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

¿Realmente necesitamos técnicas heurísticas?

Cuando enfrentamos espacios de búsqueda tan grandes como en el caso del problema del viajero, y que además los algoritmos más eficientes que existen para resolver el problema requieren tiempo exponencial, resulta obvio que las técnicas clásicas de búsqueda y optimización son insuficientes.

Ejemplos de técnicas heurísticas

- Búsqueda tabú
- Recocido simulado
- Escalando la colina

Búsqueda Tabú

- Usa una "memoria" para guiar la búsqueda.
- Algunas soluciones examinadas recientemente son "memorizadas" y se vuelven tabú (prohibidas) al tomar decisiones acerca del siguiente punto de búsqueda.
- Es determinística, aunque se le pueden agregar elementos probabilísticos.

Recocido Simulado

- Basado en el enfriamiento de los cristales.
- El horario de enfriamiento es crucial.
- Requiere de una temperatura inicial, una final y una función de variación de la temperatura.
- Es un algoritmo probabilístico de búsqueda local.

Escalando la Colina

- Se aplica a un punto a la vez (técnica local).
- Se generan varios estados posibles y se selecciona el mejor.
- No hay retroceso ni registro histórico.
- Puede quedar atrapado fácilmente en óptimos locales.
- Es un algoritmo determinístico.

Optimización Global

El objetivo principal de cualquier técnica de optimización es encontrar el óptimo (o los óptimos) globales de cualquier problema. En matemáticas, existe un área que se ocupa de desarrollar los formalismos que nos permitan garantizar la convergencia de un método hacia el óptimo global de un problema.

Optimización Global

Desgraciadamente, sólo en algunos casos limitados, puede garantizarse convergencia hacia el óptimo global.
Por ejemplo, para problemas con espacios de búsqueda convexos, las condiciones de Kuhn-Tucker son necesarias y suficientes para garantizar optimalidad global de un punto.

Optimización Global

En problemas de optimización no lineal, las condiciones de Kuhn-Tucker no son suficientes para garantizar optimalidad global. De hecho, todas las técnicas usadas para optimización no lineal pueden localizar cuando mucho óptimos locales, pero no puede garantizarse convergencia al óptimo global a menos que se usen técnicas exhaustivas o que se consideren tiempos infinitos de convergencia.

Optimización Numérica

Existen muchos tipos de problemas de optimización, pero los que nos interesan más para los fines de este curso, son de los de optimización numérica, que pueden definirse de la siguiente manera:

$$\text{Minimizar } f(\vec{x})$$

sujeta a:

$$g_i(\vec{x}) \leq 0 \qquad i = 1, ..., p$$
$$h_j(\vec{x}) = 0 \qquad j = 1, ..., n$$

Optimización Numérica

Llamaremos a $(\vec{x})$ las variables de decisión del problema, $g_i(\vec{x})$ son las restricciones de desigualdad, y $h_j(\vec{x})$ son las restricciones de igualdad. Asimismo, $f(\vec{x})$ es la función objetivo del problema (la que queremos optimizar).

Optimización Numérica

A las restricciones de igualdad y desigualdad expresadas algebraicamente, se les denomina "restricciones explícitas". En algunos problemas, existen también "restricciones implícitas", relacionadas sobre todo con las características del problema.

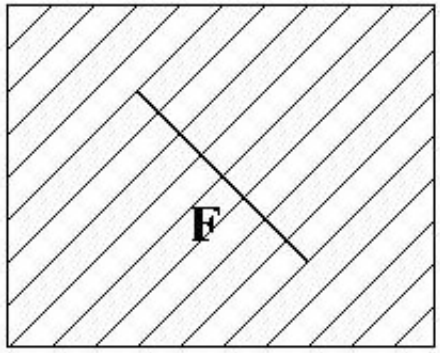
Optimización Numérica

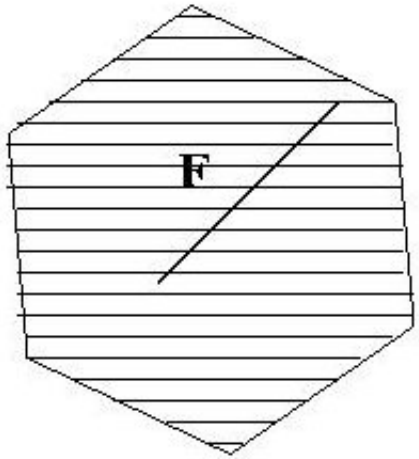
Por ejemplo, si decimos:

$$10 \leq x_1 \leq 20$$

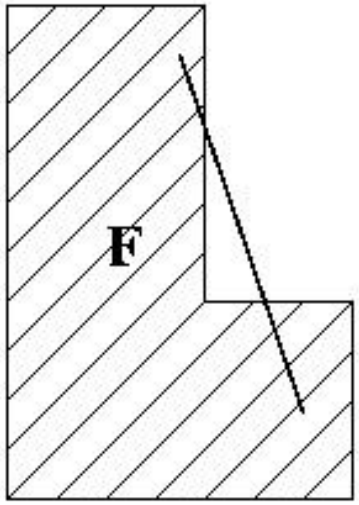
estamos definiendo que el rango de una variable de decisión debe estar contenido dentro de un cierto intervalo. De tal forma, estamos "restringiendo" el tipo de soluciones que se considerarán como válidas.

