


Facultad de Ciencias
Licenciatura en
Ciencias de la Computación

Cómputo Evolutivo

Escapando de Óptimos Locales II

M. en C. Oscar Hernández Constantino
(constantino92@ciencias.unam.mx)

Contenido de la Presentación

1. Búsqueda Local Iterada

2. Búsqueda de Vecindad Variable
(VNS,Variable Neighbourhood Search)

3. Búsqueda Local Guiada
(GLS, Guided Local Search)

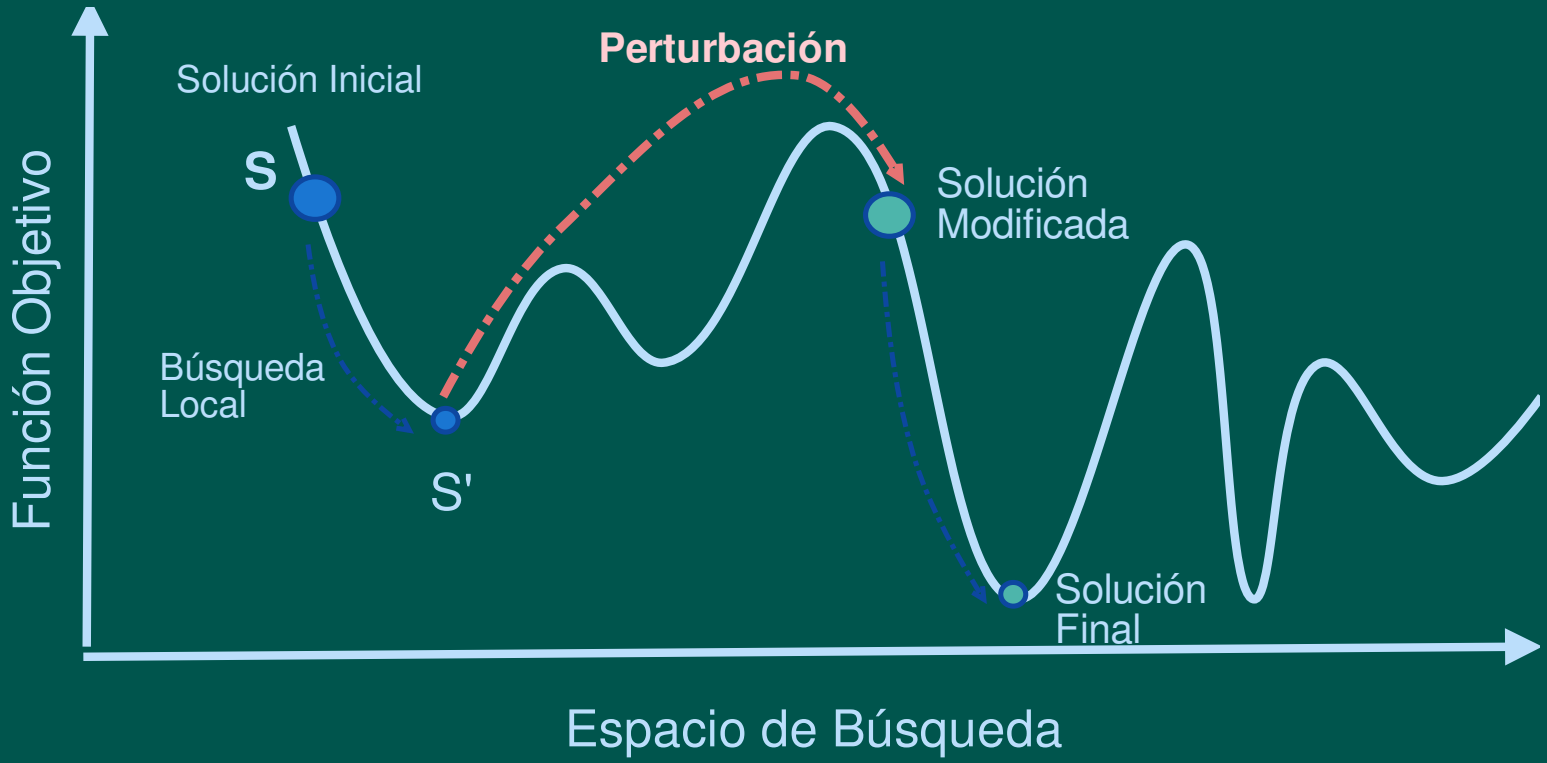
4. Búsqueda Aleatoria Voraz
(GRASP, Greedy Randomized Adaptive Search Procedure)

