Análisis y Diseño de Algoritmos.

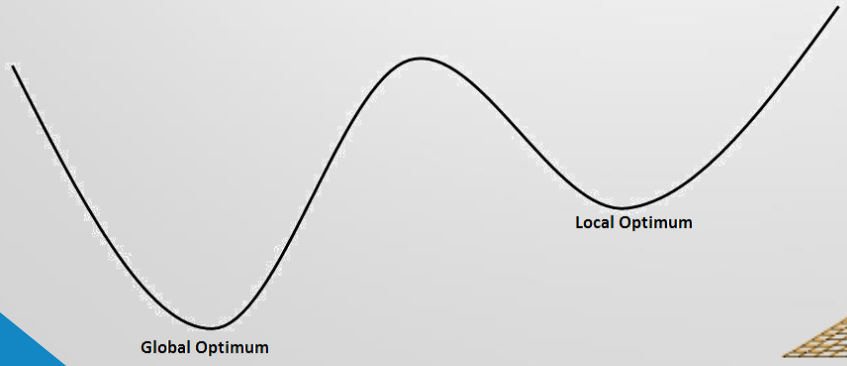
Sesión 14. 25 de Noviembre de 2015.

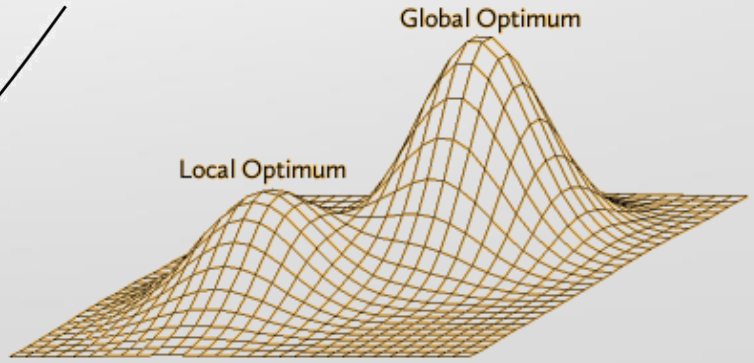
Maestría en Sistemas Computacionales.

Por: Hugo Iván Piza Dávila.

Optimización local

- El método de optimización por Hill-Climbing es muy usado por ser fácil de programar y eficiente en su ejecución.
- Sin embargo, su desventaja principal es que en muchos problemas suele caer en óptimos locales.



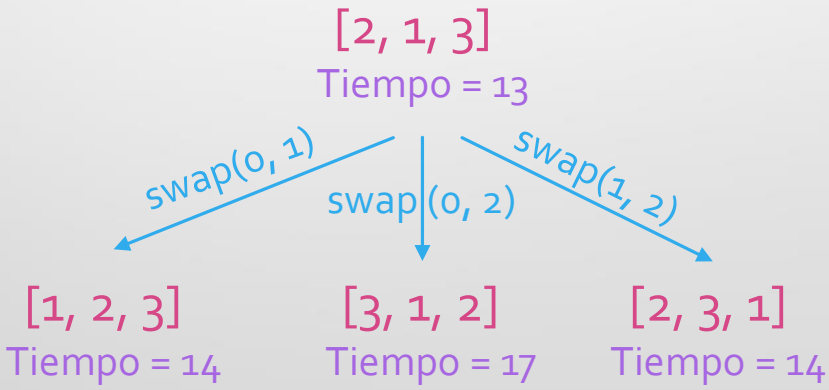


Optimización local

- Consideremos el problema del Agente Viajero.
- HC realiza intercambios de dos elementos aleatorios de la solución. Si el tiempo obtenido con el intercambio no mejoró al actual, la solución regresa como estaba.
- En el ejemplo que se trabajó (*start* = 0) el mínimo global es 10 y se obtiene con la solución [3, 2, 1].
- ¿Qué sucede si en algún momento de la ejecución del algoritmo obtenemos una de estas permutaciones?
  - [2, 1, 3] y [1, 3, 2]

