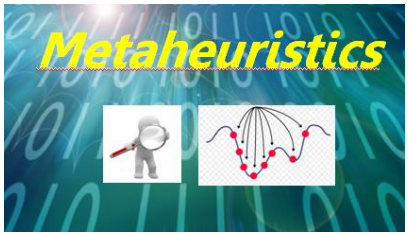
METAHEURÍSTICAS
2019 - 2020



- Tema 1. Introducción a las Metaheurísticas
- Tema 2. Modelos de Búsqueda: Entornos y Trayectorias vs Poblaciones
- Tema 3. Metaheurísticas Basadas en Poblaciones
- Tema 4: Algoritmos Meméticos
- Tema 5. Metaheurísticas Basadas en Trayectorias
- Tema 6. Metaheurísticas Basadas en Adaptación Social
- Tema 7. Aspectos Avanzados en Metaheurísticas
- Tema 8. Metaheurísticas Paralelas

Objetivos

- Conocer los fundamentos de las búsquedas basadas en trayectorias: enfriamiento simulado y búsqueda tabu.
- Entender los fundamentos termodinámicos que sirven de base al enfriamiento simulado.
- Conocer las diferentes fases que componen un algoritmo de enfriamiento simulado, así como los parámetros y componentes que lo condicionan.
- Saber aplicar el enfriamiento simulado a problemas reales.

Objetivos

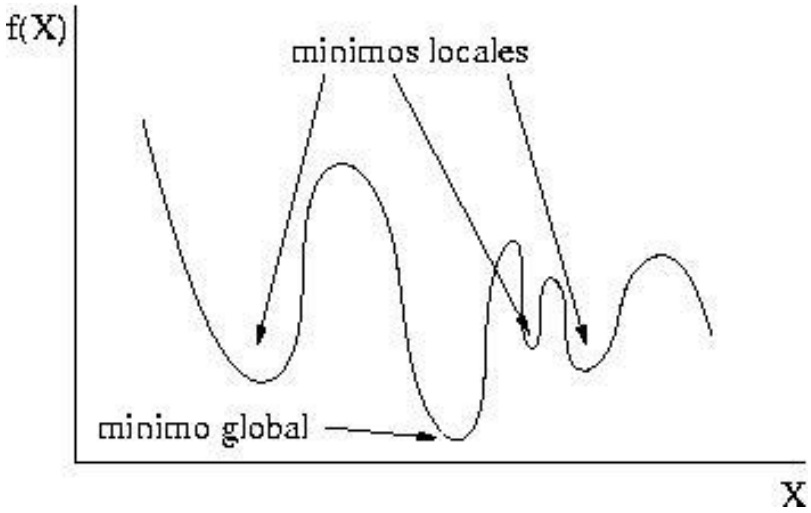
- Conocer los fundamentos de las técnicas basadas en Búsqueda Tabú.
- Distinguir el funcionamiento de la memoria de corto plazo y de la memoria de largo plazo, así como el cometido de cada una de ellas.
- Tener capacidad para, dado un problema dado, saber aplicar las técnicas de Búsqueda Tabú para su resolución.

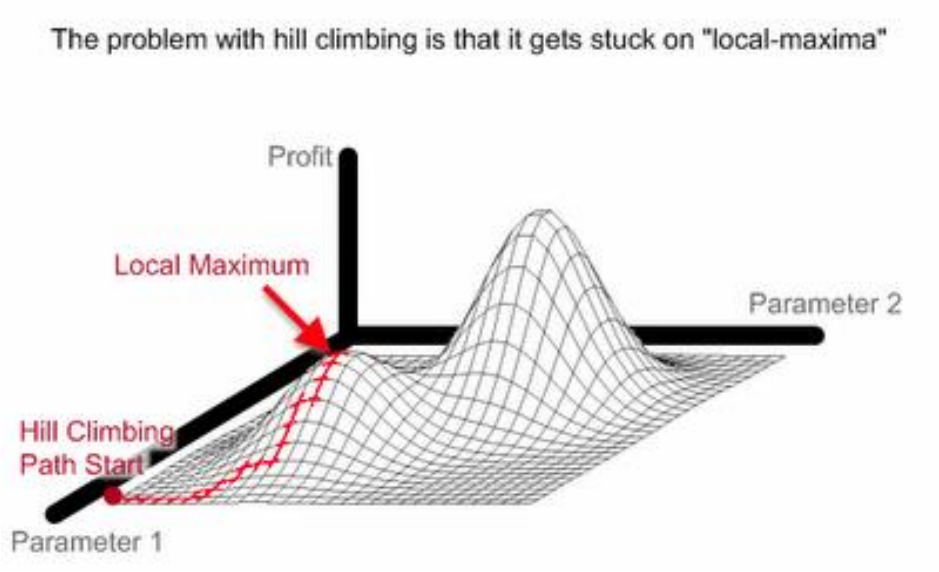
Objetivos

- Describir las características de los métodos de búsqueda local multiarranque.
- Estudiar los diferentes enfoques existentes en búsquedas de entorno variable aplicados a trayectorias multiarranque.
- Conocer los algoritmos básicos multiarranque, así como los algoritmos GRASP, ILS y VNS.
- Saber aplicar este tipo de algoritmos a problemas reales.

Problemas de la Búsqueda Local

- Suele caer en óptimos locales, que a veces están bastante alejados del óptimo global del problema





Problemas de la Búsqueda Local

SOLUCIONES: 3 opciones para salir de los óptimos locales

- Permitir movimientos de empeoramiento de la solución actual
  Ejemplo: Enfriamiento Simulado, Búsqueda Tabú

- Modificar la estructura de entornos
  Ejemplo: Búsqueda Tabú, Búsqueda en Entornos Variables: VNS, …

- Volver a comenzar la búsqueda desde otra solución inicial
  Ejemplo: Búsquedas Multiarranque, ILS, VNS, …

METAHEURÍSTICAS

TEMA 5. Métodos Basados en Trayectorias

1. Enfriamiento Simulado
2. Busqueda Tabu
3. Métodos Basados en Trayectorias Múltiples

METAHEURÍSTICAS

5.1. Algoritmos de Enfriamiento Simulado

1. Fundamentos
2. Algoritmo básico de ES
3. Parámetros y componentes
4. Aplicaciones

- *S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by Simulated Annealing, Science 220:4598 (1983) 671-680*
- *A. Díaz y otros. Optimización Heurística y Redes Neuronales. Paraninfo, 1996*
- *K.A. Dowsland, A. Díaz. Diseño de heurísticas y fundamentos del recocido simulado. Inteligencia Artificial VII:2 (2003) 93-101.*
- *B Suman, P Kumar. A survey of simulated annealing as a tool for single and multiobjective optimization. JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY 57 (10): 1143-1160 OCT 2006*
- *D. Henderson, S.H. Jacobson, A.W. Johnson. Chapter 10: The Theory and Practice of Simmulated Annealing. In: F. Glover, G.A. Kochenberber, (Eds.). Handbook of Metaheuristics. Kluwer Academics. (2003) 287-319.*

1. Fundamentos

1.1. Introducción

1.2. Modelo de Metrópolis

1.3. Analogías: Termodinámica – Optimización

1. Introducción

El Enfriamiento o Recocido Simulado es un algoritmo de búsqueda por entornos con un criterio probabilístico de aceptación de soluciones basado en Termodinámica

1.1. Introducción

Optimization by Simulated Annealing

S. Kirkpatrick [1], C. D. Gelatt Jr. [1], and M. P. Vecchi [2 1]
Research staff members at IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598
[2] Instituto Venezolano de Investigaciones Cientificas, Caracas 1010A, Venezuela

***Science* 13 May 1983:**
Vol. 220. no. 4598, pp. 671 - 680
DOI: 10.1126/science.220.4598.671

There is a deep and useful connection between statistical mechanics (the behavior of systems with many degrees of freedom in thermal equilibrium at a finite temperature) and multivariate or combinatorial optimization (finding the minimum of a given function depending on many parameters). A detailed analogy with annealing in solids provides a framework for optimization of the properties of very large and complex systems. This connection to statistical mechanics exposes new information and provides an unfamiliar perspective on traditional optimization problems and methods.

S. Kirkpatrick and C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, **Science**, Vol 220, Number 4598, pages 671-680, 1983

1.1. Introducción

- **Un modo de evitar que la búsqueda local finalice en óptimos locales, hecho que suele ocurrir con los algoritmos tradicionales de búsqueda local, es permitir que algunos movimientos sean hacia soluciones peores**

- **Pero si la búsqueda está avanzando realmente hacia una buena solución, estos movimientos "de escape de óptimos locales" deben realizarse de un modo controlado**

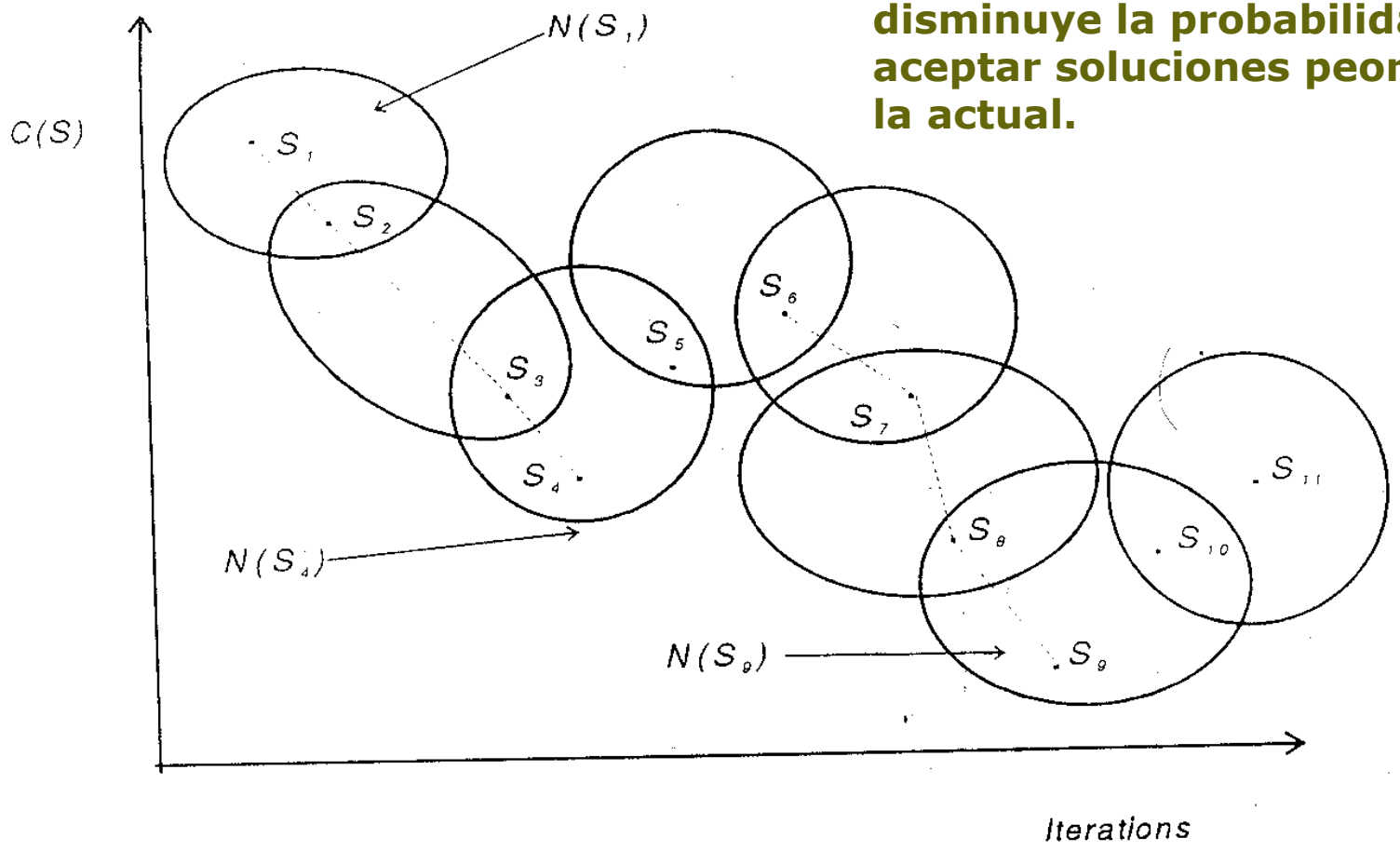
1.1. Introducción

- **En el caso del Enfriamiento Simulado (ES), esto se realiza controlando la frecuencia de los movimientos de escape mediante una función de probabilidad que hará disminuir la probabilidad de estos movimientos hacia soluciones peores conforme avanza la búsqueda (y por tanto estamos más cerca, previsiblemente, del óptimo local)**

- **Se aplica la filosofía habitual de búsqueda de diversificar al principio e intensificar al final**

1.1. Introducción

Con el aumento de iteraciones disminuye la probabilidad de aceptar soluciones peores que la actual.



1.2. Algoritmo de Metrópolis

- **El fundamento de este control se basa en el trabajo de Metrópolis (1953) en el campo de la termodinámica estadística**

- **Básicamente, Metrópolis modeló el proceso de enfriamiento simulando los cambios energéticos en un sistema de partículas conforme decrece la temperatura, hasta que converge a un estado estable (congelado). Las leyes de la termodinámica dicen que a una temperatura t la probabilidad de un incremento energético de magnitud δE se puede aproximar por**

$$P[\delta E] = \exp(-\delta E/kt)$$

siendo k una constante física denominada Boltzmann

- **En el modelo de Metrópolis, se genera una perturbación aleatoria en el sistema y se calculan los cambios de energía resultantes: si hay una caída energética, el cambio se acepta automáticamente; por el contrario, si se produce un incremento energético, el cambio será aceptado con una probabilidad indicada por la anterior expresión**

1.2. Algoritmo de Metrópolis

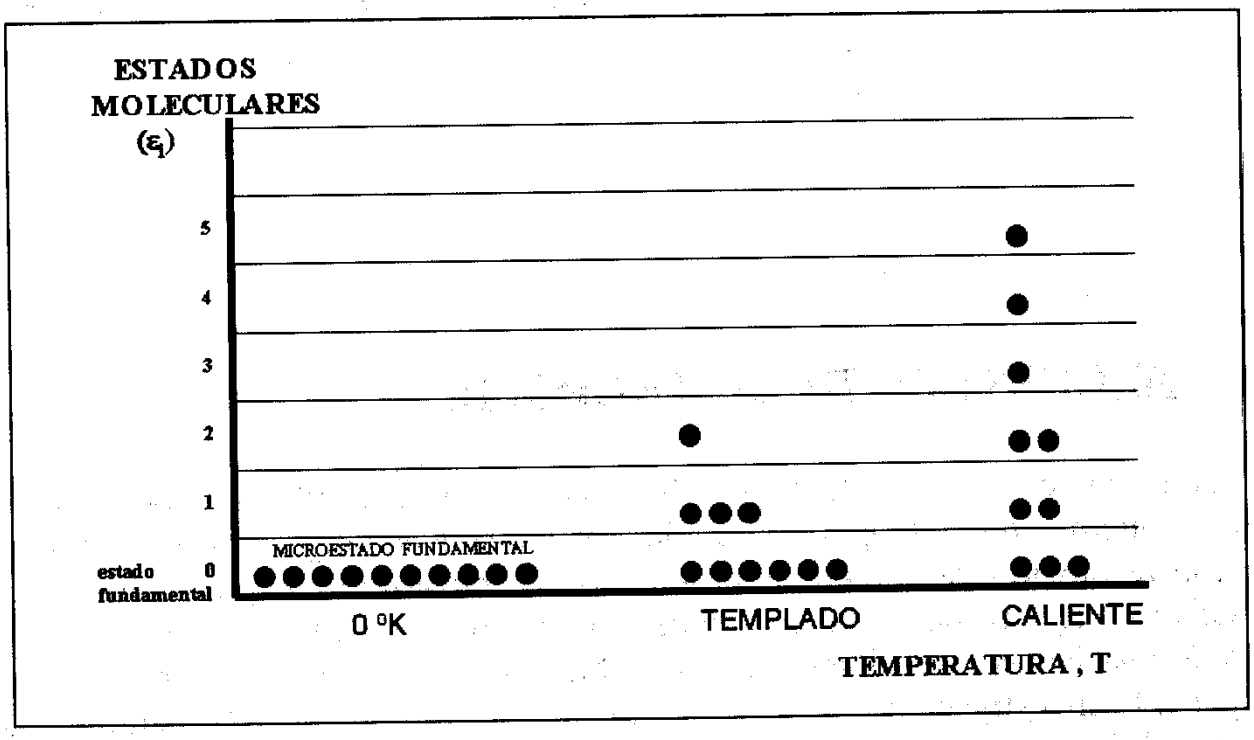
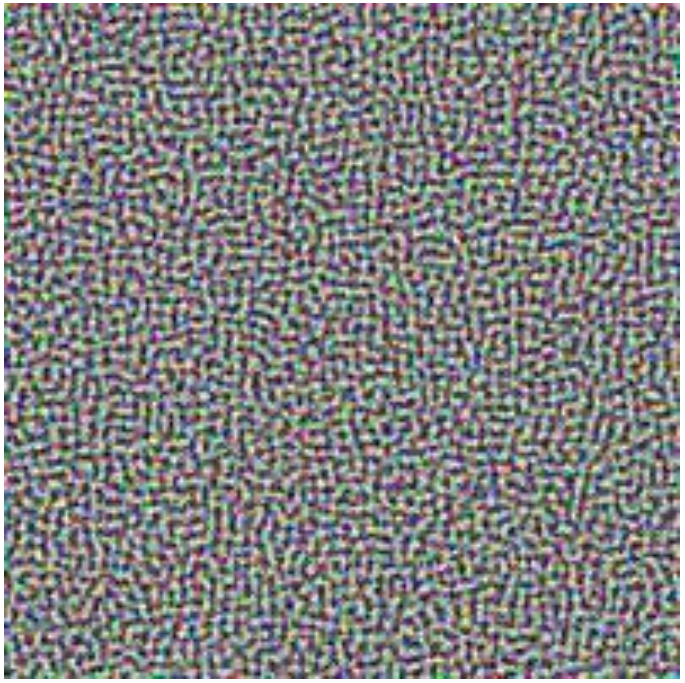
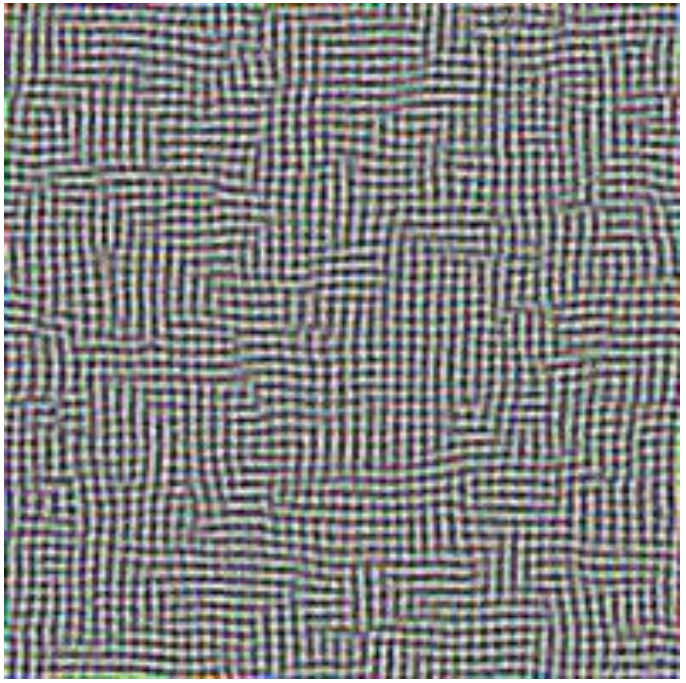


Figura 2.1. *Ejemplo de diferentes microestados según la temperatura, para N=10. En el microestado correspondiente a TEMPLADO, es n_0=6, n_1=3, n_2=1, $E= \Sigma n_i i$=5. El número de moléculas en los estados superiores decrece exponencialmente para una T fija.*

1.2. Algoritmo de Metrópolis





Example illustrating the effect of cooling schedule on the performance of simulated annealing. The problem is to rearrange the pixels of an image so as to minimize a certain potential energy function, which causes similar colours to attract at short range and repel at slightly larger distance. The elementary moves swap two adjacent pixels. The images were obtained with fast cooling schedule (left) and slow cooling schedule (right), producing results similar to amorphous and crystalline solids, respectively.

1.3. Analogías: Termodinámica - Optimización

Simulación Termodinámica	**Optimización Combinatoria**
■ Estados del sistema	■ Soluciones factibles
■ Energía	■ Coste
■ Cambio de estado	■ Solución en el entorno
■ **Temperatura**	■ **Parámetro de control**
■ Estado congelado	■ Solución heurística

S. Kirkpatrick and C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, Science, Vol 220, Number 4598, pages 671-680, 1983

2. Algoritmo básico de ES

- **El algoritmo de Enfriamiento Simulado es un método de búsqueda por entornos caracterizado por un criterio de aceptación de soluciones vecinas que se adapta a lo largo de su ejecución**

- **Hace uso de una variable llamada Temperatura, T, cuyo valor determina en qué medida pueden ser aceptadas soluciones vecinas peores que la actual**

- **La variable Temperatura se inicializa a un valor alto, denominado Temperatura inicial, T_0, y se va reduciendo cada iteración mediante un mecanismo de enfriamiento de la temperatura, $\alpha(\cdot)$, hasta alcanzar una Temperatura final, T_f**

2. Algoritmo básico de ES

- **En cada iteración se genera un número concreto de vecinos, L(T), que puede ser fijo para toda la ejecución o depender de la iteración concreta**

- **Cada vez que se genera un vecino, se aplica el criterio de aceptación para ver si sustituye a la solución actual**

- **Si la solución vecina es mejor que la actual, se acepta automáticamente, tal como se haría en la búsqueda local clásica**

- **En cambio, si es peor, aún existe la probabilidad de que el vecino sustituya a la solución actual. Esto permite al algoritmo salir de óptimos locales, en los que la BL clásica quedaría atrapada**

2. Algoritmo básico de ES

- **Esta probabilidad depende de la diferencia de costes entre la solución actual y la vecina, δ, y de la temperatura T:**

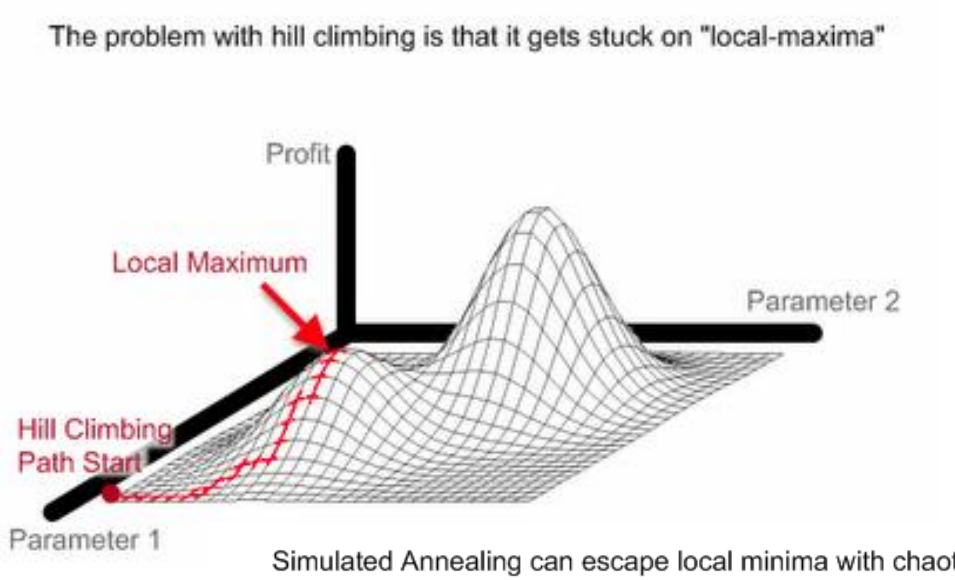
$$P_{aceptación} = \exp(-\delta/T)$$

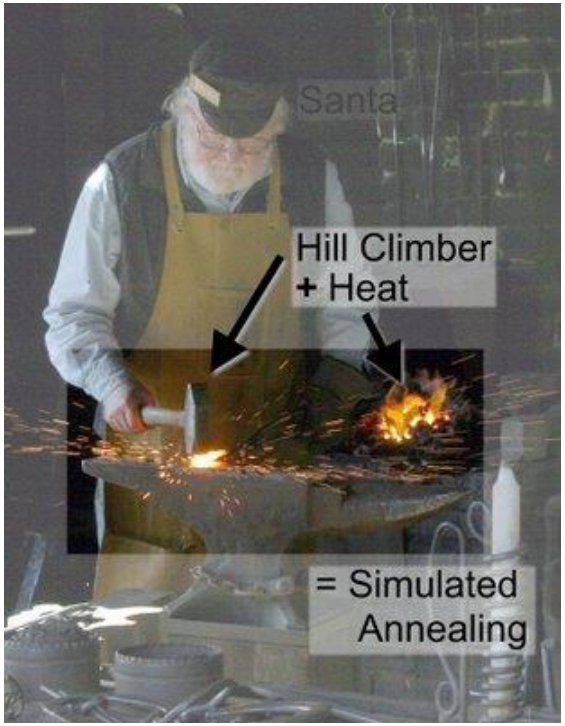
- **A mayor temperatura, mayor probabilidad de aceptación de soluciones peores. Así, el algoritmo acepta soluciones mucho peores que la actual al principio de la ejecución (exploración) pero no al final (explotación)**

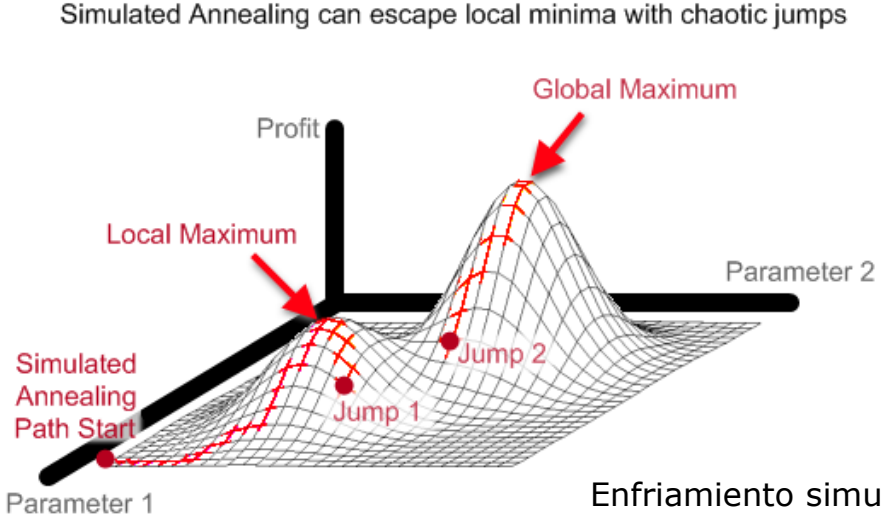
- **A menor diferencia de costes, mayor probabilidad de aceptación de soluciones peores**

- **Una vez finalizada la iteración, es decir, tras generar L(T) soluciones vecinas, se enfría la temperatura y se pasa a la siguiente iteración**