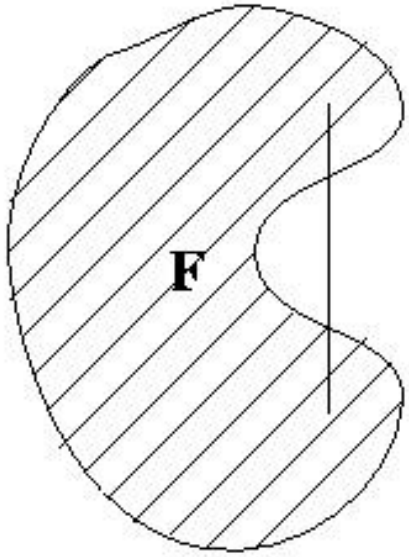
Ejemplos de espacios de búsqueda convexos





Ejemplos de espacios de búsqueda no convexos



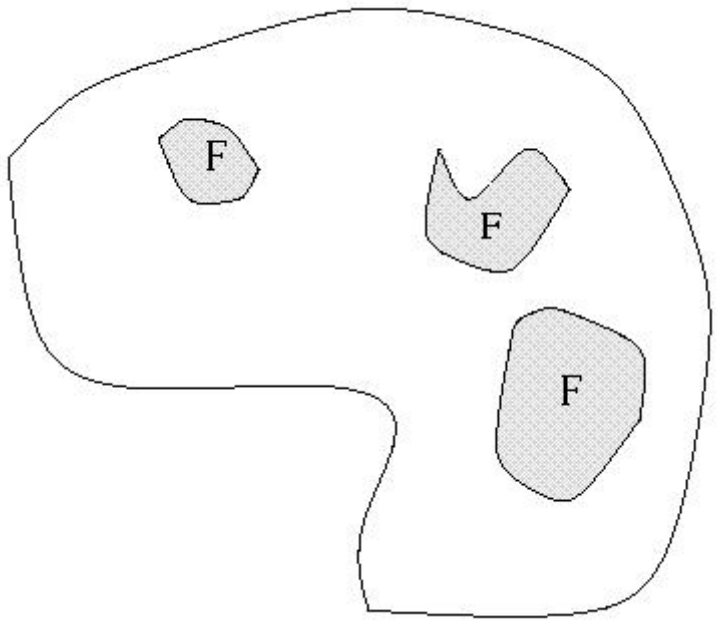


Zona factible y no factible

Todas las soluciones a un problema que satisfagan las restricciones existentes (de cualquier tipo), se consideran ubicadas dentro de la zona factible. De tal forma, podemos decir que el espacio de búsqueda de un problema se divide en la región (o zona) factible y la no factible.

Zona factible y no factible

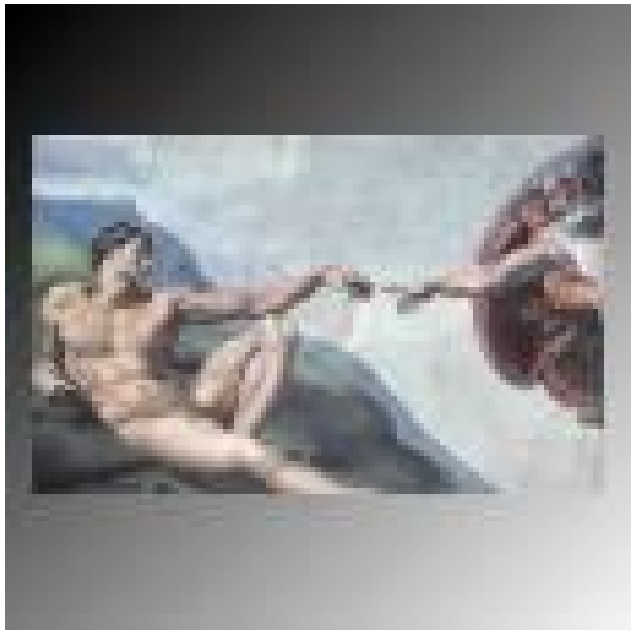
Esta imagen ilustra la diferencia entre la zona factible y no factible de un problema:



Optimización Combinatoria

Existe una clase especial de problemas que también serán de interés para este curso, en los cuales las variables de decisión son discretas y las soluciones suelen presentarse en la forma de permutaciones. A estos problemas se les denomina de "optimización combinatoria" (p. ej. el problema del viajero).

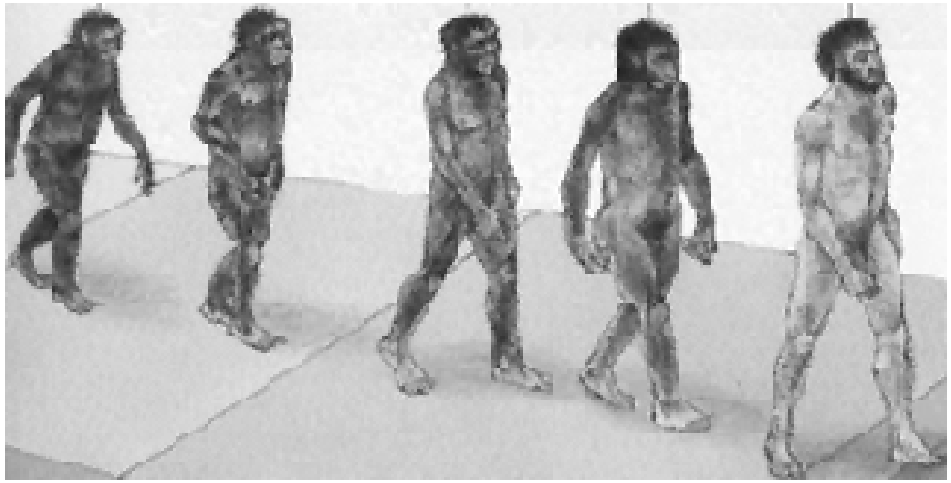
Durante muchos años, la tesis más aceptada sobre el origen de las especies fue el creacionismo: Dios creó a todas las especies del planeta de forma separada.



Además, según el creacionismo, las especies estaban jerarquizadas por Dios de tal manera que el hombre ocupaba el rango superior, al lado del creador.

