Grafos Eulerianos y Hamiltonianos

Algoritmos y Estructuras de Datos III

Grafos eulerianos

- Un circuito C en un grafo (o multigrafo) G es un circuito euleriano si C pasa por cada arista de G exactamente una vez.
- Un grafo (o multigrafo) se dice euleriano si tiene un circuito euleriano.

Teorema (Euler, 1736 + Hierholzer, 1871). Un grafo (o multigrafo) conexo es euleriano si y sólo si todos sus vértices tienen grado par.

A partir de la demostración, se puede escribir un algoritmo para construir un circuito euleriano para un grafo que tiene todos sus vértices de grado par.

Grafos eulerianos

Entrada: $G = (V, E)$ conexo con $d(v)$ par para todo $v \in V$.
Salida: Un circuito euleriano de G.

- Comenzar por cualquier vértice v y construir un ciclo Z.
- **Mientras** $E \setminus Z \neq \emptyset$ **hacer**:
  - Elegir w tal que exista $(w, u) \in Z$ y $(w, z) \in E \setminus Z$.
  - Desde w construir un ciclo D con $D \cap Z = \emptyset$.
  - $Z :=$ unir Z y D por medio de w.
- **Fin mientras**
- **Retornar** Z.

Grafos eulerianos

- Un camino euleriano en un grafo (o multigrafo) G es un camino que pasa por cada arista de G exactamente una vez.
- Un grafo orientado o digrafo, se dice euleriano si tiene un circuito orientado que pasa por cada arco de G exactamente una vez.

Teorema. Un grafo (o multigrafo) conexo tiene un camino euleriano si y sólo si tiene exactamente dos vértices de grado impar.

Teorema. Un digrafo conexo es euleriano si y sólo si para todo vértice v de G se verfica que $d_{in}(v) = d_{out}(v)$.

Problema del cartero chino (Guan, 1962)

Definición. Dado un grafo $G = (V, E)$ con longitudes asignadas a sus aristas, $\ell : E \to \mathbb{R}_+$, el problema del cartero chino consiste en encontrar un circuito de longitud mínima que pase por cada arista de G **al menos** una vez.

- Si G es euleriano, un circuito euleriano es la solución del problema del cartero chino.
- Existen algoritmos polinomiales para el problema del cartero chino cuando G es orientado o no orientado.
- Sin embargo, no se conocen algoritmos polinomiales si el grafo es mixto (tiene tanto aristas orientadas como aristas no orientadas).

Grafos hamiltonianos

- Un circuito en un grafo G es un circuito hamiltoniano si pasa por cada vértice de G una y sólo una vez.
- Un grafo se dice hamiltoniano si tiene un circuito hamiltoniano.

Grafos hamiltonianos

Teorema (condición necesaria). Sea G un grafo conexo. Si existe $W \subset V$ tal que $G \setminus W$ tiene c componentes conexas con $c > |W|$ entonces G no es hamiltoniano.

¿Es cierta la recíproca de este teorema?

Teorema (Dirac, 1952) (condición suficiente). Sea G un grafo con $n \geq 3$ y tal que para todo $v \in V$ se verifica que $d(v) \geq n/2$. Entonces G es hamiltoniano.

¿Es cierta la recíproca de este teorema?

Grafos hamiltonianos

- No se conocen condiciones necesarias y suficientes que caractericen en forma "elegante" a los grafos hamiltonianos.
- No se conocen algoritmos polinomiales para determinar si un grafo es hamiltoniano o no (algoritmos de reconocimiento).
- Más aún, se sospecha que **no existen** (!) algoritmos polinomiales para este problema (¿cómo se demuestra esto?).

El problema del viajante de comercio (TSP)

Entrada: Un grafo $G = (V, E)$ completo y una función de distancias $\ell : E \to \mathbb{R}_+$.

Salida: Un circuito hamiltoniano $C \subseteq E$ que minimice la distancia total $\ell(C) = \sum_{ij \in C} \ell(ij)$.

- Se trata de una generalización del problema de camino hamiltoniano (¿por qué?).
- Como consecuencia, no se conocen algoritmos polinomiales para resolver el TSP (¿¿por qué??).
- **Pausa filosófica**: ... entonces qué hacemos?

El problema del viajante de comercio (TSP)

Opciones inmediatas:

Si las instancias que tenemos que resolver no son muy grandes ...