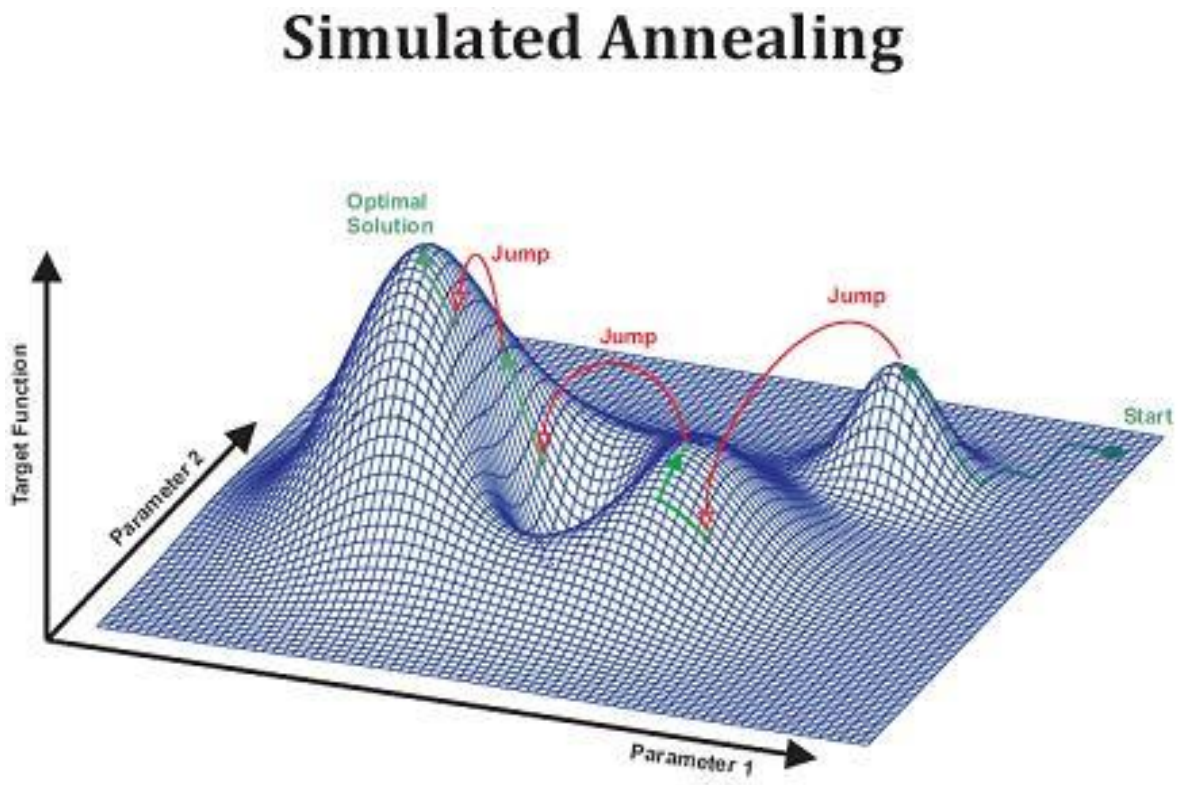
2. Algoritmo básico de ES







2. Algoritmo básico de ES



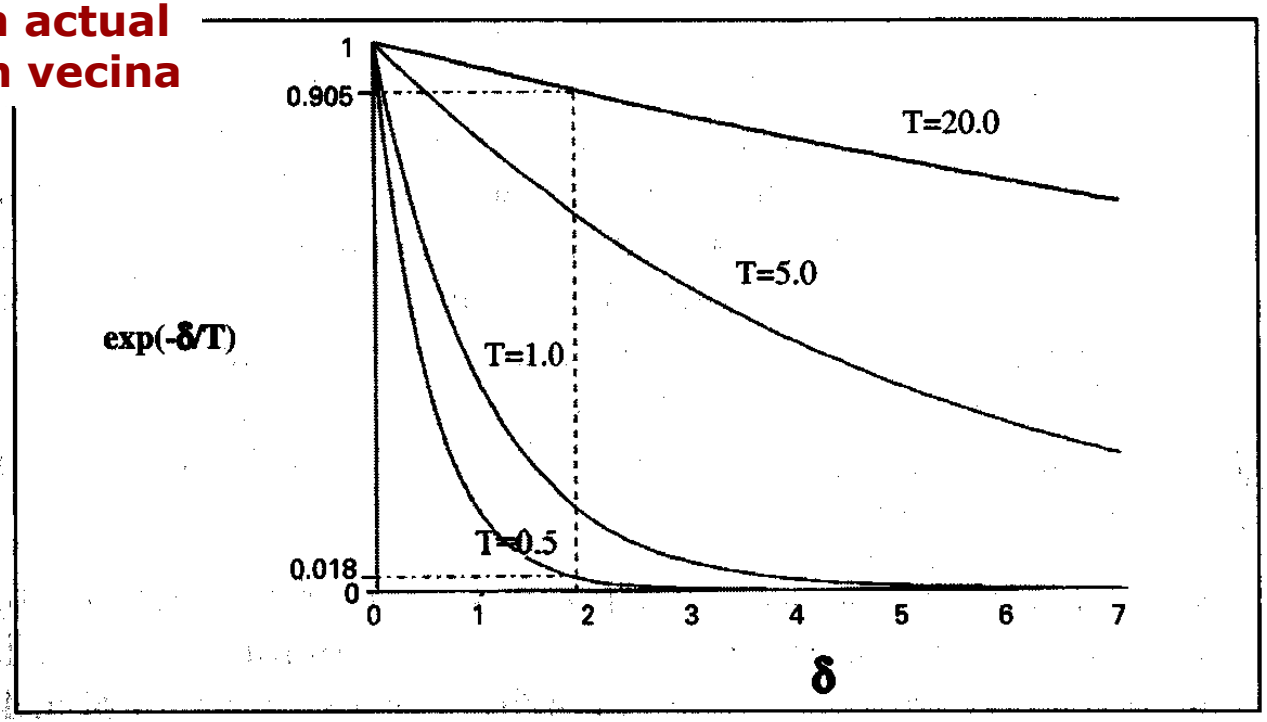
2. Algoritmo básico de ES

Procedimiento Simulated Annealing *(Δf para minimizar)*

```
Start
  T ← T0;   s ← GENERATE();   Best Solution ← s;
  Repeat
     For cont = 1 to L(T) do /* Inner loop
       Start
       s' ← NEIGHBORHOOD_OP(s); /* A single move
       Δf=f(s')−f(s);
       If ((Δf<0) or (U(0,1) ≤ exp(-Δf/k·T))) then
           s ← s';
           If COST(s) is better than COST(Best Solution)
              then Best Solution ← s;
       End
      T ← g(T); /* Cooling scheme. The classical one is geometric: T ← α·T
  until (T ≤ Tf); /* Outer loop
  Return(Best Solution);
End
```

2. Algoritmo básico de ES. Descenso de temperatura vs diferencia en la f. objetivo

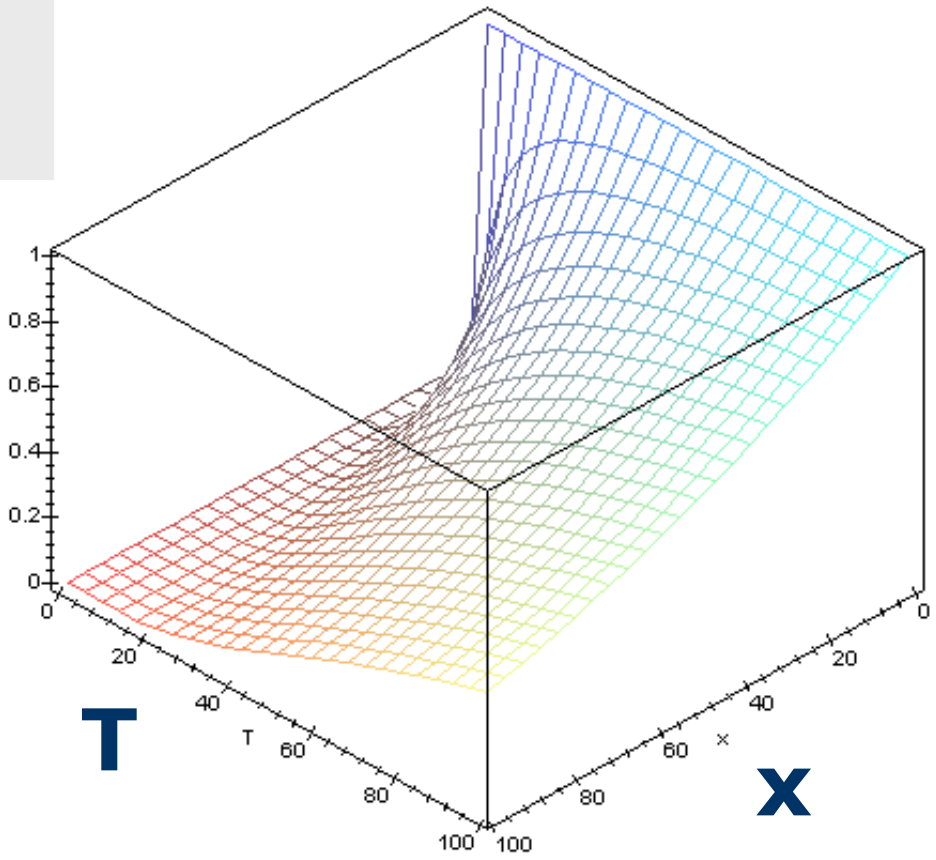
δ = C(s') – C(s)
s = solución actual
s' = solución vecina



2. *Valor del factor de Boltzmann en función de la temperatura T y de δ. Para δ=2, es 50 veces menos probable un movimiento si T=0,5 que si T=20.*

2. Algoritmo básico de ES. Descenso de temperatura vs diferencia en la f. objetivo

$$P = \exp(-x/t)$$



3. Parámetros y componentes

3.1. Representación

3.2. Generación de la solución inicial

3.3. Mecanismo de transición de soluciones

3.4. Secuencia de enfriamiento

3.1. Representación

ALGUNOS EJEMPLOS

Vector ordenado de números enteros.

Ejemplo: Problema del Viajante de Comercio

(7 6 3 1 5 4 2 8)

Vector binario.

Ejemplo: Problema de la Mochila

(0 1 0 0 1 1 1 0)

Vector de Números Reales.

Ejemplo: Problemas de Optimización con Parámetros Continuos

(2.7 3.5 4 6.27)

....

3.2. Generación de la solución inicial

- Uso de técnicas eficientes para obtenerla
- Uso de conocimiento experto

  Ejemplo: solución generada por un algoritmo *greedy*

3.3. Mecanismo de transición de soluciones

1. Generación de una nueva solución
   - Definición del conjunto de vecinos
   - Selección de un elemento de dicho conjunto

2. Cálculo de la diferencia de costos entre la solución actual y la vecina

3. Aplicación del criterio de aceptación

3.3. Mecanismo de transición de soluciones

Procedimiento Simulated Annealing

Start
  $T \leftarrow T_0$; $s \leftarrow$ GENERATE(); Best Solution $\leftarrow s$;
  Repeat
    For *cont* = 1 to *L(T)* **do** /* Inner loop
      Start
      $s' \leftarrow$ NEIGHBORHOOD_OP(s);
      $\Delta f = f(s') - f(s)$;
      If (($\Delta f < 0$) or (U(0,1) $\leq \exp(-\Delta f / k \cdot T)$)) then
        $s \leftarrow s'$;
        If COST(s) **is better than** COST(Best Solution)
          then Best Solution $\leftarrow s$;
      End
    $T \leftarrow g(T)$; /* Cooling scheme. The classical one is geometric: $T \leftarrow \alpha \cdot T$
  until ($T \leq T_f$); /* Outer loop
  Return(Best Solution);
End

Generación de una nueva solución

Cálculo de la diferencia de costos

Aplicación del criterio de aceptación

3.4. Secuencia de enfriamiento

Procedimiento Simulated Annealing

```
Start
  T ← T0;
  s ← GENERATE();   Best Solution ← s;
  Repeat
     For cont = 1 to L(T) do
       Start
       s' ← NEIGHBORHOOD_OP(s);
       Δf=f(s')−f(s);
       If ((Δf<0) or (U(0,1) ≤ exp(-Δf/k·T))) then
           s ← s';
           If COST(s) is better than COST(Best Solution)
              then Best Solution ← s;
       End
      T ← g(T);
  until (T ≤ Tf);
  Return(Best Solution);
End
```

Valor inicial del parámetro de control

Velocidad de enfriamiento L(T)

Mecanismo de enfriamiento

Condición de Parada

3.4. Secuencia de enfriamiento

1. Valor Inicial del parámetro de control (temperatura)

No parece conveniente considerar valores fijos independientes del problema.

PROPUESTA: $T_0 = (\mu / -\ln(\phi)) \cdot C(S_0)$

Tanto por uno ϕ de probabilidad de que una solución sea un μ por uno peor que la solución inicial S_0

3.4. Secuencia de enfriamiento

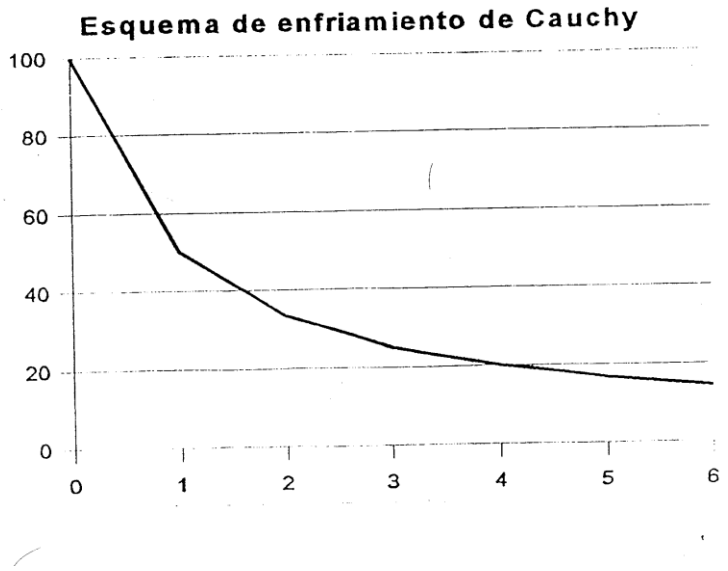
2. Mecanismo de enfriamiento

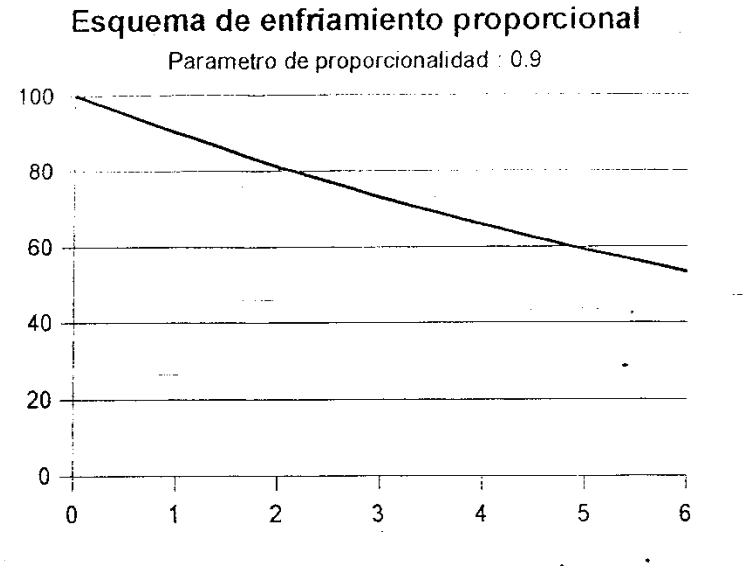
Existen varios mecanismos de enfriamiento:

- **Enfriamiento basado en sucesivas temperaturas descendentes fijadas por el usuario**
- **Enfriamiento con descenso constante de temperatura**
- **Descenso geométrico:** $T_{k+1} = \alpha \cdot T_k$
  k = nº iteración actual, α **constante cercana a 1 (usualmente,** $\alpha \in [0.8, 0.99]$**)**
- **Criterio de Boltzmann:** $T_k = T_0 / (1 + \log(k))$
- **Esquema de Cauchy:** $T_k = T_0 / (1 + k)$
- **Para controlar el número de iteraciones (Cauchy modificado):** $T_{k+1} = T_k / (1 + \beta \cdot T_k)$
  Para ejecutar exactamente M iteraciones $\Rightarrow$ $\beta = (T_o - T_f) / (M \cdot T_o \cdot T_f)$

3.4. Secuencia de enfriamiento

2. Mecanismo de enfriamiento





3.4. Secuencia de enfriamiento

3. Velocidad de enfriamiento

- L(T) debe ser suficientemente grande como para que el sistema llegue a alcanzar su estado estacionario para esa temperatura
- Lo habitual es que sea un valor fijo, pero hay una variante que permite decidir mejor cuando finalizar la iteración actual y enfriar
- Consiste en enfriar cuando se dé una de las dos situaciones siguientes:
  - Se ha generado un número máximo de vecinos (*máx_vecinos*).
  - Se han aceptado un número máximo de vecinos (*máx_éxitos*).
- Lógicamente, *máx_vecinos* tiene que ser mayor que *máx_éxitos*. Un buena proporción puede ser *máx_éxitos=0.1*máx_vecinos*

3.4. Secuencia de enfriamiento

4. Condición de parada

- En teoría, el algoritmo debería finalizar cuando T=0. En la práctica, se para:
  - cuando T alcanza o está por debajo de un valor final T_f, fijado previamente, o
  - después de un número fijo de iteraciones.

- Como es difícil dar valor de T_f, se suele usar un número fijo de iteraciones

- Una buena opción es parar cuando no se haya aceptado ningún vecino de los L(T) generados en la iteración actual (*num_éxitos=0*)

  En ese caso, es muy probable que el algoritmo se haya estancado y no vaya a mejorar la solución obtenida

  Combinando este criterio de parada y la condición de enfriamiento de los *máx_vecinos* y *máx_éxitos* se obtiene un equilibrio en la búsqueda que evita malgastar recursos

4. Aplicaciones: TSP

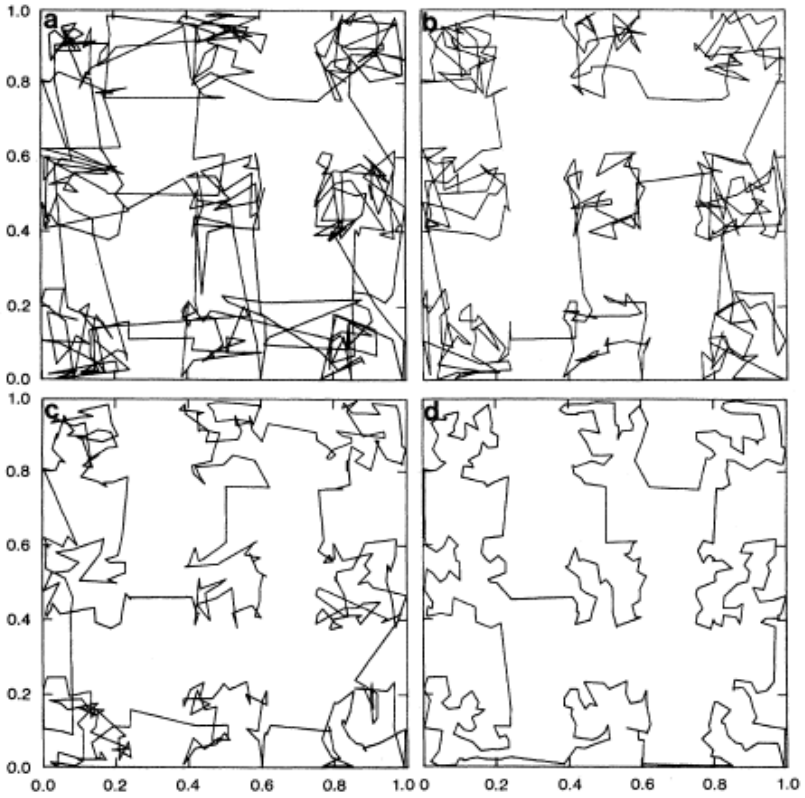
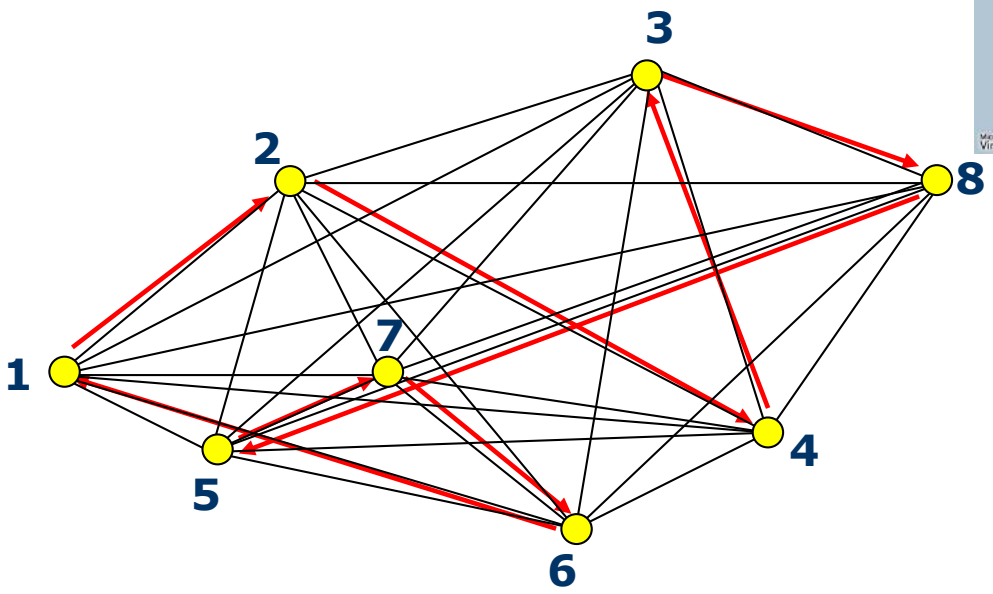


Fig. 9. Results at four temperatures for a clustered 400-city traveling salesman problem. The points are uniformly distributed in nine regions. (a) $T = 1.2$, $\alpha = 2.0567$; (b) $T = 0.8$, $\alpha = 1.515$; (c) $T = 0.4$, $\alpha = 1.055$; (d) $T = 0.0$, $\alpha = 0.7839$.

S. Kirkpatrick and C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, Science, Vol 220, Number 4598, pages 671-680, 1983.

4. Aplicaciones: TSP

- Ejemplo: Viajante de Comercio



- Representación de una solución: Camino

(1 2 4 3 8 5 7 6)

4. Aplicaciones: TSP

1. Esquema de representación: Permutación de {1, ..., n}.

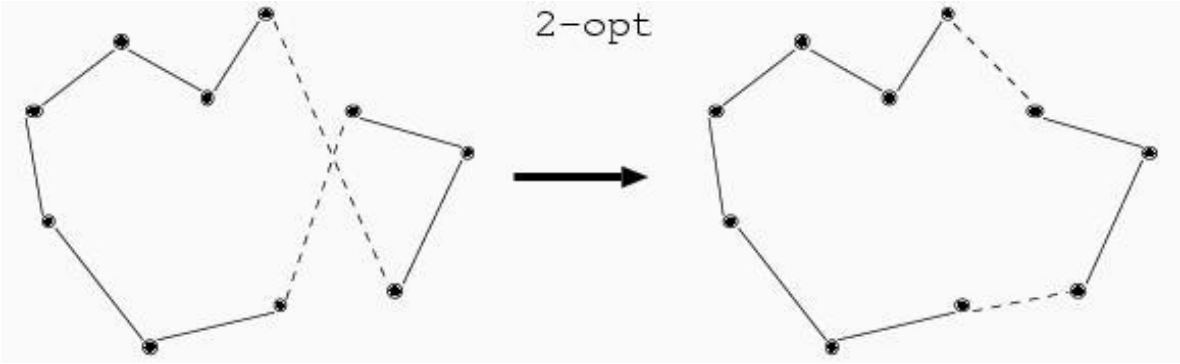
2. Función objetivo:

$$\mathbf{Min}\ C(S) = \sum_{i=1}^{n-1} \left(D[S[i], S[i+1]]\right) + D[S[n], S[1]]$$

3. Mecanismo de generación de la solución inicial: Permutación *aleatoria*.

4. Aplicaciones: TSP. Generación de una nueva solución

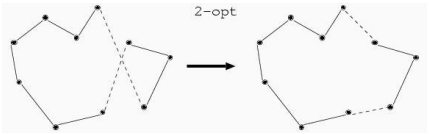
4. Operador de generación de nuevas soluciones:

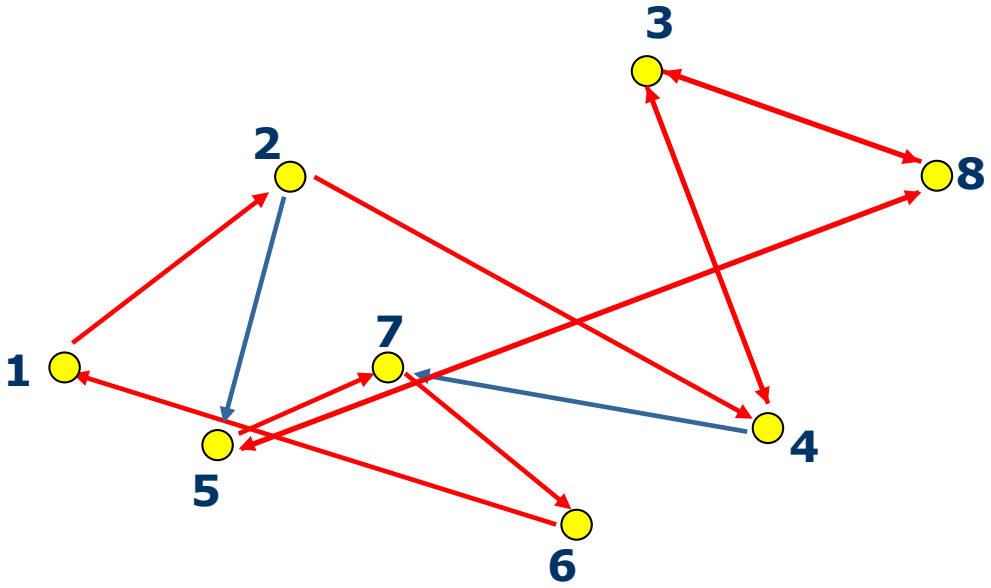


4. Aplicaciones: TSP. Generación de una nueva solución

- Inversión simple (2-Opt): se escoge una sublista y se invierte el orden

(1 2 <u>4 3 8 5</u> 7 6)
(1 2 <u>5 8 3 4</u> 7 6)