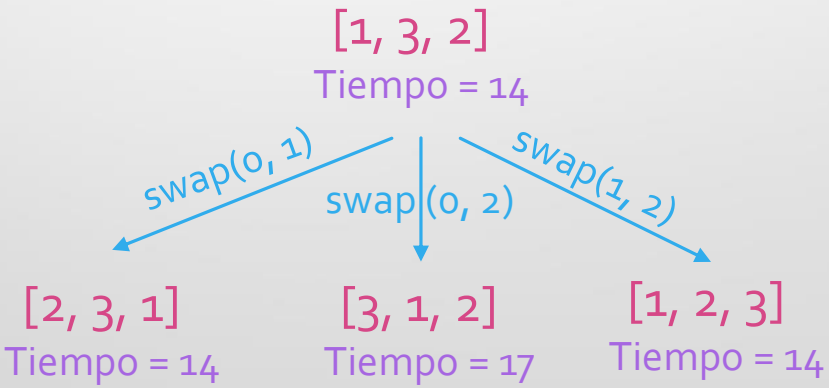
Optimización local

- El siguiente árbol muestra las tres permutaciones que se pueden obtener a partir de [2, 1, 3].
- Ninguna mejora el tiempo.
- Por lo tanto, el algoritmo se quedará con [2, 1, 3] que es la mejor solución local (entre sus vecinos).



Optimización local

- Algo semejante sucede con la solución [1, 3, 2].
- Una forma de arreglar este problema es elegir la nueva permutación si no "empeoró" el tiempo. En este caso, se pueden elegir [2, 3, 1] y [1, 2, 3] que nos llevarán a la postre a la solución óptima [3, 2, 1].
- Sin embargo, esta solución no atiende el caso anterior.



Optimización Global

- Para atender este problema, necesitamos métodos de optimización global.
- Una característica de estos métodos es que trabajan con varias soluciones a la vez que pueden influir entre sí.
- Dos métodos estocásticos de optimización global muy populares son:
  1. Algoritmos Genéticos: es uno de los tres paradigmas de la Computación Evolutiva.
  2. Particle Swarm Optimization: población de partículas que viajan por el espacio de búsqueda a una velocidad cambiante.

Computación Evolutiva

- La *computación evolutiva* comprende un conjunto de técnicas inspiradas en la teoría de la *evolución natural*:
  - Existe una población de individuos.
    - Un individuo encapsula una solución.
    - Esto disminuye las probabilidades de quedar en óptimos locales.
  - Los individuos sufren algún tipo de transformación
    - Cruza: operación entre dos individuos.
    - Mutación: semejante a lo que hace Hill-Climbing.
  - Se seleccionan los individuos más aptos, de acuerdo a una función de aptitud.

Computación Evolutiva

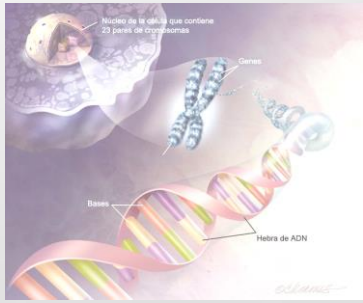
- Leclerc: el cuerpo sufre cambios orgánicos; el primero en especular en la existencia de un ancestro común entre el hombre y los simios.
- Lamarck: un cambio en el ambiente produce un cambio (hereditario) en el organismo, que conduce al mayor uso o desuso de ciertos órganos
- Darwin: los cambios en los individuos suceden de forma hereditaria, para hacer a los nuevos individuos más aptos para sobrevivir.
  - Teoría de la Combinación
  - Débil en explicar la ocurrencia de cambios repentinos en una especie
- Mendel: experimentó con chícharos y descubrió las leyes que rigen el paso de una característica de una especie a otra (herencia), incluyendo el concepto de gen *dominante* y *recesivo*.

