# Inteligencia Artificial
# Técnicas Incompletas de Búsqueda de Soluciones

Nicolás Rojas-Morales

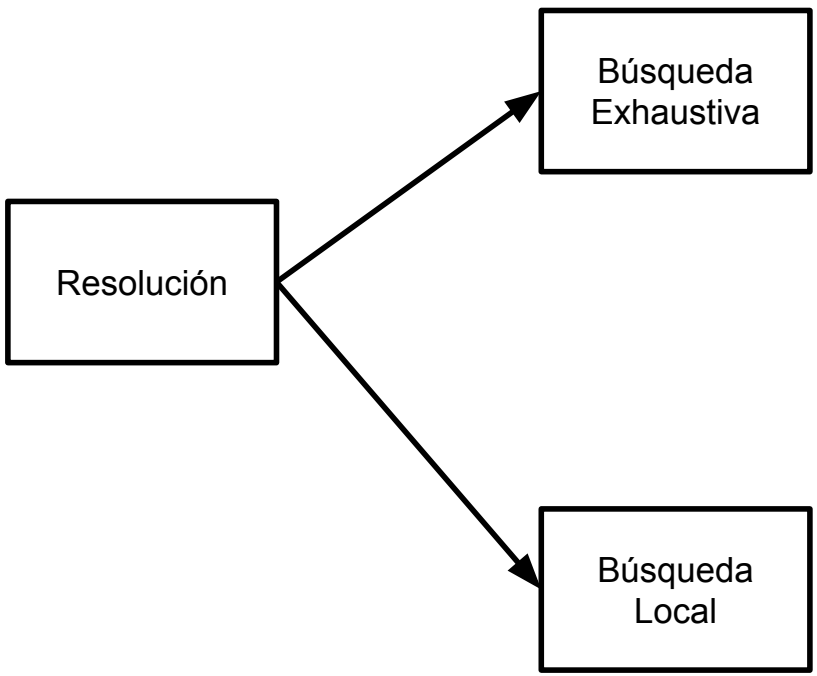
Departamento de Informática
Universidad Técnica Federico Santa María

# Conceptos Generales

# Introducción

- Algoritmos que realizan Búsqueda Exhaustiva garantizan obtener el óptimo global
- Para problemas de la vida real o de mayor complejidad (NP-Completos), no pueden encontrar solución
- Sacrificamos el óptimo global por obtener soluciones de buena calidad en un tiempo reducido

# Clasificación de Técnicas de Resolución



# Búsqueda Local/Incompleta/Alternativa

¿Qué es Heurística?

- Criterios, principios o métodos que permiten determinar entre un conjunto de posibilidades, aquella que promete ser la más eficaz para resolver un problema.

# Búsqueda Local/Incompleta/Alternativa

¿Qué es Heurística?

- Criterios, principios o métodos que permiten determinar entre un conjunto de posibilidades, aquella que promete ser la más eficaz para resolver un problema.
- Representan el compromiso entre:
  - Necesidad de utilizar criterios simples
  - Distinguir entre buenas y malas elecciones

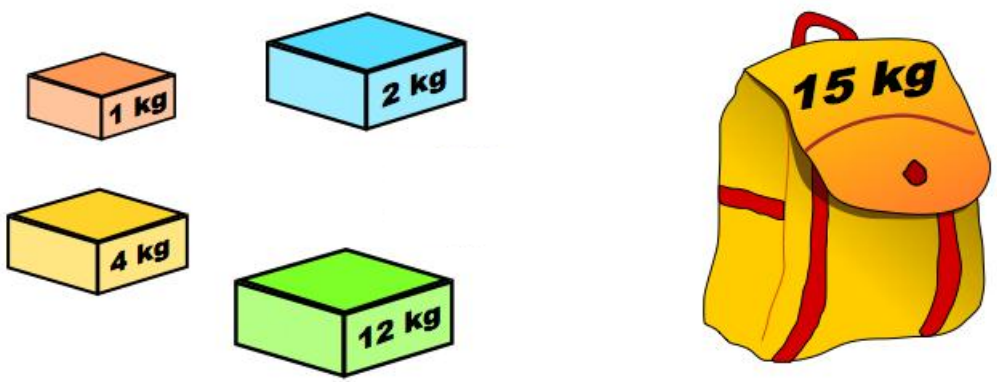
# Búsqueda Local/Incompleta/Alternativa

¿Qué es Heurística?

- Criterios, principios o métodos que permiten determinar entre un conjunto de posibilidades, aquella que promete ser la más eficaz para resolver un problema.
- Representan el compromiso entre:
  - Necesidad de utilizar criterios simples
  - Distinguir entre buenas y malas elecciones
- Un método heurístico puede ser un método empírico utilizado para guiar acciones.
- Su objetivo es encontrar soluciones (c) aceptables.
- Se utilizan porque son, en general, eficientes computacionalmente y/o fáciles de implementar.
- No son muy precisas ni predecibles.

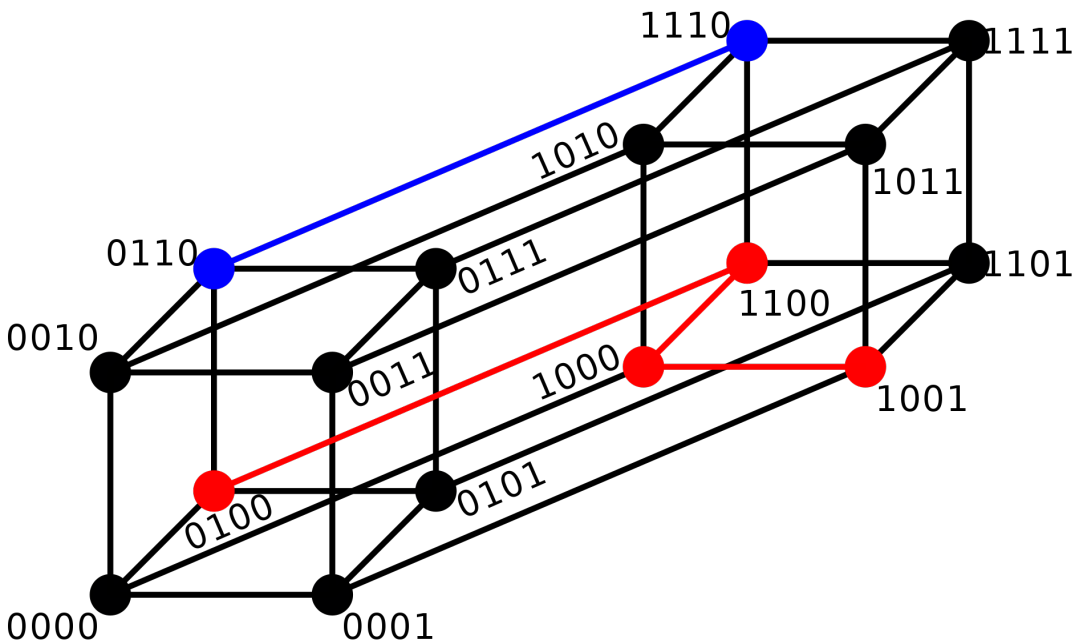
# Problema

- Grandes espacios de búsqueda $\rightarrow$ Explosión combinatoria
- Soluciones factibles e infactibles



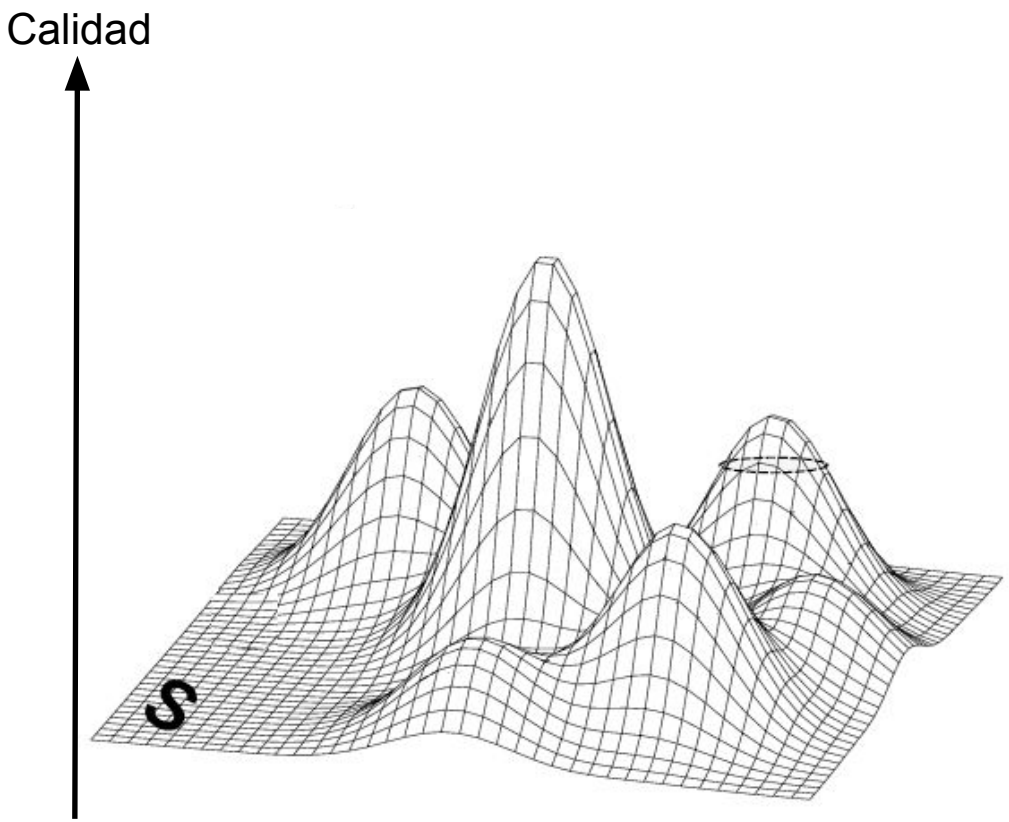
# Problema

- Grandes espacios de búsqueda → Explosión combinatoria
- Soluciones factibles e infactibles

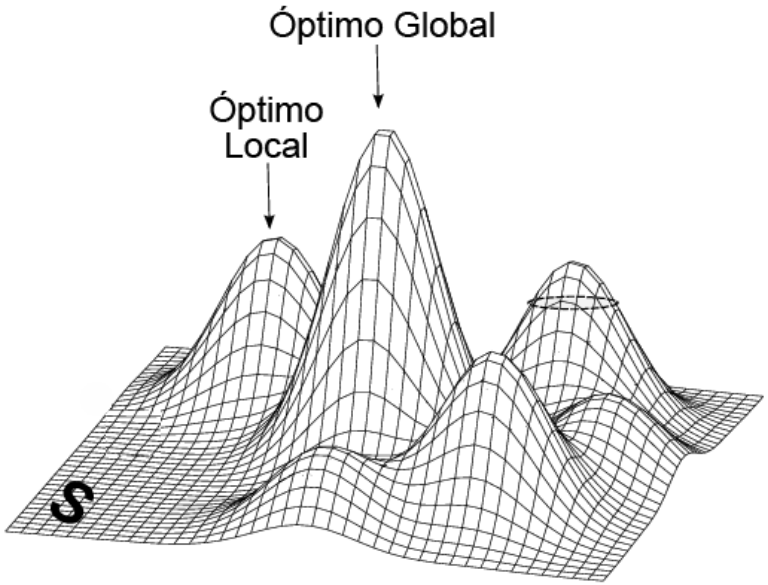


# Problema

- Grandes espacios de búsqueda $\rightarrow$ Explosión combinatoria
- Soluciones factibles e infactibles



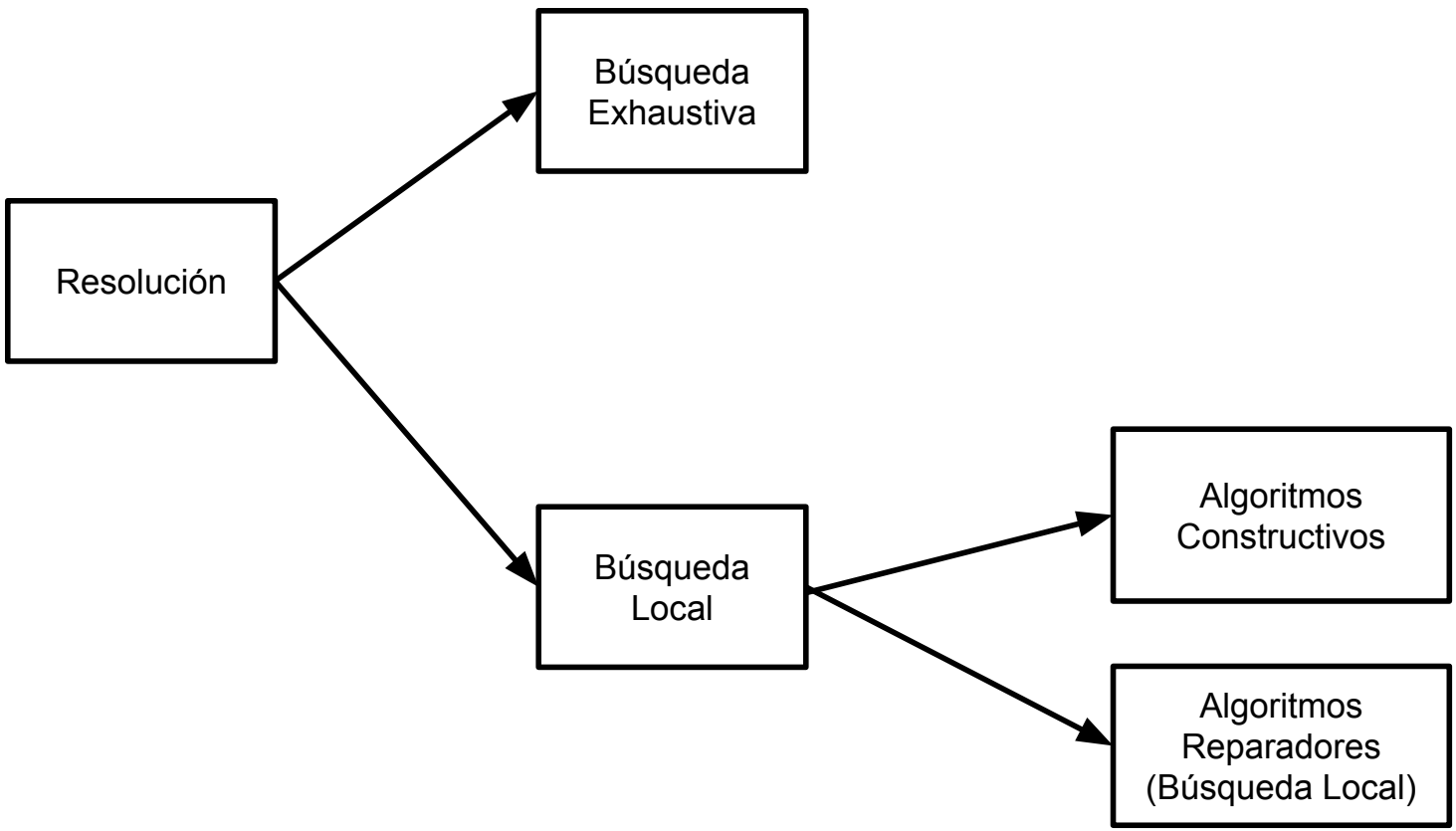
# Búsqueda Local




(problema de maximización)

- Un algoritmo que realiza búsqueda local visita diferentes regiones del espacio de búsqueda
- El objetivo de estos algoritmos es encontrar el mejor óptimo local buscando un balance entre dos aspectos:
  - **Diversificación:** el algoritmo visita diferentes regiones para identificar secciones del espacio de búsqueda que son interesantes
  - **Intensificación:** el algoritmo se enfoca en encontrar soluciones de la mejor calidad posible en la región actual del espacio de búsqueda.