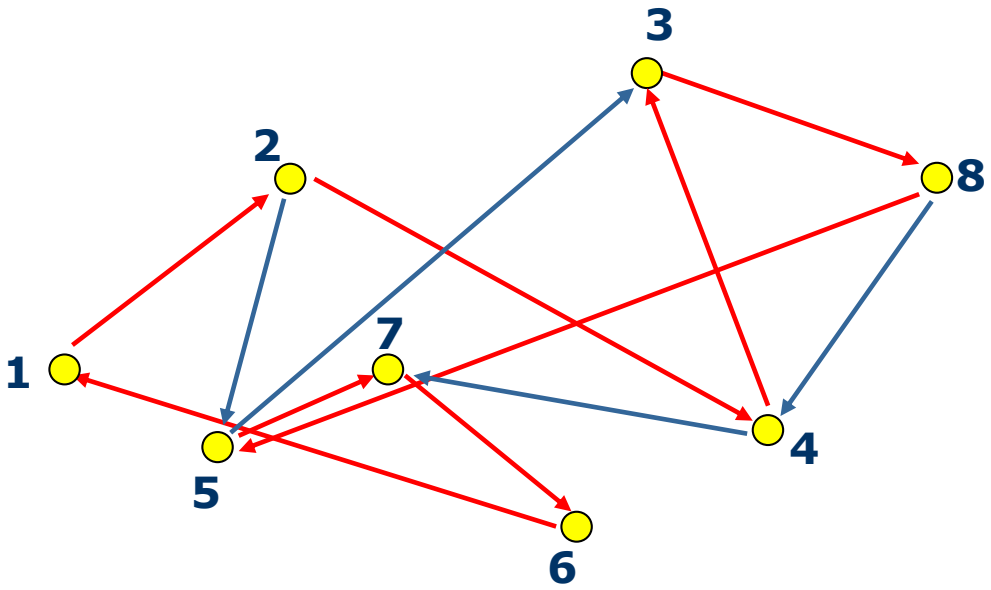


4. Aplicaciones: TSP. Generación de una nueva solución

- Intercambio: se escogen dos elementos y se intercambian

(1 2 <u>4</u> 3 8 <u>5</u> 7 6)

(1 2 <u>5</u> 3 8 <u>4</u> 7 6)



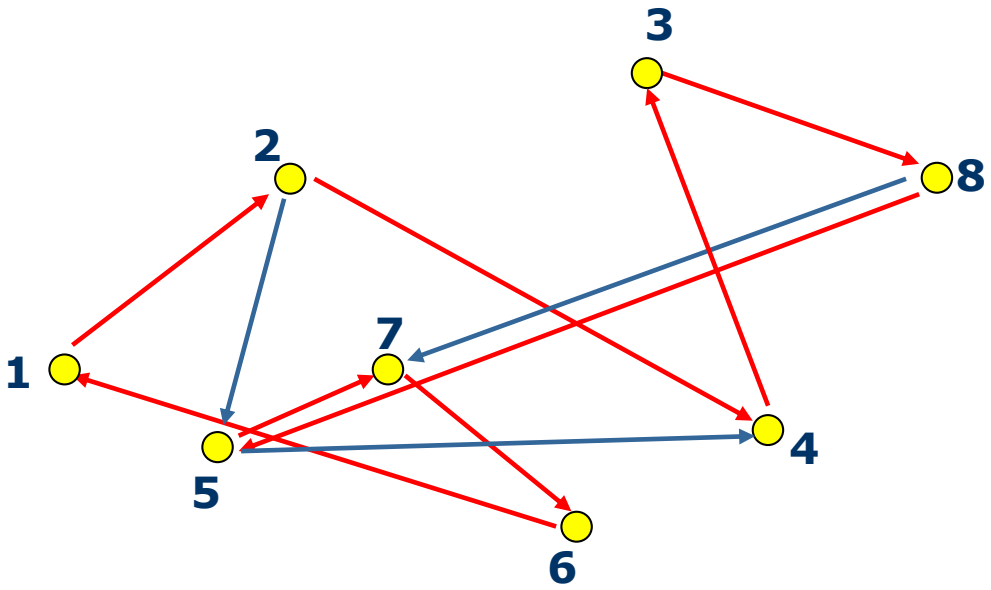
4. Aplicaciones: TSP. Generación de una nueva solución

Otro operador: Viajante de Comercio

- Posición: se elige un elemento, se extrae, y se inserta en una posición determinada

(1 2 _ 4 3 8 5 7 6)

(1 2 5 4 3 8 7 6)



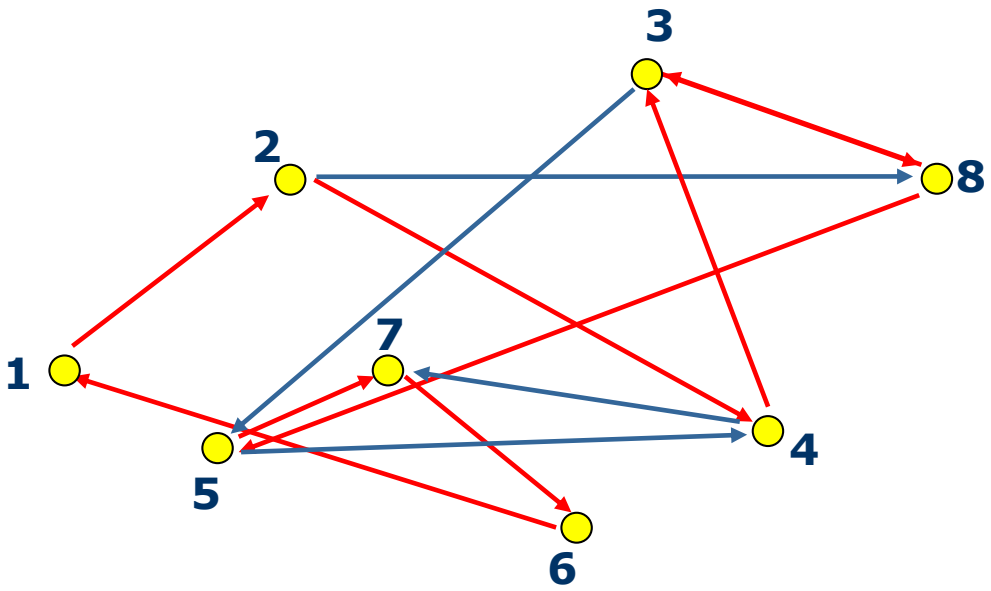
4. Aplicaciones: TSP. Generación de una nueva solución

Otro operador: Viajante de Comercio

- Sublista aleatoria: se escoge una sublista y se baraja. El tamaño de la sublista suele ser fijo y mucho menor que N

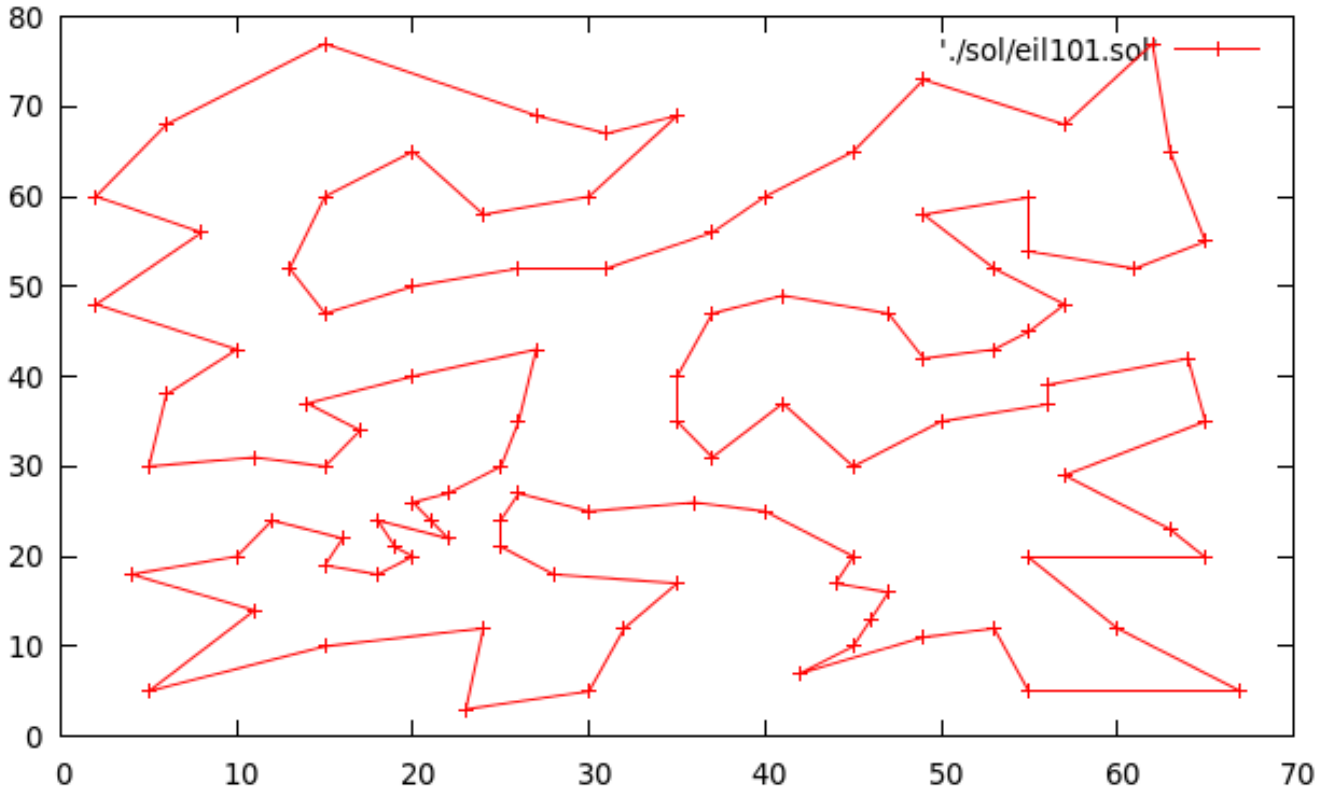
(1 2 4 3 8 5 7 6)

(1 2 8 3 5 4 7 6)



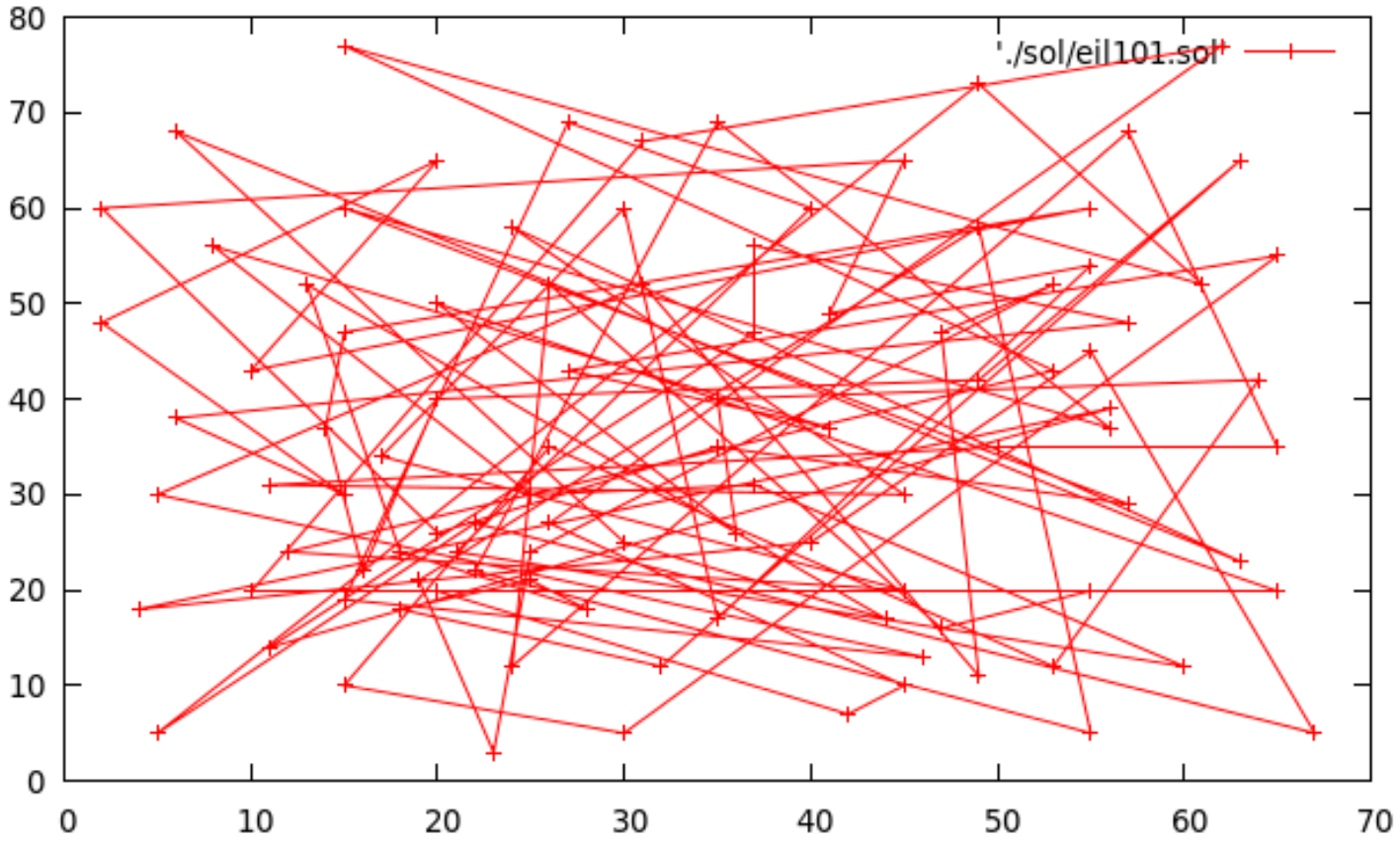
4. Aplicaciones: TSP

Problema Eil101 – Sol. Optima - 629



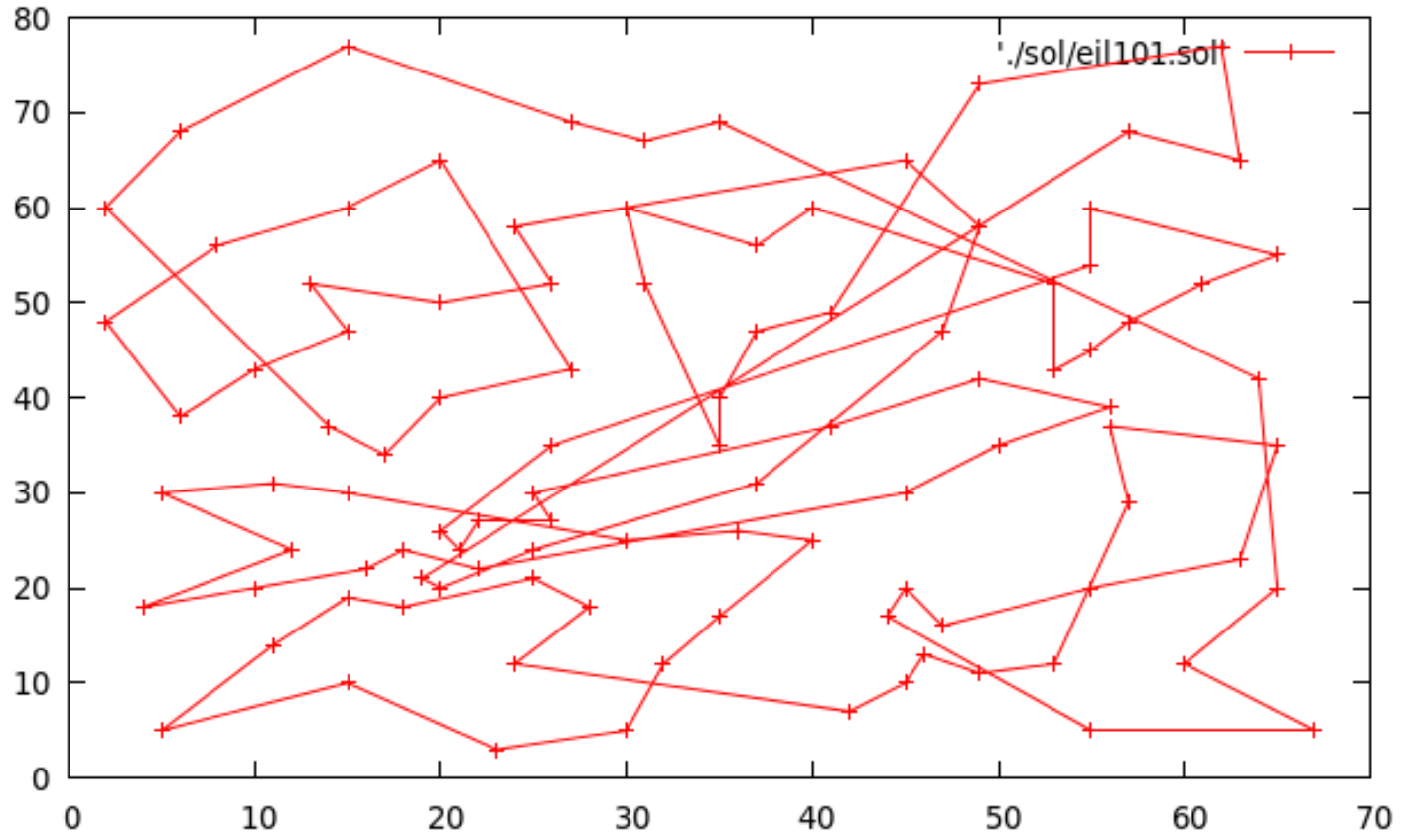
4. Aplicaciones: TSP

Problema Eil101 – Sol. Aleatoria - 3452



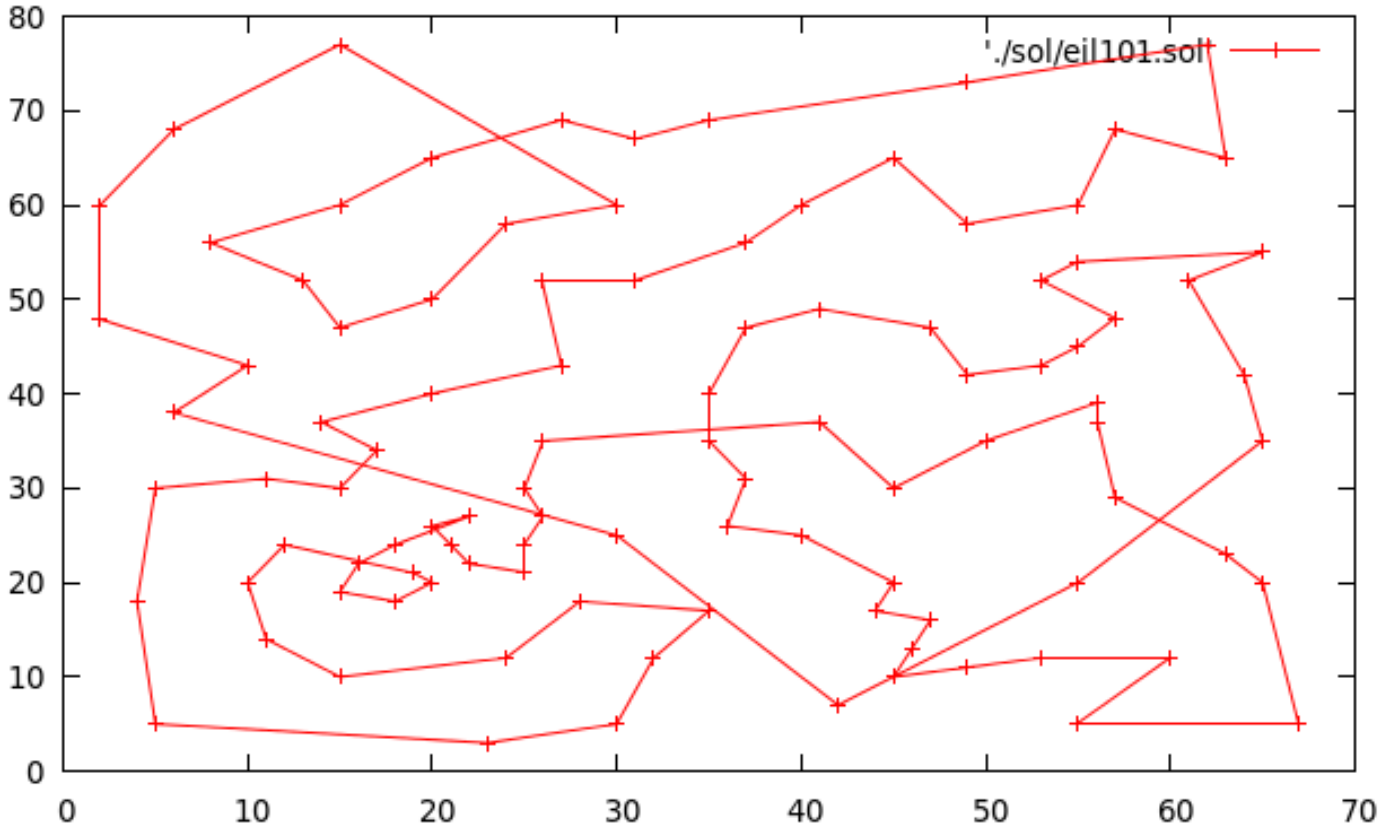
4. Aplicaciones: TSP

Problema Eil201 – Sol. Aleatoria + 2 Opt (inter-cambio de 2 arcos) – 1004 - 0.025 seg



