. Las reglas en un sistema de clasificadores forman una población de individuos que evolucionan en el tiempo.

## El Enfoque Michigan

Un sistema de clasificadores consta de los siguientes componentes:

- detector y efector,
- sistema de mensajes (entrada, salida, y listas de mensajes internos),
- sistema de reglas (población de clasificadores),
- sistema de asignación de crédito (el algoritmo *bucket brigade*), y
- procedimiento genético (reproducción de los clasificadores).

# El Enfoque Michigan

El ambiente envía un mensaje (p.ej., un movimiento en un tablero), el cual es aceptado por los detectores del sistema de clasificadores y colocado en la lista de mensajes de entrada. Los detectores decodifican el mensaje (posiblemente sub-dividiéndolo en varios mensajes) y lo colocan en la lista de mensajes internos. Los mensajes activan clasificadores; los clasificadores activados colocan mensajes en la lista de mensajes. Estos nuevos mensajes pueden activar a otros clasificadores o pueden irse a la lista de mensajes de salida. En este último caso, los efectores del sistema de clasificadores codifican estos mensajes en un mensaje de salida, el cual se regresa al ambiente. El ambiente evalúa la acción del sistema (retroalimentación del ambiente) y se usa el algoritmo del *bucket brigade* para actualizar los clasificadores.

## El Enfoque Michigan

El enfoque Michigan puede percibirse como un modelo computacional de la cognición: el conocimiento de una entidad cognitiva se expresa como una colección de reglas que sufren modificaciones en el tiempo. Podemos evaluar una unidad cognitiva completa en términos de su interacción con el ambiente; la evaluación de una sola regla (o sea, de un solo individuo) no tiene ningún sentido.

# El Enfoque Pitt

En el enfoque Pitt, cada individuo en la población codifica todo el conjunto de reglas (es decir, el individuo es una entidad cognitiva). Estos individuos compiten entre sí, de manera que los individuos débiles se mueren y los fuertes sobreviven y se reproducen (usando el mecanismo de selección y los operadores convencionales de un algoritmo genético). En otras palabras, se aplica un algoritmo genético al problema de aprendizaje. Con esto, se evita el problema de la asignación de crédito, para el cual se requiere un método heurístico (como el algoritmo del *bucket brigade*) para distribuir el crédito (positivo o negativo) entre las reglas que cooperaron para producir un comportamiento deseable o indeseable.

## El Enfoque Pitt

El enfoque Pitt, sin embargo, plantea varias preguntas interesantes. Por ejemplo: ¿debemos usar cadenas de longitud fija para representar las reglas? Esto facilita el funcionamiento del algoritmo genético, pero puede ser muy restrictivo. De hecho, pueden usarse representaciones más complejas para los sistemas de reglas (p.ej., incorporando lógica difusa, entradas que sean números reales, operadores booleanos, etc.).

## El Futuro de la Computación Evolutiva

- **Metamerismo**: El proceso en el cual una unidad estructural es duplicada un cierto número de veces y durante ese proceso se reoptimiza para otros usos.
- **Auto-adaptación**: Evitar el uso de parámetros *ad-hoc* en los algoritmos evolutivos.

## El Futuro de la Computación Evolutiva

- **Técnicas que exploten arquitecturas paralelas**: Es importante explotar al máximo las arquitecturas paralelas mediante nuevos algoritmos evolutivos. Esto traerá importantes ganancias en términos de esfuerzo computacional, sobre todo al lidiar con problemas del mundo real.

## El Futuro de la Computación Evolutiva

- **Teoría**: Pruebas de convergencia, modelos matemáticos de los algoritmos evolutivos, epístasis, diversidad, etc.
- **Entender mejor la evolución natural**: Simulaciones computacionales que permitan entender las complejas interacciones que ocurren entre los seres vivos.

## El Futuro de la Computación Evolutiva

- **Coevolución**: Muchos investigadores han dirigido sus esfuerzos a estudiar mejor la coevolución como un modelo alternativo para resolver problemas en computación evolutiva.
- **El AG sin parámetros**: Es, sin lugar a dudas, el sueño de los expertos en computación evolutiva.

## El Futuro de la Computación Evolutiva

- Las 3 grandes preguntas:
  1) ¿A qué tipo de problemas deben aplicarse los algoritmos evolutivos?
  2) ¿Cómo podemos mejorar nuestra comprensión del funcionamiento de los algoritmos evolutivos?

## El Futuro de la Computación Evolutiva

3) ¿Qué nuevas ideas pueden aplicarse a la computación evolutiva a fin de extender el paradigma (p.ej., inspiración biológica)?