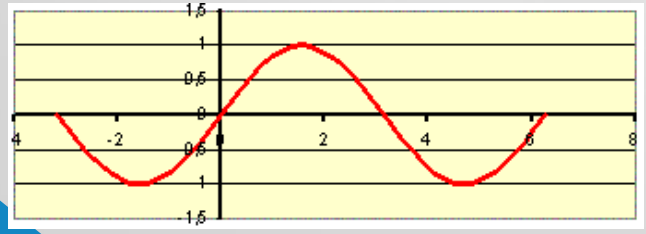
Computación Evolutiva

- Sutton: determinó que los cromosomas eran el lugar donde se almacenaban las características hereditarias; los cromosomas contienen genes.
- Neo-Darwinismo: la historia de la gran mayoría de la vida en nuestro planeta puede ser explicada a través de muchos procesos estadísticos que actúan sobre y dentro de las poblaciones y especies: la reproducción, la mutación, la competencia y la selección.
- Paradigmas de la computación evolutiva:
  - Programación Evolutiva
  - Estrategias Evolutivas
  - Algoritmos genéticos

Algoritmos Genéticos

- John H. Holland. Años 60s. Aprendizaje de máquina.
- Método estocástico de búsqueda ciega de soluciones casi-óptimas.
- Genera mucha soluciones posibles a través de generaciones
- Solución = {genotipo, fenotipo}
- Permite la exploración y explotación
  - Menos vulnerable a caer en óptimos locales
- ¿Cuándo conviene utilizarlos?
  - Funciones donde las buenas soluciones son adyacentes.



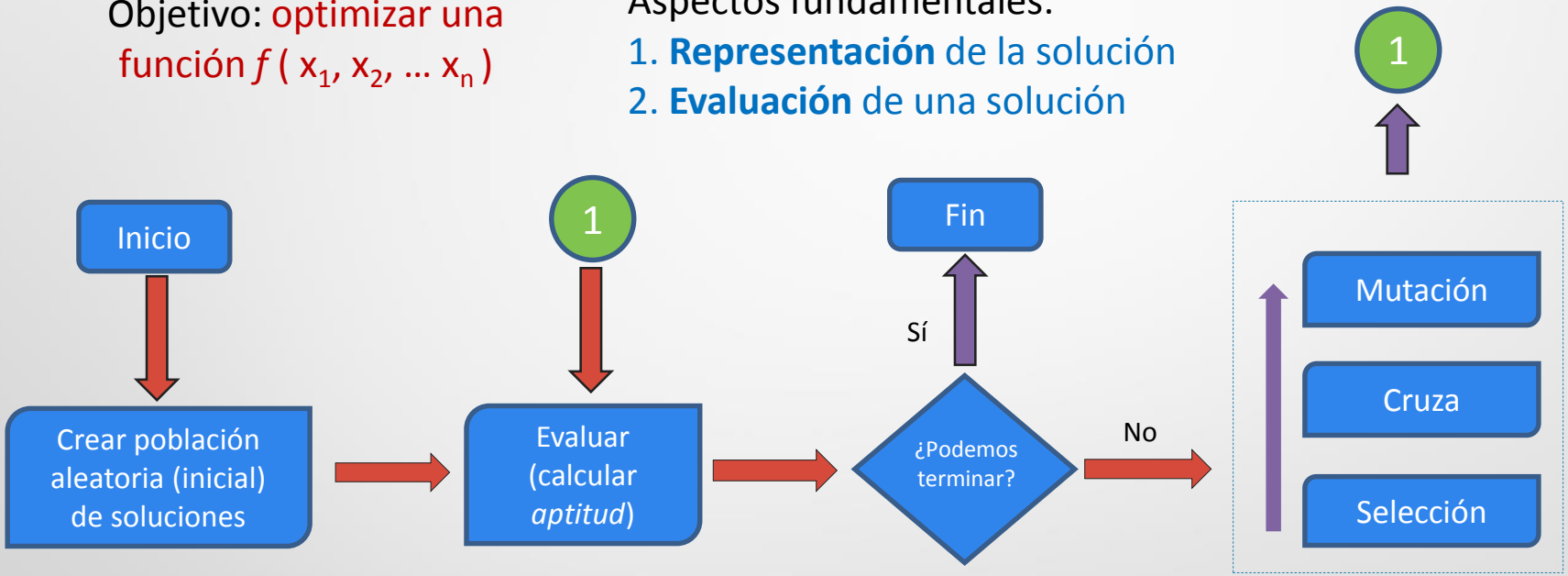


√

Algoritmos Genéticos

Objetivo: optimizar una función $f(x_1, x_2, \ldots x_n)$

Aspectos fundamentales:
1. **Representación** de la solución
2. **Evaluación** de una solución



	$(x_1, x_2, \ldots x_n)$	
1	11010100001011	0.35
2	10001010110101	0.87
n	00000101000001	0.78

Generación 1

1) La mejor solución es suficientemente buena?