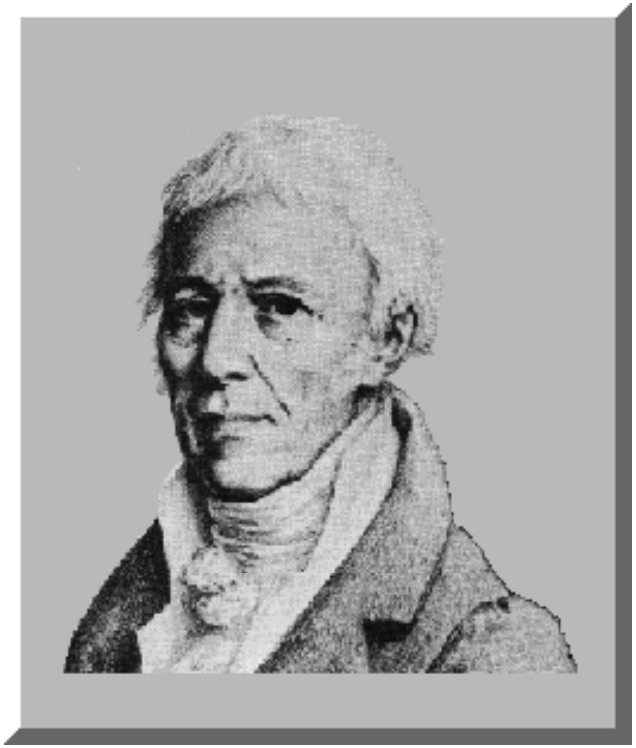
Georges Louis Leclerc (Conde de Buffon) fue tal vez el primero en especular (100 años antes que Darwin) que las especies se originaron entre sí, en una abierta oposición al "creacionismo".

Leclerc no sólo notó las similitudes entre el hombre y los simios, sino que incluso especuló sobre la existencia de un posible ancestro común entre estas 2 especies.

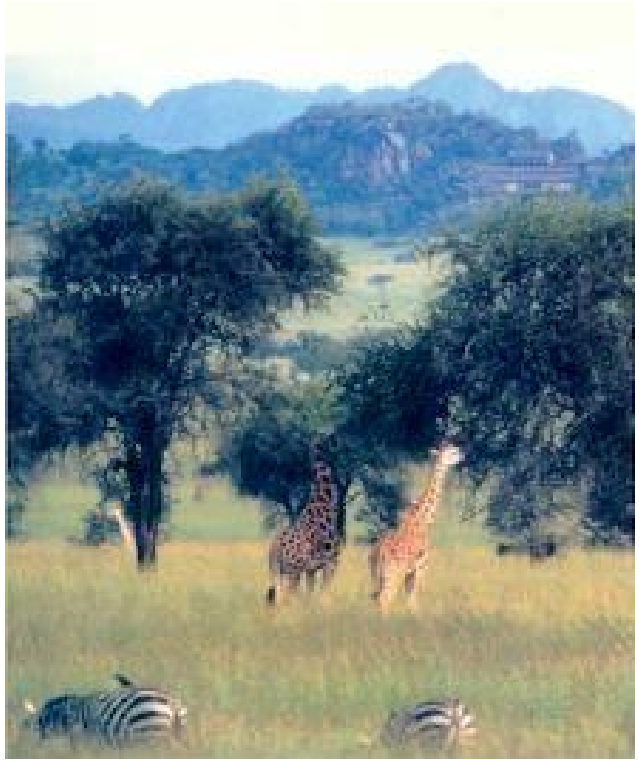


Leclerc creía en los cambios orgánicos, pero no describió un mecanismo coherente que fuera responsable por efectuarlos,sino que especuló que era el ambiente el que influía directamente sobre los organismos.

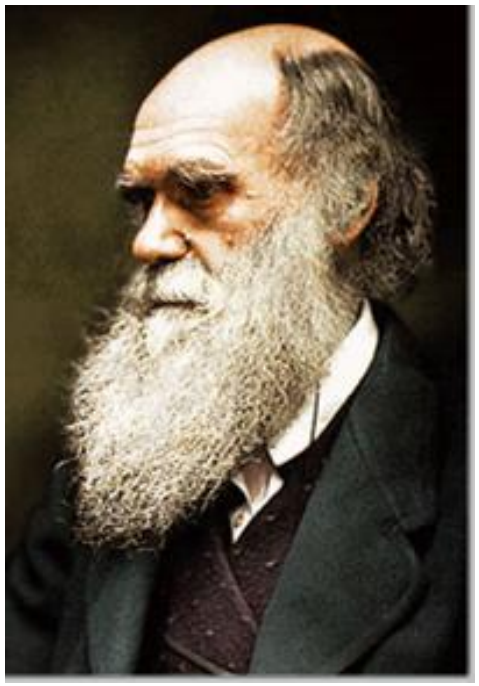
- El biólogo francés Jean-Baptiste Lamarck enunció una teoría sobre la evolución a principios del siglo XIX.



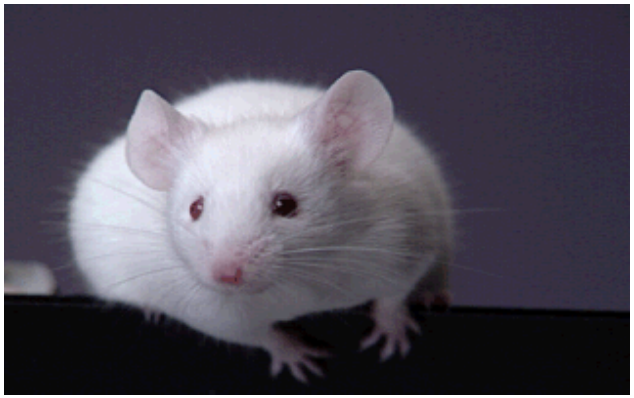
- Según Lamarck,
  las características (físicas)
  adquiridas por un organismo
  durante su vida podían ser
  transmitidas a sus descendientes.