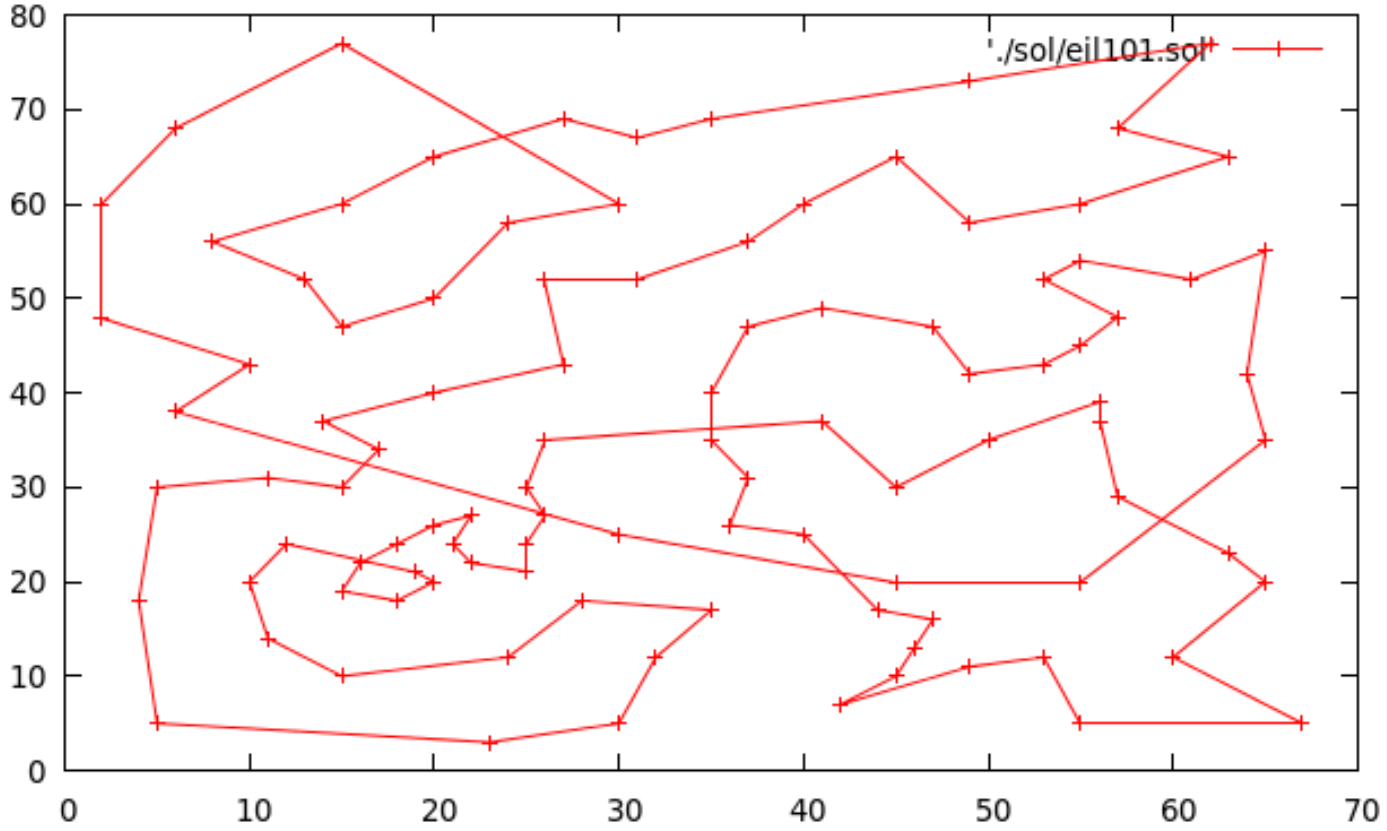
4. Aplicaciones: TSP

Problema Eil101 – Vecino más cercano – 736 – 0.016 seg



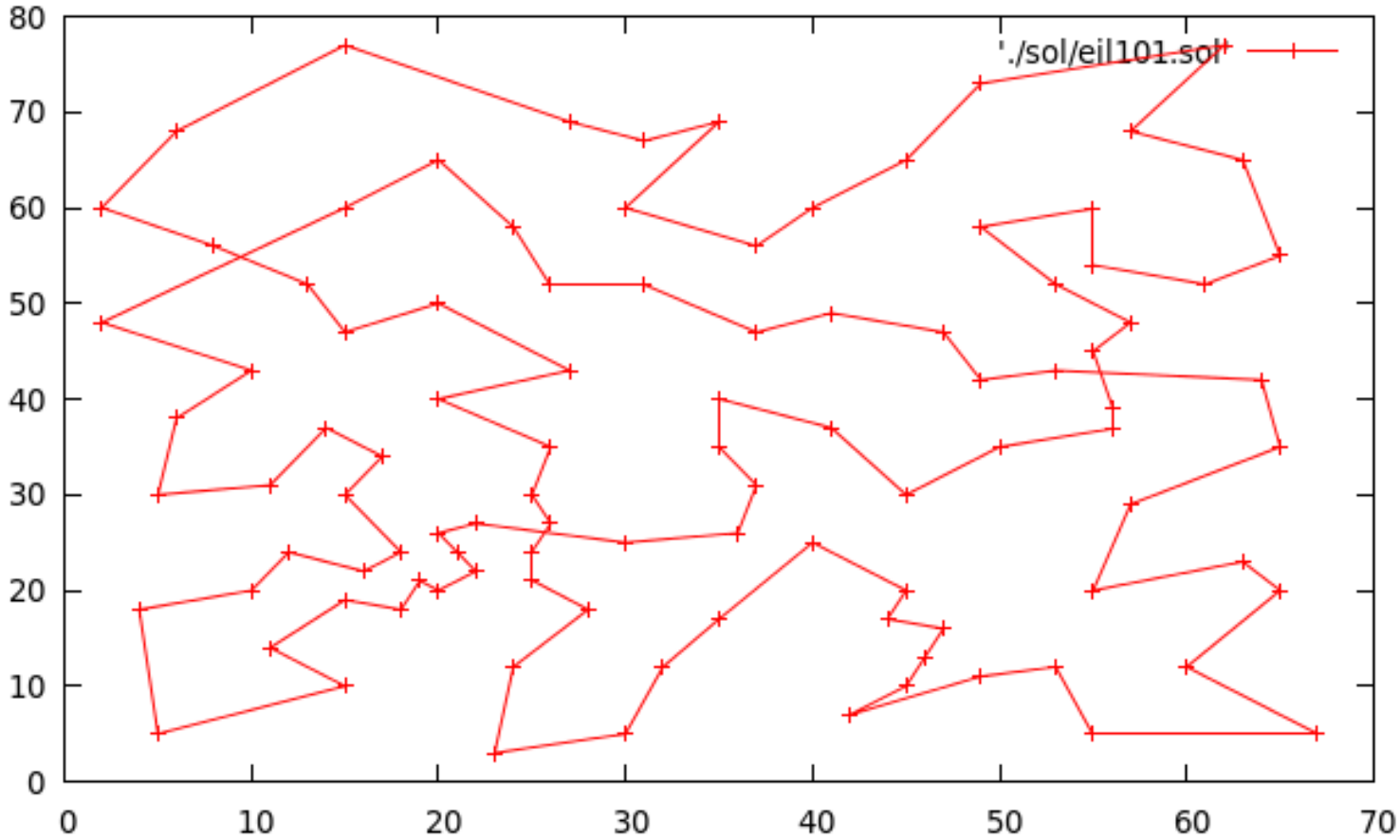
4. Aplicaciones: TSP

Problema Eil101 – Vecino más cercano + LS – 722.002 –Tiempo: 0.016 + 0.0092 seg



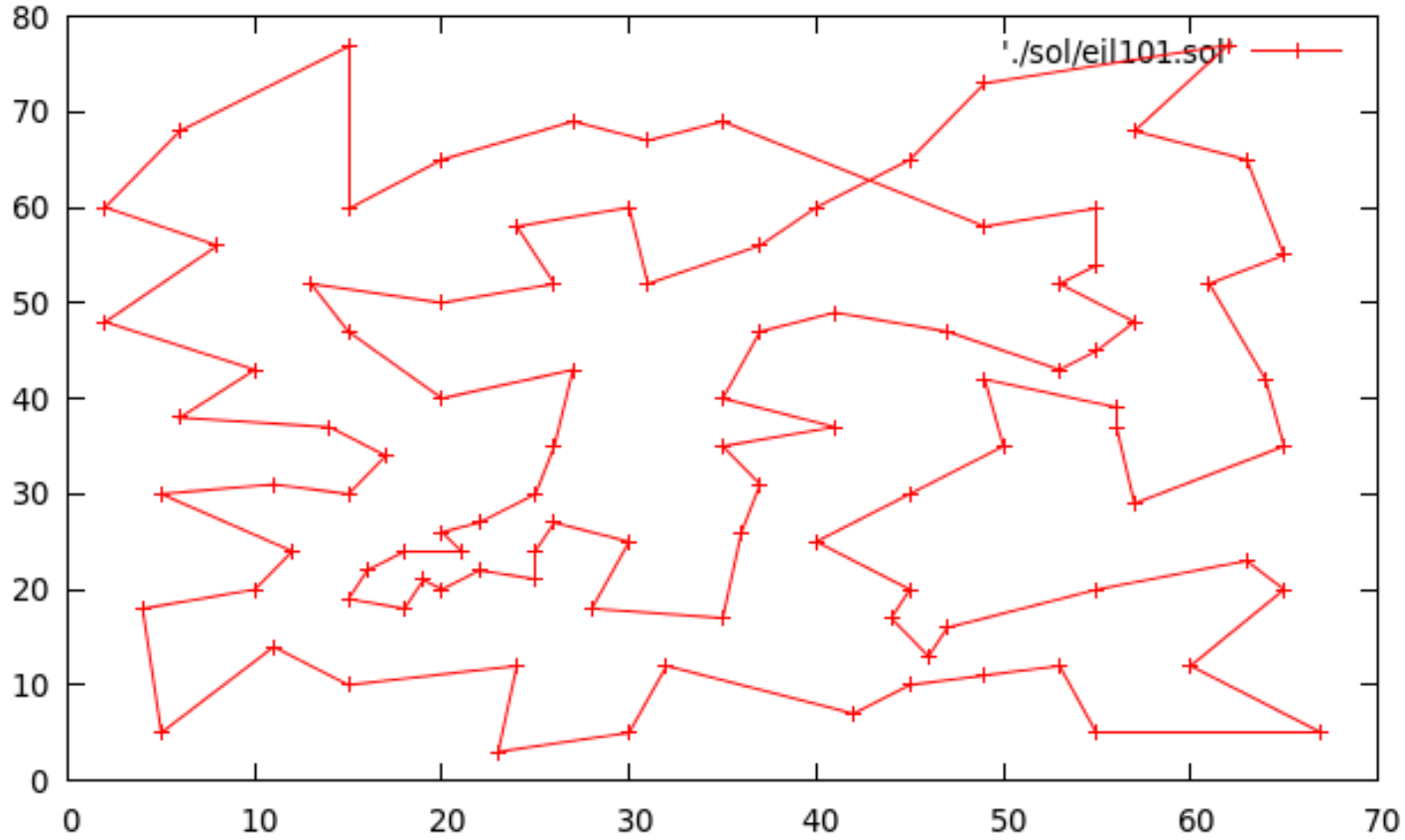
4. Aplicaciones: TSP

Problema Eil101 – SA Valor objetivo: 665.68 – 1.3 seg – 10000 iteraciones



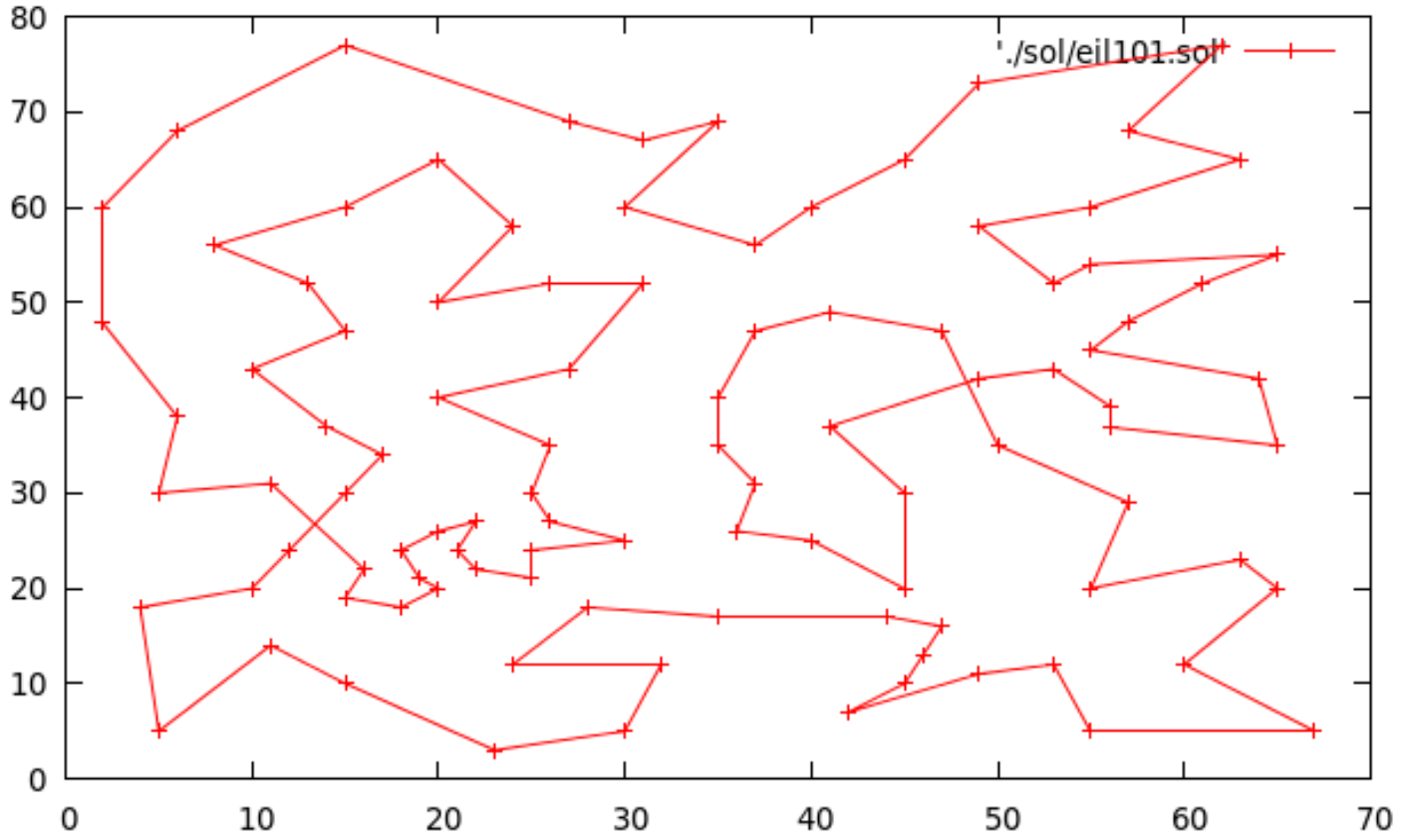
4. Aplicaciones: TSP

Problema Eil101 – SA Valor objetivo:
674.32 – 1.31 seg – 10000 iteraciones



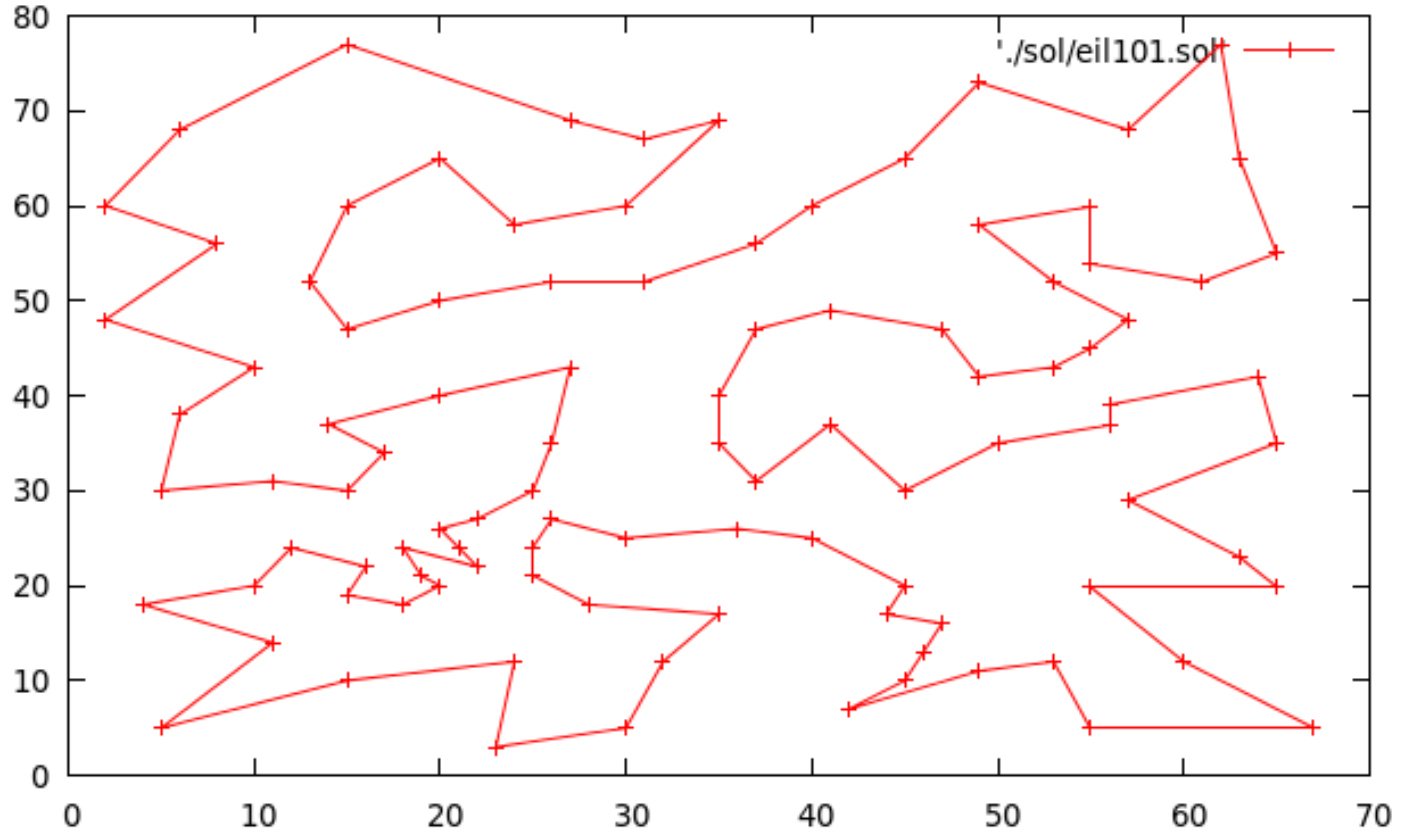
4. Aplicaciones: TSP

Problema Eil101 – SA Valor objetivo: 682.6 – 1.41 seg – 10000 iteraciones



4. Aplicaciones: TSP

Problema Eil101 – Sol. Optima - 629

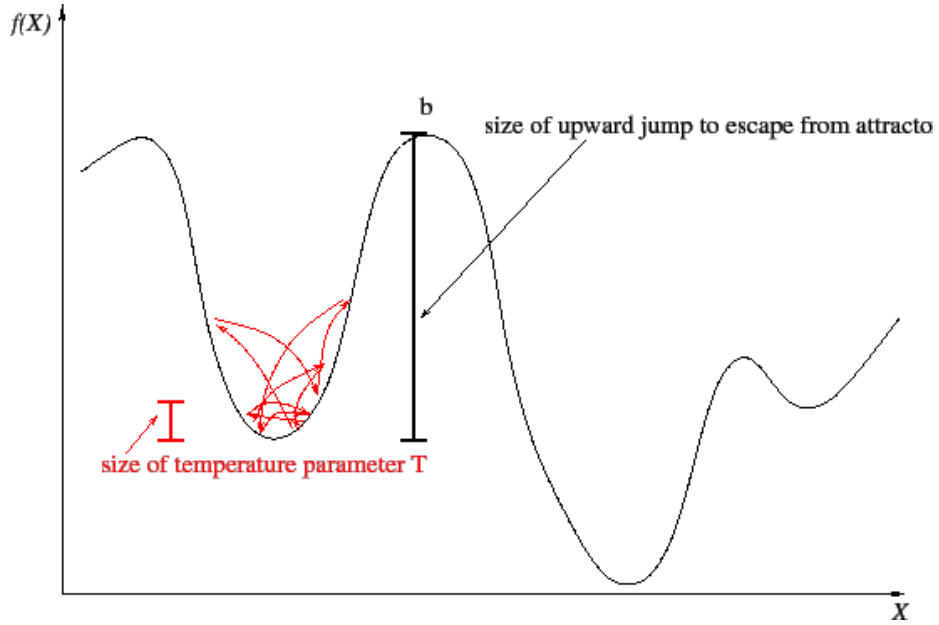


Enfriamiento Simulado

Comentarios Finales:

Técnica de uso muy fácil en muchos problemas.

Riesgos de quedar atrapado en un óptimo local



Si la temperatura es muy baja respecto al tamaño del salto entonces hay riesgos de quedar atrapado en un óptimo local.

Enfriamiento Simulado

Riesgos de quedar atrapado en un óptimo local

Δf	T	$\exp(-\Delta f/T)$		Δf	T	$\exp(-\Delta f/T)$
0.2	0.95	0.810157735		0.2	0.1	0.135335283
0.4	0.95	0.656355555		0.4	0.1	0.018315639
0.6	0.95	0.53175153		0.6	0.1	0.002478752
0.8	0.95	0.430802615		0.8	0.1	0.000335463

Δf = C(s') – C(s)
s = solución actual
s' = solución vecina

Enfriamiento Simulado. Websites

- http://www.ingber.com/#ASA:
  SA C code and materials (papers, slides, …)
- http://www.taygeta.com/annealing/simanneal.html:
  SA C and C++ codes
- http://sci2s.ugr.es/sites/default/files/files/Teaching/GraduatesCourses/Metaheuristicas/Other/binary-simulated-annealing.zip
  Enfriamiento simulado binario

METAHEURÍSTICAS

5.2. Algoritmos de Búsqueda Tabú

1. INTRODUCCIÓN
2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ
3. EJEMPLO: VIAJANTE DE COMERCIO

- *A. Díaz y otros. Optimización Heurística y Redes Neuronales. Paraninfo, 1996*
- *F. Glover, B. Melián. Búsqueda Tabú. Inteligencia Artificial VII:2 (2003) 29-47.*
- *F. Glover, M. Laguna. Tabu Search. Kluwer Academic, 1997.*
- *M. Gendreau. Chapter 2: An Introduction to Tabu Search. In: F. Glover, G.A. Kochenberber, (Eds.). Handbook of Metaheuristics. Kluwer Academics. (2003) 37-54.*

1. INTRODUCCIÓN

Problemas de la Búsqueda Local

Un mal diseño de la función objetivo puede guiar mal la búsqueda y dar lugar a que se obtengan soluciones de baja calidad

SOLUCIONES: 3 opciones para salir de los óptimos locales

- **Permitir movimientos de empeoramiento de la solución actual (Ejemplo: Enfriamiento Simulado, Búsqueda Tabú, …)**
- **Modificar la estructura de entornos (Ejemplo: Búsqueda Tabú, Búsqueda en Entornos Variables: VNS, …)**
- **Volver a comenzar la búsqueda desde otra solución inicial (Ejemplo: Búsquedas Multiarranque, ILS, Búsqueda Tabú, …)**

1. INTRODUCCIÓN

La Búsqueda Tabú es un procedimiento de búsqueda por entornos cuya característica distintiva es el uso de memoria adaptativa y estrategias especiales de resolución de problemas

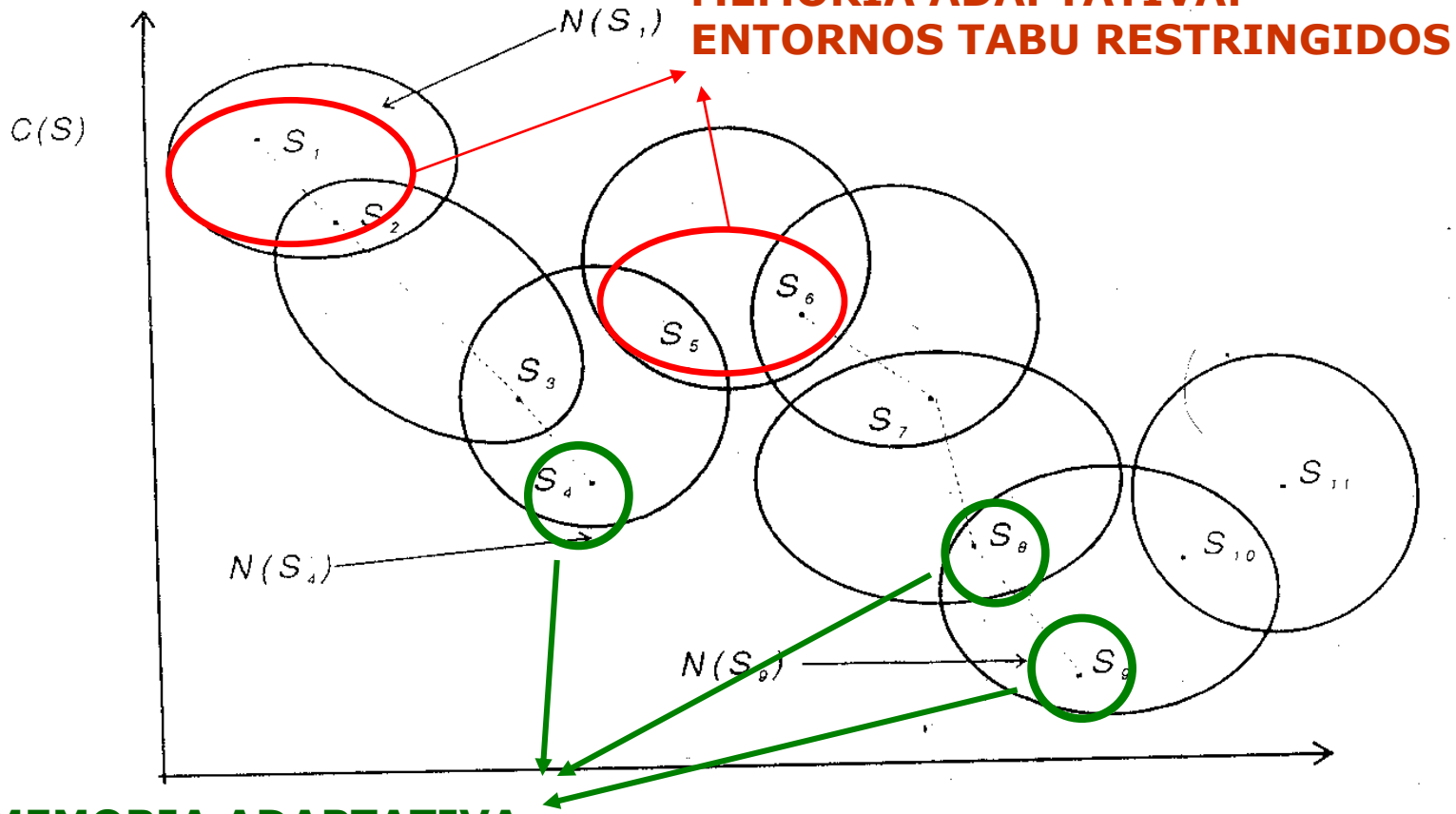
La memoria adaptativa permite:

- **restringir el entorno de búsqueda, e**
- **introducir mecanismos de reinicialización de la búsqueda mediante intensificación sobre zonas del espacio de búsqueda ya visitadas, o diversificación sobre posibles zonas del espacio de búsqueda poco visitadas**

Glover, F. "Tabu Search — Part I", *ORSA Journal on Computing* 1989 *1*: 3, 190-206.
Glover, F. "Tabu Search — Part II", *ORSA Journal on Computing* 1990 *2*: 1, 4-32.

1. INTRODUCCIÓN

MEMORIA ADAPTATIVA:
ENTORNOS TABU RESTRINGIDOS



MEMORIA ADAPTATIVA.
ESTRATEGIA ESPECIAL DE INTENSIFICACIÓN
SOBRE SOLUCIONES YA VISITADAS – Las mejores, distantes, …

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.1. Fundamentos

2.2. Memoria de Corto Plazo: Generación de Entornos Restringidos en la Búsqueda Tabú

2.3. Ejemplo de Uso de la Memoria de Corto Plazo

2.4. Memoria de Largo Plazo: Intensificación y Diversificación de la Búsqueda

2.5. Ejemplo de Uso de la Memoria de Largo Plazo

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.1. Fundamentos

La Búsqueda Tabú (TS) es una técnica de búsqueda por entornos caracterizada por dos aspectos principales:

- **Permite movimientos de empeoramiento para escapar de óptimos locales.**

  Para evitar recorridos cíclicos, incorpora un mecanismo de generación de vecinos modificado que evita la exploración de zonas del espacio de búsqueda que ya han sido visitadas: GENERACIÓN DE ENTORNOS TABÚ RESTRINGIDOS

- **Emplea mecanismos de reinicialización para mejorar la capacidad del algoritmo para la exploración-explotación del espacio de búsqueda: INTENSIFICACIÓN Y DIVERSIFICACIÓN**

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.1. Fundamentos

- **Para realizar las dos tareas anteriores, hace uso de dos estructuras de memoria adaptativas distintas:**
  - **Memoria de corto plazo o Lista tabú**
  - **Memoria de largo plazo**

La memoria de corto plazo guarda información que permite guiar la búsqueda de forma inmediata, desde el comienzo del procedimiento (GENERACIÓN DE ENTORNOS TABÚ RESTRINGIDOS)

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.1. Fundamentos

- **La memoria de largo plazo guarda información que permite guiar la búsqueda *a posteriori*, después de una primera etapa en la que se han realizado una o varias ejecuciones del algoritmo aplicando la memoria a corto plazo**

  La información guardada en esta memoria se usa para comenzar con la búsqueda desde otra solución inicial de acuerdo a dos filosofías distintas:

  - **Intensificar la búsqueda, volviendo a visitar zonas del espacio prometedoras (que contenían buenas soluciones), ya exploradas parcialmente**
  - **Diversificar la búsqueda, visitando nuevas zonas no exploradas aún**

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.2. Memoria de Corto Plazo: Generación de Entornos Restringidos en la Búsqueda Tabú

Modificación de las Estructuras de Entorno

- **La TS extiende la búsqueda local sustituyendo $E(S_{act})$ por otro entorno $E^*(S_{act}) \subset E(S_{act})$. En cada iteración, *se acepta siempre el mejor vecino de dicho entorno*, tanto si es peor como si es mejor que S_{act}**
- **La memoria de corto plazo (lista tabú) permite a la TS determinar $E^*(S_{act})$ y así organizar la manera en la que se explora el espacio**

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.2. Memoria de Corto Plazo: Generación de Entornos Restringidos en la Búsqueda Tabú

- **Las soluciones admitidas en $E^*(S_{act})$ dependen de la estructura de la lista tabú:**
  - ***Lista de soluciones tabú*: Se identifican soluciones ya visitadas y se marcan como tabú para no volver a ellas, eliminándolas del vecindario de S_{act}**
  - ***Lista de movimientos tabú*: Se eliminan del entorno todos los vecinos resultantes de aplicar sobre S_{act} un movimiento realizado anteriormente**
  - ***Lista de valores de atributos tabú*: Se eliminan del entorno todos aquellos vecinos con un par (atributo,valor) determinado que ya presentara alguna solución explorada anteriormente**

Los atributos o movimientos seleccionados son designados como "tabú-activos" y las soluciones que contienen elementos tabú-activos se convierten en "soluciones tabú"

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.2. Memoria de Corto Plazo: Generación de Entornos Restringidos en la Búsqueda Tabú

EJEMPLO:

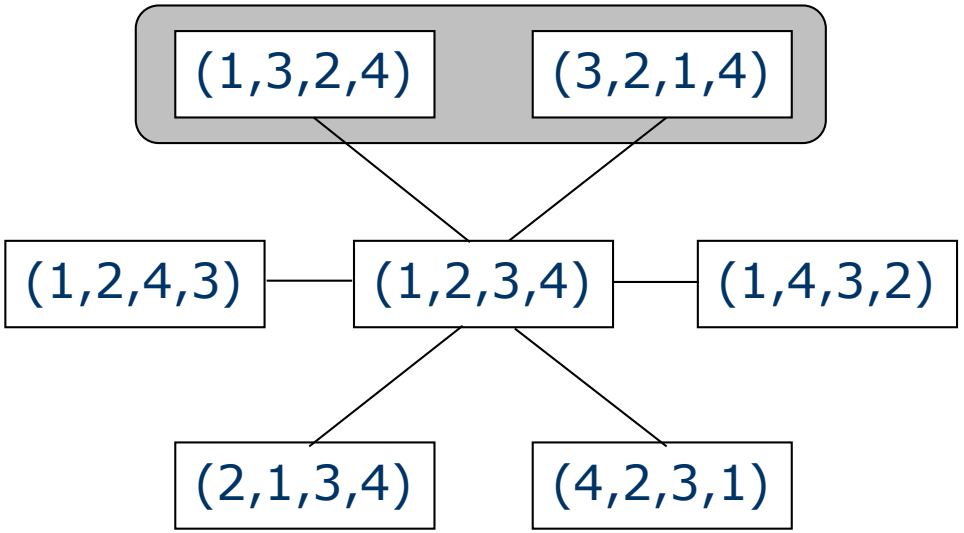
Lista tabú de soluciones:

LT = { (1,3,2,4), (3,1,2,4), (3,2,1,4) }

Operador de vecino: 2-opt S_{act} **= (1,2,3,4)**

Vecindario reducido de S_{act}**:**

$E^*(S_{act})$ = {(2,1,3,4), ~~(3,2,1,4)~~, (4,2,3,1), ~~(1,3,2,4)~~, (1,4,3,2), (1,2,4,3) }



2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.2. Memoria de Corto Plazo: Generación de Entornos Restringidos en la Búsqueda Tabú

EJEMPLO:

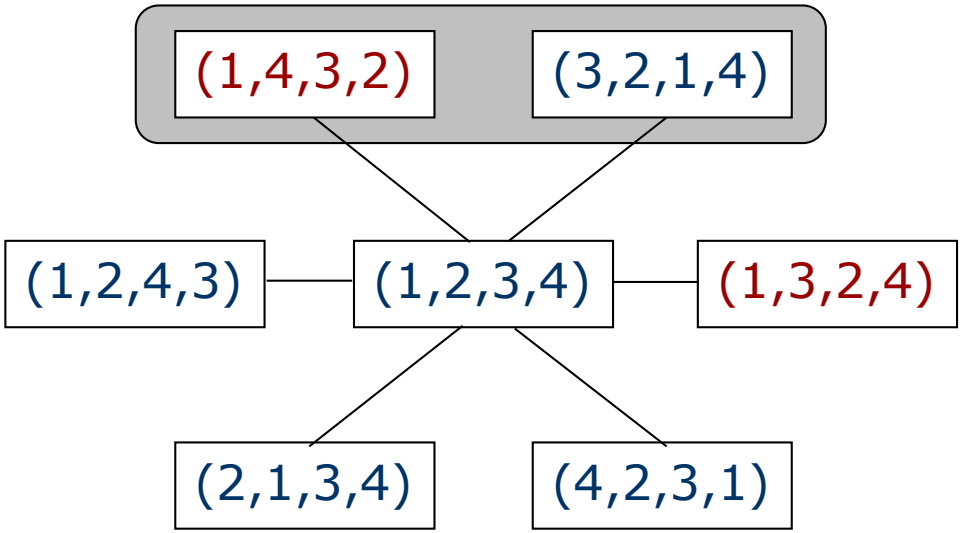
Lista tabú de movimientos:

LT = { (1,3), (2,4) }

Operador de vecino: 2-opt S_{act} **= (1,2,3,4)**

Vecindario reducido de S_{act}**:**

$E^*(S_{act})$ = {(2,1,3,4), ~~(3,2,1,4)~~, (4,2,3,1), (1,3,2,4), ~~(1,4,3,2)~~, (1,2,4,3) }



2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.2. Memoria de Corto Plazo: Generación de Entornos Restringidos en la Búsqueda Tabú

TENENCIA TABÚ

- **Un atributo/movimiento o solución que se haya incluido en la lista tabú en algún momento de la búsqueda no permanece en ella para siempre**

- **Se denomina "tenencia tabú" al intervalo de tiempo durante el que un atributo/movimiento permanece tabú-activo o una solución es tabú**

- **Este parámetro se mide en número de iteraciones. Una vez transcurrido el valor especificado, el elemento en cuestión deja de ser tabú activo y se elimina de la lista**

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.2. Memoria de Corto Plazo: Generación de Entornos Restringidos en la Búsqueda Tabú

NIVELES DE ASPIRACIÓN

- **El "criterio de aspiración" introduce un elemento importante de flexibilidad en la búsqueda tabú**
- **El estado tabú de un movimiento o atributo puede ser ignorado si se cumplen ciertas condiciones, en la forma de niveles de aspiración**
- **Por ejemplo, un vecino que sea mejor que cualquiera de las soluciones encontradas anteriormente merece ser considerado admisible, incluso aunque dicho vecino sea una solución tabú**
- **Una solución tabú dejará de serlo y se incluirá en el entorno factible si supera un cierto nivel de aspiración**

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.2. Memoria de Corto Plazo: Generación de Entornos Restringidos en la Búsqueda Tabú

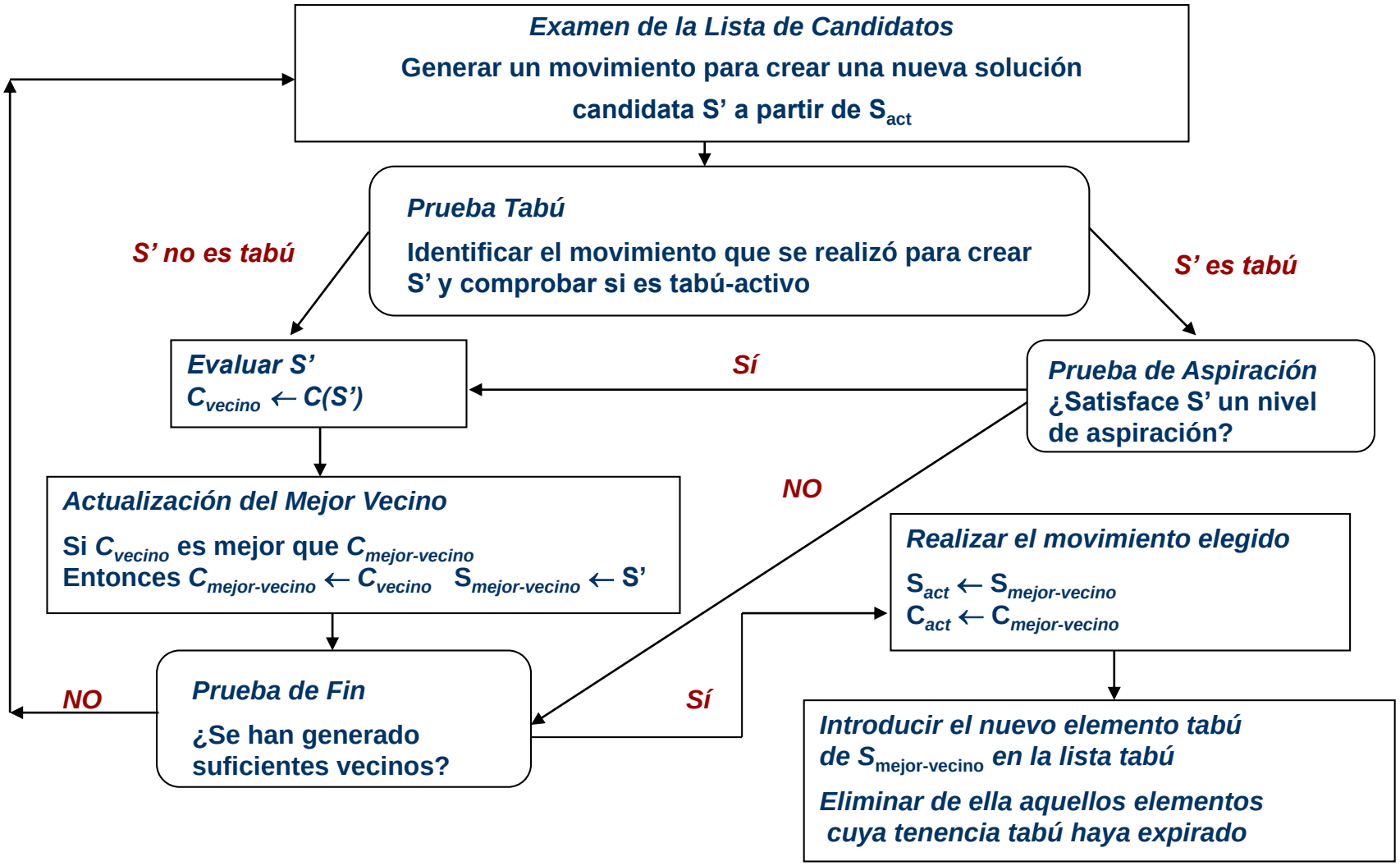
ESTRATEGIAS PARA LA LISTA DE CANDIDATOS

- **Las estrategias para la lista de candidatos se usan para restringir el número de vecinos examinados en una iteración dada, para los casos en los que $E^*(S_{act})$ es grande o la evaluación de sus elementos es costosa**

- **Se busca el mejor movimiento disponible que pueda ser determinado con una cantidad apropiada de esfuerzo**

- **Así, no se genera el entorno reducido completo sino una parte del mismo y se toma el mejor vecino. La selección adecuada de candidatos puede reducir apreciablemente los tiempos de ejecución**

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ



2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.2. Memoria de Corto Plazo: Generación de Entornos Restringidos en la Búsqueda Tabú

ESTRUCTURAS DE MEMORIA DE CORTO PLAZO

- **Sea $I = \{1, 2, ..., n\}$ un conjunto de índices para una colección de atributos y sea S una solución al problema**

- **Sea $i \in I$, el estado tabú-activo de un atributo S_i se representa por el índice i, que puede almacenarse en un vector o lista**

- **El estado tabú de $S_i = k$ se representa guardando el par ordenado $<i, k>$ en la lista**

- **Para el caso de un intercambio de dos índices $i,j \in I$, se guardaría el par $<i, j>$**

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.2. Memoria de Corto Plazo: Generación de Entornos Restringidos en la Búsqueda Tabú

- **Para facilitar la gestión de la tenencia tabú de los atributos (el número de iteraciones que son tabú-activos), lo mejor es implementar una lista circular**

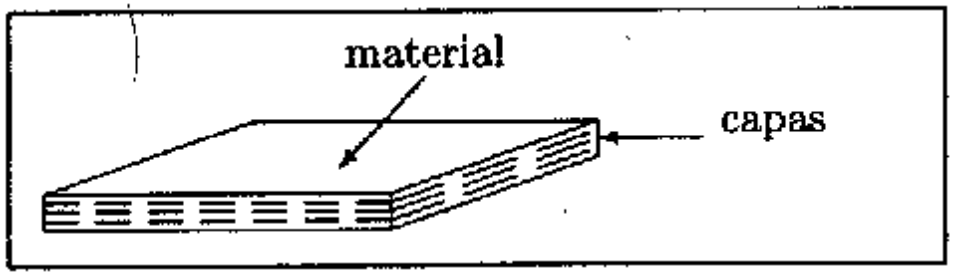
- **Cuando se llena la lista, se comienza a insertar de nuevo por el principio, borrando lo que hubiera en esas posiciones**

- **Así, la tenencia tabú de todos los atributos equivale al tamaño de la lista. La elección de un valor para la tenencia tabú suele ser experimental, siendo una función del número total de atributos (p.e., [n/3, 3·n])**

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

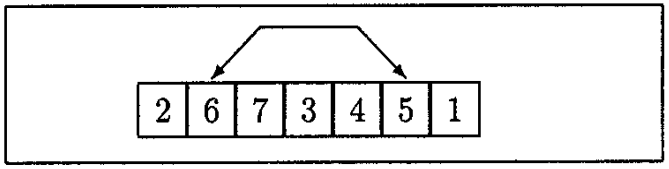
2.3. Ejemplo de Uso de la Memoria de Corto Plazo: Diseño de un Material Formado por un Número de Capas Aislantes

- **Problema de permutaciones que consiste en encontrar el orden de las capas que maximiza el valor de aislamiento total del material compuesto. Supongamos que se consideran 7 capas para un material particular y que evaluar el valor de aislamiento total de una ordenación particular es un procedimiento computacionalmente costoso. Maximizamos una función de aislamiento**

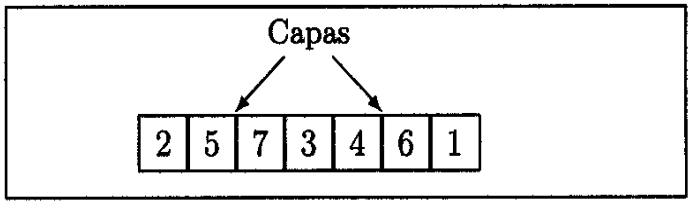


2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

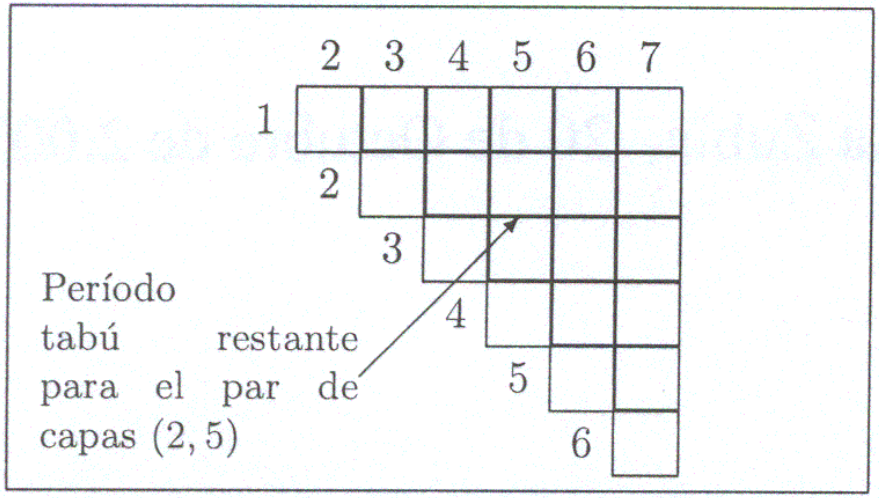
2.3. Ejemplo de Uso de la Memoria de Corto Plazo



Movimiento - Permutación



Intercambio de las capas 5 y 6

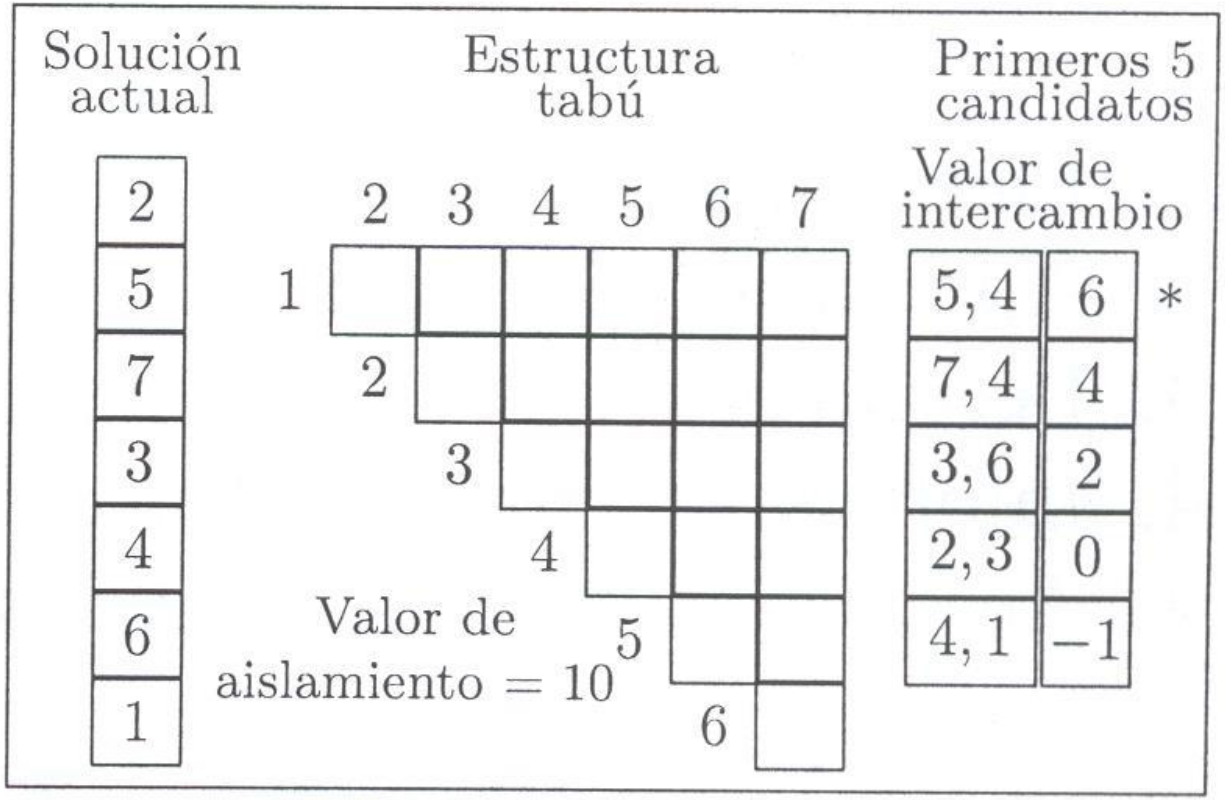


Estructura de datos de la Lista Tabú

Se impide el intercambio de las capas i y j mientras el movimiento sea tabú-activo

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

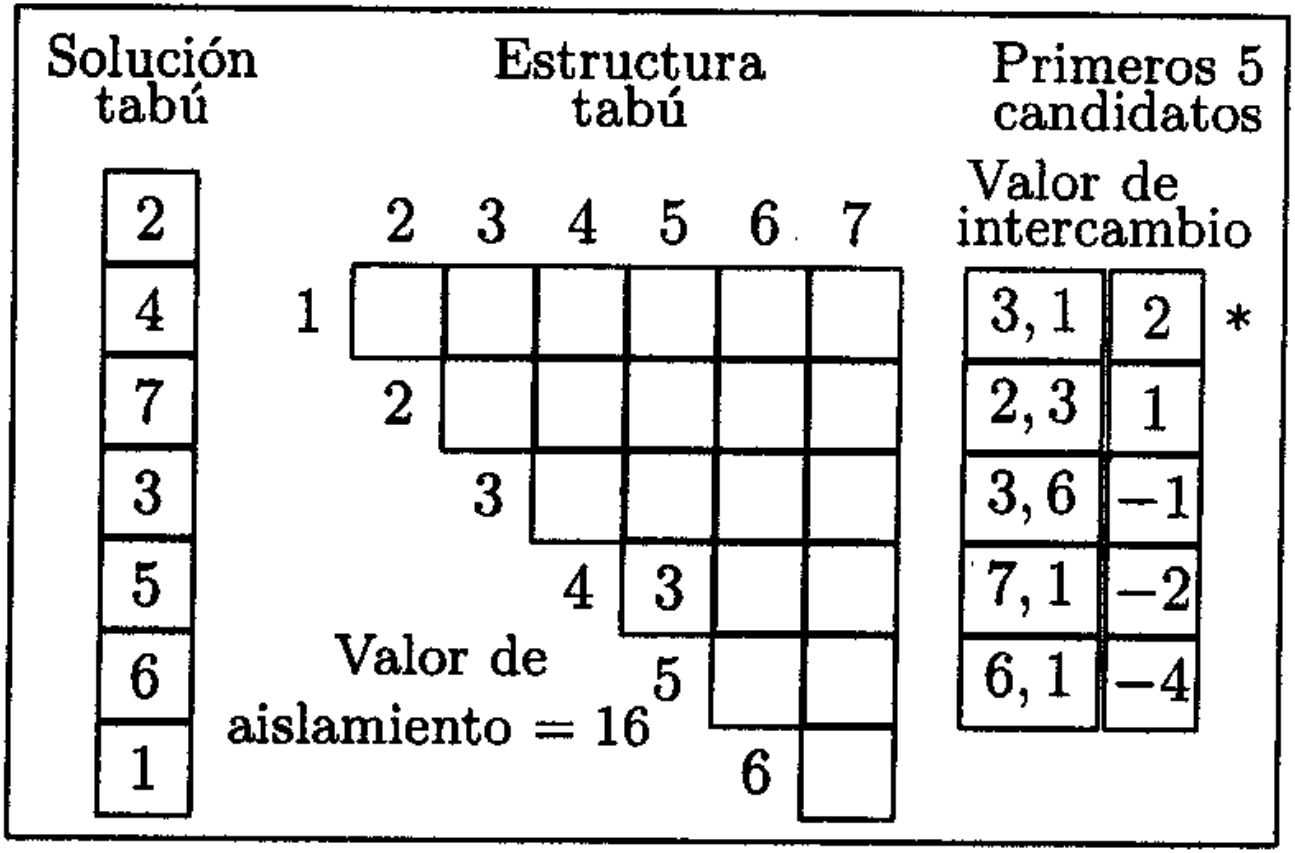
2.3. Ejemplo de Uso de la Memoria de Corto Plazo



Iteración 0

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

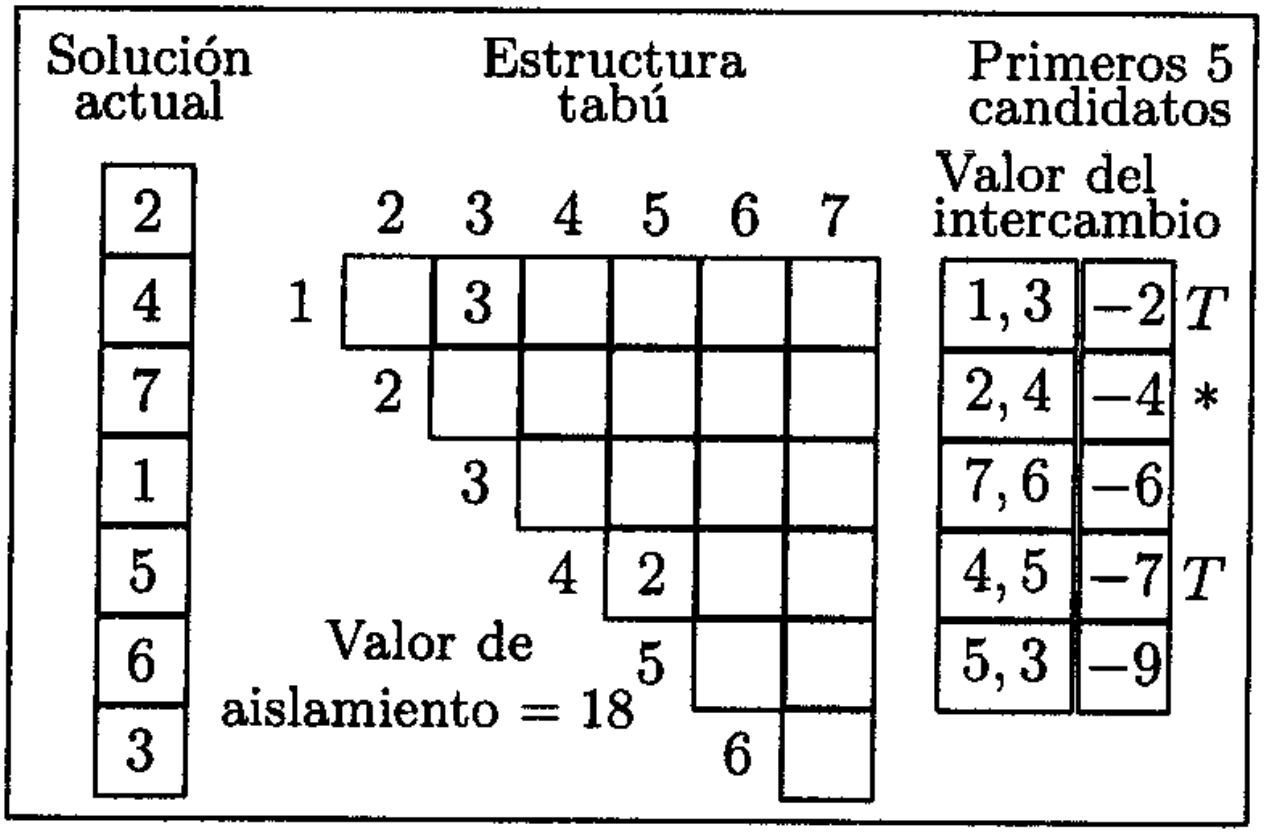
2.3. Ejemplo de Uso de la Memoria de Corto Plazo



Iteración 1 (Tenencia Tabú = 3)

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

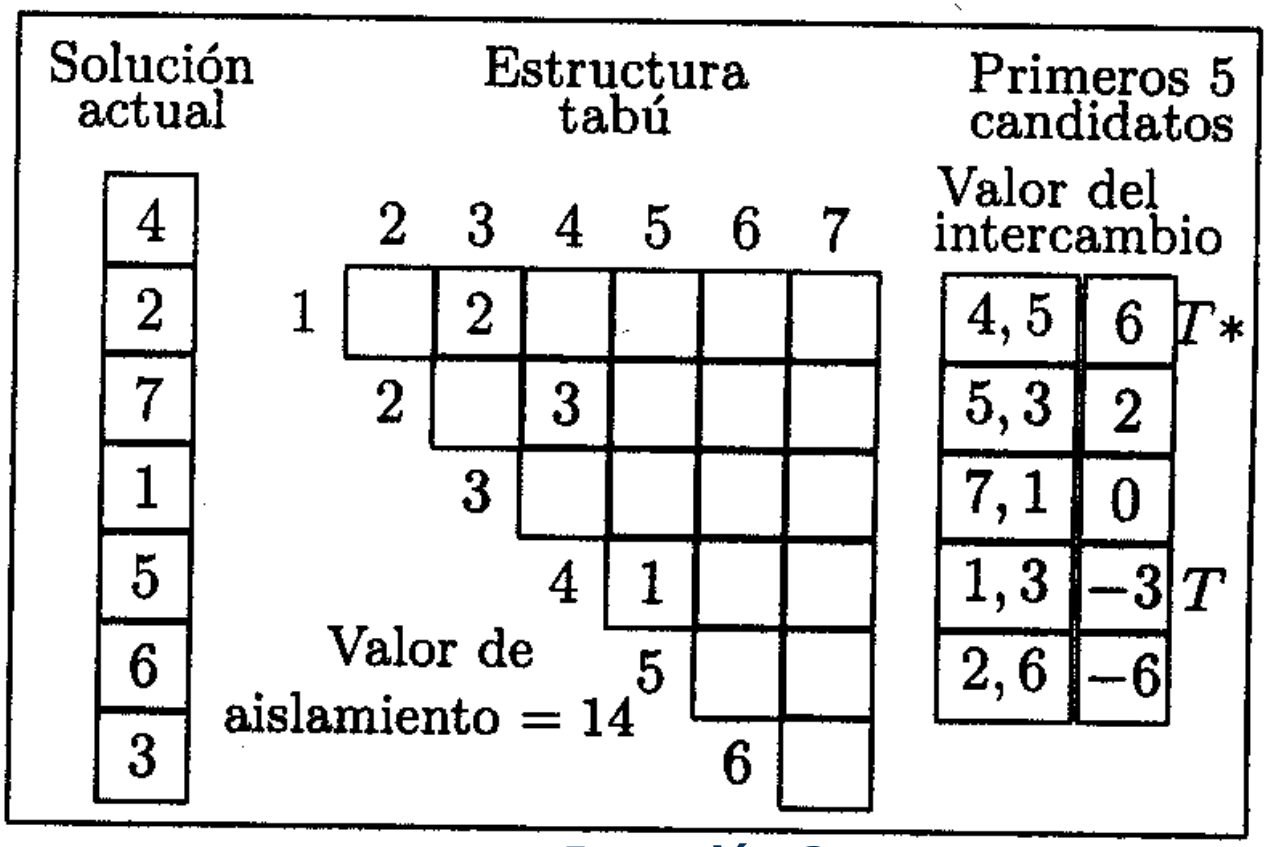
2.3. Ejemplo de Uso de la Memoria de Corto Plazo