# Clasificación de Técnicas de Resolución



# Conceptos Generales

- solución candidata (c) Se refiere a una instanciación completa (global) para el problema.
  - En estricto rigor no se debería hablar de soluciones puesto que éstas pueden ser *factibles* o *infactibles*.

# Conceptos Generales

- solución candidata (c) Se refiere a una instanciación completa (global) para el problema.
  - En estricto rigor no se debería hablar de soluciones puesto que éstas pueden ser *factibles* o *infactibles*.
- función de evaluación: Es capaz de representar cero, uno ó varios objetivos.
  - Informativa

# Conceptos Generales

- solución candidata (c) Se refiere a una instanciación completa (global) para el problema.
  - En estricto rigor no se debería hablar de soluciones puesto que éstas pueden ser *factibles* o *infactibles*.
- función de evaluación: Es capaz de representar cero, uno ó varios objetivos.
  - Informativa

# Conceptos Generales

- solución candidata (c) Se refiere a una instanciación completa (global) para el problema.
  - En estricto rigor no se debería hablar de soluciones puesto que éstas pueden ser *factibles* o *infactibles*.
- función de evaluación: Es capaz de representar cero, uno ó varios objetivos.
  - Informativa
- representación: Corresponde a la/s solución/es (c).
  - Coloreo: (V, A, V, V, A, R) (Lista Entera/Simbólica)
  - Mochila: (1, 0, 1, 0) (Lista Binaria)
  - TTP: E vs 2*(E-1) (Matriz Entera/Simbólica)

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $A$ | $C$ | $B$ | $D$ | $-C$ | $-B$ | $-D$ |
| $B$ | $D$ | $-A$ | $-C$ | $-D$ | $A$ | $C$ |
| $C$ | $-A$ | $D$ | $B$ | $A$ | $-D$ | $-B$ |
| $D$ | $-B$ | $-C$ | $-A$ | $B$ | $C$ | $A$ |

# Modelado versus Representación

- ¿Para qué modelar?
- ¿Siempre debo representar las soluciones (c) según la forma que se especificó en el modelo?

## Ejemplo: Vendedor viajero

- Modelo

$$x_{ij} = \begin{cases} 1 & \text{Si el vendedor pasa desde la ciudad } i \text{ a la ciudad } j \\ 0 & \text{si no} \end{cases}$$

versus

- Representación
  $x_i$: Visita la ciudad $x_i$ en el $i$-ésimo paso.

# Inteligencia Artificial
# Técnicas Incompletas de Búsqueda de Soluciones

Nicolás Rojas-Morales

Departamento de Informática
Universidad Técnica Federico Santa María

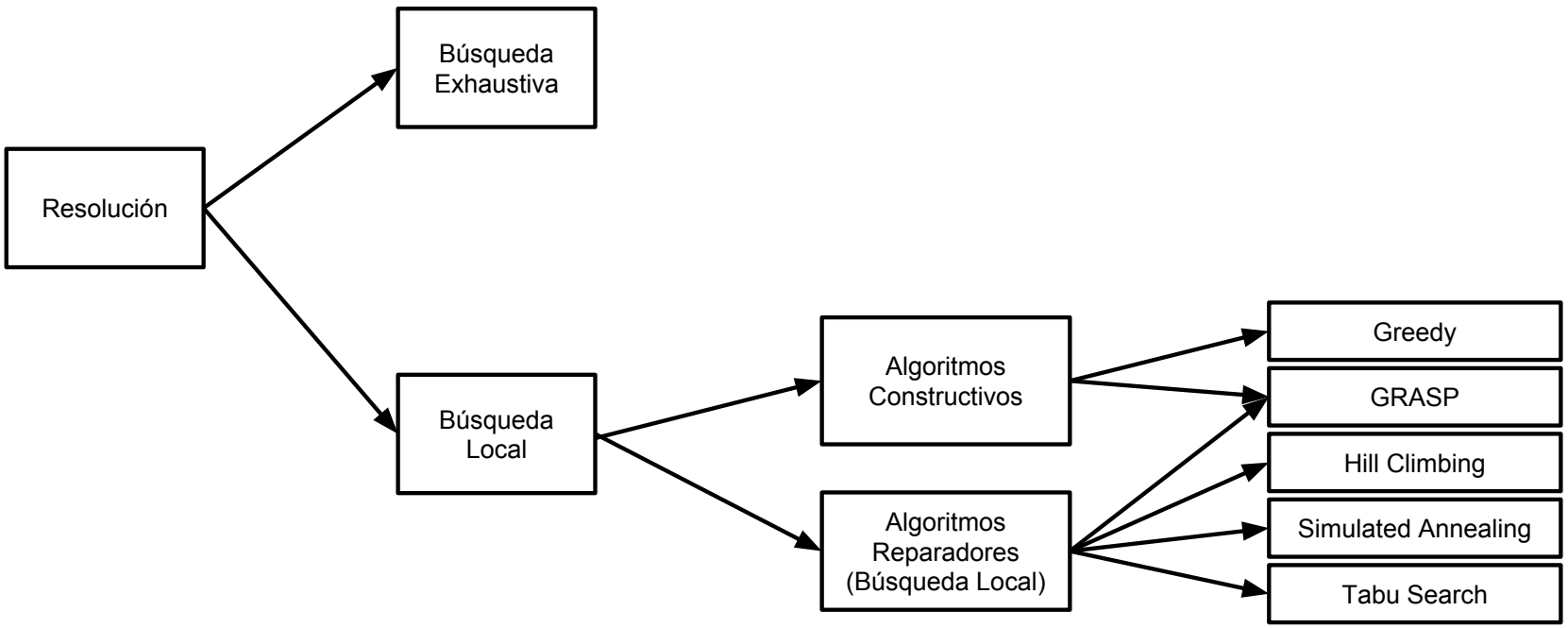
# Algoritmos Constructivos

# Algoritmos Constructivos

## Características Algoritmos Constructivos

- Generan soluciones comenzando desde una solución parcial inicialmente vacía
- Se agregan componentes a la solución parcial hasta que esté completa
- Requieren además de una definición previa de:
  - Punto de Partida: desde donde comienza a construir la solución
  - Función Miope: toma decisiones localmente óptimas guiada por la función de evaluación

# Clasificación de Técnicas de Resolución

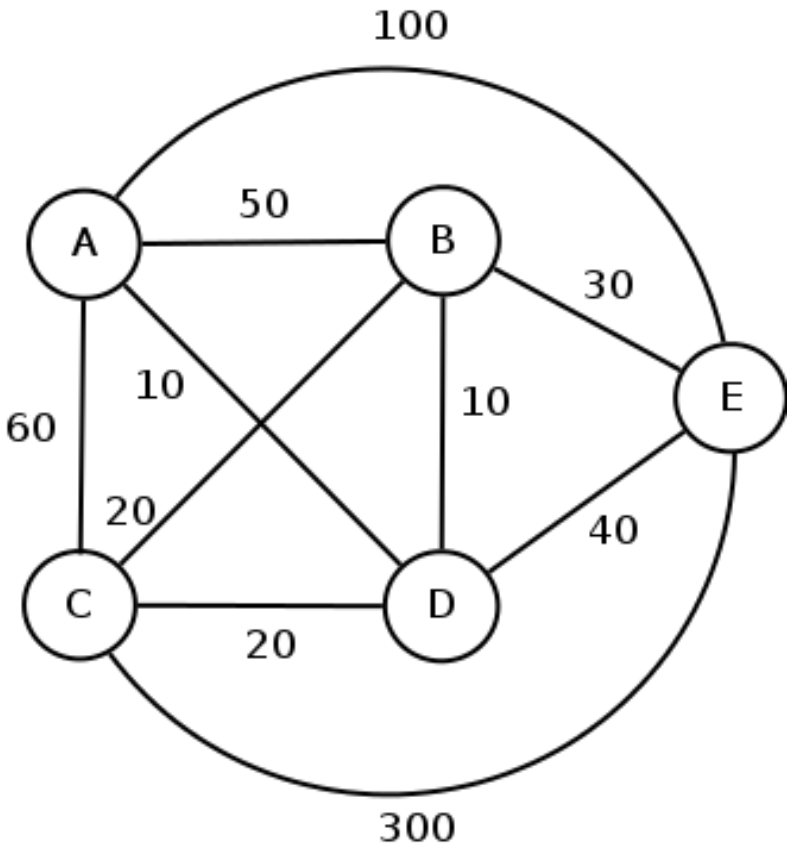


# Greedy

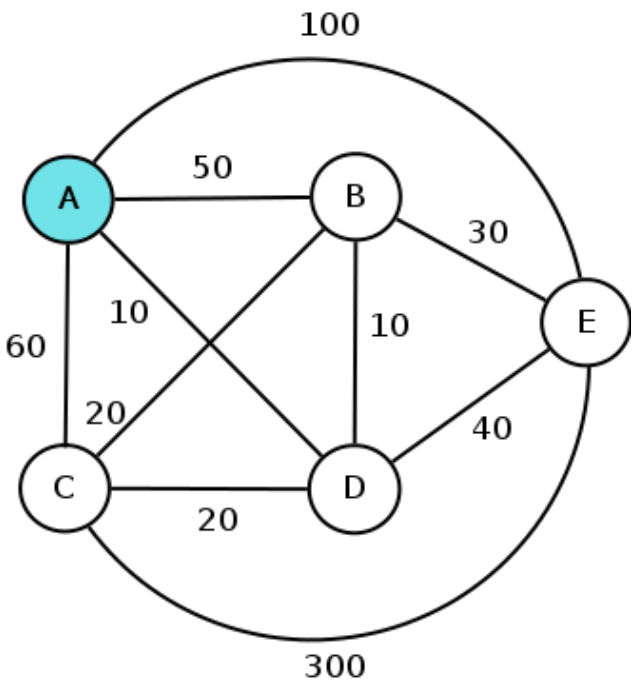
## Greedy

- Algoritmo constructivo que decide de manera localmente óptima qué componentes agregar en la solución candidata
- Para el problema del vendedor viajero:
  - Representación: Tour factible
  - Función Miope: Agregar la siguiente ciudad más cercana no visitada
  - Punto de Partida: Ciudad inicial A
  - Función de Evaluación: Largo del Tour