- El naturalista inglés Charles Darwin enunció su teoría sobre el origen de las especies en 1858, derrumbando el Lamarckismo al indicar que la evolución se origina a través de cambios aleatorios de características hereditarias, combinados con un proceso de *selección natural*.



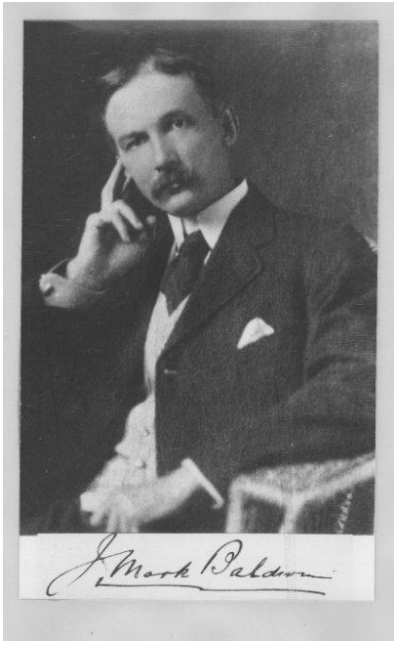
- El científico alemán August Weismann formuló en el siglo XIX una teoría denominada del *germoplasma*, según la cual la información hereditaria se transmite a través de ciertas células (llamadas germinales), mientras que otras células (llamadas somáticas) no pueden transmitir nada.



- August Weismann realizó un experimento en el que cortó las colas de un grupo de ratas durante 22 generaciones (1,592 ratas), sin que las nuevas generaciones de ratas perdieran dicha extremidad, demostrando entonces la falsedad de la hipótesis fundamental del Lamarckismo.

- El monje austriaco Johann Gregor Mendel realizó una serie de experimentos con chícharos durante una buena parte de su vida, enunciando a partir de ellos las leyes básicas que gobiernan la herencia.

- James Mark Baldwin propuso en 1896 que si el aprendizaje ayuda a la supervivencia,entonces los organismos con mayor capacidad de aprendizaje tendrán más descendientes, incrementando de esa manera la frecuencia de los genes responsables del aprendizaje.



- Baldwin planteó un impacto indirecto del aprendizaje sobre el proceso evolutivo (un individuo con mayor capacidad de aprendizaje será más apto), y no uno directo como Lamarck.

  A su propuesta se le conoce hoy en día como el "*efecto Baldwin*".


+

+


=

Neo-Darwinismo

- El pensamiento evolutivo actual gira en torno al Neo-Darwinismo, el cual establece que toda la vida en el planeta puede ser explicada a través de sólo 4 procesos:
  - Reproducción
  - Mutación
  - Competencia
  - Selección

Historia de la Computación Evolutiva

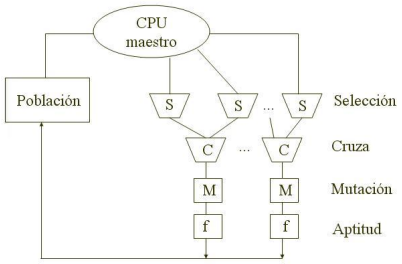
- La evolución natural ha sido vista como un proceso de aprendizaje desde los 1930s, con el trabajo de Walter B. Cannon (*The Wisdom of the Body*).

Historia de la Computación Evolutiva

- El célebre matemático Alan Mathison Turing reconoció también una conexión "obvia" entre la evolución y el aprendizaje de máquina en un artículo de 1950.

Historia de la Computación Evolutiva

- A fines de los 1950s y principios de los 1960s, el biólogo Alexander S. Fraser publicó una serie de trabajos sobre la evolución de sistemas biológicos en una computadora digital, dando la inspiración para lo que después se convertiría en el algoritmo genético.



Historia de la Computación Evolutiva

- Aproximadamente en la misma época de Fraser, el estadístico inglés George E. P. Box propuso un enfoque evolutivo para la optimización de la producción industrial.

Historia de la Computación Evolutiva

- Su técnica, denominada EVOP (*Evolutionary Operation*) sigue en uso hoy en día en la industria química.

Historia de la Computación Evolutiva

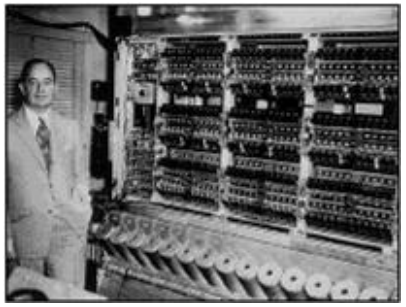
- R. M. Friedberg fue uno de los primeros científicos en intentar evolucionar programas de computadora (a fines de los 1950s). Sus experimentos no fueron muy exitosos, y originaron una avalancha de críticas de parte de los investigadores de la IA clásica.

Historia de la Computación Evolutiva

- G. J. Friedman fue tal vez el primero en proponer una aplicación de las técnicas evolutivas a la robótica: en su tesis de maestría que data de los 1950s, propuso evolucionar una serie de circuitos de control similares a las redes neuronales de hoy en día.

Historia de la Computación Evolutiva

- Nils Aall Barricelli desarrolló las que probablemente fueron las primeras simulaciones de un sistema evolutivo en una computadora digital, entre 1953 y 1956. Sus experimentos siguieron los lineamientos de una disciplina bautizada a principios de los 1980s como "vida artificial".



Historia de la Computación Evolutiva

- Hans J. Bremermann fue tal vez el primero en ver la evolución como un proceso de optimización, además de realizar una de las primeras simulaciones con cadenas binarias que se procesaban por medio de reproducción (sexual o asexual), selección y mutación, en lo que sería otro claro predecesor del algoritmo genético.