2) ¿La generación actual es mayor a MAX?

11010101010101	
10001010110101	
11001100110110	

Generación *k*

Algoritmos Genéticos

1) Crear población inicial. Crear genotipos aleatorios
2) Convertir de genotipo a fenotipo.
3) Cálculo de la aptitud. Opera a nivel fenotipo
   - Obtiene el resultado de la función objetivo y lo normaliza al rango [0…1.0]
4) Selección. De acuerdo a la aptitud
   - Es posible que un individuo no se seleccione.
   - Es posible que un individuo se seleccione más de una vez.
   - *Elitista*: el mejor individuo se selecciona sin competir.

Algoritmos Genéticos

4) Cruza o recombinación. Opera a nivel genotipo
   - Sucede con una probabilidad alta (0.8). *Elitista*: no cruza al mejor individuo.
   - Los individuos k, k + 1 se *combinan* o *aparean* y dan lugar a nuevos individuos k', k' + 1 que los sustituyen.
5) Mutación. Opera a nivel genotipo
   - Sucede con una probabilidad baja (0.2). *Elitista*: no muta al mejor individuo.
   - El individuo k sufre un cambio y da lugar al nuevo individuo k' que lo sustituye.

Algoritmos Genéticos

- Existen muchas técnicas para representación, evaluación de aptitud, selección, cruza y mutación, apropiadas para cada tipo de problema.
- En esta sesión veremos dos de cada una, apropiadas para:
  1. Optimizar una función matemática de dos variables.
  2. Atender el problema del Agente Viajero.

Representación del Genotipo

- Representación tradicional: binaria
  - Cada x_i del fenotipo está codificado en un segmento de la cadena de bits dada por el genotipo.
  - Cada segmento puede ser de diferente longitud en función del rango de valores que puede recibir cada x_i.
  - Ejemplo. Si el genotipo es de 8 bits y el rango es: [-5 .. 5], la cadena 1010 0101 codifica al fenotipo:
    - $x_1 = -5 + {}^{10}/_{15} \cdot (5 - -5) = -5 + 6.667 = 1.667$
    - $x_2 = -5 + {}^{5}/_{15} \cdot (5 - -5) = -5 + 3.333 = -1.667$

Representación del Genotipo

- De acuerdo a Holland, es preferible tener muchos genes con pocos alelos posibles (0, 1) que lo opuesto.
  - En genética es más común encontrar cromosomas largos y pocos alelos por posición.
  - Favorece la construcción de *esquemas* y con ello, la diversidad:
    - Por ejemplo, todos los individuos que siguen estos esquemas tienen aptitud alta: 0xx10x01, x11x00xx
- Una desventaja de la representación binaria es el *risco de Hamming*: dos valores adyacentes en el espacio de búsqueda difieren en su representación en más de un bit.
  - Por ejemplo, 5 y 6 difieren en dos bits: 101, 110.

Representación del Genotipo

- Si queremos codificar números reales con buena precisión, necesitamos una cadena binaria tan larga que producirá un desempeño pobre del AG.
  - Los teóricos afirman que los alfabetos pequeños (binario, digital) son más efectivos que los grandes (enteros, reales).
  - Los prácticos han mostrado con muchas aplicaciones reales que los alfabetos grandes son más eficientes.
- ¿Qué alfabeto usar para hacer eficiente el algoritmo pero no errático? Que no se pierda la diversidad.