# Greedy

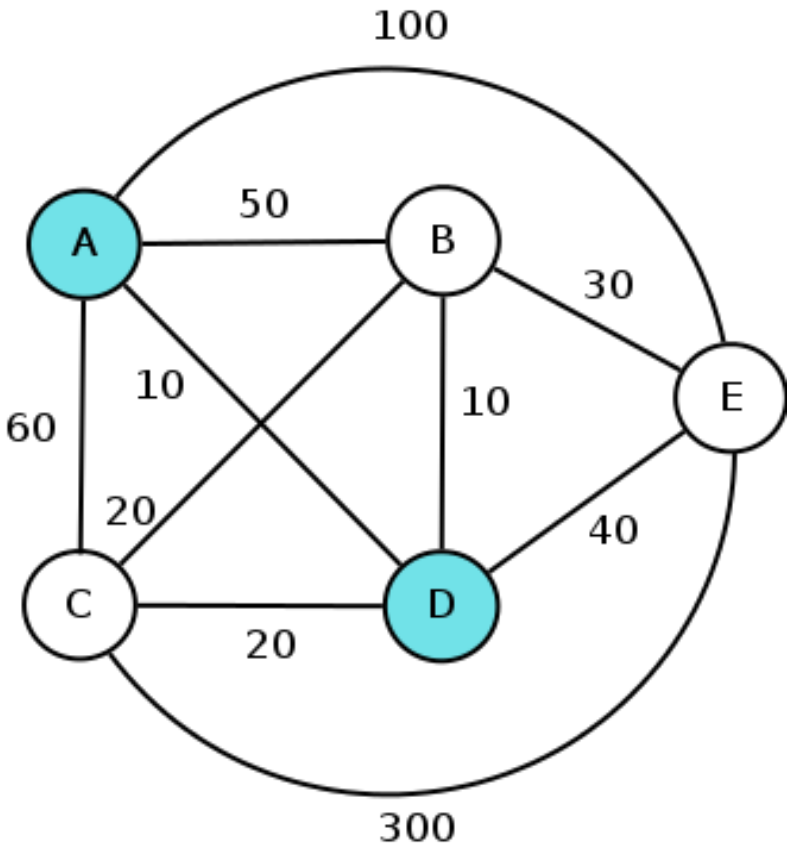


# Greedy



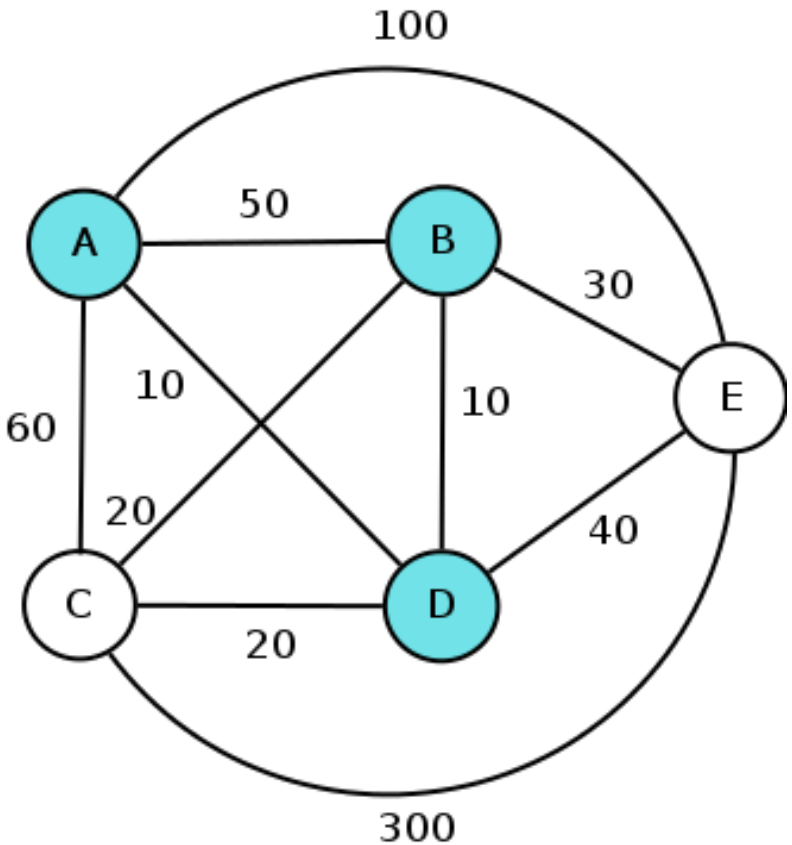

- Tour: A

# Greedy



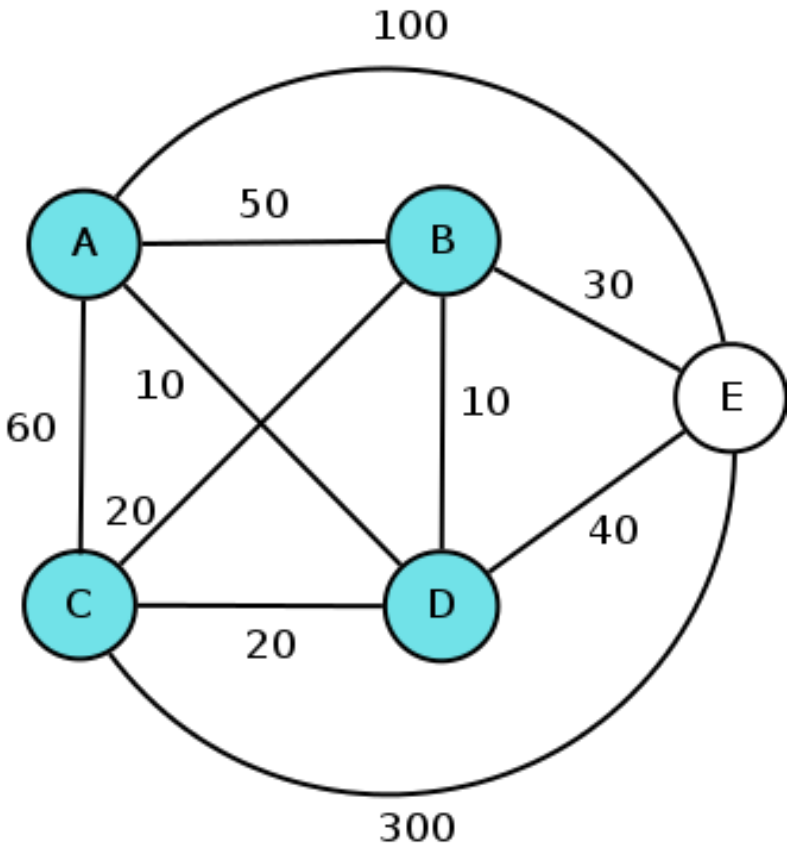

- Tour: A - D

# Greedy



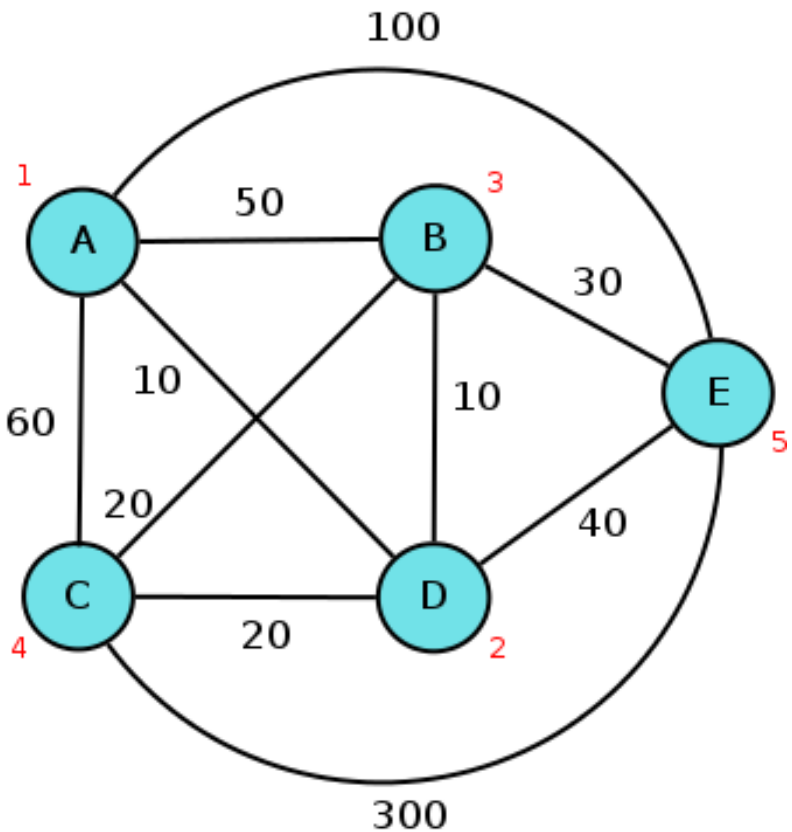

- Tour: A - D - B

# Greedy



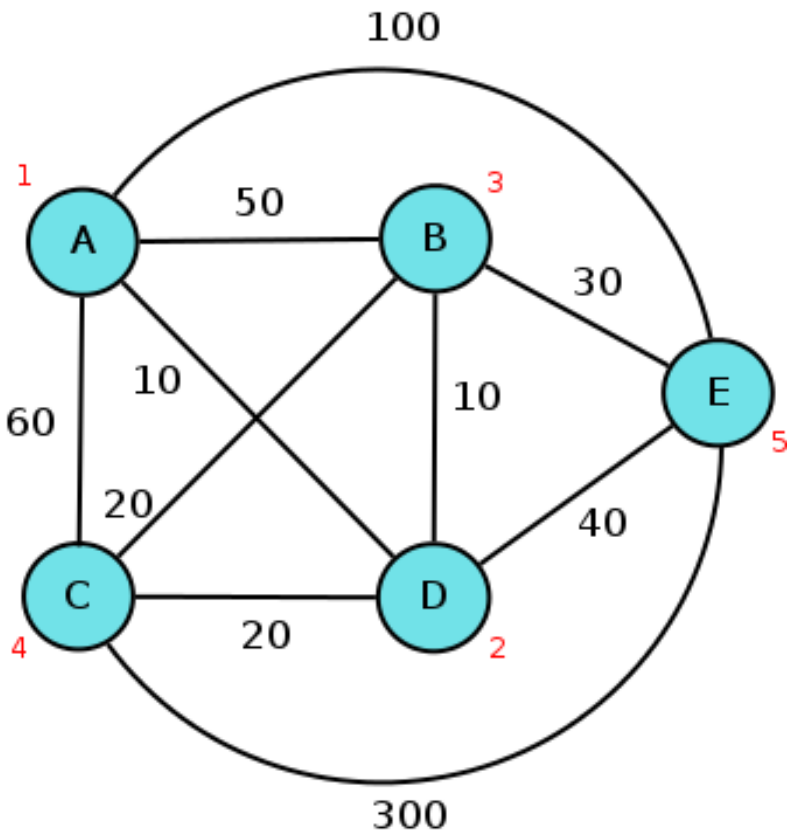

- Tour: A - D - B - C

# Greedy



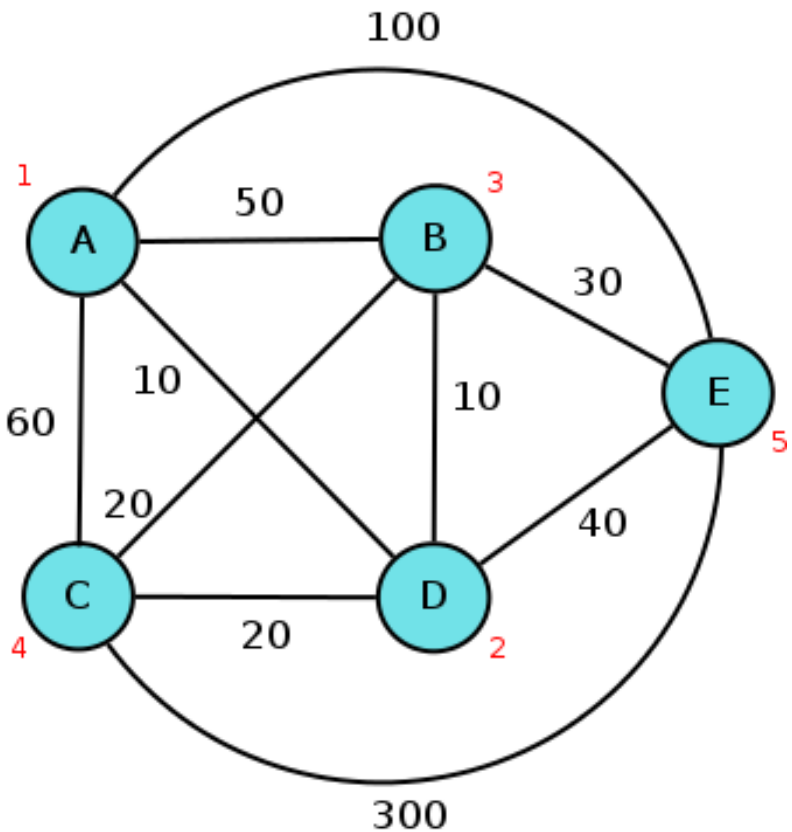

- Tour: A - D - B - C - E
- Costo Tour = 440

# Greedy




- Tour: A - D - B - C - E
- Costo Tour = 440
- Tour: **D** - **B** - C - A - E
- Costo Tour = 230

# Greedy




- Tour: A - D - B - C - E
- Costo Tour = 440
- Tour: **D** - **B** - C - A - E
- Costo Tour = 230
- Tour: A - D - E - B - C
- Costo Tour = 160

# Algoritmos basados en construcción: Mochila

$$\begin{aligned} \text{máx } & 18 \cdot x_1 + 25 \cdot x_2 + 11 \cdot x_3 + 14 \cdot x_4 \\ \text{s.a. } & 2 \cdot x_1 + 2 \cdot x_2 + x_3 + x_4 \leq 3 \\ & x_1, x_2, x_3, x_4 \in \{0, 1\} \end{aligned}$$

- Representación: Lista binaria
- Función objetivo: Ganancia total
  - Función de evaluación?
- Punto de partida: Objeto 3
- Función miope: Agregar el siguiente objeto de mayor ganancia [factible?]