Historia de la Computación Evolutiva

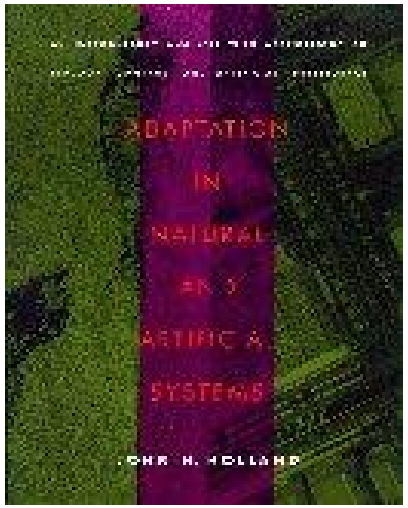
- Lawrence J. Fogel concibió el uso de la evolución simulada en la solución de problemas (sobre todo de predicción) hacia mediados de los 1960s. Su técnica se denominó "Programación Evolutiva".

Historia de la Computación Evolutiva

- Peter Bienert, Ingo Rechenberg y Hans-Paul Schwefel desarrollaron hacia mediados de los 1960s un método de ajustes discretos aleatorios inspirado en el mecanismo de mutación que ocurre en la naturaleza. Su técnica se denominó "estrategias evolutivas".

Historia de la Computación Evolutiva

- John H. Holland desarrolló a principios de los 1960s los "planes reproductivos" y "adaptativos" en un intento por hacer que las computadoras aprendan imitando el proceso de la evolución. Esta técnica sería después conocida mundialmente como el "algoritmo genético".



Historia de la Computación Evolutiva

- Michael Conrad y H. H. Pattee se cuentan entre los primeros en simular un ecosistema artificial jerárquico en el que un conjunto de organismos unicelulares estaban sujetos a una estricta ley de conservación de la materia que les inducía a competir por sobrevivir.

Historia de la Computación Evolutiva

- Conrad propuso también en los 1970s un "modelo de circuitos de aprendizaje evolutivo" en el cual especuló sobre la posibilidad de que el cerebro use el mismo tipo de mecanismos que usa la evolución para aprender. Su técnica fue uno de los primeros intentos por utilizar algoritmos evolutivos para entrenar redes neuronales.

Historia de la Computación Evolutiva

- Aunque los primeros intentos por evolucionar programas se remontan a los 1950s y 1960s, fue hasta los 1980s en que se obtuvieron resultados satisfactorios.
  Hicklin (1986) y Fujiki (1986) usaron expresiones-S en LISP para representar programas cuyo objetivo era resolver problemas de teoría de juegos.

Historia de la Computación Evolutiva

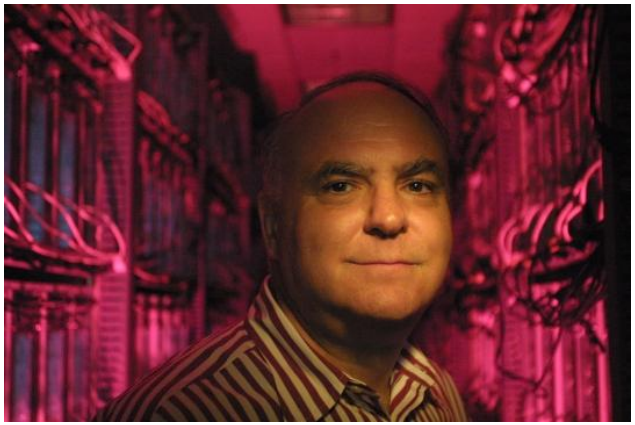
- Nichal Lynn Cramer (1985) y posteriormente, John R. Koza (1989) propusieron (de forma independiente) el uso de una representación de árbol en la que se implementó un operador de cruza para intercambiar sub-árboles entre los diferentes programas de una población generada al azar (con ciertas restricciones impuestas por la sintaxis del lenguaje de programación utilizado).

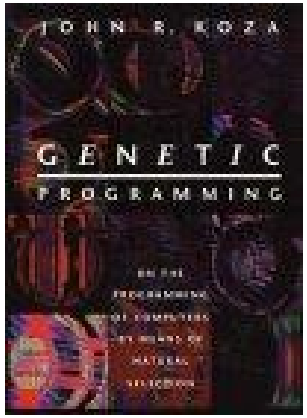
Historia de la Computación Evolutiva

- La diferencia fundamental entre el trabajo de Cramer y el de Koza es que el primero usó una función de aptitud interactiva (es decir, el usuario debía asignar a mano el valor de aptitud de cada árbol de la población), mientras el segundo logró automatizarla.

Historia de la Computación Evolutiva

- La propuesta de Koza fue la que se impuso a la larga, y más tarde se denominó "Programación Genética". Hoy en día es muy popular y cuenta con una amplia gama de aplicaciones.





Historia de la Computación Evolutiva

- Thomas S. Ray desarrolló a principios de los 1990s un simulador muy original en el que evolucionaban programas en lenguaje ensamblador, los cuales competían por ciclos de CPU de una computadora, a la vez que intentaban reproducirse (o sea, copiarse a sí mismos) en la memoria de dicha computadora.

Historia de la Computación Evolutiva

- En el simulador de Ray, denominado "Tierra", se partía de un programa único con la capacidad de auto-replicarse, al que se denominaba "ancestro". Con base en este programa se generaban "criaturas" nuevas (segmentos de código), las cuales a su vez se podían sub-dividir para dar origen a nuevas criaturas.

