Algoritmos de Búsqueda Local

Jose Antonio Lozano

Intelligent Systems Group
Departamento de Ciencias de la Computación e Inteligencia Artificial
Universidad del País Vasco–Euskal Herriko Unibertsitatea

UPV - EHU

Organización del tema

1. Búsqueda Local
2. Extensiones de la Búsqueda Local

Búsqueda local

Características

- Es el algoritmo heurístico más sencillo
- Está basado en el concepto de localidad
- Se mantiene en todo momento una posible solución al problema
- A cada paso se elige una solución cercana a la solución actual que la mejore
- El algoritmo termina cuando ninguna solución cercana mejora la actual

Búsqueda local

Características

- Es el algoritmo heurístico más sencillo
- Está basado en el concepto de localidad
- Se mantiene en todo momento una posible solución al problema
- A cada paso se elige una solución cercana a la solución actual que la mejore
- El algoritmo termina cuando ninguna solución cercana mejora la actual

Búsqueda local

Características

- Es el algoritmo heurístico más sencillo
- Está basado en el concepto de localidad
- Se mantiene en todo momento una posible solución al problema
- A cada paso se elige una solución cercana a la solución actual que la mejore
- El algoritmo termina cuando ninguna solución cercana mejora la actual

Búsqueda local

Características

- Es el algoritmo heurístico más sencillo
- Está basado en el concepto de localidad
- Se mantiene en todo momento una posible solución al problema
- A cada paso se elige una solución cercana a la solución actual que la mejore
- El algoritmo termina cuando ninguna solución cercana mejora la actual

Búsqueda local

Características

- Es el algoritmo heurístico más sencillo
- Está basado en el concepto de localidad
- Se mantiene en todo momento una posible solución al problema
- A cada paso se elige una solución cercana a la solución actual que la mejore
- El algoritmo termina cuando ninguna solución cercana mejora la actual

Búsqueda local

Seleccionar una solución inicial $e_0 \in \mathcal{E}$
Repetir
  Elegir $e \in V(e_0)$ tal que $f(e) < f(e_0)$
  Asignar e a e_0
hasta $f(e) \geq f(e_0) \ \ \forall \ e \in V(e_0)$
e_0 es la aproximación a la solución óptima

Figura: Pseudocódigo para un algoritmo de búsqueda local en un problema de minimización

Búsqueda local

Aspectos a determinar en la búsqueda local

- Solución inicial
- Conjunto de soluciones vecinas $V(e)$ de cada solución e
- Elección de la solución vecina a cada paso

Búsqueda local

Aspectos a determinar en la búsqueda local

- Solución inicial
- Conjunto de soluciones vecinas $V(e)$ de cada solución e
- Elección de la solución vecina a cada paso

Búsqueda local

Aspectos a determinar en la búsqueda local

- Solución inicial
- Conjunto de soluciones vecinas $V(e)$ de cada solución e
- Elección de la solución vecina a cada paso

Búsqueda local

Aspectos a determinar en la búsqueda local

- Solución inicial
- Conjunto de soluciones vecinas $V(e)$ de cada solución e
- Elección de la solución vecina a cada paso

Sistemas de vecinos

Características

- Formalmente un sistema de vecinos V en un espacio de búsqueda $\mathcal{E}$ es:

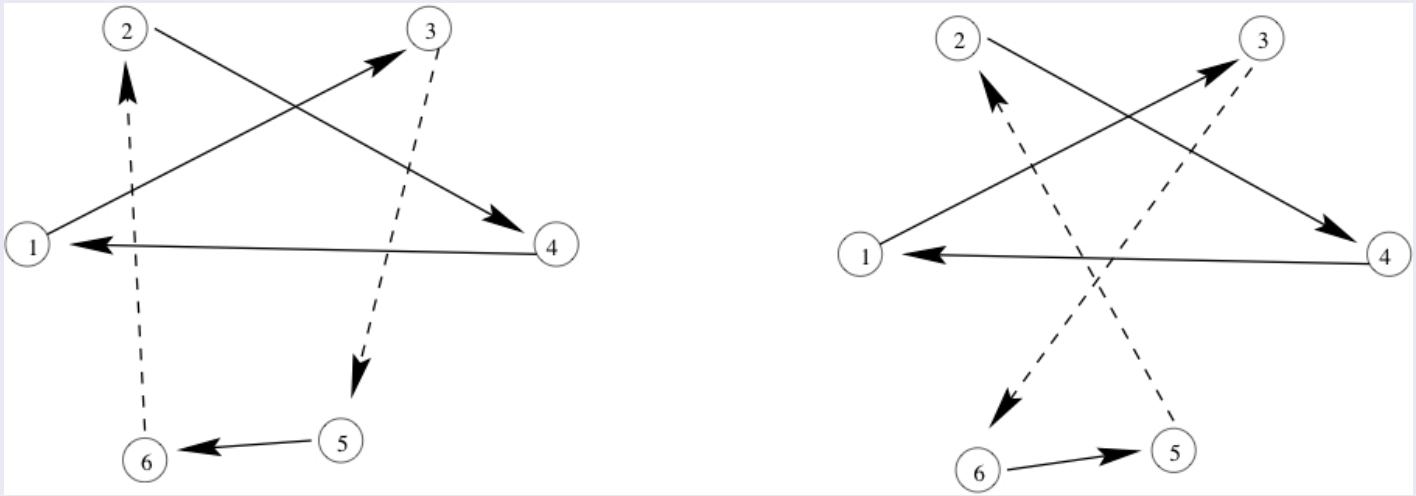
$$\begin{array}{rccc} V: & \mathcal{E} & \longrightarrow & \mathcal{P}(\mathcal{E}) \\ & e & \longrightarrow & V(e) \end{array}$$

- A cada solución e se le asigna un conjunto de soluciones *vecinas* $V(e)$

Sistema de vecinos para el TSP

Sistema de vecinos 2-opt

- Dos soluciones son vecinas si una de ellas se genera tras eliminar dos arcos no contiguos y añadir otros dos arcos que completen el ciclo



Sistema de vecinos para el TSP

Sistema de vecinos 2-opt

- En nuestra representación: (1 3 5 6 2 4) y (1 3 6 5 2 4) son vecinas:
- Tamaño del sistema de vecinos: $\frac{n(n-1)}{2} - n$
- Animación del TSP

Sistema de vecinos para el TSP

Sistema de vecinos 2-opt

- En nuestra representación: (1 3 5 6 2 4) y (1 3 6 5 2 4) son vecinas:
- Tamaño del sistema de vecinos: $\frac{n(n-1)}{2} - n$
- Animación del TSP

Sistema de vecinos para el problema de la mochila

Sistema de vecinos

- Todas las soluciones de una dada que se consigan mediante las siguientes dos operaciones:
  - Introducir un nuevo objeto
  - Introducir un nuevo objeto y eliminar otro

Sistema de vecinos para el problema de la mochila

Sistema de vecinos

- Todas las soluciones de una dada que se consigan mediante las siguientes dos operaciones:
  - Introducir un nuevo objeto
  - Introducir un nuevo objeto y eliminar otro

Sistema de vecinos para el problema de la mochila

Sistema de vecinos

- Todas las soluciones de una dada que se consigan mediante las siguientes dos operaciones:
  - Introducir un nuevo objeto
  - Introducir un nuevo objeto y eliminar otro

Elección de un sistema de vecinos

Aspectos a considerar

- Tiene que tener en cuenta la estructura del problema
- No debe ser ni muy grande ni muy pequeño, ¡sino todo lo contrario!
- Tener en cuenta en el diseño la simplificación en la evaluación de los vecinos
- Debe permitir llegar a cualquier solución desde cualquier solución

Búsqueda local

Aspectos a determinar en la búsqueda local

- Solución inicial
- Conjunto de soluciones vecinas $V(e)$ de cada solución e
- Elección de la solución vecina a cada paso