# Inteligencia Artificial
# Técnicas Incompletas de Búsqueda de Soluciones

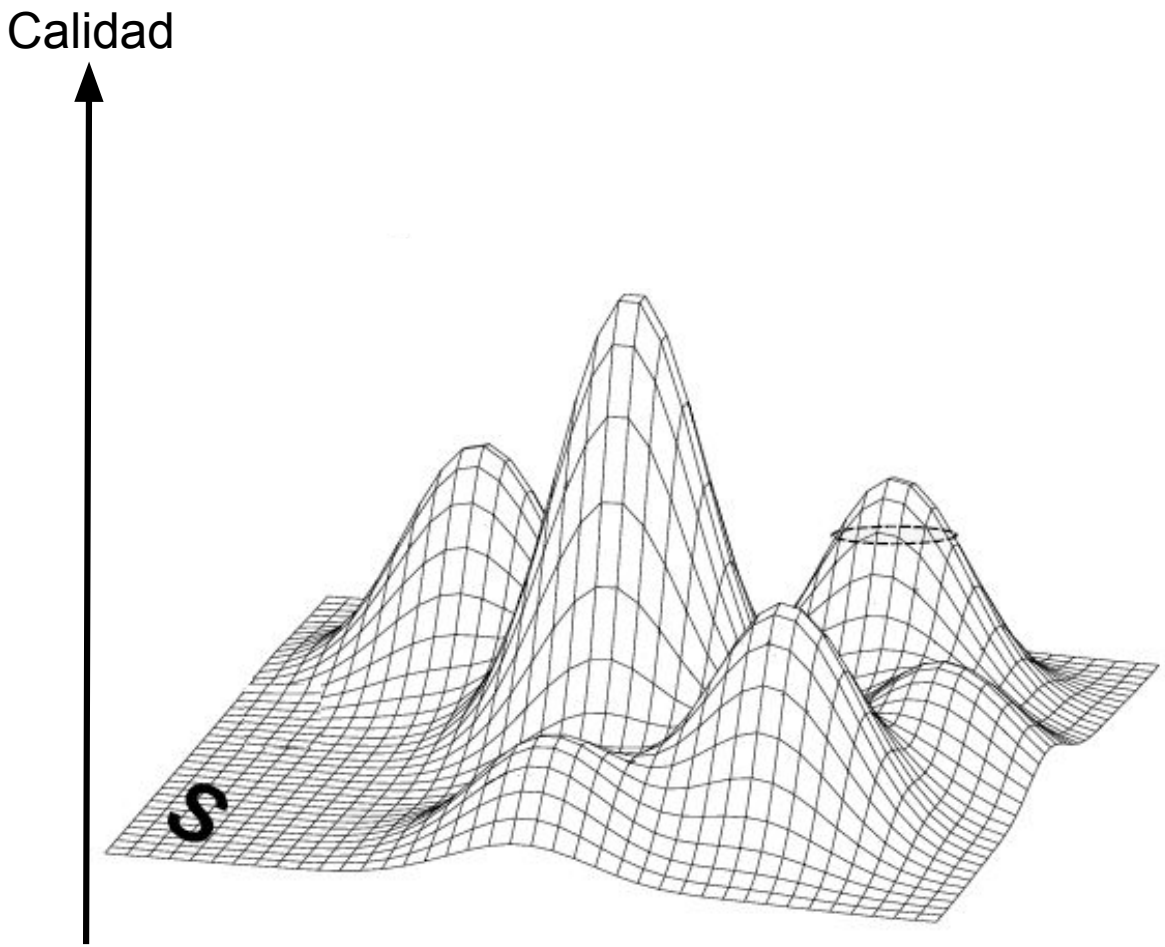
Nicolás Rojas-Morales

Departamento de Informática
Universidad Técnica Federico Santa María

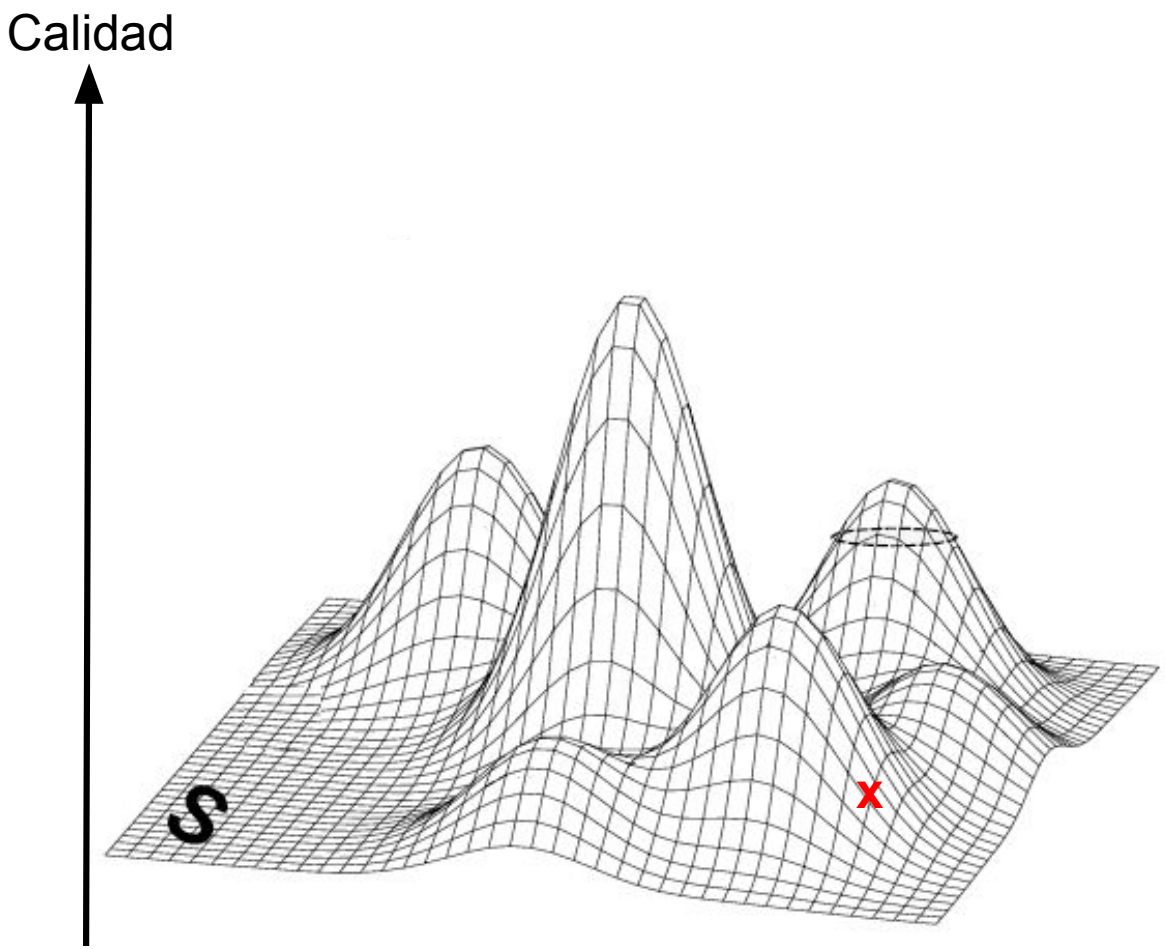
# Algoritmos Reparadores

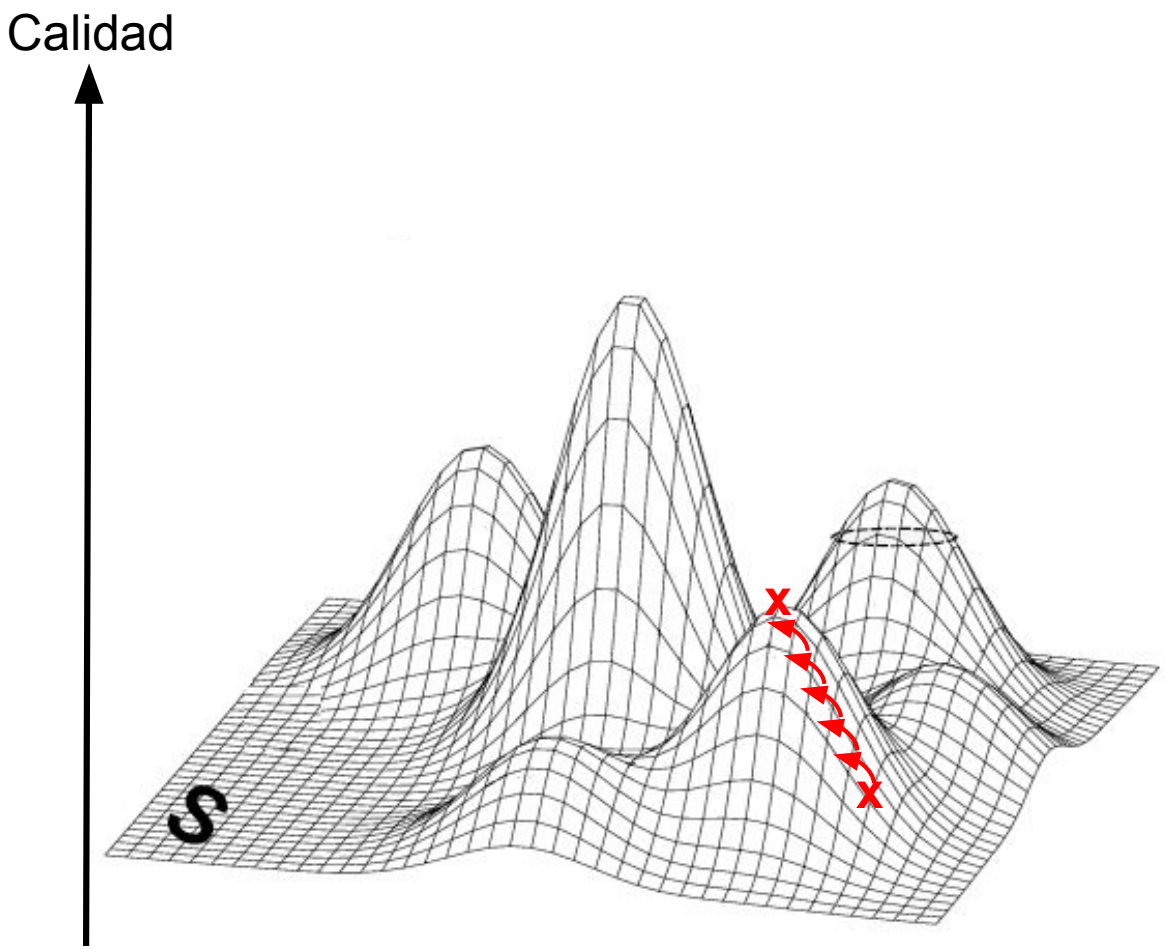
# Algoritmos basados en reparación

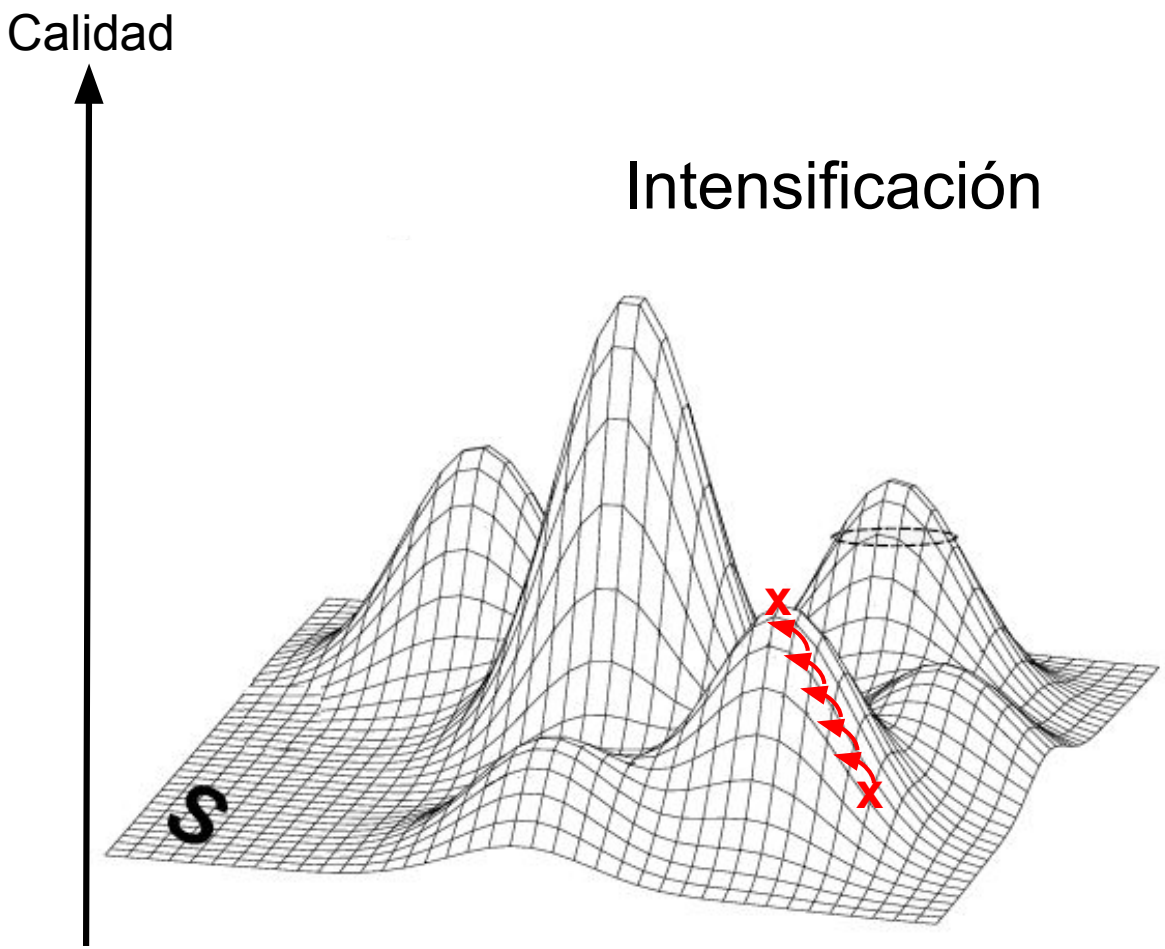
- Se mueven en el espacio de soluciones (c)
- Procesos iterativos que empiezan en una solución (c) y a través de modificaciones locales la va mejorando.
- Generalmente los algoritmos que construyen soluciones son más rápidos, pero sus resultados no siempre son buenos.