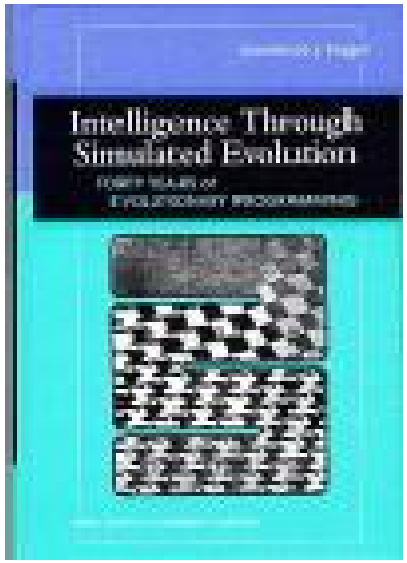
Historia de la Computación Evolutiva

- Tierra es uno de los pocos intentos por simular un ecosistema con el propósito expreso de observar los comportamientos que emergen de la dinámica evolutiva del mismo.

Tres son los paradigmas principales que conforman la computación evolutiva:

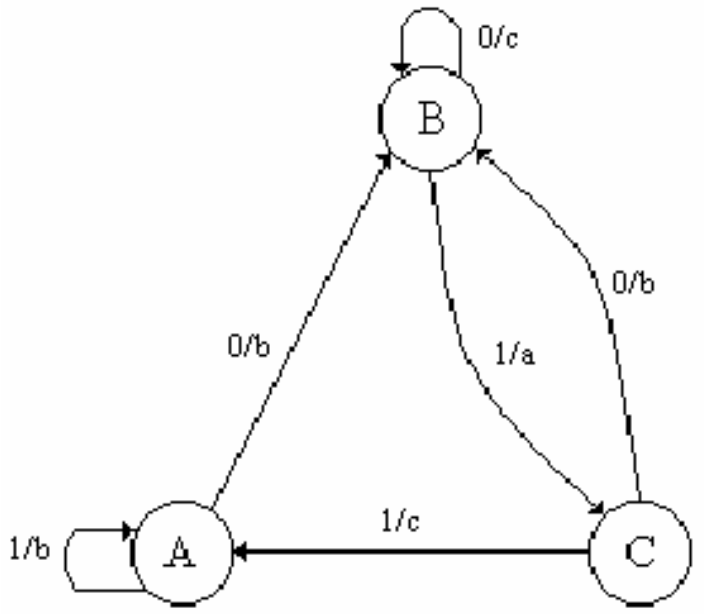
- Programación evolutiva
- Estrategias Evolutivas
- Algoritmos Genéticos

- Lawrence J. Fogel propuso en 1960 una técnica denominada "programación evolutiva", en la cual la inteligencia se ve como un comportamiento adaptativo.



Ejemplo de Programación Evolutiva

Autómata finito de 3 estados.
Los símbolos a la izquierda del "/"
son de entrada, y los de la derecha
son de salida. El estado inicial es C.



Ejemplo de Programación Evolutiva

Estado Actual	C	B	C	A	A	B
Símbolo de Entrada	0	1	1	1	0	1
Estado siguiente	B	C	A	A	B	C
Símbolo de salida	b	a	c	b	b	a

Ejemplo de Programación Evolutiva

Para este ejemplo, existen 5 tipos de mutadores: cambiar un símbolo de salida, cambiar una transición, agregar un estado, borrar un estado y cambiar el estado inicial.

Ejemplo de Programación Evolutiva

El borrado de un estado y el cambio del estado inicial sólo se permiten en máquinas de más de un estado. Las mutaciones se eligen con respecto a una distribución probabilística, que es normalmente uniforme.

- La programación evolutiva enfatiza los nexos de comportamiento entre padres e hijos, en vez de buscar emular operadores genéticos específicos (como en el caso de los algoritmos genéticos).

- El algoritmo básico de la programación evolutiva es el siguiente:

- Generar aleatoriamente una población inicial.
- Se aplica mutación.
- Se calcula la aptitud de cada hijo y se usa un proceso de selección mediante torneo (normalmente estocástico) para determinar cuáles serán las soluciones que se retendrán.

- La programación evolutiva es una abstracción de la evolución al nivel de las especies, por lo que no se requiere el uso de un operador de recombinación (diferentes especies no se pueden cruzar entre sí). Asimismo, usa selección probabilística.

- Algunas aplicaciones de la programación evolutiva son:
  - Predicción
  - Generalización
  - Juegos
  - Control automático
  - Problema del viajero
  - Planeación de rutas

- Diseño y entrenamiento de redes neuronales
- Reconocimiento de patrones

- Las estrategias evolutivas fueron desarrolladas en 1964 por un grupo de estudiantes alemanes de ingeniería encabezado por Ingo Rechenberg. Su objetivo era resolver problemas hidrodinámicos de alto grado de complejidad

- La versión original (1+1)-EE usaba un solo padre y con él se generaba un solo hijo. Este hijo se mantenía si era mejor que el padre, o de lo contrario se eliminaba (a este tipo de selección se le llama **extintiva**, porque los peores individuos tienen una probabilidad de cero de ser seleccionados).

- En la (1+1)-EE, un individuo nuevo es generado usando:

$$\vec{x}^{t+1} = \vec{x}^{t} + N(0, \vec{\sigma})$$

donde t se refiere a la generación (o iteración) en la que nos encontramos, y $N(0, \vec{\sigma})$ es un vector de números Gaussianos independientes con una media de cero y desviació estándar $\vec{\sigma}$

Ejemplo de una (1+1)-EE

Supongamos que queremos **maximizar**:

$$f(\vec{x}_1, \vec{x}_2) = 100(\vec{x}_1^2 - \vec{x}_2)^2 + (1 - \vec{x}_1)^2$$

$$\text{donde } \text{-2.048} \leq \vec{x}_1, \vec{x}_2 \leq 2.048$$

Ahora, supongamos que nuestra población consiste del siguiente individuo (generado de forma aleatoria):

$$(\vec{x}^t, \vec{\sigma}) = \text{(-1.0,1.0),(1.0,1.0)}$$

Ejemplo de una (1+1)-EE

Supongamos también que las mutaciones producidas son las siguientes:

$$\vec{x}_1^{t+1} = \vec{x}_1^t + \text{N}(0,1.0) = \text{-}1.0 + 0.61 = \text{-}0.39$$
$$\vec{x}_2^{t+1} = \vec{x}_2^t + \text{N}(0,1.0) = 1 + 0.57 = 1.57$$