Minimizar Styblinski con AG

- Recordando la función Styblinski & Tang:
  - $S(x) = 0.5(x_1^4 - 16x_1^2 + 5x_1 + x_2^4 - 16x_2^2 + 5x_2)$
  - Sujeto a: $-5.0 \leq x_1, x_2 \leq 5.0$
  - $\vec{x}^* = [-2.903535, -2.903534]$. $S(\vec{x}^*) = -78.3323314$
- Evaluación de aptitud:
  - Mientras menor sea el resultado de la función, mayor la aptitud.
  - Si se conocen el mínimo y máximo global (casi nunca), la fórmula puede seguir una regla de tres tomando esos valores.
  - En la siguiente fórmula se estima que los valores obtenidos oscilarán entre -100.0 y 1000.0: $Fitness(x_1, x_2) = 1.0 - {}^{100+S(x)}/_{1100}$

Minimizar Styblinski con AG

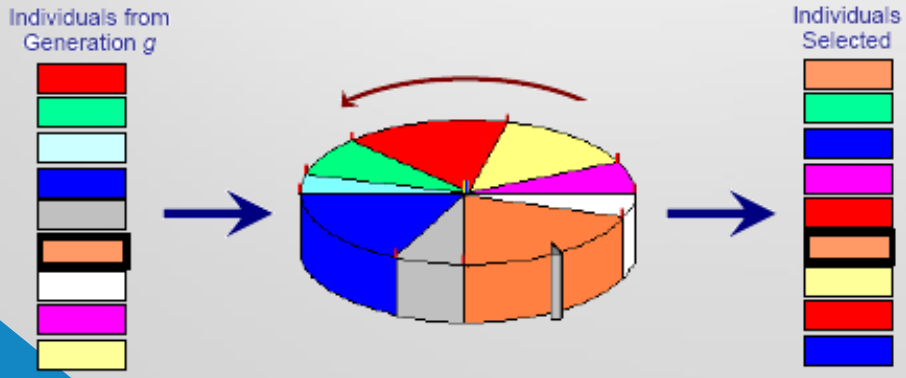
- La representación del genotipo será con 12 dígitos: los 6 dígitos más significativos para x_1 y los otros 6 para x_2.
  - Ejemplo. La cadena 123456 789012 codifica al fenotipo:
    - $x_1 = -5 + {}^{123456}/_{999999} \cdot (5 - -5) = -5 + 1.2345 = -4.8765$
    - $x_2 = -5 + {}^{789012}/_{999999} \cdot (5 - -5) = -5 + 7.8901 = \ \ 2.8901$
- El tamaño de la población suele estar en función de la longitud del genotipo.
  - En esta práctica, será 20 veces el tamaño del genotipo.

Minimizar Styblinski con AG

- La selección será por Jerarquías.
  - Los individuos se guardan en una lista ordenada de menor a mayor de acuerdo a la aptitud.
  - En función de la posición de cada individuo en dicha lista, se calcula un *valor esperado* en un rango de valores positivos. Algunos autores sugieren el rango: [0.9…1.1].
  - Valor(i) = 0.9 + (1.1 – 0.9) * $^{i}/_{N-1}$

Minimizar Styblinski con AG

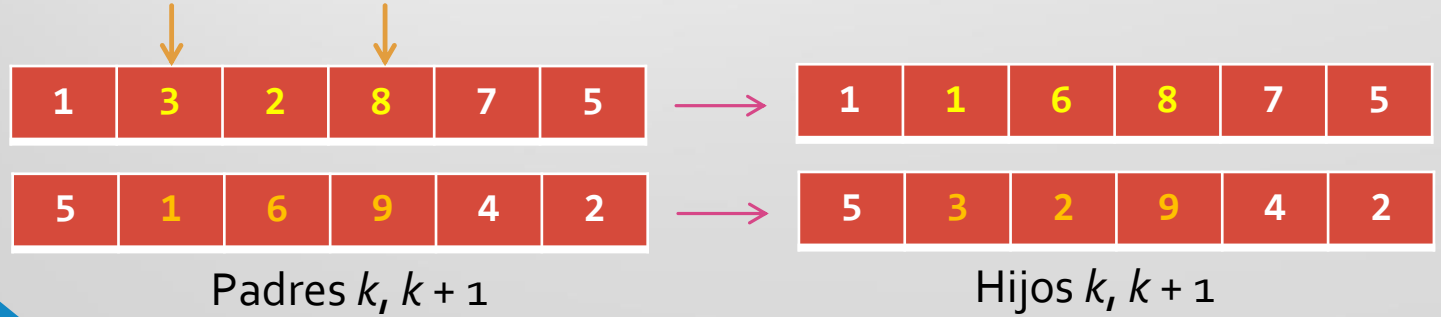
- Se ejecuta la selección por Ruleta. Repetir N veces:
  1. Generar un aleatorio entre 0 y el tamaño de la población.
  2. Elegir al individuo (de la lista original) que haga que la suma de los valores esperados supere al aleatorio.
  3. Si el número aleatorio es 1.7, el individuo elegido de la tabla es el segundo: 0.95 + 1.10 > 1.7.
     - Los más aptos tienen probabilidades de ser seleccionados más de una vez.