Búsqueda local

Aspectos a determinar en la búsqueda local

- Solución inicial
- Conjunto de soluciones vecinas $V(e)$ de cada solución e
- Elección de la solución vecina a cada paso

Elección de los vecinos

Estrategias de elección

- Se elige el mejor vecino (tratar los empates)
- Se elige el primero que mejora siguiendo una estrategia de inspección determinista
- Se elige el primero que mejora siguiendo una estrategia de inspección aleatoria
- Se elige el k-ésimo que mejora (diferentes estrategias de inspección)

Búsqueda local

Puntos a favor

- Sencillez
- Tiempo computacional

Puntos en contra

- El algoritmo termina en un óptimo local
- La solución final puede depender de la inicial (podría ser también positivo)

Búsqueda local

Puntos a favor

- Sencillez
- Tiempo computacional

Puntos en contra

- El algoritmo termina en un óptimo local
- La solución final puede depender de la inicial (podría ser también positivo)

Búsqueda local

Puntos a favor

- Sencillez
- Tiempo computacional

Puntos en contra

- El algoritmo termina en un óptimo local
- La solución final puede depender de la inicial (podría ser también positivo)

Búsqueda local

Puntos a favor

- Sencillez
- Tiempo computacional

Puntos en contra

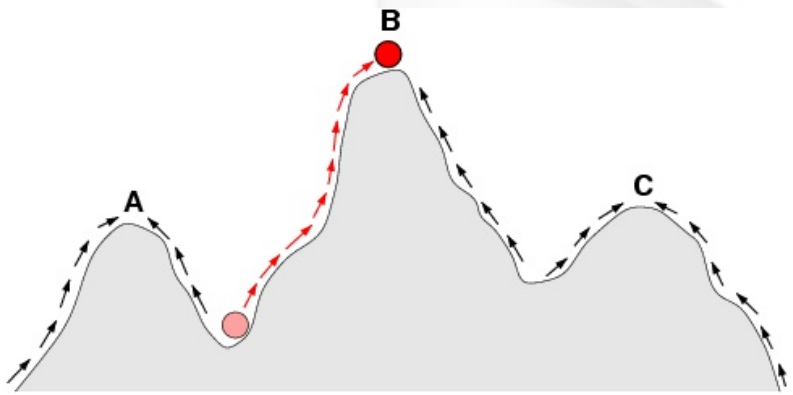
- El algoritmo termina en un óptimo local
- La solución final puede depender de la inicial (podría ser también positivo)

Óptimo local

¿Que es un óptimo local?

- Definición formal. Una solución e_{loc} se dirá que es un óptimo local si se tiene que:

$$f(e_{loc}) \leq f(e) \quad \forall e \in V(e_{loc})$$



¿Cómo mejorar la búsqueda local?

Limitaciones de la búsqueda local

- Termina en un óptimo local (no tiene porque ser global)

¿Cómo mejorar la búsqueda local?

Limitaciones de la búsqueda local

- Termina en un óptimo local (no tiene porque ser global)

Alternativas de mejora

¿Cómo mejorar la búsqueda local?

Limitaciones de la búsqueda local

- Termina en un óptimo local (no tiene porque ser global)

Alternativas de mejora

- Repetir la búsqueda comenzando en soluciones distintas (multistart)

¿Cómo mejorar la búsqueda local?

Limitaciones de la búsqueda local

- Termina en un óptimo local (no tiene porque ser global)

Alternativas de mejora

- Repetir la búsqueda comenzando en soluciones distintas (multistart)
- Aceptar soluciones que empeoren la búsqueda

¿Cómo mejorar la búsqueda local?