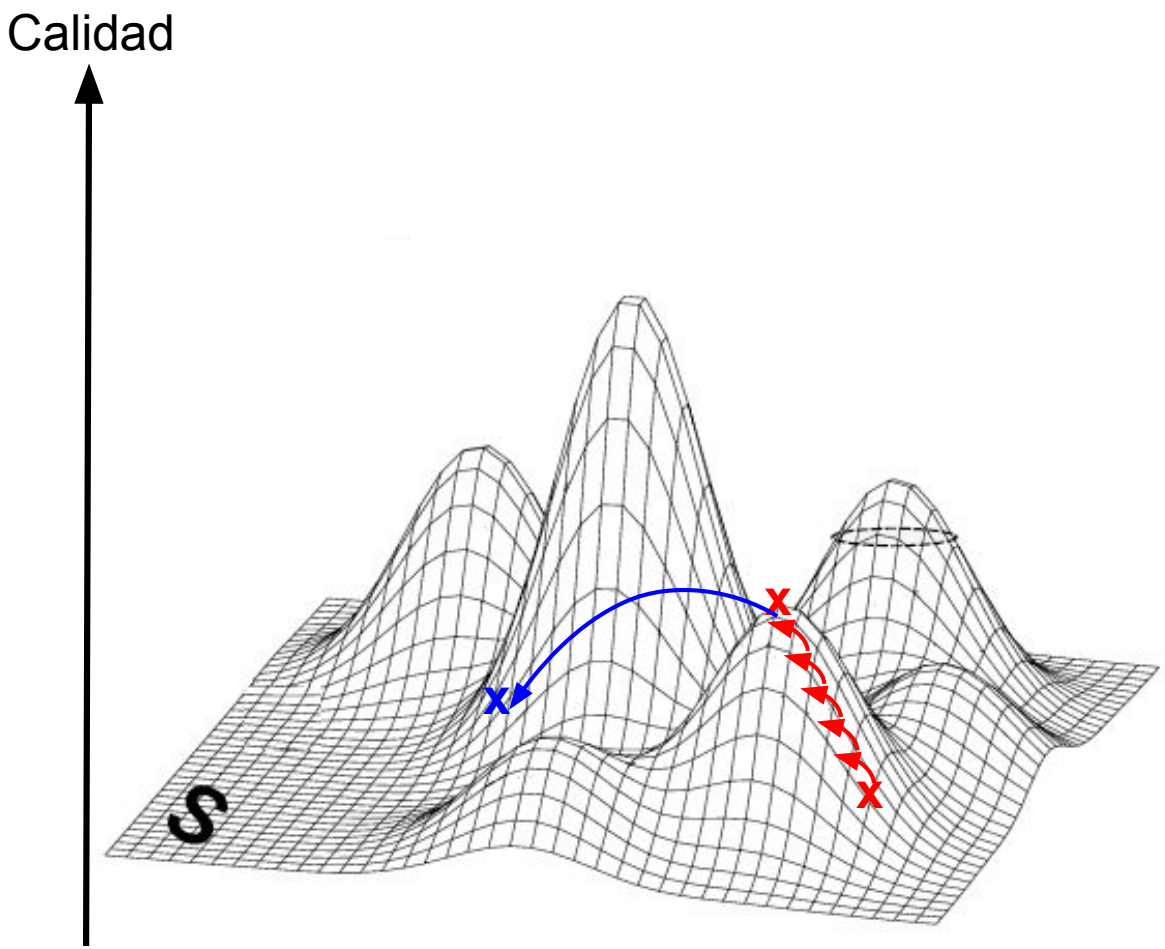


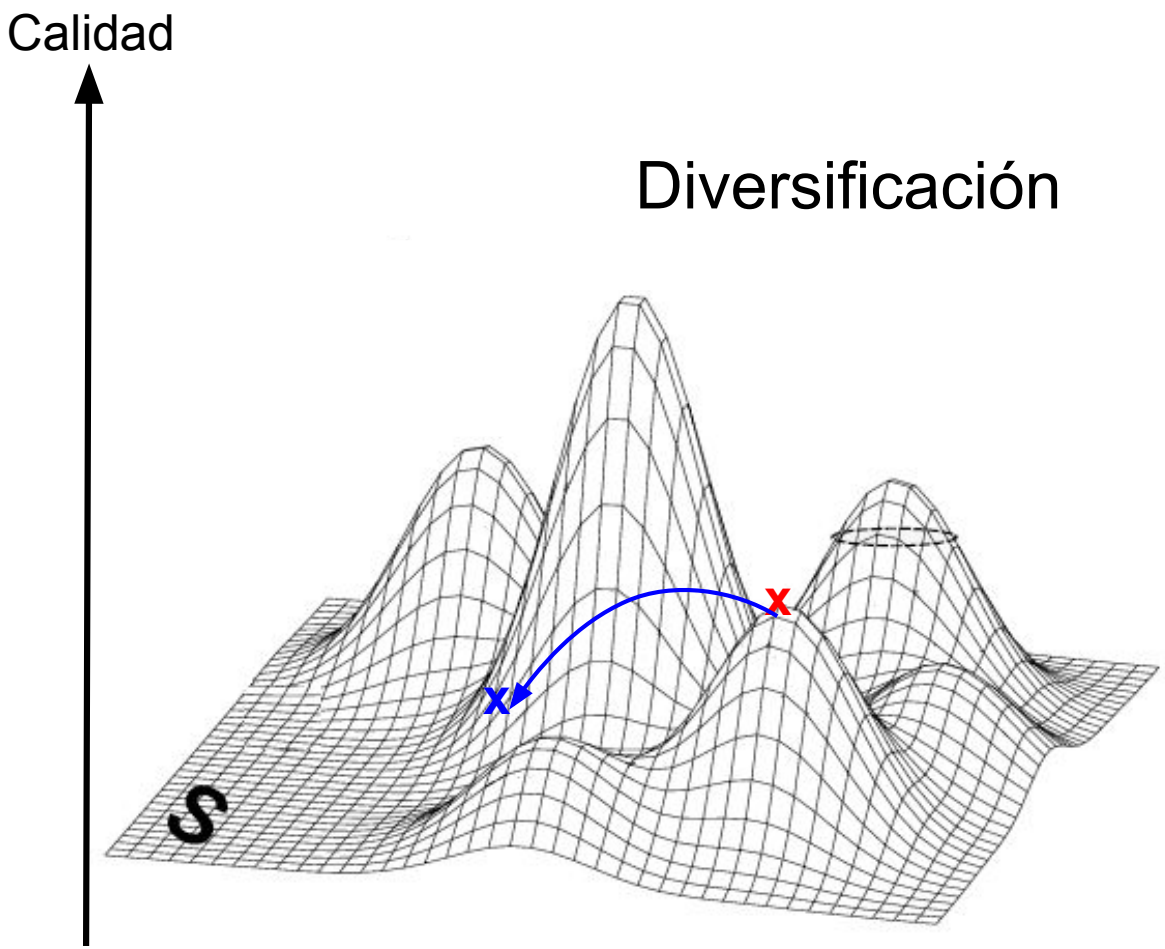

(problema de maximización)