Alternativas para evitar óptimos locales

- Iterar con diferentes soluciones
  - Multi-Inicio
  - Búsqueda Local Iterada (ILS, Iterated Local Search)
- Cambiar el paisaje (lanscape) de la función del problema
  - Cambiar la función objetivo o los datos de entrada
    - Búsqueda local guiada (GLS)
    - Métodos de Función sustituta (suavizado)
  - Usar vecindades diferentes
    - Búsqueda con Vecindades Variables (VNS, Variable Neighborhood Search)
- Aceptar vecinos que no mejoran
  - Recocido Simulado (SA, Simulated Annealing)
  - Búsqueda Tabú (TS, Tabu Search)

Búsqueda Local Iterada


Función Objetivo
Solución Inicial
Óptimo Local
Espacio de Búsqueda


Perturbación
Solución Inicial
S
Solución
Modificada
Búsqueda
Local
S'
Solución
Final
Función Objetivo
Espacio de Búsqueda

Algoritmo 1: Búsqueda Local Iterada (ILS, Iterative Local Search)

Entrada: $f :\rightarrow \mathbb{R}$

Resultado: s', mejor solución encontrada

1 $s_0 =$ inicializar Solución ;
2 $s_{actual} =$ búsqueda Local (s_0) ;
3 $s_{mejor} = s_{actual}$;
4 $t = 1$;
5 **mientras** *Condiciones de término* **hacer**
6 `// Reiniciar la búsqueda en un nuevo punto`
7 $s' =$ perturbar solución $(s_{actual},$ historia de búsqueda $)$;
8 $s_{hc} =$ búsqueda Local (s') ;
9 **si** $f(s_{hc}) < f(s_{mejor})$ **entonces**
`// Actualizar la mejor solución encontrada`
10 $s_{mejor} = s_{hc}$;
11 s_{actual} = Aplicar criterio de aceptación $(s_{actual}, s_{hc},$ memoria de búsqueda $)$;
12 $t = t + 1$
13 **devolver** s_{mejor}

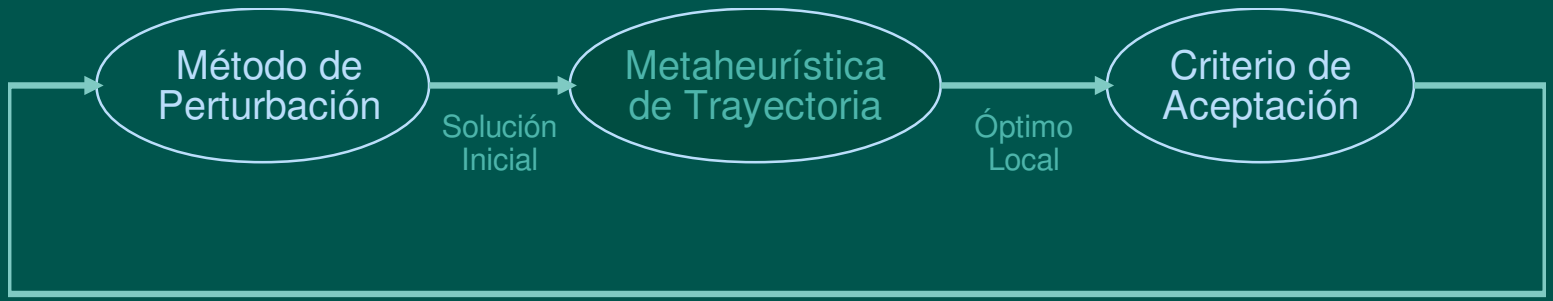
Algoritmo 2: Búsqueda por Escalada con Multi-Inicio