Limitaciones de la búsqueda local

- Termina en un óptimo local (no tiene porque ser global)

Alternativas de mejora

- Repetir la búsqueda comenzando en soluciones distintas (multistart)
- Aceptar soluciones que empeoren la búsqueda
- Modificar de forma adaptativa el sistema de vecinos

Métodos de multiarranque

Características principales

- Consisten en iterar dos pasos:
  1. Generar una solución inicial
  2. Aplicar una búsqueda local a la solución generada
- Se diferencian en como se llevan a cabo los pasos anteriores y en el criterio de parada elegido
- Algunos representantes son:
  - GRASP (greedy randomized adaptive search procedure)
  - ILS (iterated local search)

GRASP

Características principales

- El método está compuesto de dos fases:
  1. Fase constructiva. Se construye una solución inicial mediante un método que debe ser aleatorizado y adaptativo
  2. Fase de búsqueda. Partiendo de la solución anterior se realiza una búsqueda (habitualmente una búsqueda local)

GRASP para el TSP

Método constructivo

Elegir muestreando una distribución uniforme una ciudad i
Repetir $n - 1$
Calcular las k ciudades más cercanas a la última añadida (dentro de las no añadidas)

Seleccionar dentro de las k, una ciudad j con probabilidad inversamente proporcional a la distancia a la última ciudad elegida

Búsqueda local

- Algoritmo 2-opt

ILP

Características

- Basado en muestrear de forma sesgada el espacio de óptimos locales
-

Búsqueda de vecindad variable

Características

- Cuando la búsqueda local alcanza un óptimo local, una forma de salir del mismo es cambiando el sistema de vecinos
- El algoritmo se basa en considerar una sucesión de sistemas de vecinos (no necesariamente incluyentes)
- Cuando se alcanza un óptimo local se pasa al siguiente sistema de vecinos
- Si se mejora la solución entonces se vuelve al primer sistema

Búsqueda de vecindad variable

Ideas subyacentes

- Un óptimo local con una estructura de vecinos no tiene porque serlo con otra
- Un óptimo global es óptimo local con calquier estructura de vecinos
- Para muchos problemas los mínimos locales se encuentran cerca

Pseudocódigo de la búsqueda de vecindad variable

Seleccionar el conjunto de sistemas de vecinos $V_k, \quad k = 1, \ldots, k_{max}$
Elegir una solución inicial e_0
Repetir hasta que no se mejore
  Hacer $k = 1$
  Repetir hasta que $k = k_{max}$
    Encontrar la mejor solución e en $V_k(e_0)$
    Si la solución obtenida e es mejor que e_0 entonces $e_0 = e$ y $k = 1$ sino $k = k + 1$

VNS para TSP

Sistemas de vecinos

- Sucesión de sistemas de vecinos: 2-opt, 3-opt, 4-opt, etc.

Estrategias que aceptan soluciones peores

Características

- Una forma de escapar de óptimos locales es aceptando soluciones que empeoren el valor de la solución actual
- Sin embargo si no ponemos restricciones llegaríamos a una búsqueda aleatoria
- Existen básicamente dos alternativas:
  1. Algortimo de enfriamiento estadístico: se aceptan soluciones peores con cierta probabilidad que tiende a cero
  2. Búsqueda tabú: se guarda en memoria las soluciones visitadas

Simulated Annealing (algoritmo de enfriamiento estadístico)

Características

- Fué el primer metaheurístico, descubierto en los 80s
- Inspirado en el *annealing* de un sólido
- Dispone de muchos parámetros configurables que influyen de forma importante en el resultado final
- Se ha analizado de forma exhaustiva desde un punto de vista matemático, demostrándose su convergencia

Algoritmo de enfriamiento estadístico

repetir
  Elegir de forma aleatoria una solución e_{new} en $V(e_0)$
  $\delta = f(e_{new}) - f(e_0)$
  si $\delta < 0$ **entonces**
    $e_0 = e_{new}$
  sino
  elegir un número aleatorio *aleat* en $[0, 1]$
    si $e^{-\delta / c_k} > aleat$
      $e_0 = e_{new}$