- Un esquema de fuerza bruta puede no ser mala idea.
- En caso contrario, intentar con un backtracking.

Si las instancias hacen imposibles estos procedimientos ...

- Intentar una heurística golosa.
- Generar aleatoriamente muchas soluciones y quedarse con la mejor!
- Variantes más sofisticadas: búsqueda local, búsqueda tabú, simulated annealing, etc.

Algoritmos heurísticos

- Un algoritmo heurístico (también llamado una heurística) es un algoritmo que no garantiza una respuesta exacta para el problema en cuestión.
  1. Heurísticas *ad hoc* o "clásicas".
  2. Metaheurísticas.
- ¿Cuándo es conveniente recurrir a una heurística?
  1. Problemas para los cuales no se conocen algoritmos exactos eficientes.
  2. Problemas difíciles de modelar.
- ¿Cómo se evalúa una heurística?
  1. Bancos de prueba.
  2. Cotas inferiores.
  3. Garantías de optimalidad.

El problema del viajante de comercio (TSP)

Hipótesis: Las distancias cumplen la desigualdad triangular: $\ell(ij) + \ell(jk) \geq \ell(ik)$ para todo $i, j, k \in V$.

- Obtener un árbol generador mínimo $T = (V_T, E_T)$ de G.
- Duplicar las aristas de E_T, obteniendo un nuevo árbol T'.
- Encontrar un circuito euleriano C en T' (siempre existe!).
- Transformar C en un circuito hamiltoniano salteando vértices ya visitados.

El problema del viajante de comercio (TSP)

Teorema. Si ℓ_{min} es la longitud de la solución óptima del TSP para la instancia (G, ℓ) y ℓ_{heur} es la longitud de la solución generada por la heurística anterior, entonces

$$\frac{\ell_{\text{heur}}}{\ell_{\text{min}}} \leq 2.$$

- En virtud de este teorema, decimos que esta heurística es un algoritmo 2-aproximado.
- ¿Se puede mejorar?

Heurística de Cristofides (1976)

- Obtener un árbol generador mínimo $T = (V_T, E_T)$ de G.
- Sea $I \subseteq V_T$ el conjunto de vértices con grado impar en T. Encontrar un matching perfecto de peso mínimo M en el subgrafo de G inducido por I.
- Combinar las aristas de M y T para formar un multigrafo H.
- Encontrar un circuito euleriano C en H (siempre existe!).
- Transformar C en un circuito hamiltoniano salteando vértices ya visitados.

Heurística de Cristofides (1976)

Teorema (Cristofides, 1976). Si ℓ_{min} es la longitud de la solución óptima del TSP para la instancia (G, ℓ) y ℓ_{heur} es la longitud de la solución generada por la heurística anterior, entonces

$$\frac{\ell_{\text{heur}}}{\ell_{\text{min}}} \leq 3/2.$$

- ¿Se puede mejorar?
- Si las distancias ℓ son euclídeas en el plano, entonces existe un algoritmo $(1 + 1/c)$-aproximado con complejidad $O(n(\log n)^{O(c\sqrt{2})})$.

El problema del viajante de comercio (TSP)

Desde el punto de vista de algoritmos exactos, el enfoque más exitoso a la fecha está dado por algoritmos basados en programación lineal entera.

Año	Equipo	Ciudades
1954	G. Dantzig, R. Fulkerson y S. Johnson	49
1971	M. Held y R.M. Karp	64
1975	P. M. Camerini, L. Fratta y F. Maffioli	100
1977	M. Grötschel	120
1980	H. Crowder y M. W. Padberg	318
1987	M. Padberg y G. Rinaldi	532
1987	M. Grötschel y O. Holland	666
1987	M. Padberg y G. Rinaldi	2,392
1994	D. Applegate, R. Bixby, V. Chvátal y W. Cook	7,397
1998	D. Applegate, R. Bixby, V. Chvátal y W. Cook	13,509
2001	D. Applegate, R. Bixby, V. Chvátal y W. Cook	15,112
2004	D. Applegate, R. Bixby, V. Chvátal y W. Cook	24,978
2005	D. Applegate, R. Bixby, V. Chvátal y W. Cook	33,810
2006	D. Applegate, R. Bixby, V. Chvátal y W. Cook	85,900

Algoritmos de búsqueda local

- Recordemos los elementos de un problema de optimización:
  1. Conjunto S de soluciones factibles, habitualmente definido en forma implícita.
  2. Función objetivo $f : S \to \mathbb{R}$ a optimizar.
- Dada una solución factible $s \in S$, podemos considerar modificaciones pequeñas a la solución, que generan una nueva solución factible levemente modificada.
- Estas modificaciones se denominan movimientos, y las soluciones obtenidas a partir de s forman el vecindario de s con relación al movimiento considerado.