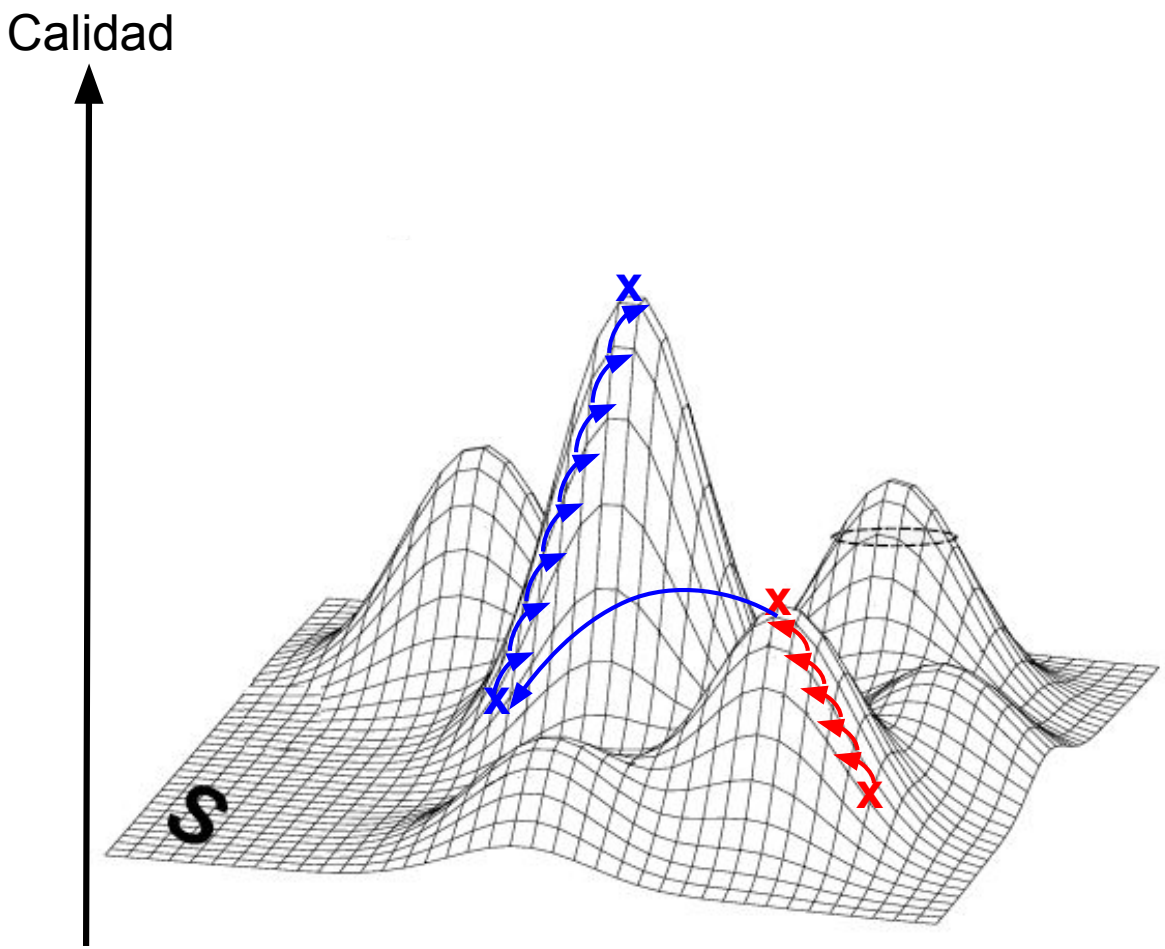

Calidad
S
X

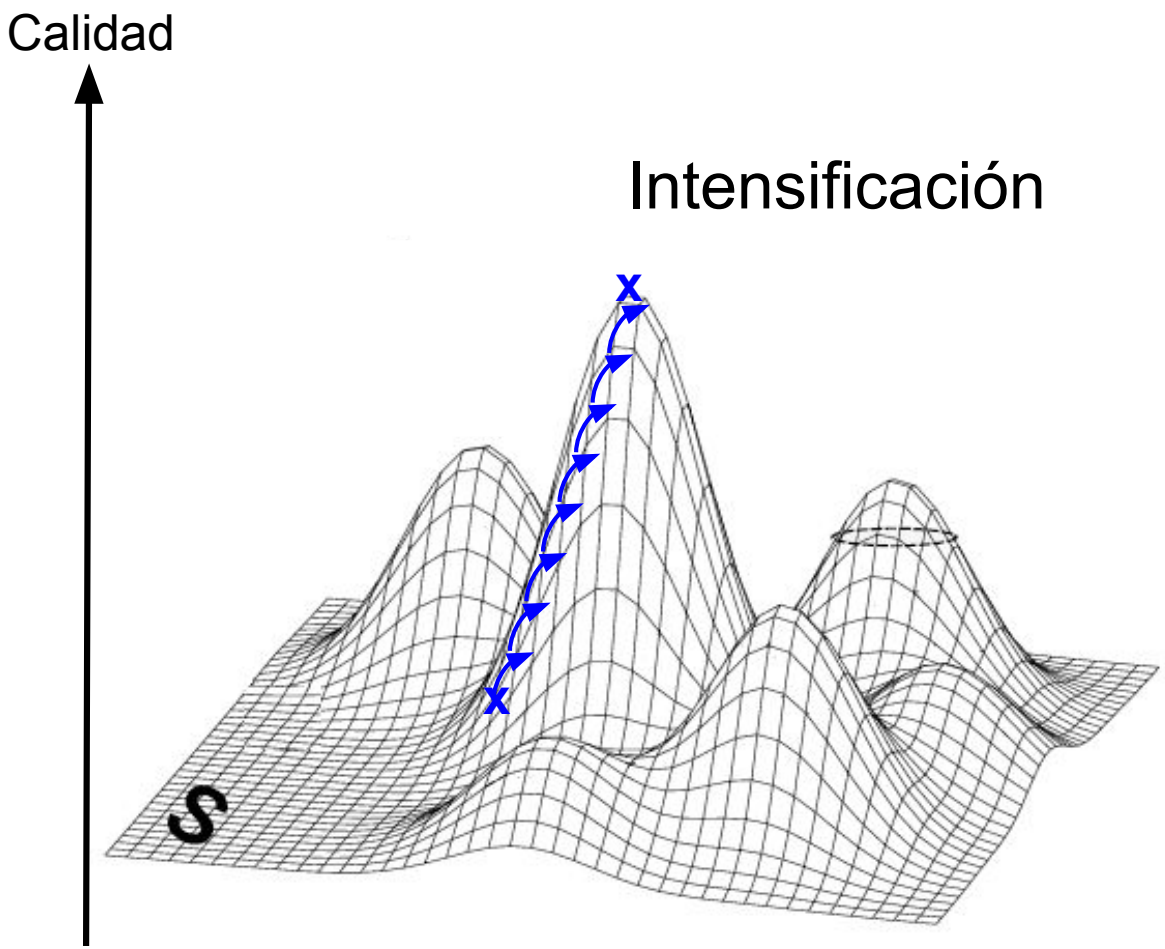

Calidad
S

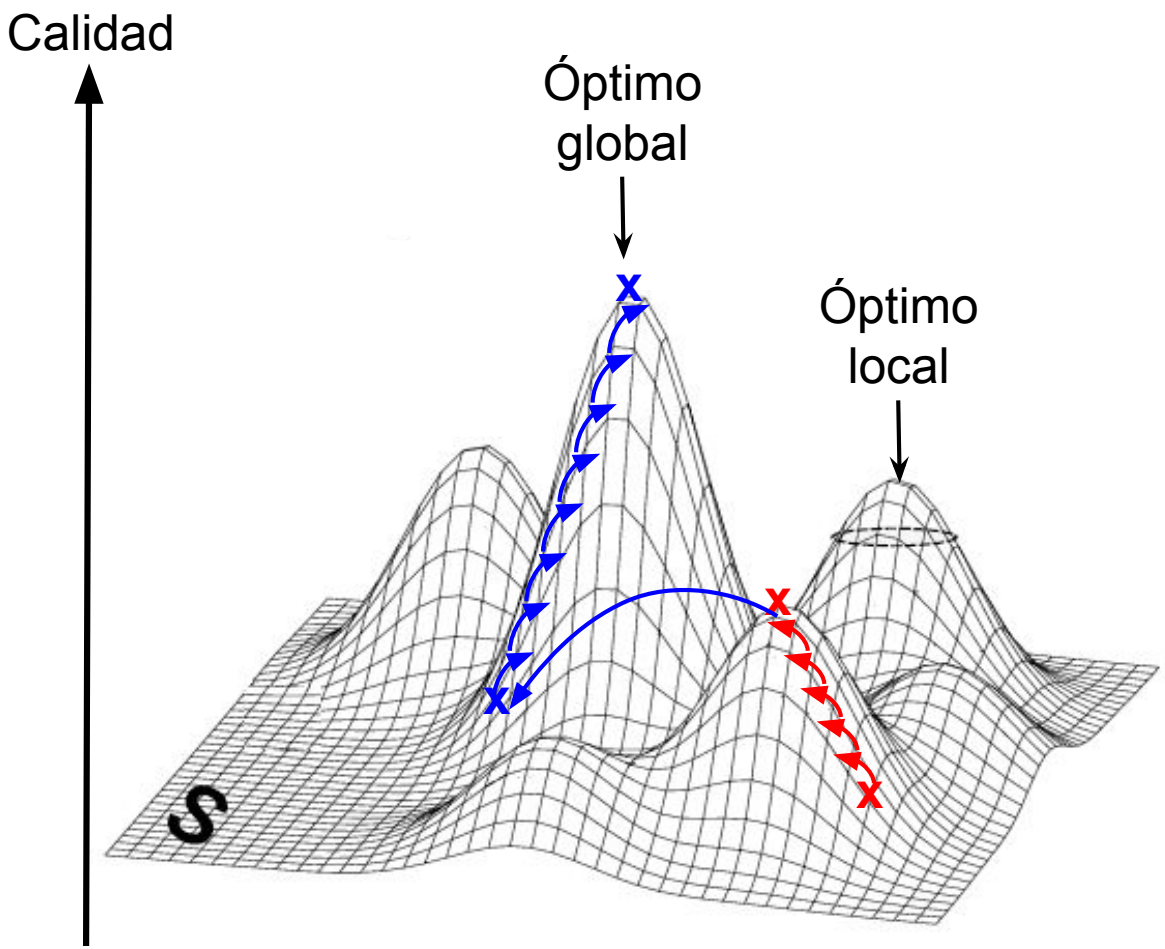

Calidad
Intensificación
S
x
x

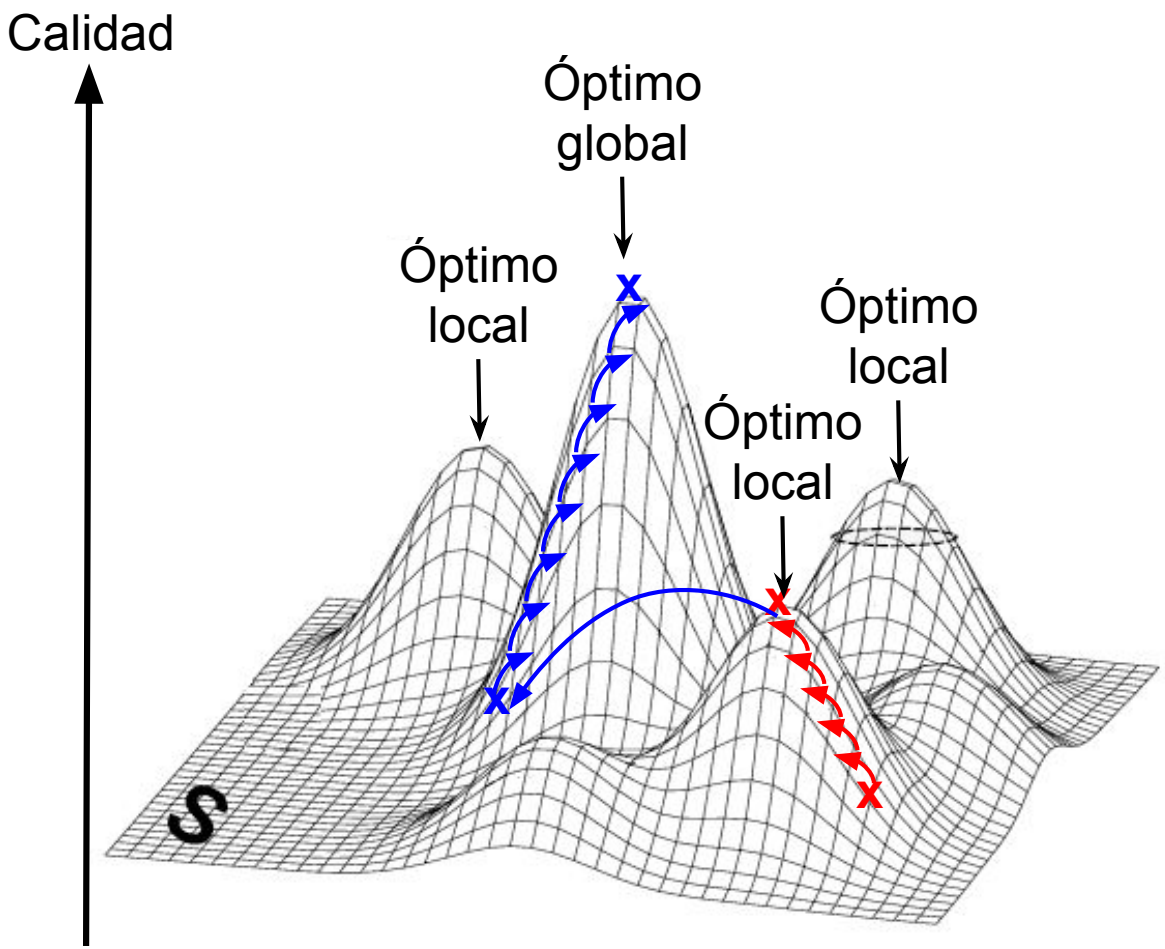

Calidad
S


Calidad
Diversificación
x
x
S


Calidad
S

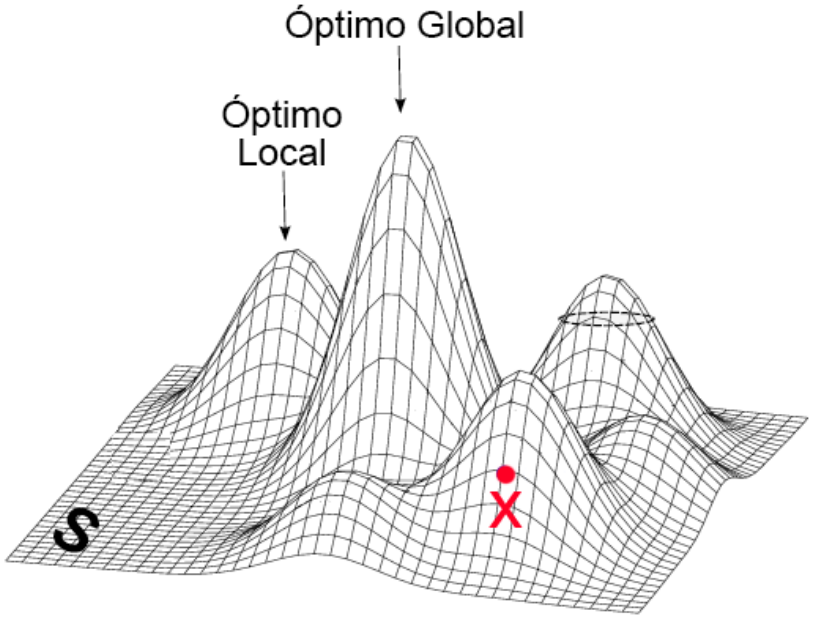

Calidad
Intensificación
S


Calidad
Óptimo
global
Óptimo
local
S


Calidad
Óptimo global
Óptimo local
Óptimo local
Óptimo local
S

# Algoritmos Reparadores

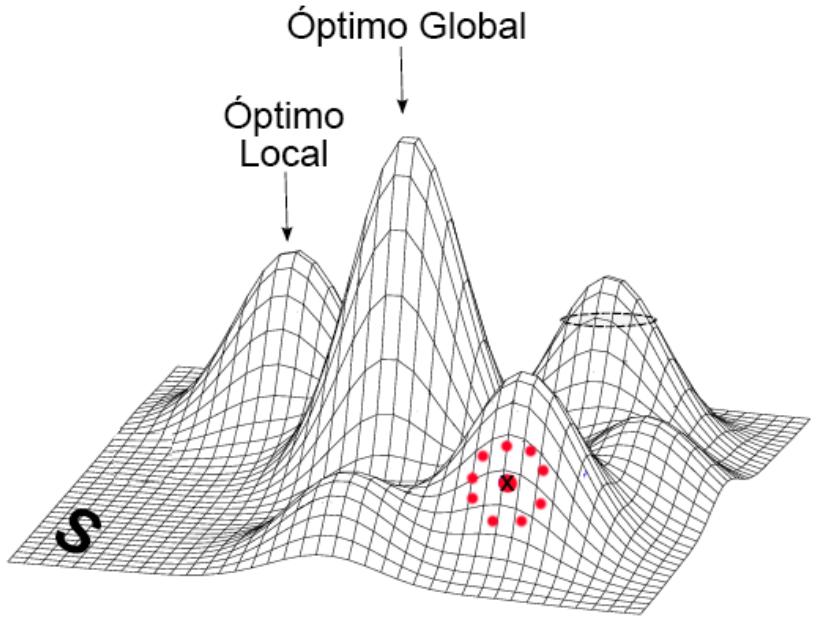
- Los algoritmos reparadores parten desde una solución inicial ($x$ en la imagen)
- Para buscar otras soluciones, podemos aplicar un **movimiento** a $x$
- Todas las soluciones cercanas a $x$, respecto a un movimiento, forman el **vecindario** de $x$
- Si el algoritmo se enfoca en mejorar la calidad de las soluciones, podemos decir que está **intensificando** la búsqueda
- Si el algoritmo intenta visitar otras regiones del espacio de búsqueda, podemos decir que está **diversificando** la búsqueda




(problema de maximización)

# Algoritmos Reparadores

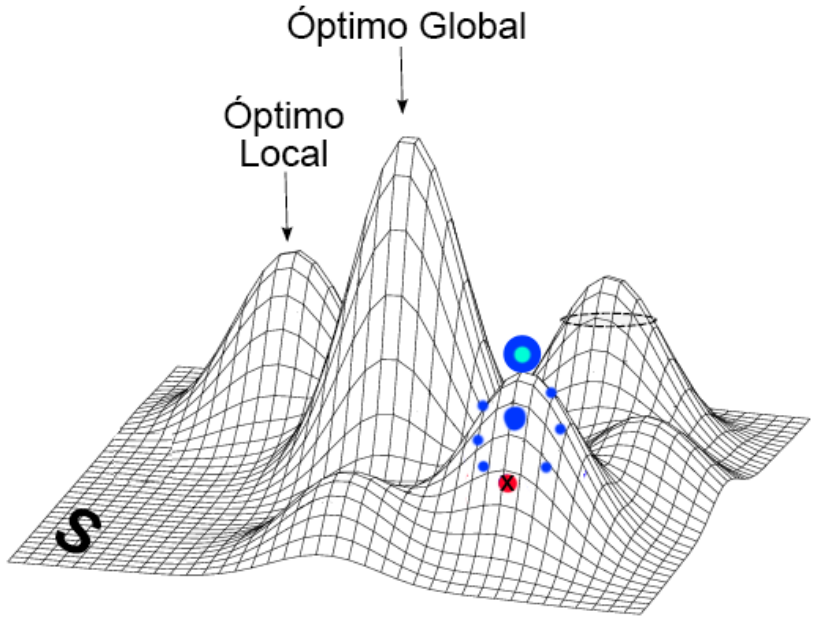
- Los algoritmos reparadores parten desde una solución inicial ($x$ en la imagen)
- Para buscar otras soluciones, podemos aplicar un **movimiento** a $x$
- Todas las soluciones cercanas a $x$, respecto a un movimiento, forman el **vecindario** de $x$
- Si el algoritmo se enfoca en mejorar la calidad de las soluciones, podemos decir que está **intensificando** la búsqueda
- Si el algoritmo intenta visitar otras regiones del espacio de búsqueda, podemos decir que está **diversificando** la búsqueda




(problema de maximización)

# Algoritmos Reparadores

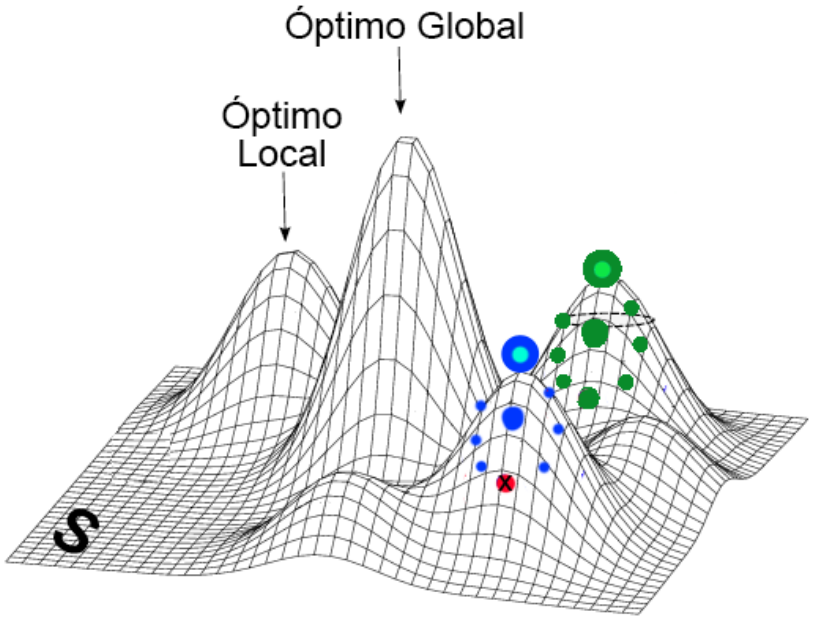
- Los algoritmos reparadores parten desde una solución inicial ($x$ en la imagen)
- Para buscar otras soluciones, podemos aplicar un **movimiento** a $x$
- Todas las soluciones cercanas a $x$, respecto a un movimiento, forman el **vecindario** de $x$
- Si el algoritmo se enfoca en mejorar la calidad de las soluciones, podemos decir que está **intensificando** la búsqueda
- Si el algoritmo intenta visitar otras regiones del espacio de búsqueda, podemos decir que está **diversificando** la búsqueda