Entrada: $MAX > 0$, máximo de iteraciones; $f :\rightarrow \mathbb{R}$

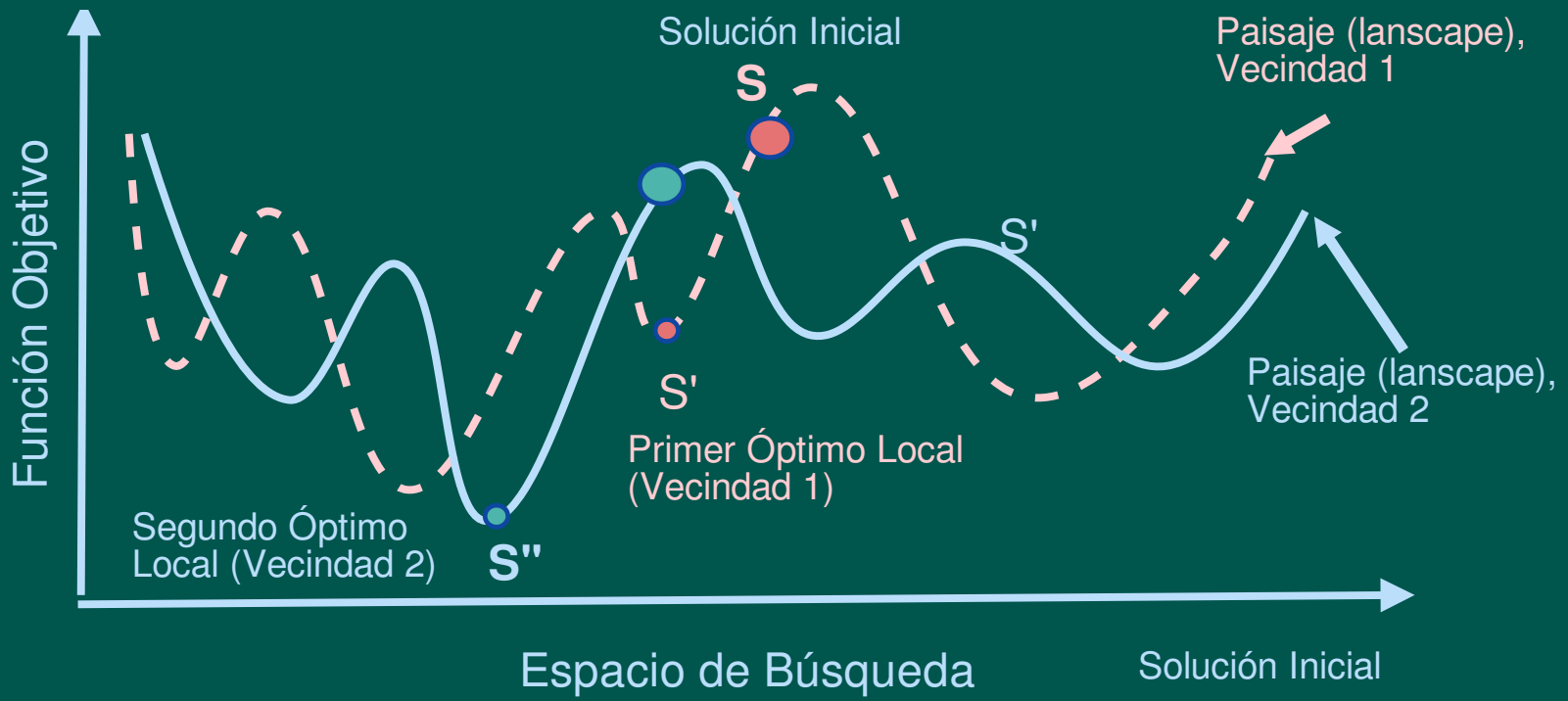
Resultado: s', mejor solución encontrada

```
1  s = inicializar Solución ;
2  s_hc = búsqueda por escalada (s) ;
3  s_mejor = s_hc ;
4  t = 1 ;
5  mientras t < MAX hacer
6      // Reiniciar la búsqueda en un nuevo punto
7      s' = inicializar solución ;
8      s'_hc = búsqueda por escalada (s) ;
9
10     si f(s') < f(s) entonces
           // Actualizar la mejor solución encontrada
11         s_mejor = s'_hc ;
12     t = t + 1
13 devolver s_mejor
```

Esquema general de la Búsqueda Local Iterada



Búsqueda de Vecindad Variable (VNS,Variable Neighbourhood Search)


Función Objetivo
Solución Inicial
S
Paisaje (lanscape),
Vecindad 1
S'
S'
Primer Óptimo Local
(Vecindad 1)
Paisaje (lanscape),
Vecindad 2
Segundo Óptimo
Local (Vecindad 2)
S"
Espacio de Búsqueda
Solución Inicial