Algoritmo de enfriamiento estadístico

Seleccionar una solución inicial e_0 de $\mathcal{E}$
Inicializar iteraciones num_iter=0
repetir
  tam_cadena=0
  repetir
    Elegir de forma aleatoria una solución e_{new} en $V(e_0)$
    $\delta = f(e_{new}) - f(e_0)$
    si $\delta < 0$ **entonces**
      $e_0 = e_{new}$
    sino
    elegir un número aleatorio *aleat* en $[0, 1]$
      si $e^{-\delta / c_k} > aleat$
        $e_0 = e_{new}$
    tam_cadena=tam_cadena+1
  hasta tam_cadena $>$ tam_max
  $c_{k+1} = w(c_k)$
  num_iter=num_iter+1
hasta num_iter $>$ iter_max

Algortimo de Enfriamiento Estadístico

Parámetros del algoritmo

- Valor inicial del parámetro de control
- Función de modificación del parámetro de control
- Tamaño de la cadena
- Criterio de parada (número de iteraciones, valor mínimo del parámetro de control)

Algortimo de Enfriamiento Estadístico

Establecimiento de los parámetros

- Esquema de enfriamiento
- Valor inicial del parámetro de control: se aceptan un % de soluciones alto
- Criterio de parada: cuando no se acepten soluciones que empeoran

Animación

http://www.biostat.jhsph.edu/∼iruczins/teaching/misc/annealing/animation.html

Búsqueda Tabú

Características básicas

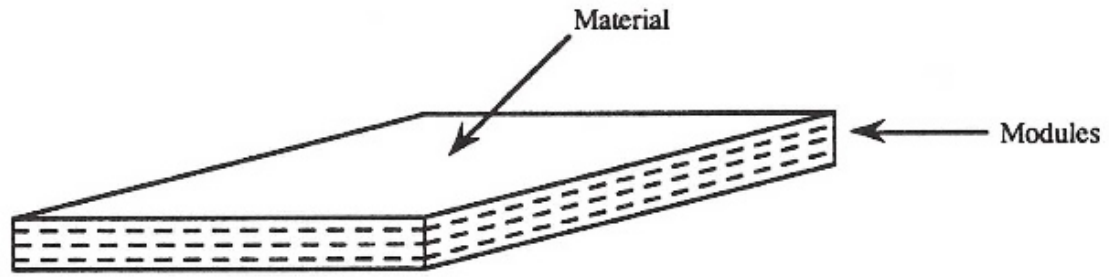
- Es un algoritmo de búsqueda local basado en el uso de memoria
- Es posible aceptar soluciones que empeoran la solución actual
- La memoria se utiliza para no repetir la trayectoria de búsqueda
- Existen dos tipos de memoria: memoria reciente y memoria a largo plazo
- En la memoria se guardan atributos de soluciones

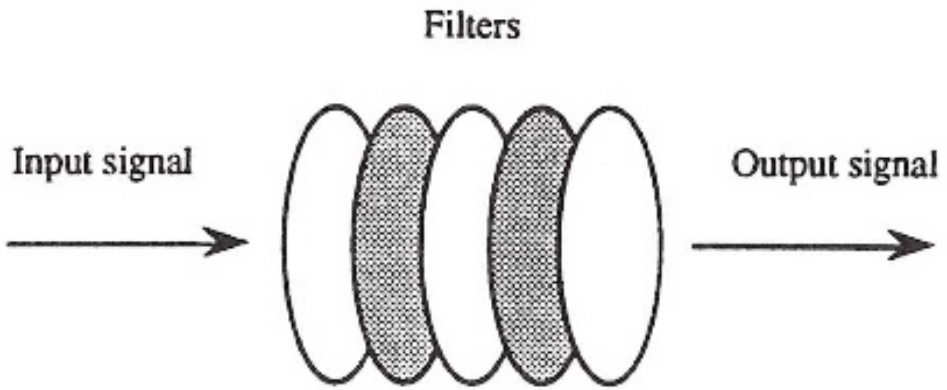
Ejemplo de Aplicación de la Búsqueda Tabú

Descripción del problema

- Optimizar la disposición de ciertos materiales de manera que se maximice el poder aislante
- La función objetivo viene dada por una caja negra
- Cada solución se representa como una permutación (disponemos de 7 materiales)