(problema de maximización)

# Algoritmos Reparadores

- Los algoritmos reparadores parten desde una solución inicial ($x$ en la imagen)
- Para buscar otras soluciones, podemos aplicar un **movimiento** a $x$
- Todas las soluciones cercanas a $x$, respecto a un movimiento, forman el **vecindario** de $x$
- Si el algoritmo se enfoca en mejorar la calidad de las soluciones, podemos decir que está **intensificando** la búsqueda
- Si el algoritmo intenta visitar otras regiones del espacio de búsqueda, podemos decir que está **diversificando** la búsqueda




(problema de maximización)

# Algoritmos Reparadores

## Definiciones

- **Movimiento**
  Transformación aplicada a una solución candidata. Altera los valores asignados a algunas variables
  - Bitflip: $1111 \rightarrow 0111$
- **Vecindario**
  El vecindario de una solución es el conjunto de soluciones generado por la aplicación del movimiento a dicha solución
  - $\mathcal{N}$ define para cada solución candidata $x \in \mathcal{S}$ un conjunto $\mathcal{N}(x) \subseteq \mathcal{S}$
  - $\mathcal{N}(x)$ son en algún sentido "cercanas" a $x$.
  - Ejemplo con Bitflip aplicado a $x = 1111$
    - $\mathcal{N}(x) = [0111, 1011, 1101, 1110]$

# Algoritmos Reparadores

## Definiciones

- **Óptimo local**
  Sea $(\mathcal{S}, f)$ un problema de optimización y $\mathcal{N}$ la función vecindario. Una solución $\hat{x} \in \mathcal{S}$ es un Óptimo Local con respecto a $\mathcal{N}$ si

$$f(\hat{x}) \geq f(x), \forall x \in \mathcal{N}(\hat{x}).$$

- Un algoritmo que realiza búsqueda local, puede estancarse en algún óptimo local sin conocer otras soluciones

# Hill-Climbing

- Algoritmo que busca ir mejorando el valor de la función de evaluación de una solución candidata, al aplicar iterativamente un movimiento
- Generalmente, comienza con una solución candidata generada de manera aleatoria
- En cada iteración genera un vecindario aplicando el movimiento a la solución actual
- Si algún vecino mejora la calidad de la solución actual, éste reemplaza la solución actual
- En caso contrario, termina la ejecución

# Hill-Climbing

- Podemos identificar dos versiones de HC:
  - Alguna Mejora (*en inglés First Improvement*): Seleccionar el primer vecino que mejora la solución candidata actual
  - Mejor Mejora (*en inglés Best Improvement*): Selecciona el mejor vecino de todos, que mejora la solución candidata actual

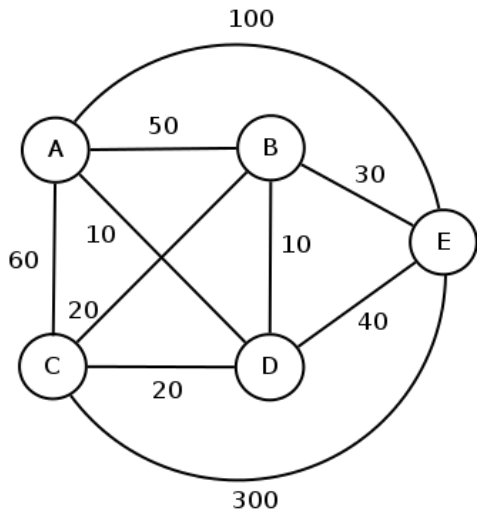
# Pseudo-código (Best Improvement)

**Procedure** hill-climbing
    $local \leftarrow FALSE$
    $s_c \leftarrow$ select a point at random
    **Repeat**
        $s_n \leftarrow$ select the best quality point in $\mathcal{N}(s_c)$
        **If** $f(s_n)$ is better than $f(s_c)$ **Then**
            $s_c \leftarrow s_n$
        **Else**
            $local \leftarrow TRUE$
    **Until** $local$
**End**

# Pseudo-código (First-Improvement Rule)

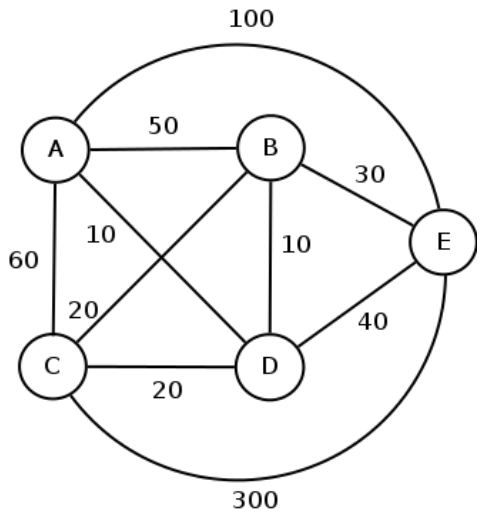
**Procedure** hill-climbing
    $local \leftarrow FALSE$
    $s_c \leftarrow$ select a point at random
    $neighbor \leftarrow 0$
    **Repeat**
        $s'_n \leftarrow$ generate a neighbor point in $\mathcal{N}(s_c)$
        $neighbor++$
        **If** $f(s'_n)$ is better than $f(s_c)$ **Then**
            $s_c \leftarrow s'_n$
            $neighbor \leftarrow 0$
        **If** $neighbor == max\_neighbors$ **Then**
            $local \leftarrow TRUE$
    **Until** local
**End**

# Hill-Climbing




Tomando como solución inicial aleatoria A-E-D-B-C con costo 230 y utilizando Mejor Mejora, su vecindario usando un intercambio entre ciudades contiguas en el tour, sería:

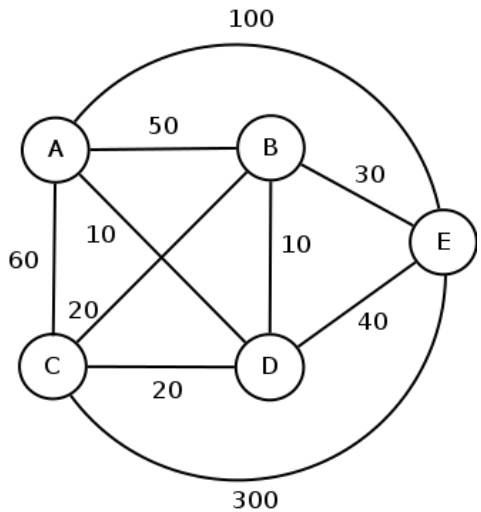
# Hill-Climbing




Tomando como solución inicial aleatoria A-E-D-B-C con costo 230 y utilizando Mejor Mejora, su vecindario usando un intercambio entre ciudades contiguas en el tour, sería:

1. E - A - D - B - C = 440

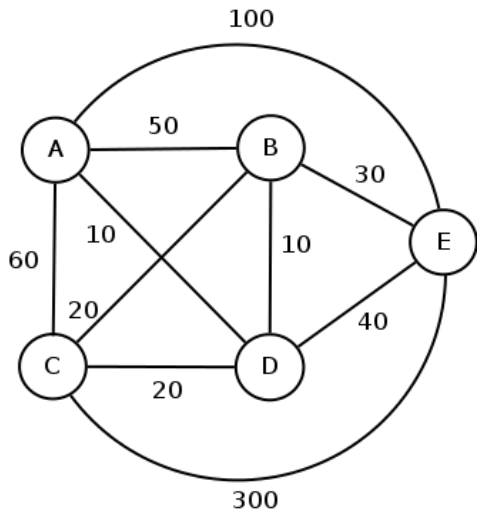
# Hill-Climbing




Tomando como solución inicial aleatoria A-E-D-B-C con costo 230 y utilizando Mejor Mejora, su vecindario usando un intercambio entre ciudades contiguas en el tour, sería:

1. E - A - D - B - C = 440
2. **A - D - E - B - C = 160**

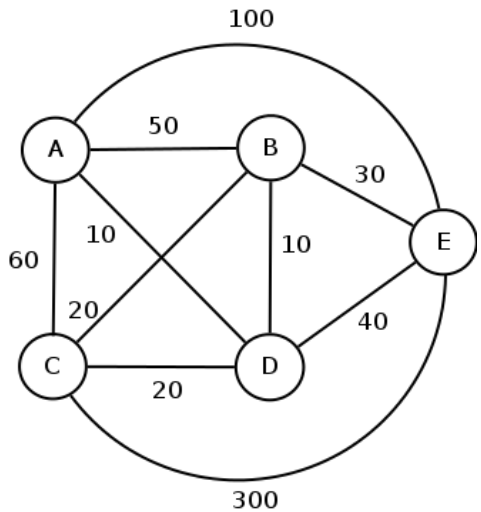
# Hill-Climbing




Tomando como solución inicial aleatoria A-E-D-B-C con costo 230 y utilizando Mejor Mejora, su vecindario usando un intercambio entre ciudades contiguas en el tour, sería:

1. E - A - D - B - C = 440
2. **A - D - E - B - C = 160**
3. A - E - B - D - C = 220

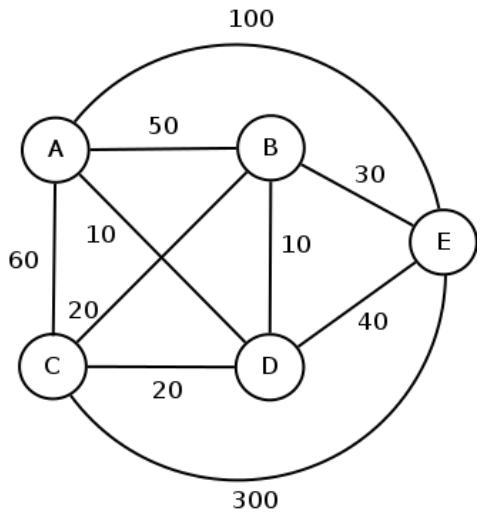
# Hill-Climbing




Tomando como solución inicial aleatoria A-E-D-B-C con costo 230 y utilizando Mejor Mejora, su vecindario usando un intercambio entre ciudades contiguas en el tour, sería:

1. E - A - D - B - C = 440
2. **A - D - E - B - C = 160**
3. A - E - B - D - C = 220
4. A - E - D - C - B = 210

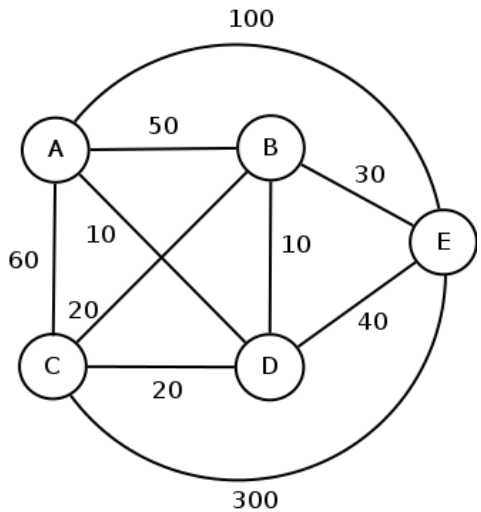
# Hill-Climbing




Tomando como solución inicial aleatoria A-E-D-B-C con costo 230 y utilizando Mejor Mejora, su vecindario usando un intercambio entre ciudades contiguas en el tour, sería:

1. E - A - D - B - C = 440
2. **A - D - E - B - C = 160**
3. A - E - B - D - C = 220
4. A - E - D - C - B = 210
5. C - E - D - B - A = 460

# Hill-Climbing




Tomando como solución inicial aleatoria A-E-D-B-C con costo 230 y utilizando Mejor Mejora, su vecindario usando un intercambio entre ciudades contiguas en el tour, sería:

1. E - A - D - B - C = 440
2. **A - D - E - B - C = 160**
3. A - E - B - D - C = 220
4. A - E - D - C - B = 210
5. C - E - D - B - A = 460

Si consideramos un algoritmo Alguna Mejora, para este caso, el primer vecino que mejora la solución es la número 2.

# Hill-Climbing

- Cuando HC encuentra la mejor solución candidata de la región, termina su ejecución
- Hill Climbing no define ninguna estrategia para escapar de óptimos locales

# Ejemplo

- Considerando el problema de la mochila

$$\begin{aligned} \text{máx } & 18 \cdot x_1 + 25 \cdot x_2 + 11 \cdot x_3 + 14 \cdot x_4 \\ \text{s.a. } & 2 \cdot x_1 + 2 \cdot x_2 + x_3 + x_4 \leq 3 \\ & x_1, x_2, x_3, x_4 \in \{0, 1\} \end{aligned}$$

- Proponga una representación acorde al problema
- Proponga un movimiento acorde al problema
- Describa el proceso de búsqueda realizado por hill-climbing[1]

[1] 1000, 0000

# Ejemplo: Hill Climbing

$$\text{Máx } 18 \cdot x1 + 25 \cdot x2 + 11 \cdot x3 + 14 \cdot x4$$

$$\text{s.a. } 2 \cdot x1 + 2 \cdot x2 + x3 + x4 \leq 3$$

$$x1, x2, x3, x4 \in \{0, 1\}$$

# Ejemplo: Hill Climbing

$$\text{Máx } 18 \cdot x1 + 25 \cdot x2 + 11 \cdot x3 + 14 \cdot x4$$

$$\text{s.a. } 2 \cdot x1 + 2 \cdot x2 + x3 + x4 \leq 3$$

$$x1, x2, x3, x4 \in \{0, 1\}$$

- Representación: Lista binaria de tamaño = número de objetos.
- Función de evaluación: Ganancia total.
  - Trabajaré solo sobre soluciones factibles, descartaré las soluciones infactibles.
- Movimiento: Bit-flip.
- Solución inicial: Aleatoria factible.

