Análisis y Diseño de Algoritmos.

Sesión 12. 11 de Noviembre de 2015.

Maestría en Sistemas Computacionales.

Por: Hugo Iván Piza Dávila.

Problema de Optimización

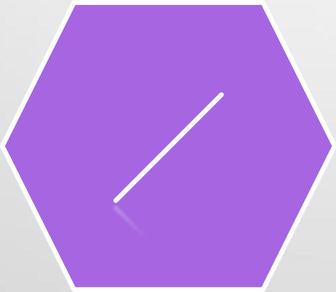
- Lo definimos formalmente así:
  - $\vec{x}^* = \arg\min_{\vec{x}} f(\vec{x})\,, tal\ que$:

    $f: \Re^n \to \Re$ es la función objetivo

    tal que: $\vec{x}_{min} \leq \vec{x} \leq \vec{x}_{max}$
- Encontrar el argumento $\vec{x} = (x_1, x_2, \ldots x_n)$ que minimice la función objetivo y se ubique dentro de la zona factible.

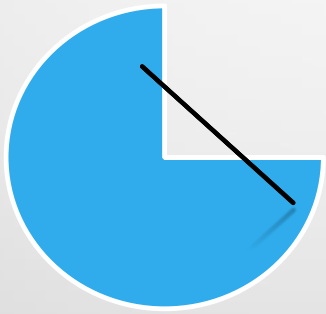
Conjuntos convexos

- **C** es un conjunto **convexo** si para dos puntos cualquiera $p_1, p_2 \in C$, el segmento rectilíneo que une estos puntos está también dentro del conjunto.
- Ejemplos:



Conjuntos convexos

- Ejemplos de conjuntos **no convexos**:

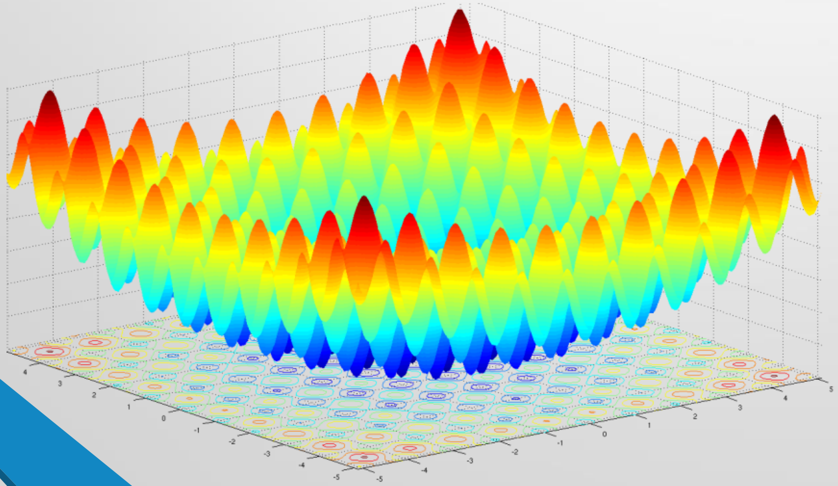


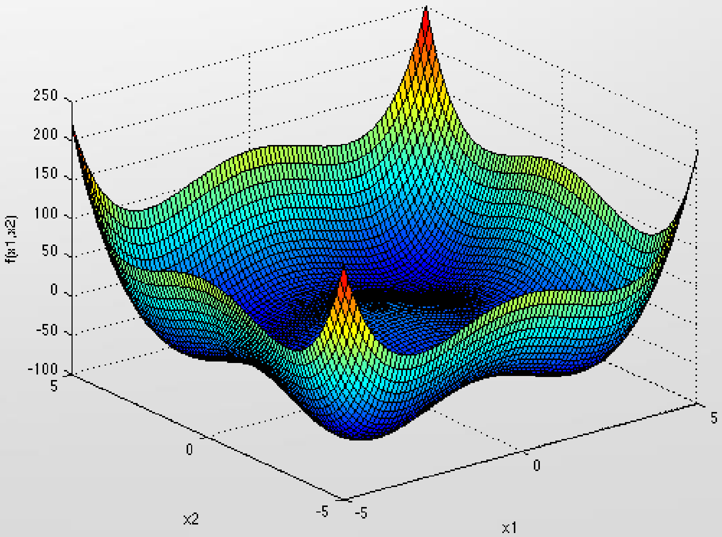
Óptimo Local vs. Global

- Un óptimo local representa la mejor solución en una zona del espacio de búsqueda (es mejor que todos sus vecinos), pero no es la mejor de todas.
- Existen algoritmos que se quedan atrapados en óptimos locales.
- Un conjunto no convexo puede tener varios óptimos locales pero sólo uno global.
- En un conjunto convexo, óptimo local = óptimo global.