Iteración 2

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

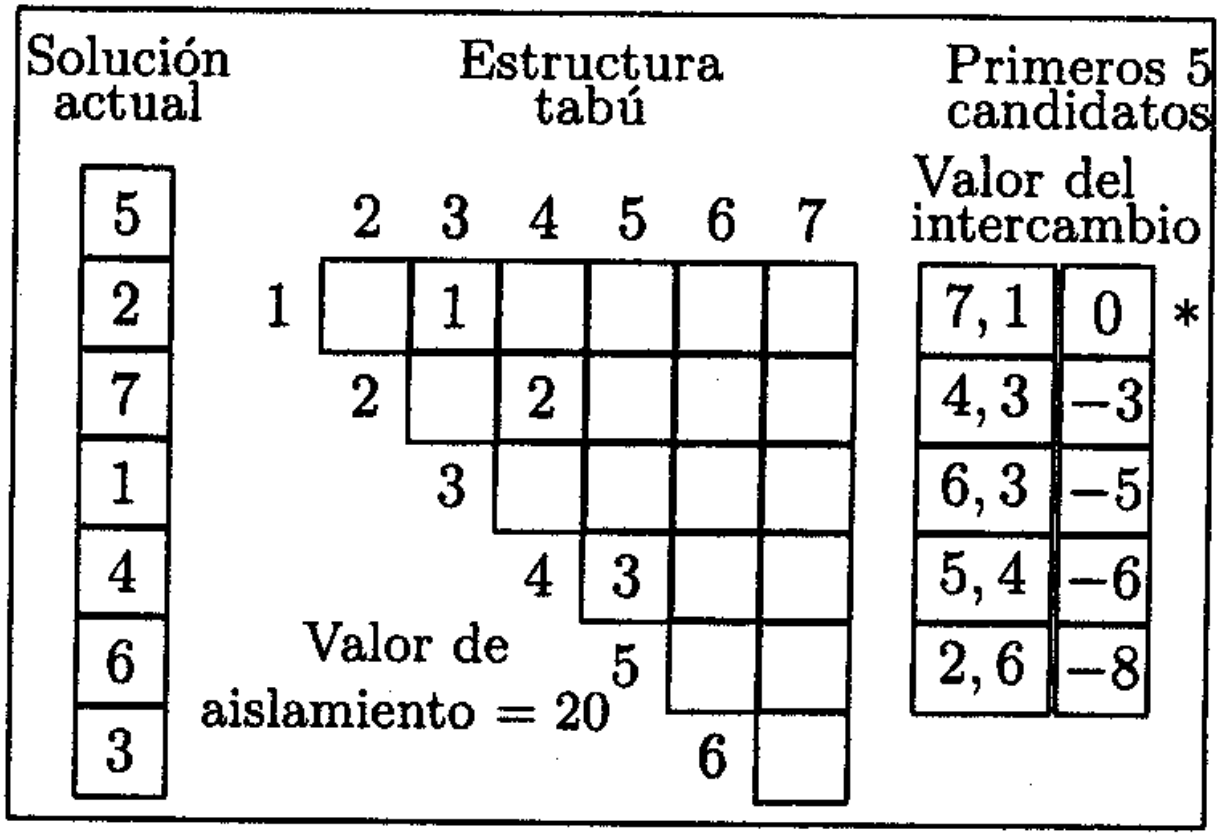
2.3. Ejemplo de Uso de la Memoria de Corto Plazo



Iteración 3

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.3. Ejemplo de Uso de la Memoria de Corto Plazo



Iteración 4

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

2.4. Memoria de Largo Plazo: Intensificación y Diversificación de la Búsqueda

- **En algunas aplicaciones, las componentes de la memoria TS de corto plazo son suficientes para producir soluciones de muy alta calidad**
- **No obstante, en general, la TS se vuelve mucho más potente incluyendo memoria de largo plazo y sus estrategias de reinicialización asociadas**
- **La memoria de largo plazo se puede usar de dos modos distintos:**
  - **Estrategias de Intensificación**
  - **Estrategias de Diversificación**
- **Una estructura muy empleada es la memoria de frecuencias, que registra el número de veces que cada valor de un atributo ha pertenecido a soluciones visitadas en la búsqueda**

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

ESTRATEGIAS DE INTENSIFICACIÓN

- **Se basan en una reinicialización de la búsqueda que efectúa un regreso a regiones atractivas del espacio para buscar en ellas más extensamente**
- **Se mantiene un registro de las mejores soluciones visitadas, insertando una nueva solución cada vez que se convierte en la mejor global**
- **Se puede introducir una medida de diversificación para asegurar que las soluciones registradas difieran una de otra en un grado deseado**

Tenemos cuatro variantes resultantes de combinar "soluciones/memoria":

- **Solución desde la que se reinicializa:**
  - **Reanudar el proceso *desde la mejor* de las soluciones registradas**
  - **Usar una pila de mejores soluciones de longitud limitada y escoger *la cabeza de la pila***
- **Restauración de la memoria de corto plazo:**
  - ***Almacenar la memoria de corto plazo en el momento que se encontró cada una de las mejores soluciones para restaurarla cuando se reanude la búsqueda desde dicha solución***
  - ***Borrar la memoria de corto plazo* para iniciar la búsqueda desde cero**

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

Enfoque simple de intensificación en TS

Realizar una serie de iteraciones haciendo uso de la memoria de corto plazo

↓

Crear e ir manteniendo una lista de k soluciones élite (por ejemplo, k=5, ..., 20)

↓

Cuando el algoritmo se estanque (es decir, cuando la proporción de aceptación de nuevas soluciones mejores caiga por debajo de un umbral):

Si se llegó al límite de iteraciones o la lista de soluciones élite está vacía, Parar

si no, escoger una solución élite y eliminarla de la lista

↓

Continuar la ejecución del algoritmo de TS desde la solución escogida (opcionalmente restaurar la memoria de corto plazo)

Si durante las iteraciones actuales se encuentran soluciones válidas para la estrategia de selección de soluciones élite, incluirlas en la lista (reemplazando a otras si la lista está llena)

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

ESTRATEGIAS DE DIVERSIFICACIÓN

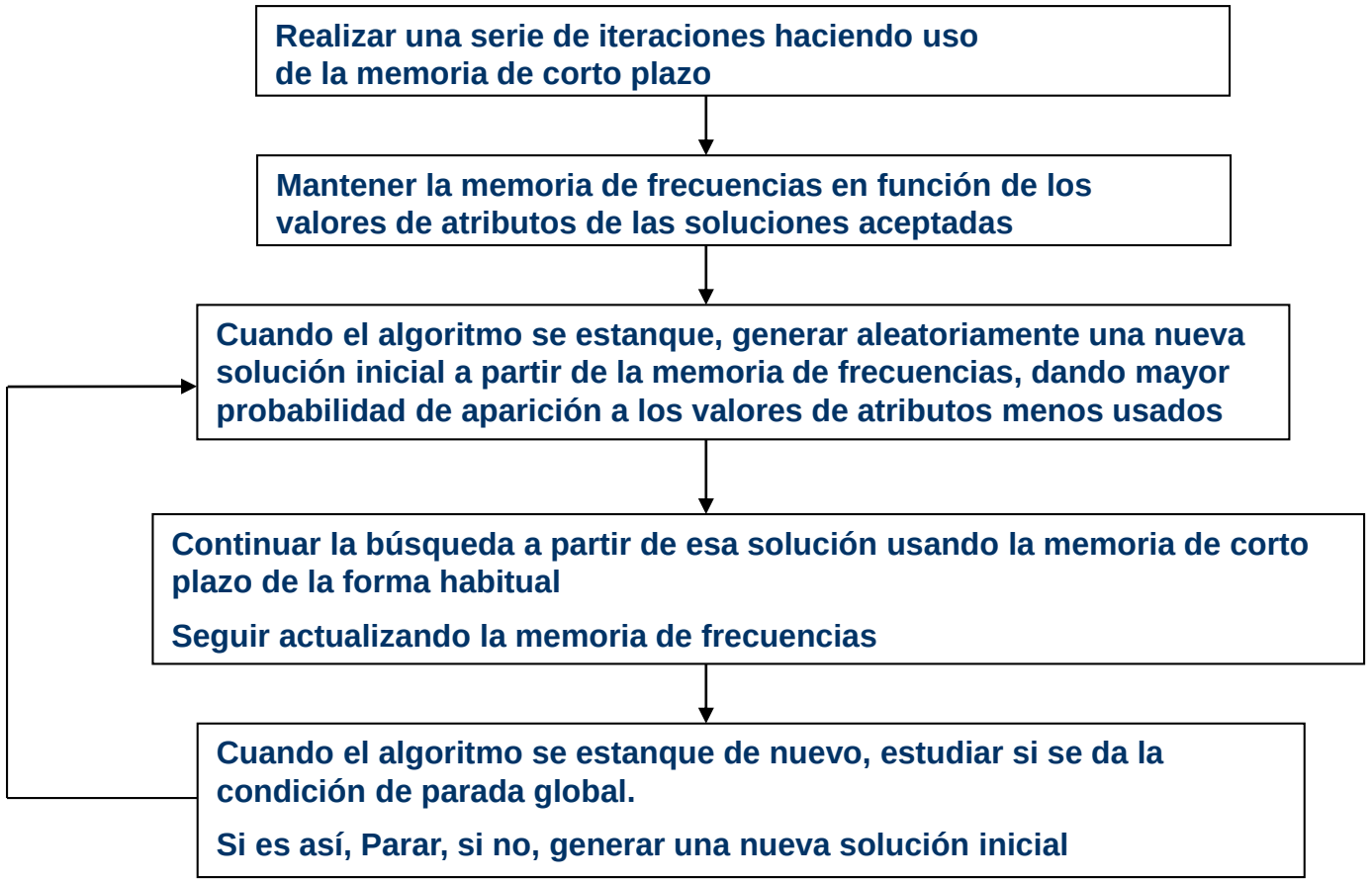
- Conducen la búsqueda hacia nuevas regiones del espacio de búsqueda no exploradas aún
- La búsqueda se reinicializa cuando se estanca, partiendo de una solución no visitada
- Esta solución se genera a partir de la memoria de frecuencias, dando mayor probabilidad de aparición a los valores menos habituales

OSCILACIÓN ESTRATÉGICA

- Interacción efectiva entre intensificación y diversificación consistente en escoger aleatoriamente una u otra cuando la búsqueda se estanque

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

Enfoque simple de diversificación en TS



2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

ESTRUCTURAS DE MEMORIA DE LARGO PLAZO (FRECUENCIAS)

- **Si las variables son binarias, se puede usar un vector de dimensión n, para almacenar el número de veces que cada variable tomó el valor 0 (ó 1)**

- **Si son enteras, se utiliza una matriz bidimensional, como contador de las veces que la variable i toma el valor k: M[i,k]**

- **Si son permutaciones de orden, se puede utilizar una matriz bidimensional, como contador de las veces que el valor i ha ido seguido del j: M[i,j]**

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ

USO DE LA MEMORIA DE FRECUENCIAS

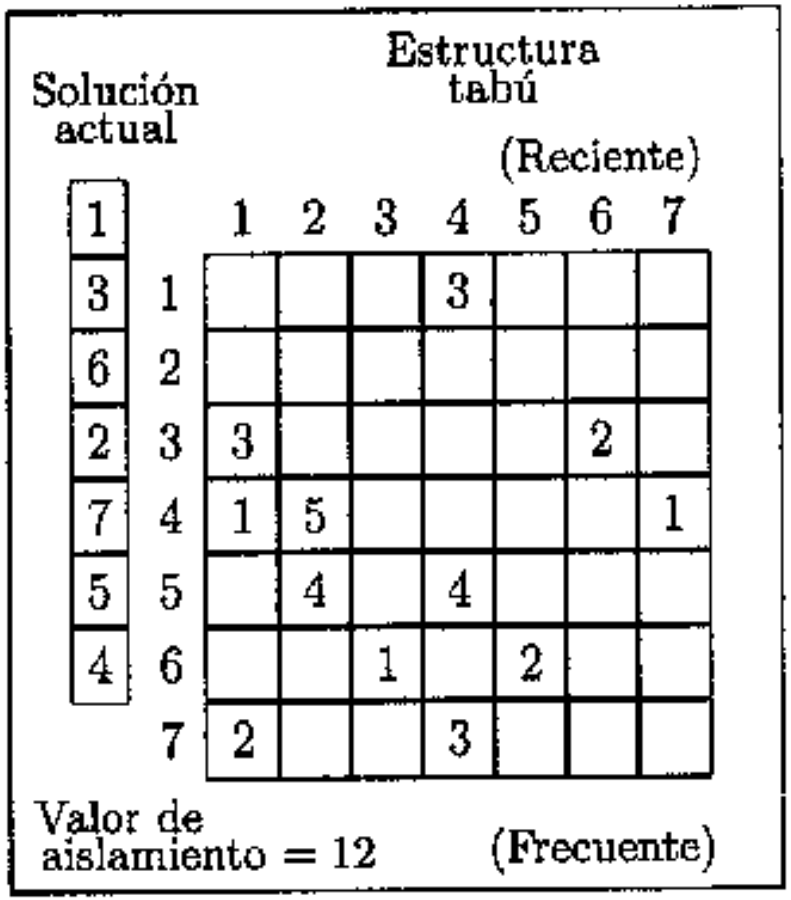
- **Existen dos posibilidades:**

1) **Generar directamente la nueva solución inicial a partir de la información almacenada en la memoria de frecuencias M, dando mayor probabilidad de aparición a los valores menos habituales**

2) **Usar la información almacenada en M para modificar temporalmente el caso del problema, potenciando los valores heurísticos de los atributos menos usados en la búsqueda**

   Aplicar un algoritmo greedy sobre ese caso modificado para generar la solución inicial. Restaurar el caso original del problema antes de continuar con la búsqueda

2. LA ESTRUCTURA DE LA BÚSQUEDA TABÚ



Estructura de Memoria Complementaria: Memoria a Largo Plazo

Problema: Diseño de un Material Formado por un Número de Capas Aislantes

Ejemplo de estructura de memoria con diagonal superior como lista tabú, y diagonal inferior midiendo la frecuencia de aparición de las capas del material en las posiciones correspondientes, para poder diversificar la búsqueda en futuras reinicializaciones

Al reinicializar, se generaría una nueva solución en la que en cada posición se le daría más probabilidad a la capa que menos frecuentemente la hubiera ocupado en el pasado

3. EJEMPLO: VIAJANTE DE COMERCIO

1. **Generación de la solución inicial**: aleatoria
2. **Esquema de representación**: Representación de orden mediante permutación {1,…,n}
3. **Operador de generación de vecinos**: seleccionar dos ciudades e intercambiarlas (2-opt)
4. **Función objetivo (minimización)**:

$$C(S) = \sum_{i=1}^{n-1} \left(D\left[S\left[i\right], S\left[i+1\right]\right]\right) + D\left[S\left[n\right], S\left[1\right]\right]$$

Los vecinos se evalúan con el cálculo optimizado de la función objetivo:

C(S') = C(S) – D(S[i-1],S[i]) – D(S[i],S[i+1]) - D(S[j-1],S[j]) –
D(S[j],S[j+1]) + D(S[i-1],S[j]) + D(S[j],S[i+1]) +
D(S[j-1],S[i]) + D(S[i],S[j+1])

3. EJEMPLO: VIAJANTE DE COMERCIO

5. **Lista tabú (L)**: Cada vez que se acepta una solución, se almacena en la lista tabú el movimiento que provocó dicha solución

Para ello, se almacenan las dos ciudades intercambiadas y las nuevas posiciones que ocupan (i,j,Pos(i),Pos(j)). No se permiten intercambios que den lugar a que la ciudad i ocupe la posición Pos(i), o la ciudad j la posición Pos(j)

Ejemplo:

1	2	3	4	5	6	7	8
(1	**2**	**4**	**3**	**8**	**5**	**7**	**6)**
(1	**2**	**5**	**3**	**8**	**4**	**7**	**6)**

L = L – {movimento más antiguo} + {(4,5,6,3)}

Para decidir si un movimiento es tabú activo, hay que tener en cuenta que (i,j,Pos(i),Pos(j)) = (j,i,Pos(j),Pos(i))

3. EJEMPLO: VIAJANTE DE COMERCIO

6. **Estrategia de selección de vecinos**: Examinar 50 movimientos (todos distintos) y escoger el mejor de acuerdo a los criterios tabú

7. **Criterio de aspiración**: Tener menor coste que la mejor solución obtenida hasta el momento

8. **Estrategias de reinicialización**:
   a) Generar una solución aleatoria y continuar el algoritmo a partir de ella (diversificación)
   b) Volver a la mejor solución obtenida hasta el momento y continuar el algoritmo a partir de ella (intensificación)
   c) Generar una nueva solución *greedy* a partir de la memoria a largo plazo y continuar el algoritmo a partir de ella (diversificación controlada)

3. EJEMPLO: VIAJANTE DE COMERCIO

- Memoria a largo plazo: Se usa una matriz simétrica *frec* que almacena el número de veces que cada par de ciudades han estado consecutivas en alguna de las soluciones aceptadas durante la búsqueda

  Para reinicializar provocando diversidad, se aplica el algoritmo *greedy* sobre una matriz de distancias modificada de forma que valores altos de frecuencia entre un par de ciudades supongan un valor alto de distancia entre ellas:

$$d(i,j) = d(i,j) + \mu \cdot (d_{\max} - d_{\min}) \cdot \frac{frec(i,j)}{frec_{\max}}$$

  siendo d_{max} y d_{min} la mayor y menor distancia existente entre dos ciudades, y $frec_{max}$ la frecuencia más alta existente en la matriz de memoria a largo plazo. El parámetro μ dicta el grado de alteración de la distancia (p.e., μ=0,3)

  La alteración de distancias sólo se usa para generar la solución diversa con el *greedy* y no afecta a la función objetivo en la búsqueda posterior

3. EJEMPLO: VIAJANTE DE COMERCIO

- Hay distintas posibilidades para la lista tabú del algoritmo de BT para el Viajante de Comercio. La elección de una de ellas da lugar a un algoritmo distinto:

  1. Vector (i,j,Pos(i),Pos(j)): Vector que evita cualquier intercambio que dé lugar a una solución en la que las ciudades i y j ocupen las posiciones Pos(i) y Pos(j) respectivamente

  2. Vector (i,j,Pos(i),Pos(j)): Mismo vector que evita un intercambio que dé lugar a una solución en la que la ciudad i ocupe la posición Pos(i) **o** la j ocupe la posición Pos(j)

  3. Vector (i,Pos(i)): Para evitar que la ciudad i vuelva a la posición Pos(i)

  4. Ciudad i: Para evitar que la ciudad i se mueva a la izquierda de su posición actual

3. EJEMPLO: VIAJANTE DE COMERCIO

5. Ciudad i: Para evitar que la ciudad i se mueva (cambie su posición)
6. Vector (j,Pos(j)): Para evitar que la ciudad j vuelva a la posición Pos(j)
7. Ciudad j: Para evitar que la ciudad j se mueva a la derecha de su posición actual
8. Ciudad j: Para evitar que la ciudad j se mueva (cambie su posición)
9. Ciudades i y j: Para evitar que ambas ciudades cambien su posición actual

- En las condiciones 3 a 9 se supone que las ciudades i y j están situadas de tal forma que Pos(i)<Pos(j). La condición 1 es la menos restrictiva y la condición 9 la más restrictiva. El nivel restrictivo de las condiciones 3, 4 y 5 es creciente

METAHEURÍSTICAS

5.3. Métodos Basados en Trayectorias Múltiples

1. Introducción a la Búsqueda Multiarranque
2. Algoritmos Multiarranque Básicos
3. Modelos Multiarranque
4. Algoritmo GRASP
5. Algoritmos de Búsqueda Local Reiterativos Basados en Óptimos: ILS
6. Búsqueda de Entorno Variable: VNS
7. Aplicaciones

METAHEURÍSTICAS

5.3. Métodos Basados en Trayectorias Múltiples

BIBLIOGRAFÍA

- *F. Glover, G.A. Kochenberber. Handbook of Metaheuristics. Kluwer Acad., 2003. Cap. 12. Multi-start Methods, Rafael Martí, 355-368.* Cap. 8. Greedy Randomized Adaptive Search Procedure. (M.G.C. Resende, C.S. Ribeiro), 331-240. Cap. 6. Variable Neighborhood Search, P.Hansen, N. Mladenovic, 145-184. Cap. 11. Iterated Local Search, H.R. Lourenço, O.C. Martin, T. Stützle, 321-353.
- *R. Martí, J. Marcos Moreno. Métodos Multiarranque. Inteligencia Artificial 19 (2003) 49-60.*
- P. Hansen, N. Mladenovic, J.A. Moreno. Búsqueda de Entorno Variable. Inteligencia Artificial 19 (2003) 77-92.
- *M.G.C. Resende, J.L. González, GRASP: Procedimientos de Búsqueda Miopes, aleatorizados y adaptativos. Inteligencia Artificial 19 (2003) 61-76*
- *T.A. Feo, M.G.C. Resende. Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization 6 (1995) 109-133*
- T. Stützle, 1998. Local Search Algorithms for Combinatorial Problems- Analysis, Improvements and New Applications. PhD Thesis, Darmstadt, University of Technology, Department of Computer Science.
- N. Mladenovic, P. Hansen. Variable Neighborhood Search. Computers and Operations Research 24(11) (1997) 1097-1100.

1. INTRODUCCIÓN A LA BÚSQUEDA MULTIARRANQUE

Problemas de la Búsqueda Local

Suele caer en óptimos locales, que a veces están bastante alejados del óptimo global del problema

SOLUCIONES: 3 opciones para salir de los óptimos locales

- **Permitir movimientos de empeoramiento de la solución actual**
  (Ejemplo: Enfriamiento Simulado, Búsqueda Tabú, …)
- **Modificar la estructura de entornos**
  (Ejemplo: Búsqueda Tabú, Búsqueda en Entornos Variables: VNS, …)
- **Volver a comenzar la búsqueda desde otra solución inicial (Ejemplo: Búsquedas Multiarranque, GRASP, ILS,VNS, …)**

1. INTRODUCCIÓN A LA BÚSQUEDA MULTIARRANQUE

- **Una Búsqueda con Arranque Múltiple es un algoritmo de búsqueda global que itera las dos etapas siguientes:**
  - **Generación de una solución inicial: Se genera una solución S de la región factible**
  - **Búsqueda Local: Se aplica una BL desde S para obtener una solución optimizada S'**
- **Estos pasos se repiten hasta que se satisfaga algún criterio de parada**
- **Se devuelve como salida del algoritmo la solución S' que mejor valor de la función objetivo presente**
- **La Búsqueda Multiarranque Básica se caracteriza porque las soluciones iniciales se generan de forma aleatoria**

1. INTRODUCCIÓN A LA BÚSQUEDA MULTIARRANQUE

Procedimiento Búsqueda con Arranque Múltiple

COMIENZO

S_{act} ← Genera Solución () (ETAPA 1)

***Mejor_Solución* ← S_{act}**

REPETIR

S' ← Búsqueda Local (S_{act}) (ETAPA 2)

Sl S' es mejor que *Mejor_Solución* ENTONCES

***Mejor_Solución* ← S'**

S_{act} ← Genera Solución () (ETAPA 1)

HASTA (criterio de parada)

Devolver *Mejor_Solución*

FIN

1. INTRODUCCIÓN A LA BÚSQUEDA MULTIARRANQUE

- **En algunas aplicaciones, la Etapa 1 se limita a la simple generación aleatoria de las soluciones, mientras que en otros modelos se emplean sofisticados métodos de construcción que consideran las características del problema de optimización para obtener soluciones iniciales de calidad**

- **En cuanto a la Etapa 2, se puede emplear una búsqueda local básica, o procedimientos de búsqueda basados en trayectorias más sofisticados**

- **En cuanto a la condición de parada, se han propuesto desde criterios simples, como el de parar después de un número dado de iteraciones, hasta criterios que analizan la evolución de la búsqueda. En muchas de las propuestas que se encuentran en la literatura especializada se fija un número de iteraciones de la búsqueda local**

2. ALGORITMOS MULTIARRANQUE BÁSICOS

Búsqueda Multiarranque Básica

- **El algoritmo multiarranque más básico que podemos considerar es aquel en el que las soluciones iniciales se generan al azar en la región factible del problema, y la etapa de búsqueda se realiza mediante algún procedimiento de búsqueda local BL**
- **Este método converge al óptimo global del problema con probabilidad 1 cuando el número de puntos generados tiende a infinito**
- **El procedimiento es muy ineficiente puesto que se pueden generar muchos puntos cercanos entre sí, de modo que al aplicarles el procedimiento de búsqueda BL se obtenga repetidamente el mismo óptimo local**

3. MODELOS MULTIARRANQUE

- **Existen múltiples propuestas de metaheurísticas que se pueden considerar como técnicas multiarranque:**
  - **Métodos constructivos de la solución inicial**
    - **Construcción *greedy*: Algoritmos GRASP**
    - **Algoritmos Basados en Colonias de Hormigas: ACO (Tema 6)**
  - **Métodos iterativos mediante modificación de la solución encontrada**
    - **ILS: Búsqueda Local Reiterativa**
    - **VNS: Búsqueda de Entorno Variable**
  - **Hibridaciones entre técnicas poblacionales de exploración/ combinación de soluciones y métodos de búsqueda local (Tema 4)**
    - **Algoritmos Meméticos / Algoritmos Genéticos con BL**

4. ALGORITMO GRASP
4.1. Introducción

- **Un algoritmo GRASP es un método multiarranque, en el que cada iteración consiste en la construcción de una solución *greedy* aleatorizada y la aplicación de una búsqueda local que toma dicha solución como punto inicial de la búsqueda**
- **Este procedimiento se repite varias veces y la mejor solución encontrada sobre todas las iteraciones GRASP se devuelve como salida del algoritmo**

PROCEDIMIENTO ITERATIVO:

GREEDY-ALEATORIZADO-ADAPTATIVO

+

BUSQUEDA LOCAL

- ***T.A. Feo, M.G.C. Resende. Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization 6 (1995) 109-133***

4. ALGORITMO GRASP
4.1. Introducción

Procedimiento GRASP

Procedimiento GRASP

Repetir Mientras (no se satisfaga el criterio de parada)

S ← Construcción Solución Greedy Aleatorizada ()

S' ← Búsqueda Local (S)

Actualizar (S', *Mejor_Solución*)

Devolver (*Mejor_Solución*)

FIN-GRASP

4. ALGORITMO GRASP
4.2. Descripción

Construcción de la Solución Inicial en GRASP

- **Cuando se utiliza la función de selección para construir una solución *greedy*, se crea una lista de restringida de candidatos con un número determinado de mejores candidatos**
- **Se realiza una selección aleatoria de un candidato de la lista**
- **Se adapta la función de selección para recalcular la nueva lista de candidatos para el siguiente paso del proceso constructivo**

4. ALGORITMO GRASP
4.2. Descripción

Construcción de la Solución Inicial en GRASP

Procedimiento Construcción-Greedy-Aleatorizada ()

S = {}

Repetir Mientras (no se haya construido la solución)

Crear Lista Restringida de Candidatos (LRC)

s ← selección-aleatoria-elemento (LRC)

S ← S ∪ {s}

Adaptar la función de selección **%Actualizar el conjunto de candidatos**

Devolver S

Fin-Procedimiento

4. ALGORITMO GRASP
4.2. Descripción

Construcción de la Solución Inicial en GRASP

- **La variante más habitual consiste en incluir en la LRC los / mejores candidatos de acuerdo a la función de selección. El parámetro / controla la diversidad de generación de soluciones. Puede ser fijo o variable**
- **LRC adaptativa (tamaño variable): Existen variantes que incluyen en la LRC todos los candidatos con un valor de selección por encima de un umbral de calidad μ = cmejor $\pm\alpha\cdot$ (cmejor- cpeor) (cmejor=coste mejor candidato; cpeor=coste peor candidato). Esto provoca que la LRC sea adaptativa y tenga un tamaño variable en cada iteración de la etapa de generación**
- **Otras variantes incluyen un sesgo en la LRC con una probabilidad mayor de selección de los mejores candidatos**
- **También se combina la construcción al azar con la construcción *greedy*, alternando ambos procedimientos aleatoriamente y en secuencias de candidatos**

4. ALGORITMO GRASP
4.2. Descripción

Búsqueda Local en GRASP

- **La búsqueda local desempeña un papel importante en GRASP ya que sirve para buscar soluciones localmente óptimas en regiones prometedoras del espacio de soluciones**

- **En el contexto de GRASP, se han utilizado esquemas de búsqueda local clásicos, así como otros más sofisticados (enfriamiento simulado, búsqueda tabú, etc.)**

- **Aunque los algoritmos *greedy* pueden producir soluciones iniciales razonables para búsquedas locales, su principal desventaja es su falta de diversidad, de ahí las propuestas de LRC para generar más diversidad en la primera etapa del GRASP y mejorar las posibilidades de la búsqueda local.**

4. ALGORITMO GRASP
4.2. Descripción

Ejemplo de Diversidad en las Soluciones Iniciales

Problema: Satisfacción de cláusulas (**1391 variables, 3126 cláusulas, 7025 literales**)

Distribución de las soluciones en 100000 ejecuciones de la primera etapa del GRASP

size RCL	solution values										
	3116	3117	3118	3119	3120	3121	3122	3123	3124	3125	3126
1									100000		
2							151	6053	93796		
4						75	1676	17744	80503	2	
8				1	50	750	6566	31257	61336	35	5
16				16	282	2485	13274	38329	45547	42	25
32		1	3	72	635	4196	16455	37937	40479	164	58
64		4	18	177	1213	5933	19553	37666	34832	441	163
128		4	36	269	1716	7324	21140	37186	34832	679	281
256	1	5	35	304	1980	7867	21792	36725	29027	1575	689

Greedy

Fig. 4. Sample distributions of GRASP iteration solutions.

size RCL	1	2	4	8	16	32	64	128	256
mean (3120 +)	4.00	3.94	3.79	3.53	3.27	3.14	3.00	2.91	2.89

Fig. 5. Means of sample distributions of GRASP iteration solutions.