Ejemplo de Aplicación de la Búsqueda Tabú





Ejemplo de Aplicación de la Búsqueda Tabú

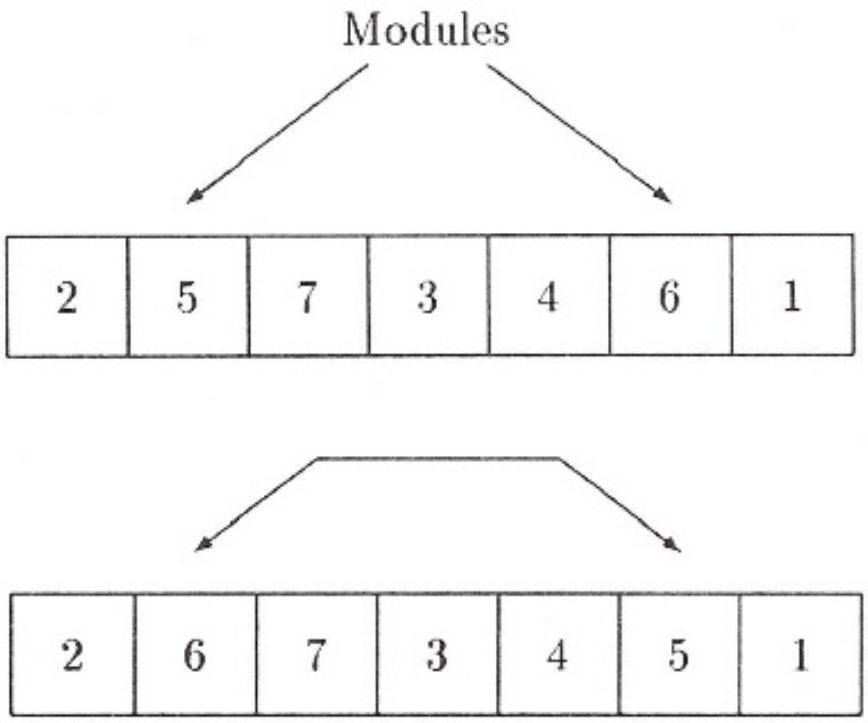
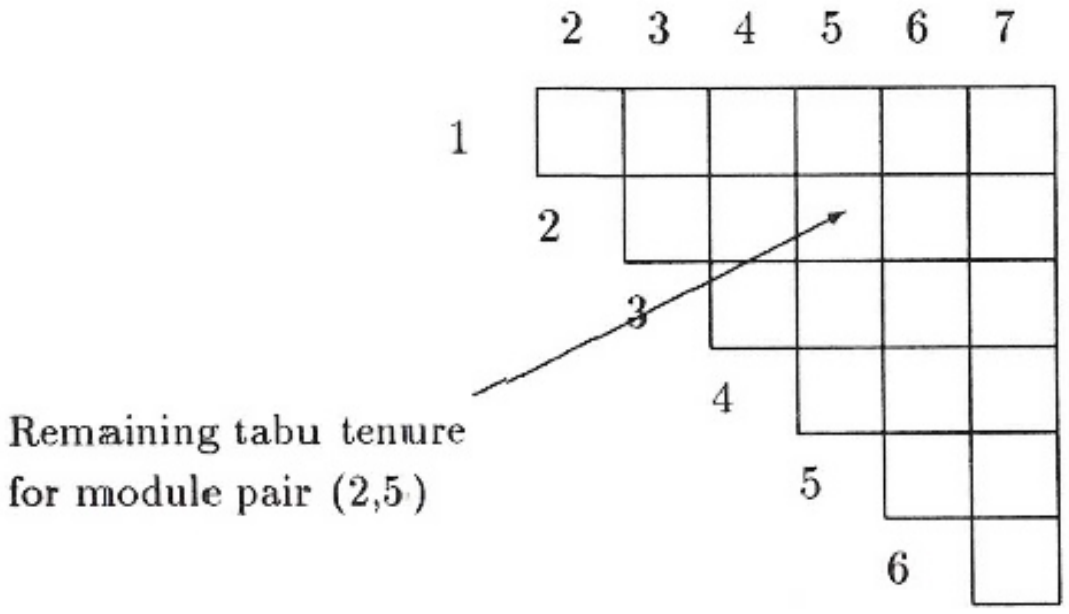