Óptimo Local vs. Global

- Rastrigin tiene muchos máximos y mínimos locales.
- Styblinski-Tang tiene cuatro mínimos locales pero sólo un mínimo global.





Técnicas clásicas de Optimización

- Existen muchas técnicas clásicas para resolver problemas de optimización con ciertas características.
  - Optimización lineal: Método Simplex
  - Optimización no lineal: Descenso Empinado, Gradiente Conjugado, Newton-Raphson
- Problema principal: requieren información que no siempre está disponible.
  - Gradiente Conjugado: primera derivada de la función objetivo.
  - Newton: la segunda derivada de la función objetivo.

Técnicas clásicas de Optimización

- Por lo tanto, si la función objetivo no es diferenciable varios de estos métodos no pueden aplicarse.
- Peor aún, en algunos problemas del mundo real, la función objetivo no está disponible en forma explícita.
- Es importante conocer las técnicas clásicas porque, cuando el problema puede adecuarse a ellas, son la mejor opción para resolverlas: en tiempo y precisión.

Técnicas Heurísticas

- Heurística (griego) es: *encontrar* o *descubrir.*
- Una heurística es una técnica que busca soluciones buenas a un costo computacional razonable.
- No hay garantías que la solución encontrada sea *optima* o *factible*.
- Ejemplos:
  - Búsqueda Tabú — Recocido Simulado
  - Hill-Climbing — Montecarlo

Técnicas Heurísticas

- ¿Cuándo recurrir a ellas?

1. Las técnicas clásicas no se pueden adaptar al problema por alguna de las razones antes vistas.
2. El mejor algoritmo que lo resuelve no es polinomial.
   - Lidiamos con espacios de búsqueda grandes como el caso del ***Agente Viajero***, y los algoritmos más eficientes conocidos utilizan tiempo exponencial $(N-1)!$

Hill-Climbing

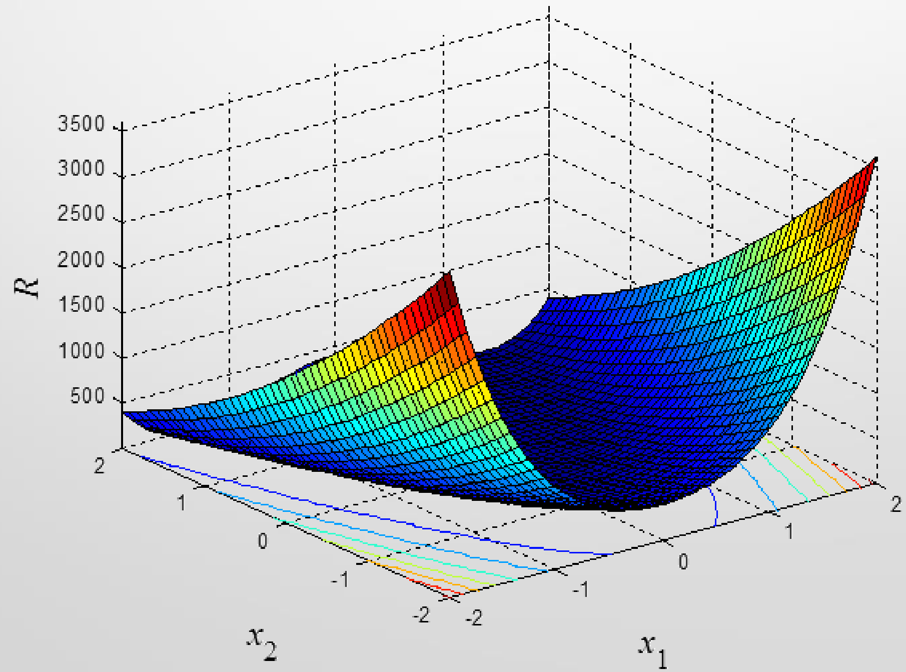
- Heurística muy popular por ser fácil de entender e implementar.
- En muchos problemas entrega buenos resultados en un tiempo razonable.
- Trabaja con una solución cuya *aptitud* va mejorando conforme avanza el algoritmo.
- La *aptitud* es un valor numérico que expresa qué tan buena o mala es la solución. Normalmente está en el rango [0.0..1.0].
- Puede quedar atorado en óptimos locales.