4. ALGORITMO GRASP
4.3. Extensiones

Reencadenamiento de Trayectorias

- **Esta técnica (*path relinking*) se propuso inicialmente para explorar las trayectorias entre soluciones elite (conjunto de las mejores soluciones obtenidas en varias ejecuciones de la búsqueda tabú u otro algoritmo)**

- **Usando una o más soluciones elite, se exploran las trayectorias en el espacio de soluciones que conducen a otras soluciones elite para buscar mejores soluciones**

- **Para generar trayectorias, los movimientos se seleccionan para introducir atributos en la solución actual que estén presentes en la solución elite guía**

GRASP Híbrido

- **Se introduce una mutación (perturbación fuerte) a la solución encontrada tras la etapa de la Búsqueda Local y se repite la optimización**

Procedimiento GRASP Híbrido

Repetir Mientras (no se satisfaga el criterio de parada)

S ← Construcción Solución Greedy Aleatorizada ()

Repetir Mientras (se decida continuar)

S' ← Búsqueda Local (S)

Actualizar (S', Mejor_Solución)

S ← Mutar Solución (S')

Devolver (Mejor_Solución)

FIN-GRASP-Híbrido

4. ALGORITMO GRASP
4.4. Ejemplo: Viajante de Comercio

- Construcción de soluciones *greedy* aleatorizadas en el Viajante de Comercio:
  - Se basa en la misma regla heurística empleada en el algoritmo *greedy*: seleccionar la siguiente ciudad entre las más cercanas
  - Cada vez que se debe seleccionar una nueva ciudad en el recorrido a partir de la ciudad actual, se construye una lista LRC con las ciudades más cercanas (menor coste) a ella
  - Se selecciona aleatoriamente una ciudad de la lista y se vuelve a construir una nueva lista para la nueva ciudad

4. ALGORITMO GRASP
4.4. Ejemplo: Viajante de Comercio

- Ejemplo de ejecución de la primera etapa:
  - Instancia con 6 ciudades. Se comienza en la ciudad 1. LRC de tamaño 3.

	Distancias								
	1	2	3	4	5	6	LRC	Uniforme	Solución
1	-	23	54	17	132	41	{4, 2, 6}	6	(1 6 - - - -)
6	-	48	31	142	39	-	{3, 5, 2}	3	(1 6 3 - - -)
3	-	12	-	45	28	-	{2, 5, 4}	5	(1 6 3 5 - -)
5	-	59	-	22	-	-	{4, 2}	2	(1 6 3 5 2 -)
2	-	-	-	100	-	-	{4}	4	(1 6 3 5 2 4)

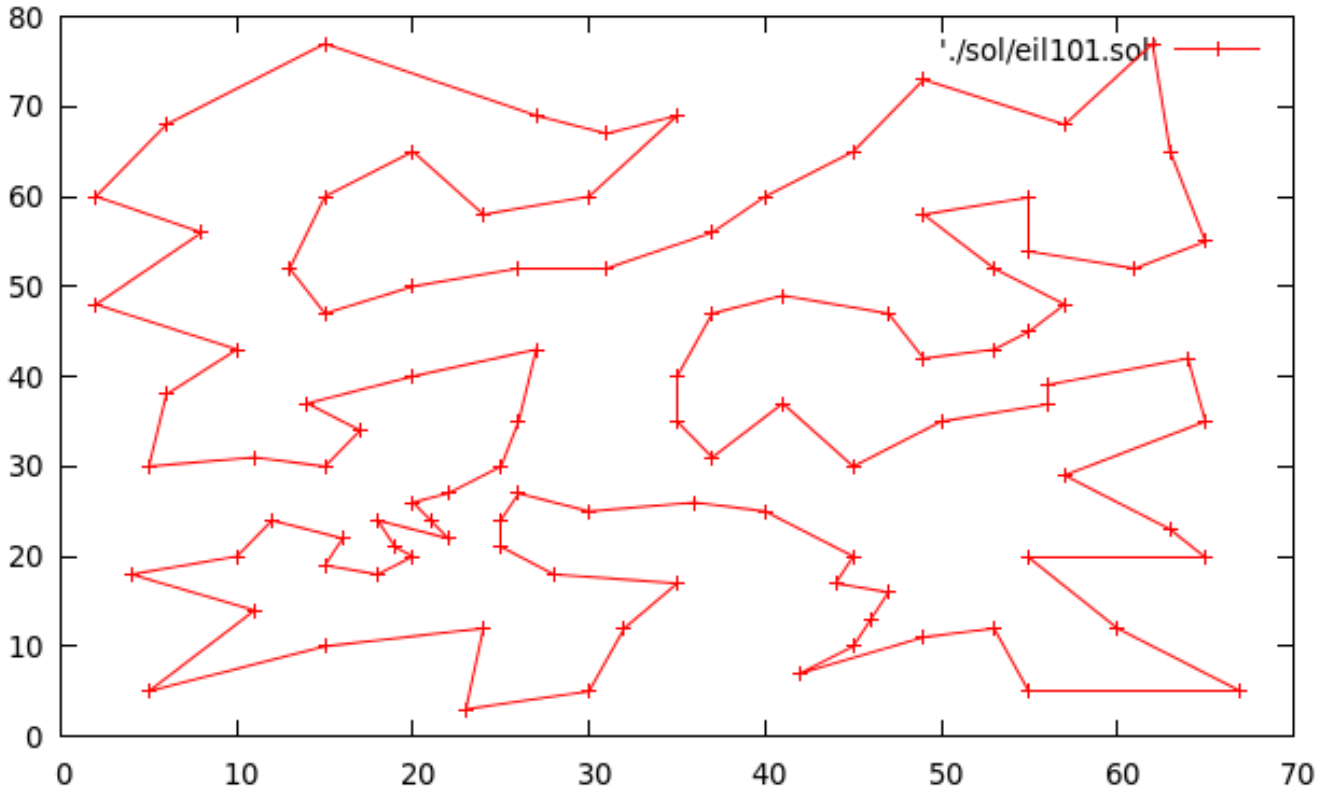
4. ALGORITMO GRASP
4.4. Ejemplo: Viajante de Comercio

- Mutación en GRASP Híbrido:
  - Debe realizarse una perturbación mayor que la que provoca el operador de vecino empleado en la búsqueda local (intercambio, 2-opt)
  - Para ello, cada vez que se realiza una mutación se aplica la modificación por sublista aleatoria de tamaño fijo ($s=n/4$) consistente en seleccionar una cadena consecutiva de elementos de tamaño s y alterar aleatoriamente sus asignaciones

(1 2 4 3 8 5 7 6)
(1 2 8 3 5 4 7 6)

4. ALGORITMO GRASP
4.4. Ejemplo: Viajante de Comercio

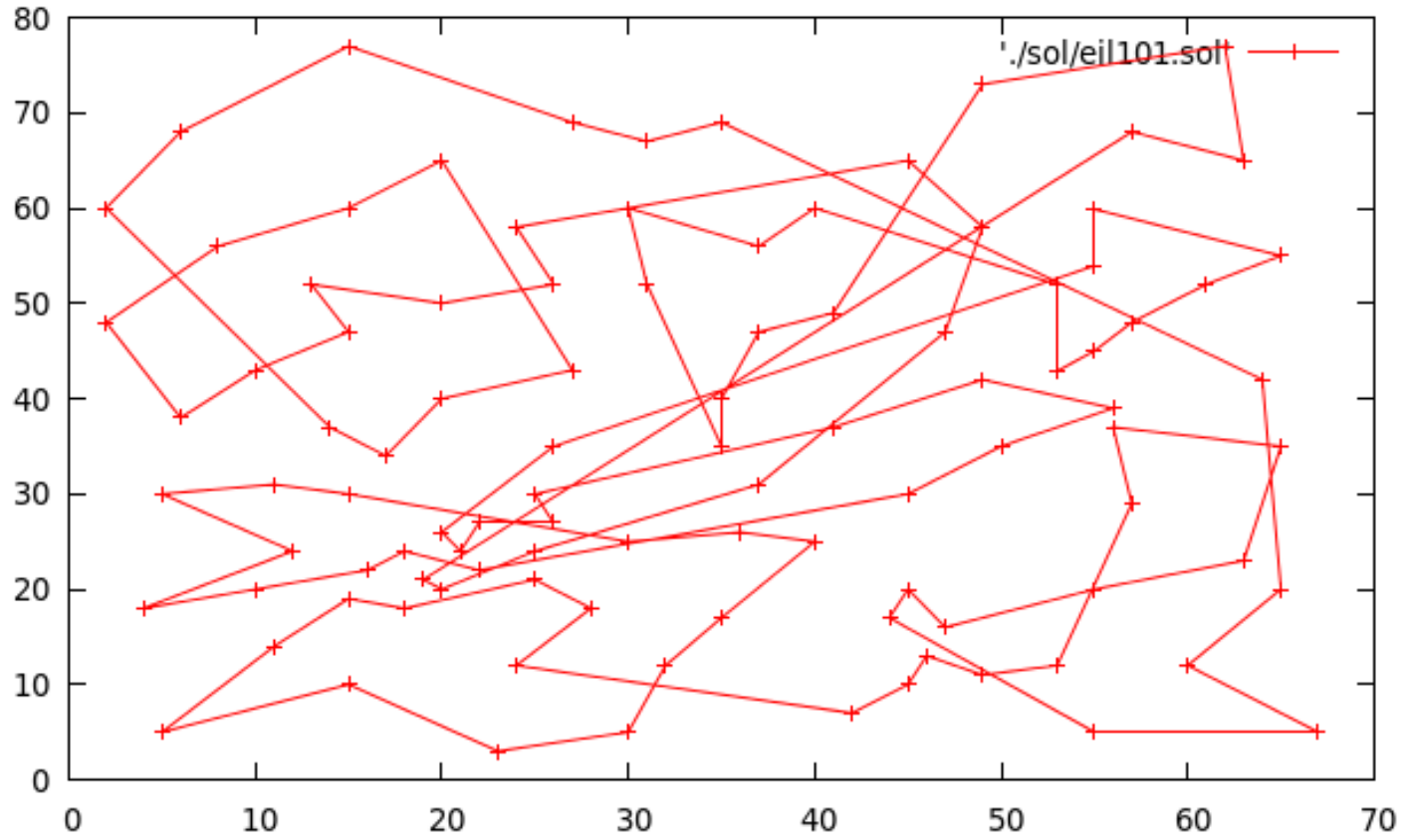
Problema Eil101 – Sol. Optima - 629



4. ALGORITMO GRASP

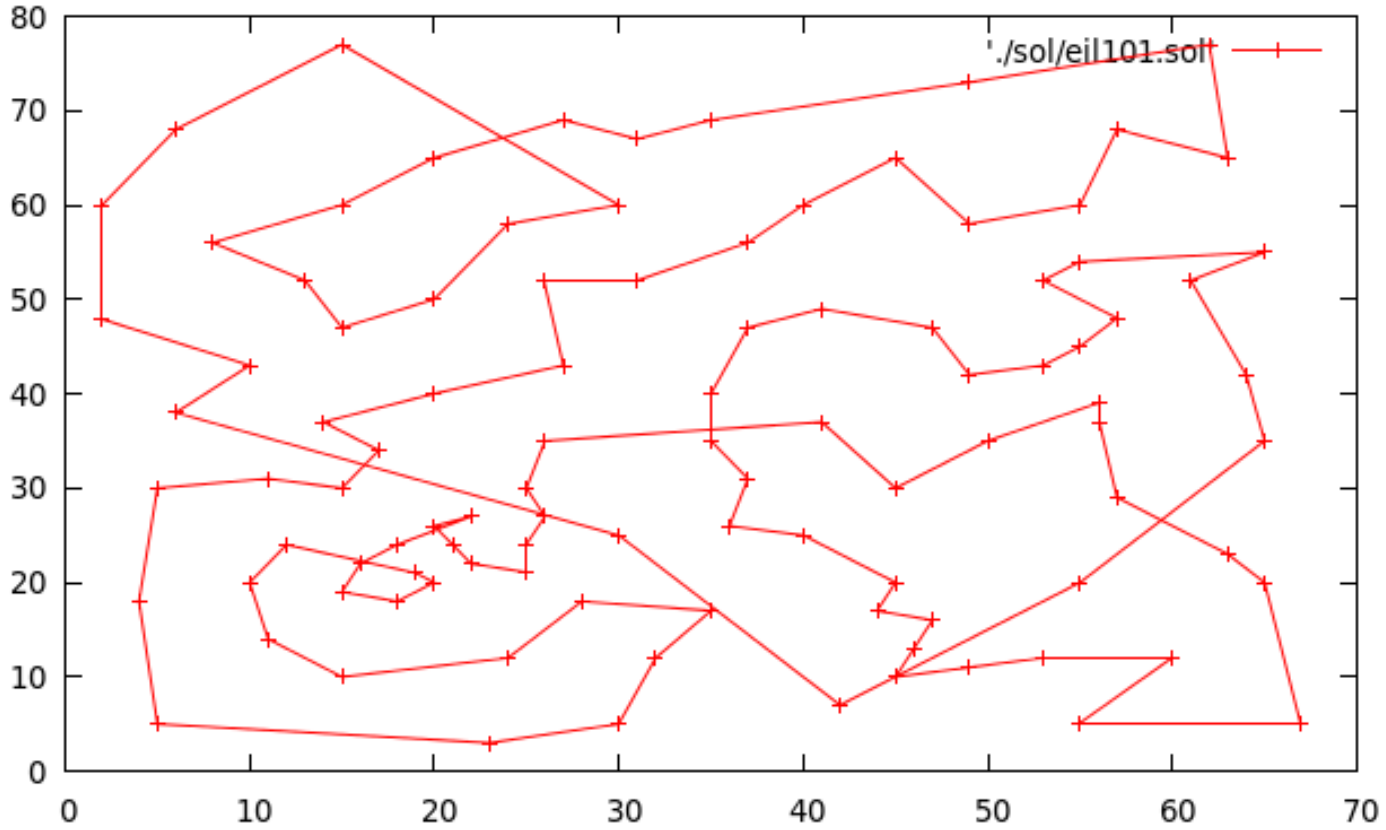
4.4. Ejemplo: Viajante de Comercio

Problema Eil101 – Sol. Aleatoria + inter- cambio de 2 posiciones – 1004 - 0.025 seg



4. ALGORITMO GRASP
4.4. Ejemplo: Viajante de Comercio

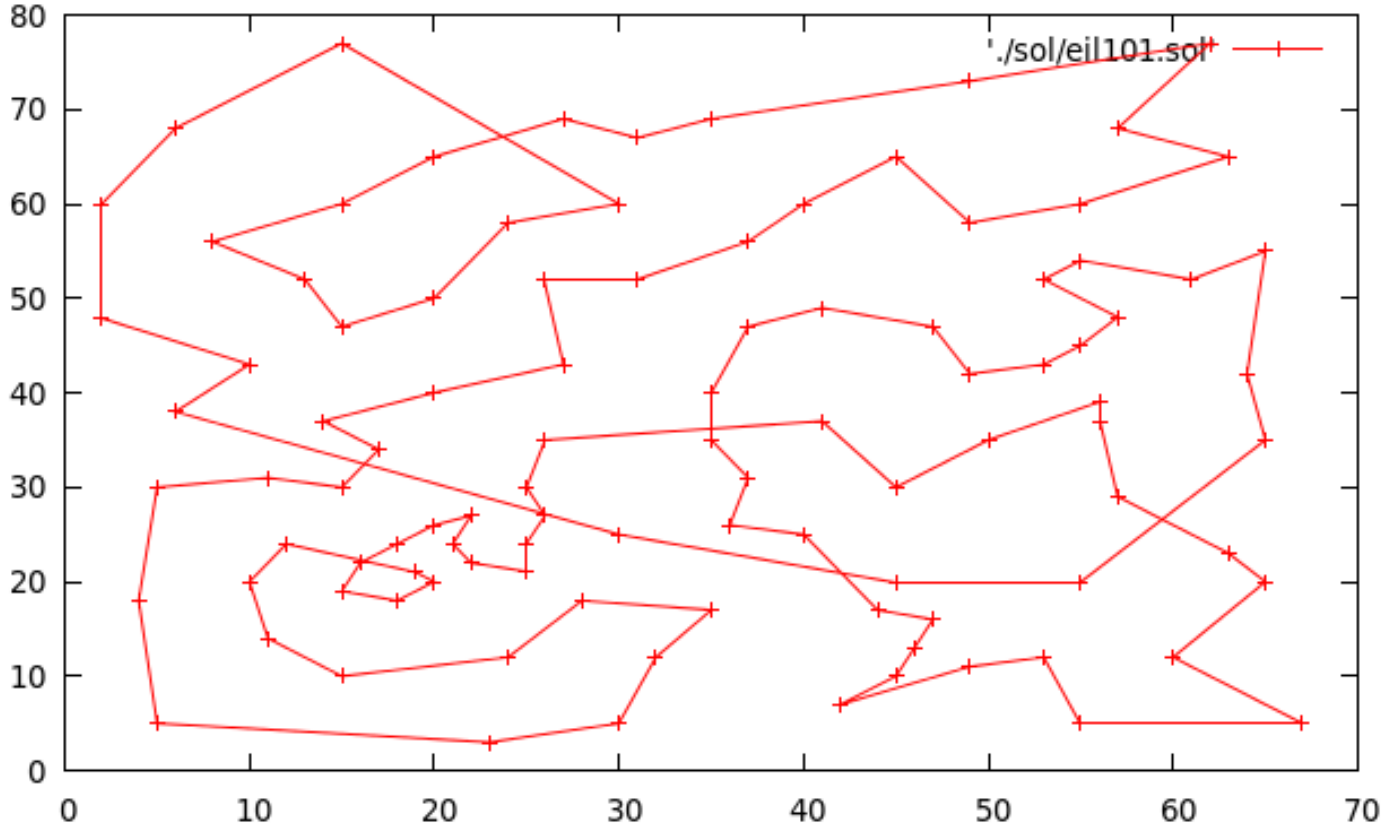
Problema Eil101 – Vecino más cercano – 736 – 0.016 seg



4. ALGORITMO GRASP

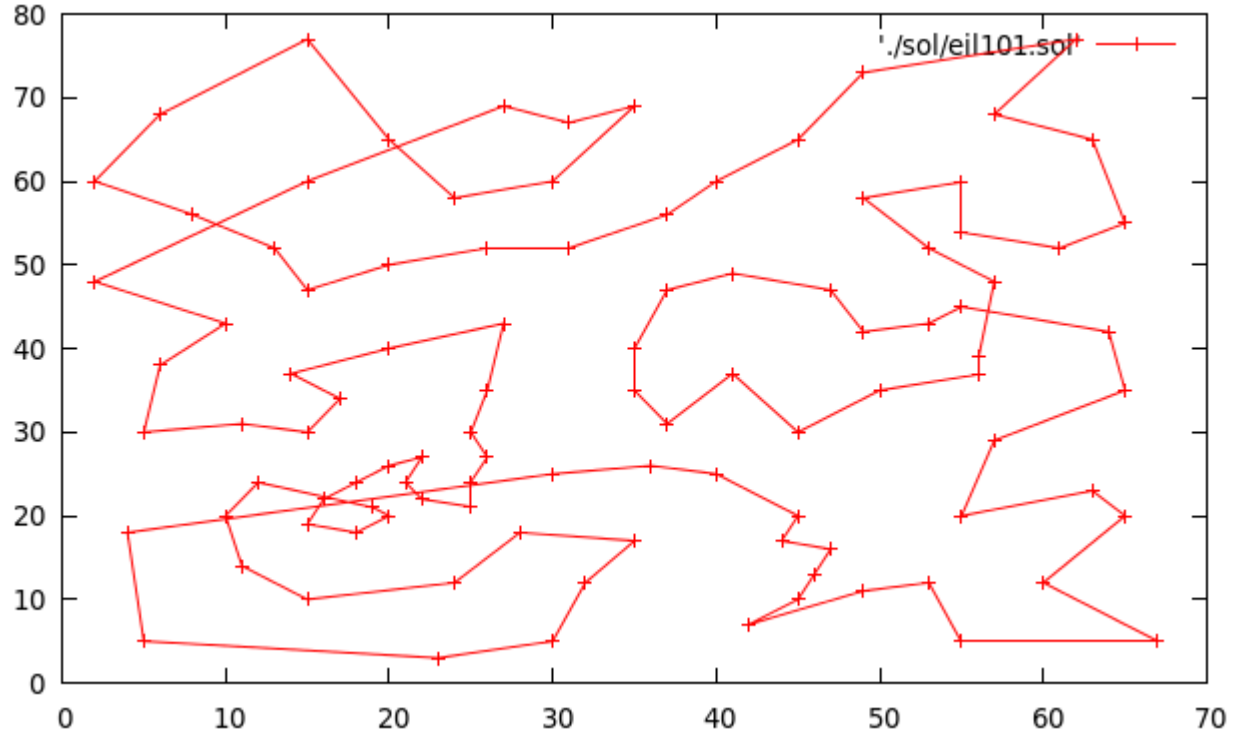
4.4. Ejemplo: Viajante de Comercio

Problema Eil101 – Vecino más cercano + LS – 722.002 – 0.016 + 0.0092 seg



4. ALGORITMO GRASP
4.4. Ejemplo: Viajante de Comercio

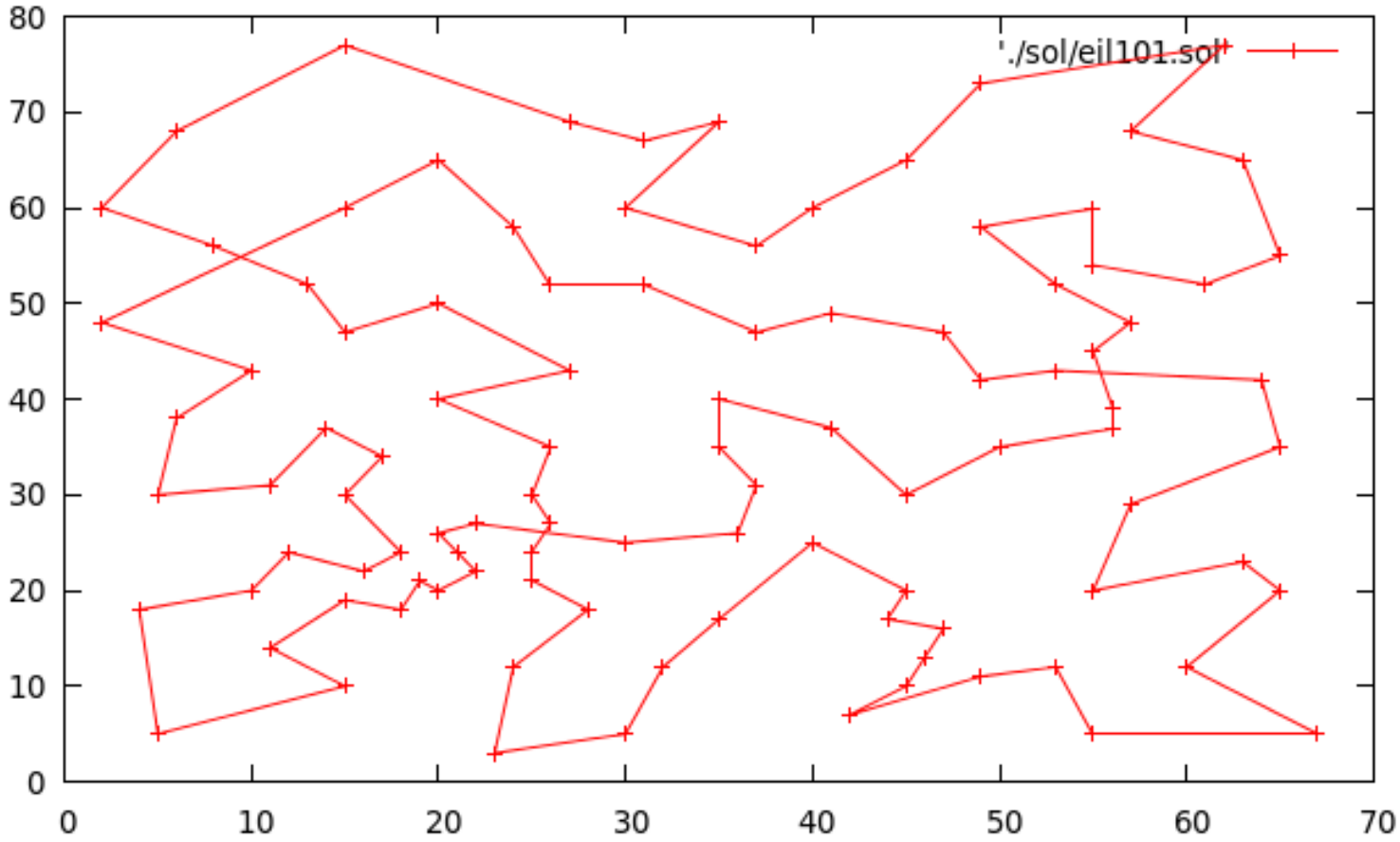
Problema Eil101 – GRASP + LS – 300 iteraciones – 677 -1.37 segs



Nota: En cada iteración al construir una solución greedy se escoge como lista de candidatos aquellos con valor de la función greedy no mayor que 1.1 del valor del mejor candidato.Local search 2-OPT del primer mejor tras la construcción.