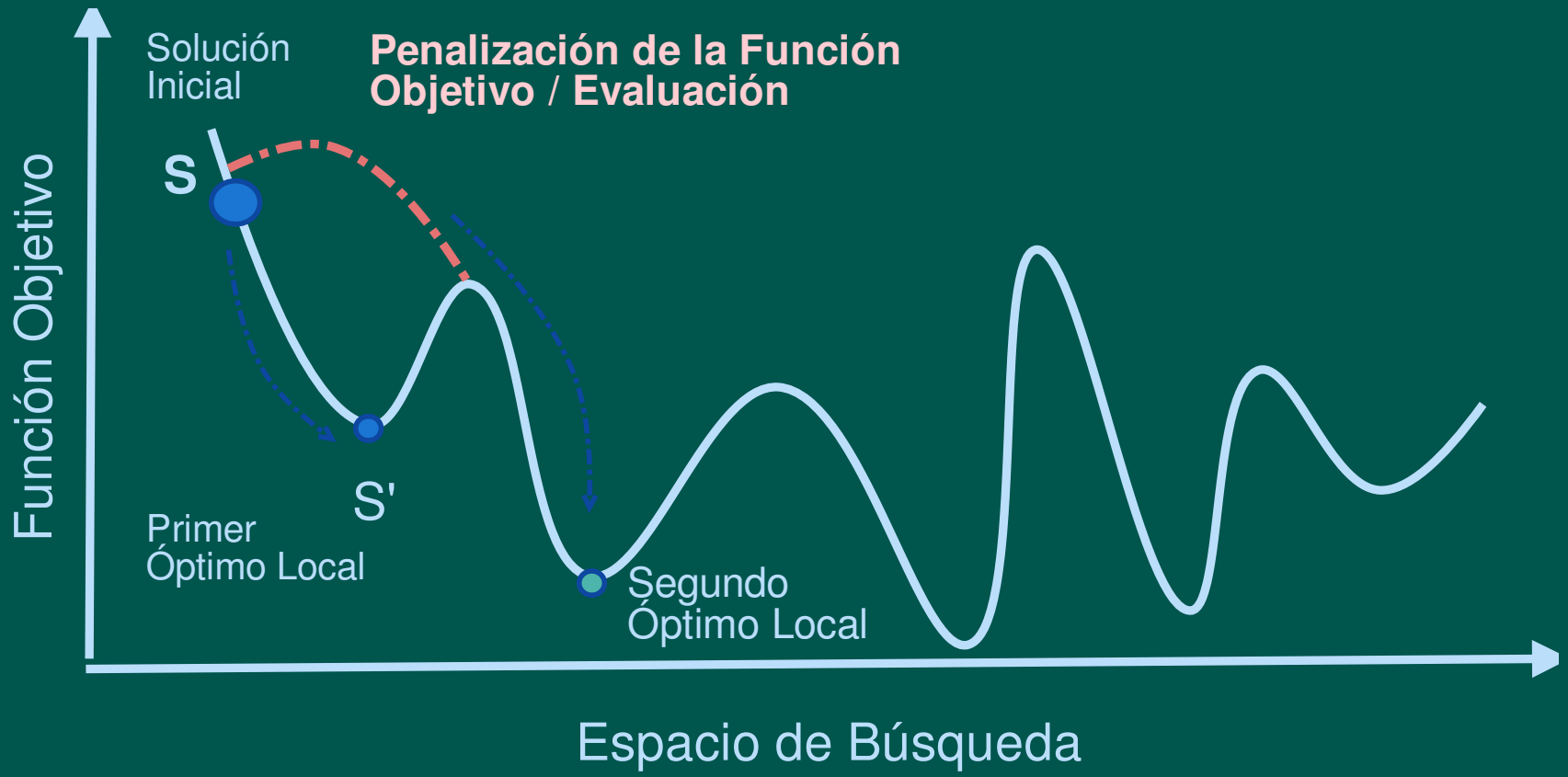
Algoritmo 3: VNS

Entrada: $f :\rightarrow \mathbb{R}$; s_{ini}, solución inicial; un conjunto N_k de estructuras de vecindad, $k = 1, \ldots, k_{max}$

Resultado: s, mejor solución encontrada

```
1  s = s_ini ;
2  mientras Condiciones de término hacer
3      k = 1 ;
4      mientras k <= k_max hacer
5          Seleccionar un vecino s' de manera aleatoria de la k-ésima vecindad de
           N_k(s) ;
6          s'_hc = búsqueda local (s') ;
7          si f(s'_hc) < f(s) entonces
8              s = s'_hc ;
9              Continuar la búsqueda con N_1 ; k = 1 ;
10         en otro caso
11             k = k + 1 ;
12 devolver s
```

Búsqueda Local Guiada
(GLS, Guided Local Search)


Solución
Inicial
Penalización de la Función
Objetivo / Evaluación
S
S'
Primer
Óptimo Local
Segundo
Óptimo Local
Función Objetivo
Espacio de Búsqueda

Algoritmo 4: Búsqueda Local Guiada

Entrada: $f :\rightarrow \mathbb{R}$; s_{ini}, solución inicial; *M_Trayectoria*; λ; características I; costos c;

Resultado: s', mejor solución encontrada

1 $s = s_{ini}$;
2 $p_i = 0$ /* Inicializar penalización */ ;
3 **mientras** *Condiciones de término* **hacer**
4 $s' = M\_Trayectoria(s)$;
5 **para cada** *característica i de* s' **hacer**
6 $u_i = \frac{c_i}{1+p_i}$ // Calcular utilidad ;
7 $u_j = \max_{i=1,\ldots,m}(u_i)$
8 $p_j = p_j + 1$ // cambiar la función objetivo penalizando la característica j ;
9 **devolver** s

Búsqueda Aleatoria Voraz (GRASP, Greedy Randomized Adaptive Search Procedure)

Algoritmo 5: GRASP

Entrada: $f :\rightarrow \mathbb{R}$; *numIters*, Número máximo de iteraciones;
Random_Greedy, Heurística constructiva aleatoria;

Resultado: s', mejor solución encontrada

1 **mientras** *Condiciones de término* **hacer**
2 $s = Random\_Greedy(seed)$;
3 $s' =$ Búsqueda Local (s) ;
4 **devolver** s