i	Aptitud	Valor
0	0.2	0.90
1	0.7	1.10
2	0.4	0.96
3	0.5	1.03

Minimizar Styblinski con AG

- La cruza será Uniforme de 2 puntos. La probabilidad será 0.8.
- Ejecutar lo siguiente por cada par de individuos adyacentes excepto el mejor individuo y su vecino (Elitista):
  1. Si un aleatorio en el rango [0..1] es mayor que 0.8, continuar con el siguiente par de individuos.
  2. Obtener dos índices aleatorios i_1, i_2, tal que $i_1 < i_2$:
  3. Desde $i = i_1$ hasta i_2:
     - Si un aleatorio en el rango [0..1] es menor que 0.5, intercambiar los alelos en la posición i de los dos individuos.



Minimizar Styblinski con AG

- La mutación será Uniforme. La probabilidad será 0.2.
- Ejecutar lo siguiente por cada individuo excepto el mejor (Elitista):
  1. Si un aleatorio en el rango [0..1] es mayor que 0.2, continuar con el siguiente individuo.
  2. Seleccionar una posición aleatoria del genotipo.
  3. Calcular un nuevo valor aleatorio en el rango permitido [0..9].

Agente Viajero con AG

- Parámetros generales (ajustables posteriormente):
  - Probabilidad de cruza = 0.8
  - Probabilidad de mutación = 0.2
  - Generaciones: número de nodos x 10,000
  - Tamaño de la población: ½ número de nodos, garantizando que el resultado sea par.
  - Tamaño del genotipo: número de nodos – 1.

Agente Viajero con AG

- Representación:
  - Genotipo = Fenotipo: arreglo de enteros que guarda la ruta.
- Evaluación de aptitud:
  - Ejecutar el algoritmo que calcula la distancia de la ruta, normalizando el resultado al rango [0.0…1.0]:
    - A mayor distancia, menor aptitud: $1 - {}^{\text{Distancia}}/_{\text{MAX}}$
  - Asignar como distancia máxima de una ruta (MAX) el peso máximo que puede tener una arista multiplicado por el número de nodos del grafo.

Agente Viajero con AG

- La selección será por Ruleta.
  - Cada individuo tendrá un atributo valor esperado que será la aptitud dividida entre la suma de todas las aptitudes, multiplicado por el tamaño de la población.

Aptitud	Valor
0.2	4(0.2/1.8) = 0.444
0.7	4(0.7/1.8) = 1.555
0.4	4(0.4/1.8) = 0.888
0.5	4(0.5/1.8) = 1.111

  - Este valor se asigna cada vez que se actualiza la aptitud.
  - Después se sigue el algoritmo explicado antes para seleccionar los individuos que pasan a la siguiente generación.
- Ruleta es más eficiente que Jerarquías porque no ordena. Sin embargo, los valores pueden ser tan contrastantes (0.4 y 1.5) que reduce la exploración en espacios de búsqueda donde hay individuos poco aptos pero que puedan llevarnos al óptimo global: convergencia prematura.

Agente Viajero con AG

- La mutación será por Intercambio Recíproco seleccionando dos índices aleatorios (como se hizo con Hill-Climbing).

- La cruza será por Order Crossover con dos índices aleatorios.
  - Se copia la subcadena elegida en el individuo hijo
  - Los demás espacios se llenan con los valores que no están en la subcadena, y en el orden en que aparecen en el otro padre.