Figure 3.4: Swap of modules 5 and 6

Ejemplo de Aplicación de la Búsqueda Tabú



Ejemplo de Aplicación de la Búsqueda Tabú

Iteration 0 (*Starting point*)

Current solution

2	5	7	3	4	6	1

Insulation Value=10

Tabu structure

	2	3	4	5	6	7
1						
2						
3						
4						
5						
6						

All entries zero

Top 5 candidates

Swap	Value	
5,4	6	*
7,4	4	
3,6	2	
2,3	0	
4,1	-1	

Ejemplo de Aplicación de la Búsqueda Tabú

Iteration 1

Current solution

2	4	7	3	5	6	1

Insulation Value=16

Tabu structure

	2	3	4	5	6	7
1						
2						
3						
4	3					
5						
6						

Top 5 candidates

Swap	Value	
3,1	2	*
2,3	1	
3,6	-1	
7,1	-2	
6,1	-4	

Ejemplo de Aplicación de la Búsqueda Tabú

Iteration 2

Current solution

2	4	7	1	5	6	3

Insulation Value=18

Tabu structure

	2	3	4	5	6	7
1		3				
2						
3						
4		2				
5						
6						

Top 5 candidates

Swap	Value	
1,3	-2	T
2,4	-4	*
7,6	-6	
4,5	-7	T
5,3	-9	

Ejemplo de Aplicación de la Búsqueda Tabú

Iteration 3

Current solution

4	2	7	1	5	6	3

Insulation Value=14

Tabu structure

	2	3	4	5	6	7
1		2				
2			3			
3						
4				1		
5						
6						

Top 5 candidates

Swap	Value	
4,5	6	T*
5,3	2	
7,1	0	
1,3	-3	T
2,6	-6	

Ejemplo de Aplicación de la Búsqueda Tabú

Iteration 4

Current solution

5	2	7	1	4	6	3

Insulation Value=20

Tabu structure

	2	3	4	5	6	7
1		1				
2			2			
3						
4				3		
5						
6						

Top 5 candidates

Swap	Value	
7,1	0	*
4,3	-3	
6,3	-5	
5,4	-6	T
2,6	-8	

Ejemplo de Aplicación de la Búsqueda Tabú

Iteration 26
Current solution

1	3	6	2	7	5	4

Insulation Value=12

Tabu structure
(Recency)

	1	2	3	4	5	6	7
1				3			
2							
3	3					2	
4	1	5					1
5		4		4			
6			1		2		
7	2			3			

(Frequency)

Top 5 candidates

Swap	Value	Penalized Value	
1,4	3	3	T
2,4	-1	-6	
3,7	-3	-3	*
1,6	-5	-5	
6,5	-4	-6	

El Problema del Orden Lineal

Definición del problema y codificación de la solución

- Dada una matriz $n \times n$, $E = [e_{ij}]$, se trata de hallar la permutación filas y columnas p que maximiza la suma de los valores en la parte triangular superior:

$$C(p) = \sum_{j=1}^{n-1} \sum_{j=i+1}^{n} e_{p_i p_j}$$

- Una solución viene dada por una permutación $(p_1, \ldots, p_n)$

El Problema del Orden Lineal

Sistemas de vecinos, medida de influencia y atributos tabu

- Sistema de vecinos: se consideran n sistemas de vecinos, N^j con $j = 1, \ldots, n$
- N^j está compuesto de todas las soluciones que se consiguen al cambiar el j-ésimo índice a otra posición
- Medida de influencia: $w_j = \sum_{i \neq j} e_{ij} + e_{ji}$
- El índice j correspondiente al sistema de vecinos N^j elegido permanece tabú cierto número de iteraciones (además guardamos en $freq(j)$ el número de veces utilizado)