4. ALGORITMO GRASP
4.4. Ejemplo: Viajante de Comercio

Problema Eil101 – SA . 665.68 – 1.3 seg – 10000 iteraciones (1000 soluciones bucle interno)



3. MODELOS MULTIARRANQUE

- **Existen múltiples propuestas de metaheurísticas que se pueden considerar como técnicas multiarranque:**
  - **Métodos constructivos de la solución inicial**
    - **Construcción *greedy*: Algoritmos GRASP**
    - **Algoritmos Basados en Colonias de Hormigas: ACO (Tema 6)**
  - **Métodos iterativos mediante modificación de la solución encontrada**
    - **ILS: Búsqueda Local Reiterativa**
    - **VNS: Búsqueda de Entorno Variable**
  - **Hibridaciones entre técnicas poblacionales de exploración/ combinación de soluciones y métodos de búsqueda local (Tema 4)**
    - **Algoritmos Meméticos / Algoritmos Genéticos con BL**

5. ALGORITMOS DE BÚSQUEDA LOCAL REITERATIVOS BASADOS EN ÓPTIMOS: ILS

5.1. Algoritmo ILS

5.2. Modelo ILS Basado en Poblaciones

5.1. Algoritmo ILS

- **La ILS está basada en la aplicación repetida de un algoritmo de Búsqueda Local a una solución inicial que se obtiene por mutación de un óptimo local previamente encontrado**

- **Propuesta inicialmente en la Tesis Doctoral de Thomas Stützle:**

T. Stützle, 1998. Local Search Algorithms for Combinatorial Problems- Analysis, Improvements and New Applications. PhD Thesis, Darmstadt, University of Technology, Department of Computer Science.

5. ALGORITMOS DE BÚSQUEDA LOCAL REITERATIVOS BASADOS EN ÓPTIMOS: ILS

5.1. Algoritmo ILS

- **La aplicación de la ILS necesita de la definición de cuatro componentes:**

  1. **Una *solución inicial* (usualmente, aleatoria)**
  2. **Un *procedimiento de modificación* (mutación) que aplica un cambio brusco sobre la solución actual para obtener una solución intermedia**
  3. **Un procedimiento de *Búsqueda Local***
  4. **Un *criterio de aceptación* que decide a qué solución se aplica el procedimiento de modificación**

- **En la mayoría de las aplicaciones, la mutación se aplica a la mejor solución encontrada**

H.R. Lourenço, O.C. Martin, T. Stützle, Iterated Local Search. En: F.Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics. Kluwer Academic Publishers, 2003, pp. 321-353.

H.H. Hoos, T. Stützle. Stochastic Local Search. Morgan Kaufmann, 2004.

5. ALGORITMOS DE BÚSQUEDA LOCAL REITERATIVOS BASADOS EN ÓPTIMOS: ILS

5.1. Algoritmo ILS

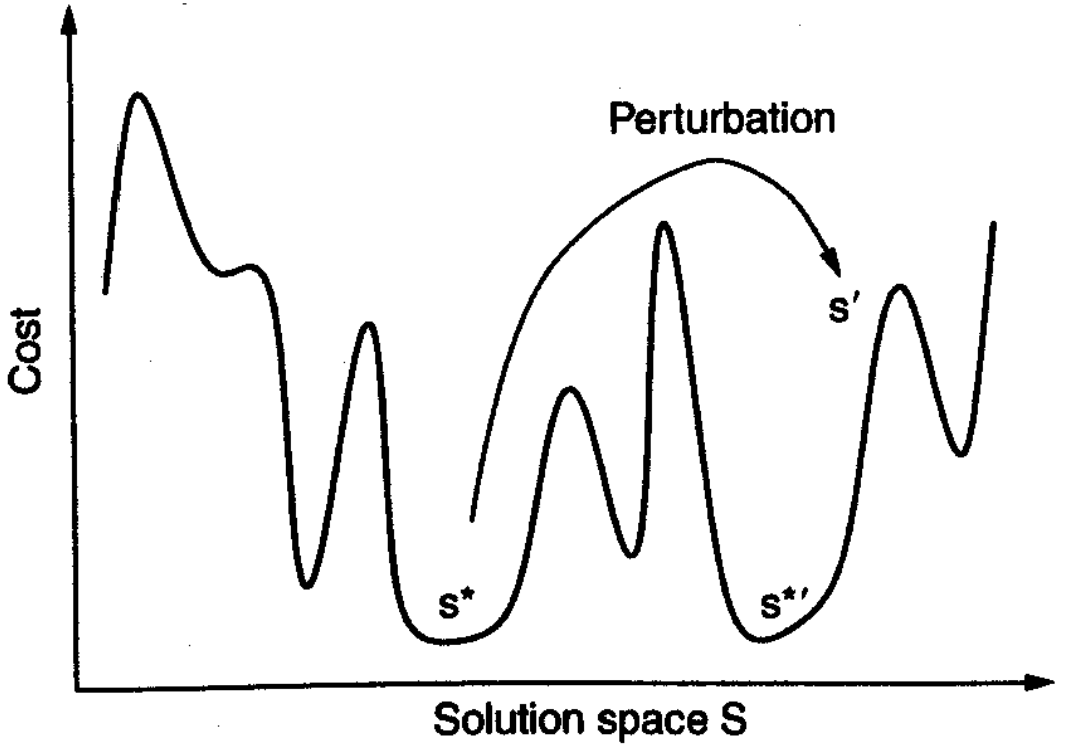


Figura: Representación del funcionamiento de la ILS

Caída en el óptimo local s*. La perturbación/mutación conduce a s'. Después de aplicar la Búsqueda Local, se encuentra un nuevo óptimo s*' que es mejor que s*

5. ALGORITMOS DE BÚSQUEDA LOCAL REITERATIVOS BASADOS EN ÓPTIMOS: ILS

5.1. Algoritmo ILS

Procedimiento Búsqueda Local Reiterada (ILS)

Comienzo-ILS
  S_o ← Generar-Solución-Inicial
  S ← Búsqueda Local (S_o)
  Repetir
    S' ← Modificar (S, historia) **%Mutación**
    S'' ← Búsqueda Local (S')
    S ← Criterio-Aceptación (S, S'', historia)
    Actualizar (S, *Mejor_Solución*)
  Hasta (Condiciones de terminación)
Devolver *Mejor_Solución*
Fin-ILS

Modelo General del Algoritmo ILS

5. ALGORITMOS DE BÚSQUEDA LOCAL REITERATIVOS BASADOS EN ÓPTIMOS: ILS

5.1. Algoritmo ILS

La solución inicial:

se elige aleatoriamente o mediante una heurística constructiva, como *greedy*

La "historia" se utiliza para influenciar en:

- **la modificación aplicada a la solución actual, o**
- **la solución a modificar, haciendo uso de listas de las mejores soluciones o cualquier otro tipo de información**

Criterios de Aceptación:

- **Criterio del Mejor**
  - **Criterio-Aceptación (S, S'', historia) = Mejor (S, S'')**
  - **Este criterio favorece la intensificación**
- **Criterio RW (Random walk)**
  - **Criterio-Aceptación (S, S'', historia) = RW (S, S'') = S''**
  - **Este criterio favorece la diversificación sobre la intensificación**
- **Criterios Intermedios**
  - **S = Reinicializar (S, S'', historia) - Cualquier otro criterio de aceptación, método de Enfriamiento Simulado, etc.**
  - **Si el algoritmo no mejora la solución durante it_o iteraciones, se asume que se ha llegado a un óptimo local y se reinicializa parcialmente la solución (mutación fuerte)**

5. ALGORITMOS DE BÚSQUEDA LOCAL REITERATIVOS BASADOS EN ÓPTIMOS: ILS

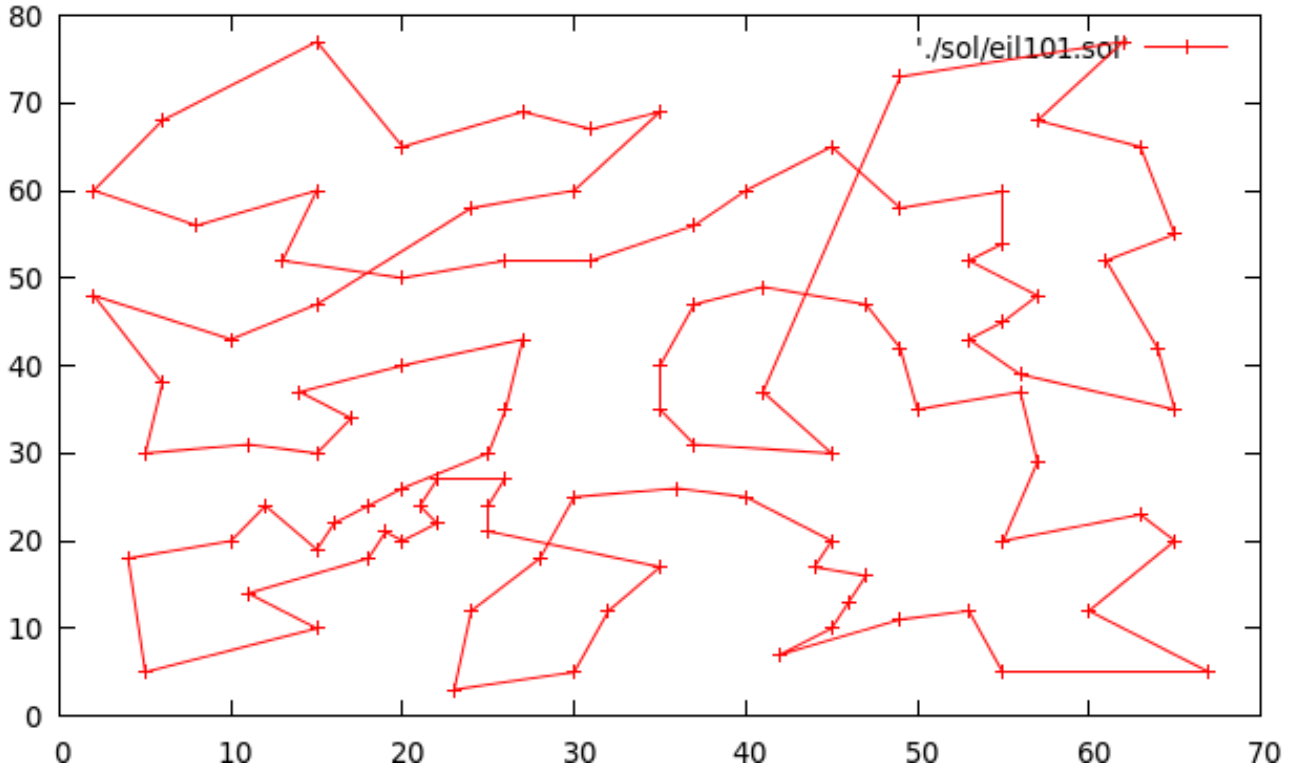
5.2. Modelo ILS Basado en Poblaciones

- **MODELO REEMPLAZAR EL PEOR**
  - **Se genera una población de soluciones iniciales**
  - **Se aplica una ILS en paralelo a partir de cada una de ellas, con la única peculiaridad de que las BL se ejecutan sólo durante *it* iteraciones**
  - **Cada *rb* iteraciones, se reemplaza la peor solución encontrada por la mejor**
  - **La motivación asociada a este esquema es ir concentrando gradualmente la búsqueda alrededor de la mejor solución de la población**

- **MODELO ESTRATEGIA (μ+λ)**
  - **Se generan μ soluciones iniciales**
  - **Se obtienen λ hijos a partir de ellas mediante la mutación**
  - **Se aplica la BL a cada hijo y se seleccionan las μ mejores soluciones obtenidas para formar la siguiente población**
  - **Para evitar la convergencia local se pueden seleccionar los μ mejores atendiendo a la distancia entre ellos**

5. ALGORITMO ILS
Ejemplo: Viajante de Comercio

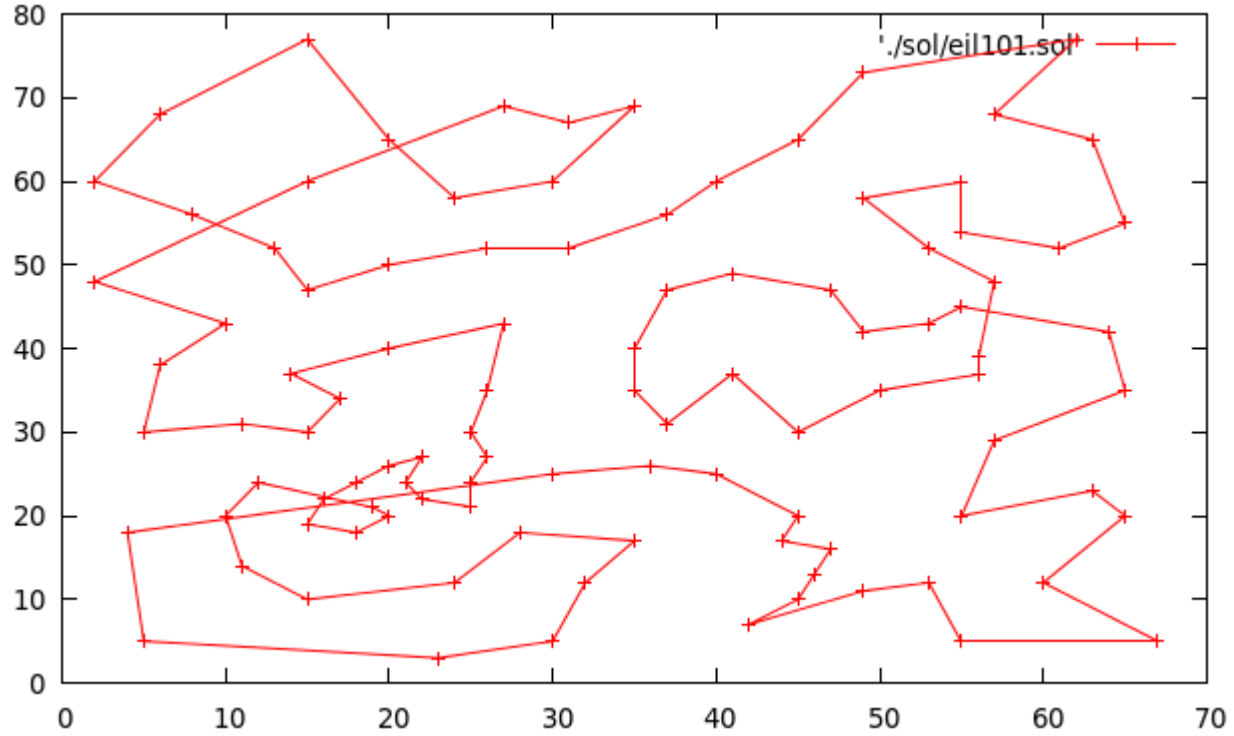
Problema Eil101 – ILS + LS – 120 iteraciones – 690 -1.49 segs



Nota: En cada iteración se produce una mutación aleatoria en un sector aleatorio de la solución y después se le aplica LS. Local search del primer mejor tras la construcción.

5. ALGORITMO ILS
Ejemplo: Viajante de Comercio

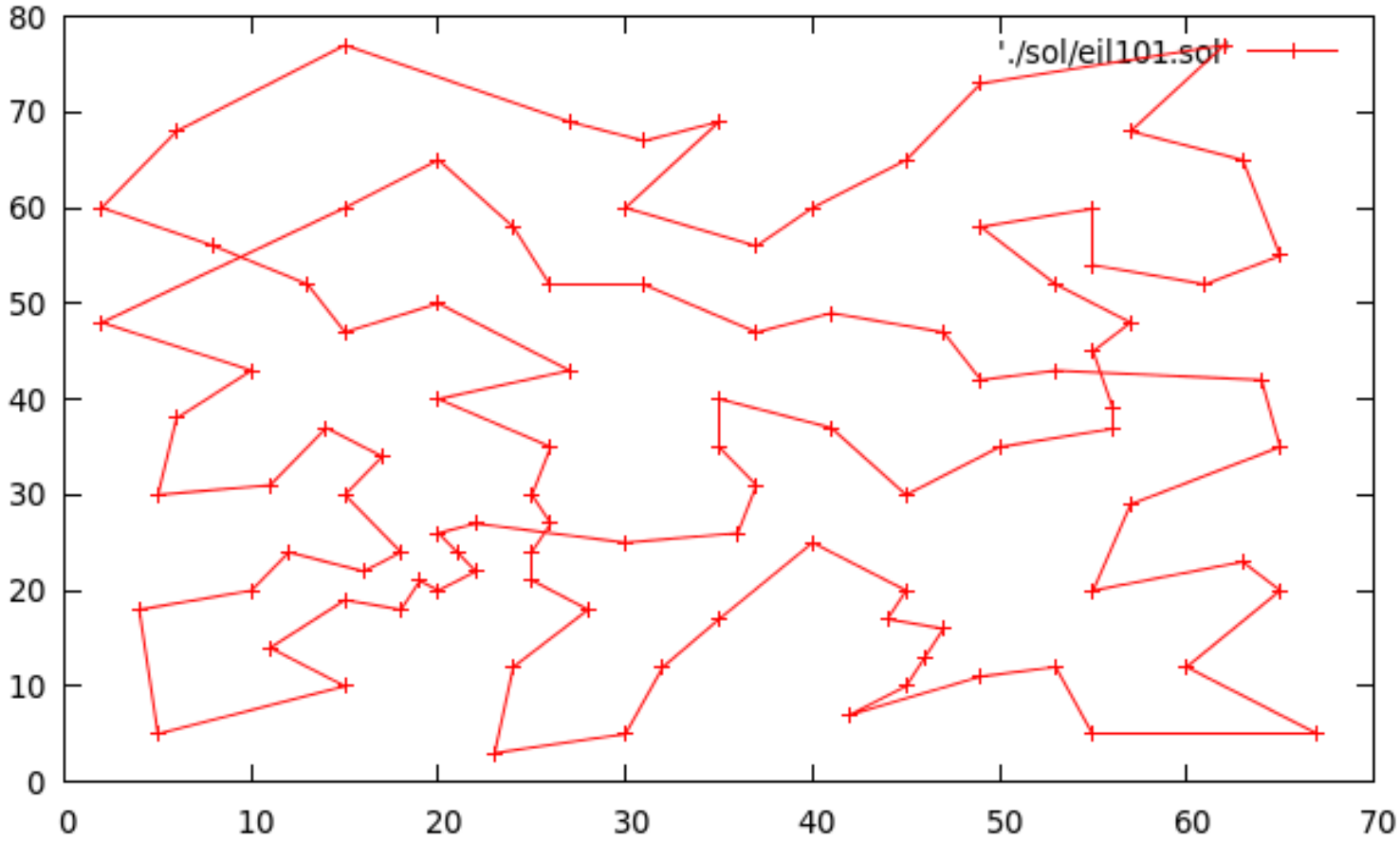
Problema Eil101 – GRASP + LS – 300 iteraciones – 677 -1.37 segs



Nota: En cada iteración al construir una solución greedy se escoge como lista de candidatos aquellos con valor de la función greedy no mayor que 1.1 del valor del mejor candidato.Local search del primer mejor tras la construcción.

5. ALGORITMO ILS
Ejemplo: Viajante de Comercio

Problema Eil101 – SA . 665.68 – 1.3 seg – 10000 iteraciones (1000 soluciones bucle interno)



6. BÚSQUEDA DE ENTORNO VARIABLE: VNS

6.1. Introducción
6.2. Modelo VNS Básico
6.3. Otros Modelos de Entornos Variables
6.4. VNS versus ILS

6.1. Introducción

- **La Búsqueda de Entorno Variable (VNS) es una metaheurística para resolver problemas de optimización cuya idea básica es el cambio sistemático de entorno dentro de una búsqueda local (aumentando el tamaño cuando la búsqueda no avanza)**

N. Mladenovic, P. Hansen, Variable Neighborhood Search. Computers & Operations Research 24:11 (1997) 1097-1100.

6. BÚSQUEDA DE ENTORNO VARIABLE: VNS

6.1. Introducción

La VNS está basada en tres hechos simples:

1. **Un mínimo local con una estructura de entornos no lo es necesariamente con otra**
2. **Un mínimo global es mínimo local con todas las posibles estructuras de entornos**
3. **Para muchos problemas, los mínimos locales con la misma o distinta estructura de entorno están relativamente cerca**

Los hechos 1 a 3 sugieren el empleo de varias estructuras de entornos en las búsquedas locales para abordar un problema de optimización

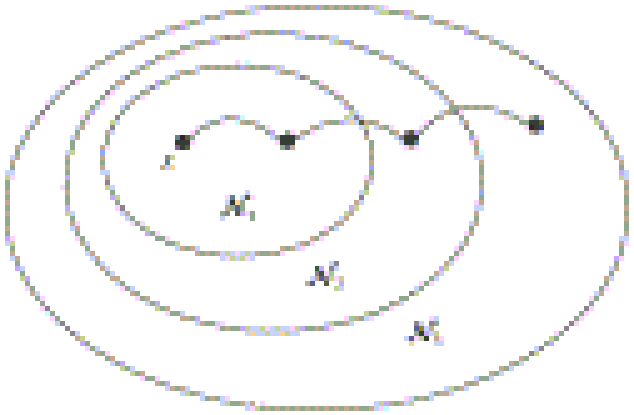


Figura: Secuencia Encajada de Entornos

6. BÚSQUEDA DE ENTORNO VARIABLE: VNS

6.1. Introducción

Problemas de la Búsqueda Local

Suele caer en óptimos locales, que a veces están bastante alejados del óptimo global del problema

SOLUCIONES: 3 opciones para salir de los óptimos locales

- **Permitir movimientos de empeoramiento de la solución actual (Ejemplo: Enfriamiento Simulado, Búsqueda Tabú, …)**
- **Modificar la estructura de entornos (Ejemplo: Búsqueda Tabú, Búsqueda en Entornos Variables: VNS, …)**
- **Volver a comenzar la búsqueda desde otra solución inicial (Ejemplo: Búsquedas Multiarranque, GRASP, ILS, VNS …)**

6. BÚSQUEDA DE ENTORNO VARIABLE: VNS

6.1. Introducción

Existen dos variantes:

- ***Búsqueda Descendente Basada en Entornos Variables (VND):***

  Algoritmo de BL del mejor cuyo operador de vecino cambia de entorno (ampliándolo) cuando el mejor vecino generado es peor que la solución actual (extensión de la búsqueda local clásica)

- ***Búsqueda Basada en Entornos Variables (VNS):***

  Algoritmo ILS en el que el operador de mutación cambia de entorno cuando la solución obtenida tras aplicar la BL es peor que la solución actual

6. BÚSQUEDA DE ENTORNO VARIABLE: VNS

6.2. Modelo VNS Básico

- Sea E_k ($k = 1, ..., k_{máx}$) un conjunto finito de estructuras de vecindario (entorno) preseleccionadas, y sea $E_k(S)$ el conjunto de soluciones del entorno k-ésimo de S

- VNS aplica progresivamente una BL sobre una solución S' obtenida a partir de una mutación de la actual S, realizada de acuerdo al tipo de entorno utilizado en cada iteración $E_k(S)$

- Si la última BL efectuada resultó efectiva, es decir, si la solución obtenida tras ella, S'', mejoró la solución actual, S, se pasa a trabajar con el entorno primero E_1.

- En caso contrario, se pasa al siguiente entorno ($k \leftarrow k+1$) para provocar una perturbación mayor y alejar la nueva solución de inicio de la BL, S', de la zona del espacio de búsqueda en la que está situada la actual S

6. BÚSQUEDA DE ENTORNO VARIABLE: VNS

6.2. Modelo VNS Básico

Procedimiento Búsqueda Basada en Entornos Variables (VNS)

Comienzo-VNS
  S ← Generar-Solución-Inicial
  k ← 1
  Repetir mientras (k <= k_{max})
    S' ← Mutación-en-E_k (S)
    S'' ← Búsqueda Local (S')
    Si S'' mejor que S entonces
      S ← S''; k ← 1
    si no k ← k+1
  Fin-repetir
Devolver S
Fin-VNS

Selección de Estructuras de Vecindario

Es posible seleccionar diferentes heurísticas para utilizar en cada iteración en la que se aplica la BL

Existen diferentes posibilidades:

- **Cambiar los parámetros de los métodos existentes en cada iteración**
- **Utilizar movimientos de diferente tamaño k para generar vecindarios que aumentan de tamaño de acuerdo al aumento del parámetro k**
- **Combinar las estrategias previas**

6. BÚSQUEDA DE ENTORNO VARIABLE: VNS

6.3. Otros Modelos de Entornos Variables

En la literatura se han propuesto diversas formas de extender la VNS para dotarla de algunas características adicionales:

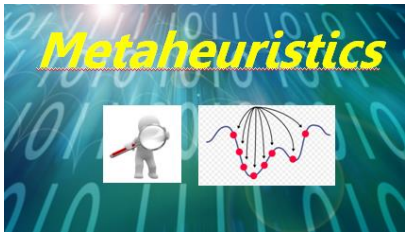
- **VNS General (GVNS): Utiliza una VND como búsqueda local**
- **VNS con Descomposición (VNDS): Extiende a VNS en dos niveles basados en la descomposición del problema trabajando sobre subconjuntos de variables**
- **VNS Sesgada (SVNS): Se afronta la exploración de zonas alejadas. Degenera, en algún sentido, en una heurística de arranque múltiple en la que se realizan iterativamente BLs desde soluciones generadas al azar**
- **Hibridaciones de VSN con Búsqueda Tabú, GRASP, Multiarranque clásico, …**

6. BÚSQUEDA DE ENTORNO VARIABLE: VNS

6.4. VNS versus ILS

- **La mayor diferencia entre ILS y VNS radica en la filosofía subyacente en las dos metaheurísticas**
  - **ILS tiene explícitamente el objetivo de construir un camino en el conjunto de soluciones optimales locales**
  - **VNS se deriva desde la idea de cambiar sistemáticamente de entorno a lo largo de la búsqueda**

METAHEURÍSTICAS
2019 - 2020



- Tema 1. Introducción a las Metaheurísticas
- Tema 2. Modelos de Búsqueda: Entornos y Trayectorias vs Poblaciones
- Tema 3. Metaheurísticas Basadas en Poblaciones
- Tema 4: Algoritmos Meméticos
- Tema 5. Metaheurísticas Basadas en Trayectorias
- Tema 6. Metaheurísticas Basadas en Adaptación Social
- Tema 7. Aspectos Avanzados en Metaheurísticas
- Tema 8. Metaheurísticas Paralelas