Ejemplo de una (1+1)-EE

Ahora, comparamos al padre con el hijo:

Padre:

$$f(\vec{x}^t) = \text{f(-1.0,1.0)} = 4.0$$

Hijo:

$$f(\vec{x}^{t+1}) = \text{f(-0.39,1.57)} = 201.416$$

Dado que:

$$201.416 > 4.0$$

el hijo reemplazará al padre en la siguiente generación.

- Ingo Rechenberg (1973) introdujo el concepto de población, al proponer una estrategia evolutiva llamada $(\mu + 1) - EE$, en la cual hay μ padres y se genera un solo hijo, el cual puede reemplazar al peor padre de la población (selección extintiva).

- Schwefel (1975) introdujo el uso de múltiples hijos en las denominadas $(\mu + \lambda) - EEs$ y $(\mu, \lambda) - EEs$. La notación se refiere al mecanismo de selección utilizado:

- a) En el primer caso, los μ mejores individuos obtenidos de la unión de padres e hijos sobreviven.
- b) En el segundo caso, sólo los μ mejores hijos de la siguiente generación sobreviven.

Convergencia de las Estrategias Evolutivas

- Rechenberg formuló una regla para ajustar la desviación estándar de forma determinística durante el proceso evolutivo de tal manera que el procedimiento convergiera hacia el óptimo.

Convergencia de las Estrategias Evolutivas

Esta regla se conoce como la "regla del éxito 1/5", y en palabras dice:

> *" La razón entre mutaciones exitosas y el total de mutaciones debe ser 1/5. Si es más, entonces debe incrementarse la desviación estándar. Si es menos, entonces debe decrementarse".*

Convergencia de las Estrategias Evolutivas

Formalmente:

$$\sigma(t) = \begin{cases} \sigma(t-n)/c, & \text{si } p_s > 1/5 \\ \sigma(t-n) \cdot c, & \text{si } p_s < 1/5 \\ \sigma(t-n), & \text{si } p_s = 1/5 \end{cases} \quad (1)$$

donde n es el número de dimensiones, t es la generación, p_s es la frecuencia relativa de mutaciones exitosas medida sobre intervalos de (por ejemplo) $10 \cdot n$ individuos, y c=0.817 (este valor fue derivado teóricamente por Schwefel). $\sigma(t)$ se ajusta cada n mutaciones.

Convergencia de las Estrategias Evolutivas

- Thomas Bäck (1996) derivó una regla de 1/7 para las $(\mu+\lambda)$-EEs.

Auto-Adaptación

- En las estrategias evolutivas se evoluciona no sólo a las variables del problema, sino también a los parámetros mismos de la técnica (es decir, las desviaciones estándar). A esto se le llama "auto-adaptación".

Auto-Adaptación

- Los padres se mutan usando las siguientes fórmulas:

$$\sigma'(i) = \sigma(i) \times exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)) x'(i) = x(i) + N(0, \sigma'(i))$$

donde τ y τ' son constantes de proporcionalidad que están en función de n.

- Los operadores de recombinación de las estrategias evolutivas pueden ser:

- **Sexuales**: el operador actúa sobre 2 individuos elegidos aleatoriamente de la población de padres.
- **Panmíticos**: se elige un solo padre al azar, y se mantiene fijo mientras se elige al azar un segundo padre (de entre toda la población) para cada componente de sus vectores.

- Las estrategias evolutivas simulan el proceso evolutivo al nivel de los individuos, por lo que la recombinación es posible. Asimismo, usan normalmente selección determinística.

Estrategias Evolutivas vs Programación Evolutiva

- La Programación Evolutiva usa normalmente selección estocástica, mientras que las estrategias evolutivas usan selección determinística.
- Ambas técnicas operan a nivel fenotípico.

Estrategias Evolutivas vs Programación Evolutiva

- La programación evolutiva es una abstracción de la evolución al nivel de las especies, por lo que no se requiere el uso de un operador de recombinación (diferentes especies no se pueden cruzar entre sí). En contraste, las estrategias evolutivas son una abstracción de la evolución al nivel de un individuo, por lo que la recombinación es posible.

- Algunas aplicaciones de las estrategias evolutivas son:

- Problemas de ruteo y redes
- Bioquímica
- Optica
- Diseño en ingeniería
- Magnetismo

- Los algoritmos genéticos (denominados originalmente "planes reproductivos") fueron desarrollados por John H. Holland a principios de los 1960s. Su motivación principal fue el aprendizaje de máquina.

- El algoritmo genético enfatiza la importancia de la cruza sexual (operador principal) sobre el de la mutación (operador secundario), y usa selección probabilística.

- El algoritmo básico es el siguiente:

- Generar (aleatoriamente) una población inicial
- Calcular aptitud de cada individuo
- Seleccionar (probabilísticamente) con base en aptitud
- Aplicar operadores genéticos (cruza y mutación) para generar la siguiente población
- Ciclar hasta que cierta condición se satisfaga

- La representación tradicional es la cadena binaria del tipo:

$$\underbrace{0110}_{Cadena1}\ \underbrace{1101}_{Cadena2}\ \underbrace{0011}_{Cadena3}\ \underbrace{1001}_{Cadena4}$$

- A la cadena se le llama "cromosoma". A cada posición de la cadena se le denomina "gene" y al valor dentro de esta posición se le llama "alelo".