El Problema del Orden Lineal

Esquema básico de la búsqueda tabú

- Intensificación (uso de memoria reciente)
- Diversificación (uso de memoria basada en frecuencia)
- Estas fases se repiten MaxGlo iteraciones

El Problema del Orden Lineal

Intensificación

- Se elige un índice j con probabilidad proporcional a w_j
- Se halla la mejor solución en N^j
- Este proceso termina después de MaxInt iteraciones sin mejora
- Al final se aplica una búsqueda local que garantice un óptimo local

El Problema del Orden Lineal

Diversificación

- Se elige un índice j con probabilidad inversamente proporcional a $freq(j)$
- Se halla la mejor solución en N^j
- Este proceso se repite MaxDiv iteraciones

El Problema del Orden Lineal

Intensificación adicional: path relinking

- Se mantienen un conjunto de soluciones élite *EltSol*
- Al final de un periodo de intensificación se trata de modificar la solución obtenida con las contenidas en *EltSol*

El Problema del Orden Lineal

Diversificación a largo plazo

- Se calcula la posición media de cada índice j, $\alpha(j)$ en las soluciones élite y las soluciones visitadas en la última intensificación
- Cada índice se inserta en la posición $n - \alpha(j)$
- Este proceso se ejecuta cada MaxLong iteraciones globales sin mejora

El Problema del Orden Lineal: Resultados I

Table 3. Comparison of TS variants with LOLIB instances.				
	TS	**TS_PR**	**TS_LD**	**TS_LOP**
Obj. Function	22,040,159.4	22,040,160.9	22,041,257.7	22,041,261.5
Deviation	0.04%	0.04%	0.00%	0.00%
Num. of Opt.	30	30	44	47
CPU seconds	0.33	0.54	0.67	0.93

Table 4. Comparison of TS variants with Stanford GraphBase instances.				
	TS	**TS_PR**	**TS_LD**	**TS_LOP**
Obj. Function	6,032,093.76	6,032,546.88	6,033,122.75	6,033,124.09
Deviation	0.018%	0.010%	0.001%	0.001%
Num. of Best	35	40	59	66
CPU seconds	1.16	2.29	2.65	4.13

Table 5. Comparison of TS variants with random (0, 25000) instances.				
	TS	**TS_PR**	**TS_LD**	**TS_LOP**
Obj. Function	129,223,009	129,223,369	129,255,824	129,269,367.5
Deviation	0.065%	0.065%	0.038%	0.027%
Num. of Best	25	41	20	33
CPU seconds	10.79	17.94	13.07	20.19

El Problema del Orden Lineal: Resultados II

Table 6. LOLIB problems (49 instances).						
	Greedy	**Greedy-10**	**Becker**	**CK**	**CK-10**	**TS_LOP**
Value	22,033,729.49	22,038,090.39	20,375,556.16	22,018,008,35	22,040,892.14	22,041,261.51
Deviation	0.15%	0.02%	8.95%	0.15%	0.02%	0.00%
No. of Optimal	11	22	0	11	27	47
CPU seconds	0.01	0.08	0.02	0.10	1.06	0.93

Table 7. Stanford GraphBase problems (75 instances).						
	Greedy	**Greedy-10**	**Becker**	**CK**	**CK-10**	**TS_LOP**
Value	6,022,126.63	6,032,440.56	5,909,898.24	6,028,562.89	6,032,591.57	6,033,124.09
Deviation	0.18%	0.01%	2.04%	0.08%	0.01%	0.00%
No. of Best	3	20	0	4	22	70
CPU seconds	0.06	0.55	0.20	1.45	16.33	4.09

Table 8. Random (0, 25000) problems (75 instances).						
	Greedy	**Greedy-10**	**Becker**	**CK**	**CK-10**	**TS_LOP**
Value	128,729,161.6	128,981,141	125,587,971.7	128,663,947.3	128,919,838	129,269,367.5
Deviation	0.47%	0.23%	3.08%	0.53%	0.28%	0.00%
No. of Best	0	2	0	0	0	73
CPU seconds	0.12	1.21	0.80	10.67	108.44	20.19