Hill-Climbing

1. Crea un solución aleatoria dentro de la zona factible.
2. Evalúa la aptitud de la solución.
3. Define un incremento aleatorio.
4. Evalúa la aptitud obtenida con el incremento.
5. Si la aptitud obtenida es mejor que la anterior
    a. A la solución actual se le suma el incremento.
    b. El incremento se hace más fino.
6. Regresar al paso 3 mientras no se cumpla la condición de terminación:
    - Se ejecutaron N generaciones.
    - La aptitud es mayor que máximo definido.
    - Se ejecutaron n generaciones consecutivas sin cambios en la aptitud.

Función Rosenbrock

- $R(x) = 100(x_2 - {x_1}^2)^2 + (1 - x_1)^2$
- $\vec{x}_{min} = [-5, -5],\ \vec{x}_{max} = [5, 5]$
- Se conoce: $\vec{x}^* = [1, 1].\ R(\vec{x}^*) = 0.0$



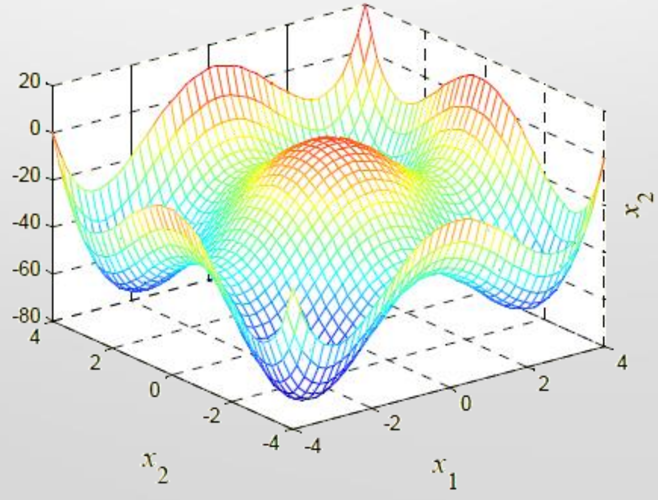
HC para Rosenbrock

1. $\vec{x}$ comienza con valores aleatorios en $[\vec{x}_{min} .. \vec{x}_{max}]$
2. Mientras menor sea R($\vec{x}$), mayor será la aptitud:
   - Aptitud = (1.0 + R($\vec{x}$))$^{-1}$. Aptitud $\in$ [0.0..1.0]
3. El incremento $\delta\vec{x}$ estará en el rango [-½$\Delta\vec{x}$ .. ½$\Delta\vec{x}$]:
   - $\delta\vec{x}$ = -½ + $\Delta\vec{x}$ + rand() · $\Delta\vec{x}$
4. Conforme la aptitud crezca, $\Delta\vec{x}$ será menor:
   - $\Delta\vec{x}$ = ($\vec{x}_{max}$ - $\vec{x}_{min}$) · (1 – bestFitness)
   - Inicialmente, $\Delta\vec{x}$ = ($\vec{x}_{max}$ - $\vec{x}_{min}$), porque bestFitness = 0.

- La corrida es **exitosa** si: $|\vec{x}^* - \vec{x}| \leq 0.001$.

Función Styblinski and Tang

- $S(x) = 0.5(x_1^4 - 16x_1^2 + 5x_1 + x_2^4 - 16x_2^2 + 5x_2)$
- Sujeto a: $-5.0 \leq x_1, x_2 \leq 5.0$
- $\vec{x}^* = [-2.903535, -2.903534]$. $S(\vec{x}^*) = -78.3323314$
  - ¿Cómo calcular la aptitud a partir de $S(x)$?
- ¿Qué tan frecuente es exitoso Hill-Climbing?



HC para el Agente Viajero

- Crear una permutación aleatoria de nodos.
- Repetir N generaciones
  - Calcular dos índices aleatorios diferentes.
  - Intercambiar los elementos en tales índices.
  - Si la solución intercambiada no mejoró la anterior, volver a la solución anterior.
- ¿Es mejor que un algoritmo que genere N combinaciones aleatorias y registre el mejor tiempo?