# Hill Climbing

| iteracion | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local |
|---|---|---|---|---|
| 1 | 1000 (18) | | | FALSE |

# Hill Climbing

| iteracion | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local |
|---|---|---|---|---|
| 1 | 1000 (18) | | | FALSE |
| | | 0000 (0) | | |

# Hill Climbing

| iteracion | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local |
|---|---|---|---|---|
| 1 | 1000 (18) | | | FALSE |
| | | 0000 (0) | | |
| | | ~~1100(43)~~ | | |

# Hill Climbing

| iteracion | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local |
|---|---|---|---|---|
| 1 | 1000 (18) | | | FALSE |
| | | 0000 (0) | | |
| | | ~~1100(43)~~ | | |
| | | 1010(29) | | |

# Hill Climbing

| iteracion | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local |
|---|---|---|---|---|
| 1 | 1000 (18) | | | FALSE |
| | | 0000 (0) | | |
| | | ~~1100(43)~~ | | |
| | | 1010(29) | | |
| | | 1001(32) | | |

# Hill Climbing

| iteracion | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local |
|---|---|---|---|---|
| 1 | 1000 (18) | | | FALSE |
| | | 0000 (0) | | |
| | | ~~1100(43)~~ | | |
| | | 1010(29) | | |
| | | 1001(32) | | |
| | | | 1001(32) | |

# Hill Climbing

| iteracion | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local |
|---|---|---|---|---|
| 1 | 1000 (18) | | | FALSE |
| | | 0000 (0) | | |
| | | ~~1100(43)~~ | | |
| | | 1010(29) | | |
| | | 1001(32) | | |
| | | | 1001(32) | |
| 2 | 1001 (32) | | | |

# Hill Climbing

| iteracion | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local |
|---|---|---|---|---|
| 1 | 1000 (18) | | | FALSE |
| | | 0000 (0) | | |
| | | ~~1100(43)~~ | | |
| | | 1010(29) | | |
| | | 1001(32) | | |
| | | | 1001(32) | |
| 2 | 1001 (32) | | | |
| | | 0001 (14) | | |

# Hill Climbing

| iteracion | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local |
|---|---|---|---|---|
| 1 | 1000 (18) | | | FALSE |
| | | 0000 (0) | | |
| | | ~~1100(43)~~ | | |
| | | 1010(29) | | |
| | | 1001(32) | | |
| | | | 1001(32) | |
| 2 | 1001 (32) | | | |
| | | 0001 (14) | | |
| | | ~~1101(57)~~ | | |

# Hill Climbing

| iteracion | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local |
|---|---|---|---|---|
| 1 | 1000 (18) | | | FALSE |
| | | 0000 (0) | | |
| | | ~~1100(43)~~ | | |
| | | 1010(29) | | |
| | | 1001(32) | | |
| | | | 1001(32) | |
| 2 | 1001 (32) | | | |
| | | 0001 (14) | | |
| | | ~~1101(57)~~ | | |
| | | ~~1011(43)~~ | | |

# Hill Climbing

| iteracion | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local |
|---|---|---|---|---|
| 1 | 1000 (18) | | | FALSE |
| | | 0000 (0) | | |
| | | ~~1100(43)~~ | | |
| | | 1010(29) | | |
| | | 1001(32) | | |
| | | | 1001(32) | |
| 2 | 1001 (32) | | | |
| | | 0001 (14) | | |
| | | ~~1101(57)~~ | | |
| | | ~~1011(43)~~ | | |
| | | 1000(18) | | |

# Hill Climbing

| iteracion | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local |
|---|---|---|---|---|
| 1 | 1000 (18) | | | FALSE |
| | | 0000 (0) | | |
| | | ~~1100(43)~~ | | |
| | | 1010(29) | | |
| | | 1001(32) | | |
| | | | 1001(32) | |
| 2 | 1001 (32) | | | |
| | | 0001 (14) | | |
| | | ~~1101(57)~~ | | |
| | | ~~1011(43)~~ | | |
| | | 1000(18) | | |
| | | | 1000(18) | TRUE |

# Hill-Climbing

- Cuando HC encuentra la mejor solución candidata de la región, termina su ejecución
- Hill Climbing no define ninguna estrategia para escapar de óptimos locales
- Una solución a este problema es Hill Climbing + Restart

# Hill-Climbing + Restart

- La idea es re-comenzar el algoritmo con una solución nueva cuando éste se ha quedado estancado.

# Pseudo-código

**Procedure** hill-climbing
    $t \leftarrow 0$
    initialize $s_{best}$
    **Repeat**
        $local \leftarrow FALSE$
        $s_c \leftarrow$ select a point at random
        **Repeat**
            $s_n \leftarrow$ select the best quality point in $\mathcal{N}(s_c)$
            **If** $f(s_n)$ is better than $f(s_c)$ **Then**
                $s_c \leftarrow s_n$
            **Else**
                $local \leftarrow TRUE$
        **Until** $local$
        $t \leftarrow t + 1$
        **if** $f(s_c)$ is better than $f(s_{best})$ **then**
            $s_{best} \leftarrow s_c$
    **Until** $t = MAX$

# Ejemplo: Hill Climbing + Restart

$$\text{Máx } 18 \cdot x1 + 25 \cdot x2 + 11 \cdot x3 + 14 \cdot x4$$

$$\text{s.a. } 2 \cdot x1 + 2 \cdot x2 + x3 + x4 \leq 3$$

$$x1, x2, x3, x4 \in \{0, 1\}$$

# Ejemplo: Hill Climbing + Restart

$$\text{Máx } 18 \cdot x1 + 25 \cdot x2 + 11 \cdot x3 + 14 \cdot x4$$

$$\text{s.a. } 2 \cdot x1 + 2 \cdot x2 + x3 + x4 \leq 3$$

$$x1, x2, x3, x4 \in \{0, 1\}$$

- Representación: Lista binaria de tamaño = número de objetos.
- Función de evaluación: Ganancia total.
  - Trabajaré solo sobre soluciones factibles, descartaré las soluciones infactibles.
- Movimiento: Bit-flip.
- Solución inicial: Aleatoria factible.

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |
| | | | | 1001(32) | | |
| | 2 | 1001 (32) | | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |
| | | | | 1001(32) | | |
| | 2 | 1001 (32) | | | | |
| | | | 0001 (14) | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |
| | | | | 1001(32) | | |
| | 2 | 1001 (32) | | | | |
| | | | 0001 (14) | | | |
| | | | ~~1101(57)~~ | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |
| | | | | 1001(32) | | |
| | 2 | 1001 (32) | | | | |
| | | | 0001 (14) | | | |
| | | | ~~1101(57)~~ | | | |
| | | | ~~1011(43)~~ | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |
| | | | | 1001(32) | | |
| | 2 | 1001 (32) | | | | |
| | | | 0001 (14) | | | |
| | | | ~~1101(57)~~ | | | |
| | | | ~~1011(43)~~ | | | |
| | | | 1000(18) | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |
| | | | | 1001(32) | | |
| | 2 | 1001 (32) | | | | |
| | | | 0001 (14) | | | |
| | | | ~~1101(57)~~ | | | |
| | | | ~~1011(43)~~ | | | |
| | | | 1000(18) | | | |
| | | | | 1000(18) | TRUE | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |
| | | | | 1001(32) | | |
| | 2 | 1001 (32) | | | | |
| | | | 0001 (14) | | | |
| | | | ~~1101(57)~~ | | | |
| | | | ~~1011(43)~~ | | | |
| | | | 1000(18) | | | |
| | | | | 1000(18) | TRUE | |
| | | | | | | 1001 (32) |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |
| | | | | 1001(32) | | |
| | 2 | 1001 (32) | | | | |
| | | | 0001 (14) | | | |
| | | | ~~1101(57)~~ | | | |
| | | | ~~1011(43)~~ | | | |
| | | | 1000(18) | | | |
| | | | | 1000(18) | TRUE | |
| | | | | | | 1001 (32) |
| 2 | 1 | 0000(0) | | | FALSE | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |
| | | | | 1001(32) | | |
| | 2 | 1001 (32) | | | | |
| | | | 0001 (14) | | | |
| | | | ~~1101(57)~~ | | | |
| | | | ~~1011(43)~~ | | | |
| | | | 1000(18) | | | |
| | | | | 1000(18) | TRUE | |
| | | | | | | 1001 (32) |
| 2 | 1 | 0000(0) | | | FALSE | |
| | | | 1000(18) | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |
| | | | | 1001(32) | | |
| | 2 | 1001 (32) | | | | |
| | | | 0001 (14) | | | |
| | | | ~~1101(57)~~ | | | |
| | | | ~~1011(43)~~ | | | |
| | | | 1000(18) | | | |
| | | | | 1000(18) | TRUE | |
| | | | | | | 1001 (32) |
| 2 | 1 | 0000(0) | | | FALSE | |
| | | | 1000(18) | | | |
| | | | 0100(25) | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |
| | | | | 1001(32) | | |
| | 2 | 1001 (32) | | | | |
| | | | 0001 (14) | | | |
| | | | ~~1101(57)~~ | | | |
| | | | ~~1011(43)~~ | | | |
| | | | 1000(18) | | | |
| | | | | 1000(18) | TRUE | |
| | | | | | | 1001 (32) |
| 2 | 1 | 0000(0) | | | FALSE | |
| | | | 1000(18) | | | |
| | | | 0100(25) | | | |
| | | | 0010(11) | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |
| | | | | 1001(32) | | |
| | 2 | 1001 (32) | | | | |
| | | | 0001 (14) | | | |
| | | | ~~1101(57)~~ | | | |
| | | | ~~1011(43)~~ | | | |
| | | | 1000(18) | | | |
| | | | | 1000(18) | TRUE | |
| | | | | | | 1001 (32) |
| 2 | 1 | 0000(0) | | | FALSE | |
| | | | 1000(18) | | | |
| | | | 0100(25) | | | |
| | | | 0010(11) | | | |
| | | | 0001(14) | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |
| | | | | 1001(32) | | |
| | 2 | 1001 (32) | | | | |
| | | | 0001 (14) | | | |
| | | | ~~1101(57)~~ | | | |
| | | | ~~1011(43)~~ | | | |
| | | | 1000(18) | | | |
| | | | | 1000(18) | TRUE | |
| | | | | | | 1001 (32) |
| 2 | 1 | 0000(0) | | | FALSE | |
| | | | 1000(18) | | | |
| | | | 0100(25) | | | |
| | | | 0010(11) | | | |
| | | | 0001(14) | | | |
| | | | | 0100(25) | | |
| | 2 | 0100 (25) | | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 1000 (18) | | | FALSE | - - - - (0) |
| | | | 0000 (0) | | | |
| | | | ~~1100(43)~~ | | | |
| | | | 1010(29) | | | |
| | | | 1001(32) | | | |
| | | | | 1001(32) | | |
| | 2 | 1001 (32) | | | | |
| | | | 0001 (14) | | | |
| | | | ~~1101(57)~~ | | | |
| | | | ~~1011(43)~~ | | | |
| | | | 1000(18) | | | |
| | | | | 1000(18) | TRUE | |
| | | | | | | 1001 (32) |
| 2 | 1 | 0000(0) | | | FALSE | |
| | | | 1000(18) | | | |
| | | | 0100(25) | | | |
| | | | 0010(11) | | | |
| | | | 0001(14) | | | |
| | | | | 0100(25) | | |
| | 2 | 0100 (25) | | | | |
| | | | ~~1100 (43)~~ | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 2 | 2 | 0100 (25) | | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 2 | 2 | 0100 (25) | | | | |
| | | | ~~1100 (43)~~ | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 2 | 2 | 0100 (25) | | | | |
| | | | ~~1100 (43)~~ | | | |
| | | | 0000(0) | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 2 | 2 | 0100 (25) | | | | |
| | | | ~~1100 (43)~~ | | | |
| | | | 0000(0) | | | |
| | | | 0110(36) | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 2 | 2 | 0100 (25) | | | | |
| | | | ~~1100 (43)~~ | | | |
| | | | 0000(0) | | | |
| | | | 0110(36) | | | |
| | | | 0101(39) | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 2 | 2 | 0100 (25) | | | | |
| | | | ~~1100 (43)~~ | | | |
| | | | 0000(0) | | | |
| | | | 0110(36) | | | |
| | | | 0101(39) | | | |
| | | | | 0101(39) | | |
| 2 | 3 | 0101 (39) | | | | |
| | | | ~~1101 (57)~~ | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 2 | 2 | 0100 (25) | | | | |
| | | | ~~1100 (43)~~ | | | |
| | | | 0000(0) | | | |
| | | | 0110(36) | | | |
| | | | 0101(39) | | | |
| | | | | 0101(39) | | |
| 2 | 3 | 0101 (39) | | | | |
| | | | ~~1101 (57)~~ | | | |
| | | | 0001(14) | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 2 | 2 | 0100 (25) | | | | |
| | | | ~~1100 (43)~~ | | | |
| | | | 0000(0) | | | |
| | | | 0110(36) | | | |
| | | | 0101(39) | | | |
| | | | | 0101(39) | | |
| 2 | 3 | 0101 (39) | | | | |
| | | | ~~1101 (57)~~ | | | |
| | | | 0001(14) | | | |
| | | | ~~0111(50)~~ | | | |

# Hill Climbing + Restart

| restart | it | $s_c$ | $\mathcal{N}(s_c)$ | $s_n$ | local | $s_{best}$ |
|---|---|---|---|---|---|---|
| 2 | 2 | 0100 (25) | | | | |
| | | | ~~1100 (43)~~ | | | |
| | | | 0000(0) | | | |
| | | | 0110(36) | | | |
| | | | 0101(39) | | | |
| | | | | 0101(39) | | |
| 2 | 3 | 0101 (39) | | | | |
| | | | ~~1101 (57)~~ | | | |
| | | | 0001(14) | | | |
| | | | ~~0111(50)~~ | | | |
| | | | 0100(25) | | TRUE | |
| | | | | | | 0101 (39) |

# Hill-Climbing + Restart

- Desventaja de H-C W/ Restart: Pérdida de información valiosa del proceso de búsqueda actual.
- Una alternativa es aceptar movimientos que empeoran la calidad de la solución actual.
- La desventaja de aceptar soluciones de peor calidad es que se pueden producir ciclos en la búsqueda.