Algoritmos de búsqueda local

- **Ejemplo.** Consideremos el problema de programar tareas independientes en dos máquinas:
  1. **Datos:** Cantidad k de máquinas, conjunto $T = \{1, \ldots, n\}$ de tareas, y tiempo $t_i \in \mathbb{R}_+$ de procesamiento de cada tarea $i \in T$.
  2. **Objetivo:** Determinar en qué máquina se realiza cada tarea y el orden entre las tareas de cada máquina, de modo tal de minimizar el tiempo de finalización de la última tarea.
- El orden entre las tareas no es importante! Posibles movimientos:
  1. Cambiar la máquina de una tarea (y ponerla al final de la nueva máquina).
  2. Intercambiar dos tareas de máquinas distintas.

Algoritmos de búsqueda local

- Un algoritmo de búsqueda local está dado por los siguientes pasos:
  1. Construir una solución factible $s \in S$.
  2. Si existe una solución $s' \in N(s)$ con $f(s') < f(s)$, asignar $s := s'$ y repetir este paso.
  3. En caso contrario, retornar s.
- La solución retornada no tiene ningún vecino con mejor función objetivo, y se denomina un óptimo local.
- No necesariamente es la solución óptima!

Algoritmos de búsqueda local

- Para implementar un algoritmo de búsqueda local, se deben definir los siguientes elementos:
  1. Cómo se representa una solución factible.
  2. Cómo se construye la primera solución factible del algoritmo.
  3. Qué vecindario se utiliza durante el algoritmo.
- Este tipo de algoritmos se denomina una metaheurística. Una vez definidos estos detalles, tenemos una heurística para el problema en cuestión.
- **Ejemplo.** Búsqueda local para el problema de programación de tareas independientes en procesadores iguales.
  1. Construimos una solución asignando aleatoriamente una máquina a cada tarea.
  2. Como movimiento para generar el vecindario, cambiamos una tarea de una máquina a otra.

Algoritmos de búsqueda local

- Si buscar el mejor vecino tiene una complejidad demasiado alta, se puede interrumpir **mejorVecino()** cuando se encuentra un vecino mejor que la solución actual.
- Se puede usar más de un vecindario en secuencia. Cuando la solución es un óptimo local para el primer vecindario, se continúa con el segundo vecindario y viceversa.
- GRASP (*greedy randomized adaptive search procedure*). Ejecutar la búsqueda local más de una vez, si la construcción de la solución inicial es aleatoria o se puede cambiar en forma arbitraria.

Ejemplo: VRP

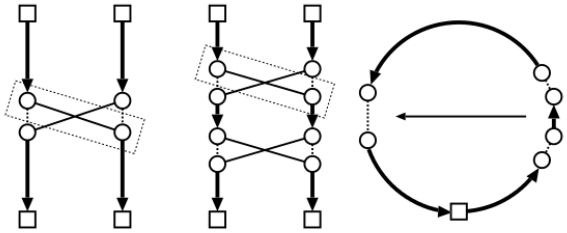
- Recordemos la definición del problema de ruteo de vehículos:

  Ruteo de vehículos (VRP)
  1. **Entrada:** Un conjunto de n clientes con su matriz de distancias $A \in \mathbb{R}^{n \times n}$. Una cantidad c_i a entregar en cada cliente $i = 1, \ldots, n$. Una cantidad m de camiones y la capacidad C de cada camión. Un número real $k \in \mathbb{R}$.
  2. **Salida:** ¿Existe un conjunto de m rutas que pase por todos los clientes, tal que la suma de las capacidades de los clientes en cada ruta no excede la capacidad C del camión, y tal que la distancia total recorrida es menor o igual que k?

- ¿Cómo está definida una solución factible?
- ¿Cómo se puede definir un vecindario para este problema?

Ejemplo: VRP

- Posibles vecindarios:



(a) 2-opt* (b) Cross exchange (c) Intra-route