- Para poder aplicar el Algoritmo Genético se requiere de los 5 componentes básicos siguientes:

- Una representación de las soluciones potenciales del problema
- Una forma de crear una población inicial de posibles soluciones (normalmente un proceso aleatorio)
- Una función de evaluación que juegue el papel del ambiente, clasificando las soluciones en términos de su "aptitud"

- Operadores genéticos que alteren la composición de los hijos que se producirán para las siguientes generaciones
- Valores para los diferentes parámetros que utiliza el Algoritmo Genético (tamaño de la población, probabilidad de cruza, probabilidad de mutación, número máximo de generaciones, etc.)

Algoritmos Genéticos vs otras técnicas evolutivas

- El AG usa selección probabilística al igual que la Programación Evolutiva, y en contraposición a la selección determinística de las Estrategias Evolutivas.
- El AG usa representación binaria para codificar las soluciones a un problema, por lo cual se evoluciona el genotipo y no el fenotipo como en la Programación Evolutiva o las Estrategias Evolutivas.

Algoritmos Genéticos vs otras técnicas evolutivas

- El operador principal en el AG es la cruza, y la mutación es un operador secundario. En la Programación Evolutiva, no hay cruza y en las Estrategias Evolutivas es un operador secundario.

Algoritmos Genéticos vs otras técnicas evolutivas

- Ha sido demostrado (Rudolph, 1994) que el AG requiere de elitismo para poder converger al óptimo.
- Los AGs no son, normalmente, auto-adaptativos.

- Algunas aplicaciones de los AGs son las siguientes:

- Optimización (estructural, de topologías, numérica, combinatoria, etc.)
- Aprendizaje de máquina (sistemas clasificadores)
- Bases de datos (optimización de consultas)
- Reconocimiento de patrones (por ejemplo, imágenes)
- Generación de gramáticas (regulares, libres de contexto, etc.)
- Planeación de movimientos de robots
- Predicción

- Para simular el proceso evolutivo en una computadora se requiere:

- Codificar las estructuras que se replicarán.
- Operaciones que afecten a los "individuos".
- Una función de aptitud.
- Un mecanismo de selección.

Diferencias de las técnicas evolutivas con respecto a las tradicionales

- Las técnicas evolutivas usan una población de soluciones potenciales en vez de un solo individuo, lo cual las hace menos sensibles a quedar atrapadas en mínimos/máximos locales.
- Las técnicas evolutivas no necesitan conocimientos específicos sobre el problema que intentan resolver.

Diferencias de las técnicas evolutivas con respecto a las tradicionales

- Las técnicas evolutivas usan operadores probabilísticos, mientras las técnicas tradicionales utilizan operadores determinísticos.
- Aunque las técnicas evolutivas son estocásticas, el hecho de que usen operadores probabilísticos no significa que operen de manera análoga a una simple búsqueda aleatoria.

Ventajas de las Técnicas Evolutivas

- Simplicidad Conceptual.
- Amplia aplicabilidad.
- Superiores a las técnicas tradicionales en muchos problemas del mundo real.
- Tienen el potencial para incorporar conocimiento sobre el dominio y para hibridizarse con otras técnicas de búsqueda/optimización.

Ventajas de las Técnicas Evolutivas

- Pueden explotar fácilmente las arquitecturas en paralelo.
- Son robustas a los cambios dinámicos.
- Generalmente pueden auto-adaptar sus parámetros.
- Capaces de resolver problemas para los cuales no se conoce solución alguna.

Comparaciones entre Técnicas

	Estrategias Evolutivas	Programación Evolutiva	Algoritmo Genético
Representación	Real	Real	Binaria
Función de Aptitud	Valor de la Función Objetivo	Valor de la Función Objetivo ajustada	Valor de la Función Objetivo ajustada
Auto-Adaptación	Desviaciones Estándar y ángulos de rotación	Ninguna Varianzas (PE-estándar), Coeficientes de correlación (meta-PE)	Ninguna
Mutación	Gaussiana, operador principal	Gaussiana, operador único	Inversión de bits, operador secundario

Comparaciones entre Técnicas

	Estrategias Evolutivas	Programación Evolutiva	Algoritmo Genético
Recombinación	Discreta e intermedia, sexual y panmítica, importante para la auto-adaptación	Ninguna	Cruza de z-puntos, cruza uniforme, únicamente sexual, operador principal
Selección	Determinística, extintiva o basada en la preservación	Probabilística, extintiva	Probabilística, basada en la preservación
Restricciones	Restricciones arbitrarias de desigualdad	Ninguna	Límites simples mediante el mecanismo de codificación
Teoría	Velocidad de Convergencia para casos especiales, (1+1)-ES, (1+λ)-ES, Convergencia Global para $(\mu + \lambda)$-ES	Velocidad de Convergencia para casos especiales, (1+1)-PE, Convegencia Global para meta-PE	Teoría de los Esquemas, Convergencia Global para la versión elitista

Críticas a las Técnicas Evolutivas

- Criticadas en sus orígenes (1960s) por los investigadores de la IA simbólica. Se creía que una simple búsqueda aleatoria podía superarlas.
- La programación automática fue considerada una "moda pasajera" en IA y el enfoque evolutivo fue visto como "un intento más" por lograr lo imposible.

Críticas a las Técnicas Evolutivas

- Actualmente, todavía muchas personas creen que un AG funciona igual que una técnica "escalando la colina" que comienza de varios puntos. Se ha demostrado que esto no es cierto.

Críticas a las Técnicas Evolutivas

- Las técnicas sub-simbólicas (redes neuronales y computación evolutiva) gozan de gran popularidad entre la comunidad científica en general, excepto por algunos especialistas de IA clásica que las consideran "mal fundamentadas" e "inestables".