DISEÑO Y ANÁLISIS DE ALGORITMOS

Algoritmos constructivos y búsquedas por entornos

Max-mean dispersion problem

J. Marcos Moreno-Vega
Christopher Expósito Izquierdo
Belén Melián Batista

Objetivo:
Proponer, implementar y evaluar algoritmos constructivos y búsquedas por entornos para el Max-Mean Dispersion Problem.

Tareas:
Además de las tareas descritas en el presente documento, los alumnos tendrán que realizar las modificaciones que se planteen durante la corrección de la práctica. Asimismo, tendrán que responder a un cuestionario de preguntas tipo test sobre los contenidos teóricos de los algoritmos constructivos y búsquedas por entornos.

Corrección:
Semana del 27 de abril al 1 de mayo.

Evaluación:
Código fuente y memoria: hasta 5 puntos; si el día de la corrección falta algún código o este es incorrecto, la práctica se calificará como No apta.

Modificación propuesta el día de la corrección: hasta 3 puntos.

Cuestionario sobre los contenidos teóricos: hasta 2 puntos.

Lenguaje de programación:
Java o C++.

Max-mean dispersion problem

Sea dado un grafo completo $G = (V, E)$, donde V es el conjunto de vértices ($|V| = n$) y E es el conjunto de aristas ($|E| = n(n-1)/2$. Cada arista $(i, j) \in E$ tiene asociada una distancia o afinidad $d(i, j)$. En el max-mean dispersion problem se desea encontrar el subconjunto de vértices $S \subseteq V$ que maximiza la dispersión media dada por

$$md(S) = \frac{\sum_{i,j \in S} d(i, j)}{|S|}$$

Un constructivo voraz

Un algoritmo constructivo voraz muy sencillo para este problema parte del subconjunto, S, formado por los vértices conectados por la arista con mayor afinidad. A continuación, añade a este subconjunto, iterativamente y mientras sea posible, el vértice que mayor incremento positivo produce en la función objetivo. El pseudocódigo de este algoritmo se muestra a continuación.

Algoritmo constructivo voraz

1: Seleccionar la arista (i, j) con mayor afinidad;
2: S = $\{i, j\}$;
3: **repeat**
4: $S^* = S$;
5: Obtener el vértice k que maximiza $md(S \cup \{k\})$;
6: **if** $md(S \cup \{k\}) \geq md(S)$ **then**
7: $S = S \cup \{k\}$;
8: **until** $(S^* = S)$
9: Devolver S^*;

Figura 1: Algoritmo constructivo voraz

Implementación

Las instancias del problema se suministrarán en un fichero de texto con el siguiente formato: en la primera fila se encuentra el número de vértices, n; a continuación, se enumeran las afinidades, $d(i, j)$, entre los pares de vértices (se asume que las afinidades son simétricas, es decir, que $d(i, j) = d(j, i)$, $\forall i, j \in V$. Además, $d(i, i) = 0$, $\forall i \in V$).

Por ejemplo, si $n = 4$, un fichero de texto para el problema podría contener los siguiente datos (solo la primera columna; la segunda describe qué representa cada dato):

4	número de vértices
-3,90	$d(1,2) = d(2,1)$
-4,27	$d(1,3) = d(3,1)$
5,66	$d(1,4) = d(4,1)$
5,95	$d(2,3) = d(3,2)$
9,53	$d(2,4) = d(4,2)$
-6,91	$d(3,4) = d(4,3)$

Tareas

a) Diseñar e implementar el algoritmo constructivo voraz descrito en la figura 1.

b) Diseñar e implementar un nuevo algoritmo voraz para el problema.

c) Diseñar e implementar un GRASP para el problema.

d) Diseñar e implementar un Método Multiarranque para el problema.

e) Diseñar e implementar una Búsqueda por Entorno Variable para el problema.

Qué debe presentar el alumno

a) Código fuente, debidamente comentado, y fichero ejecutable.

b) Una memoria en formato pdf en la que se describan brevemente los algoritmos diseñados enumerando las estructuras de datos usadas, las estructuras de entorno empleadas en las correspondientes búsquedas y cualquier elemento necesario para comprender el diseño propuesto.

c) Tablas o gráficas de resultados que muestren el comportamiento de los algoritmos sobre diferentes instancias del problema. A continuación se muestran un modelo de tablas de resultados que el alumno puede usar. No obstante, se trata solo de un modelo que puede modificarse si se cree que otro es mejor.

Algoritmo voraz

Problema	n	Ejecución	md	CPU
ID_1		1		
ID_1		2		
ID_1		3		
ID_1		4		
ID_1		5		
ID_2		1		
ID_2		2		
ID_2		3		
ID_2		4		
ID_2		5		
...	...	...	...	...

Tabla 1: Algoritmo voraz. Tabla de resultados

Multiarranque

Problema	n	Ejecución	md	CPU
ID_1		1		
ID_1		2		
ID_1		3		
ID_1		4		
ID_1		5		
ID_2		1		
ID_2		2		
ID_2		3		
ID_2		4		
ID_2		5		
...	...	...	...	...

Tabla 2: Multiarranque. Tabla de resultados

GRASP					
Problema	n	$\|LRC\|$	Ejecución	md	CPU
ID_1		2	1		
ID_1		2	2		
ID_1		2	3		
ID_1		2	4		
ID_1		2	5		
ID_1		3	1		
ID_1		3	2		
ID_1		3	3		
ID_1		3	4		
ID_1		3	5		
ID_2		2	1		
ID_2		2	2		
ID_2		2	3		
ID_2		2	4		
ID_2		2	5		
ID_2		3	1		
ID_2		3	2		
ID_2		3	3		
ID_2		3	4		
ID_2		3	5		
...	...	...	...	...	...

Tabla 3: GRASP. Tabla de resultados

VNS					
Problema	m	k_{max}	Ejecución	md	CPU
ID_1		2	1		
ID_1		2	2		
ID_1		2	3		
ID_1		2	4		
ID_1		2	5		
ID_1		3	1		
ID_1		3	2		
ID_1		3	3		
ID_1		3	4		
ID_1		3	5		
ID_2		2	1		
ID_2		2	2		
ID_2		2	3		
ID_2		2	4		
ID_2		2	5		
ID_2		3	1		
ID_2		3	2		
ID_2		3	3		
ID_2		3	4		
ID_2		3	5		
...	...	...	...	...	...

Tabla 4: VNS. Tabla de resultados