Grafos Eulerianos y Hamiltonianos

Algoritmos y Estructuras de Datos III

Grafos eulerianos

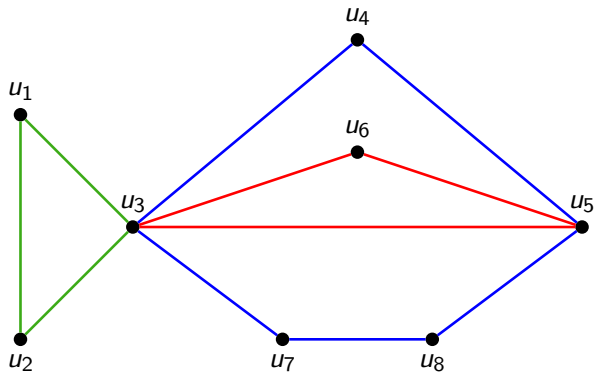
Definiciones:

- Un circuito C en un grafo (o multigrafo) G es un **circuito euleriano** si C pasa por todos las aristas de G una y sólo una vez.
- Un **grafo euleriano** es un grafo que tiene un circuito euleriano (o multigrafo).

Teorema: Un grafo (o multigrafo) conexo es euleriano si y sólo si todos sus nodos tienen grado par.

A partir de la demostración del teorema de Euler se puede escribir un algoritmo para construir un circuito euleriano para un grafo que tiene todos sus nodos de grado par.

Grafos eulerianos



Grafos eulerianos

Entrada: $G = (V, X)$ conexo con todos los nodos de grado par.

`comenzar por cualquier nodo` v `y construir un ciclo` Z
mientras `exista` $e \in X \setminus Z$ **hacer**
    `elegir` w `tal que existe` $(w, u) \in Z$ `y` $(w, z) \in X \setminus Z$
    `desde` w `construir un ciclo` D `con` $D \cap Z = \emptyset$
    $Z :=$ `unir` Z `y` D `por medio de` w
fin mientras
retornar Z

¿Cuál es la complejidad de este algoritmo?

Grafos eulerianos

Definiciones:

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

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

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

Problema del cartero chino (Guan, 1962)

Definición: Dado un grafo $G = (V, X)$ con longitudes asignadas a sus aristas, $l : X \to \mathbb{R}^{\geq 0}$, el **problema del cartero chino** consiste en encontrar un circuito que pase por cada arista de G **al menos** una vez de longitud mínima.

- Si G es euleriano, un circuito euleriano es la solución del problema del cartero chino.
- Hay algoritmos polinomiales para el problema del cartero chino cuando G es orientado o no orientado.
- Pero no se conocen algoritmos polinomiales (el problema no está computacionalmente resuelto) si el grafo es mixto (algunas aristas orientados y otros no).

Grafos hamiltonianos

Definiciones:

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

No se conocen buenas caracterizaciones para grafos hamiltonianos.

¿Cómo intentar construir un circuito hamiltoniano?

No se conocen algoritmos polinomiales para decidir si un grafo es hamiltoniano o no.

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) (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?

Metaheurísticas

- Heurísticas clásicas.
- Metaheurísticas o heurísticas "modernas".

¿Cuándo usarlas?

- Problemas para los cuales no se conocen buenos algoritmos exactos.
- Problemas difíciles de modelar.

¿Cómo se evalúan?

- Problemas test.
- Problemas reales.
- Problemas generados al azar.
- Cotas.

Problema del viajante de comercio (TSP)

Definición: Dado un grafo $G = (V, X)$ con longitudes asignadas a las aristas, $l : X \to \mathbb{R}^{\geq 0}$, queremos determinar un circuito hamiltoniano de longitud mínima.

- No se conocen algoritmos polinomiales para resolver el problema del viajante de comercio.
- Tampoco se conocen algoritmos ϵ-aproximados polinomiales para el TSP general (si se conocen cuando las distancias son euclideanas).
- Es **el** problema de optimización combinatoria más estudiado.

Heurísticas y algoritmos aproximados para el TSP

Heurística del vecino más cercano

elegir un nodo v
$orden(v) := 0$
$S := \{v\}$
$i := 0$
mientras $S \neq V$ **hacer**
  $i := i + 1$
  elegir la arista (v, w) *más barata* con $w \notin S$
  $orden(w) := i$
  $S := S \cup \{w\}$
  $v := w$
fin mientras
retornar *orden*

¿Cuál es la complejidad de este algoritmo?

Heurísticas y algoritmos aproximados para el TSP

Heurísticas de inserción

$C :=$ un circuito de longitud 3
$S :=$ {nodos de C}
mientras $S \neq V$ **hacer**
  ELEGIR un nodo $v \notin S$
  $S := S \cup \{v\}$
  INSERTAR v en C
fin mientras
retornar C

¿Cómo ELEGIR?

$\rightsquigarrow$ **variantes de la heurística de inserción**

¿Cómo INSERTAR?

Heurísticas y algoritmos aproximados para el TSP

Heurísticas de inserción

Para INSERTAR el nodo v elegido:

- Sea $c_{v_i v_j}$ es el costo o la longitud de la arista (v_i, v_j).
- Elegimos dos nodos consecutivos en el circuito v_i, v_{i+1} tal que

$$c_{v_i v} + c_{v v_{i+1}} - c_{v_i v_{i+1}}$$

  sea mínimo.
- Insertamos v entre v_i y v_{i+1}.

Heurísticas y algoritmos aproximados para el TSP

Heurísticas de inserción

Podemos ELEGIR el nuevo nodo v para agregar al circuito tal que:

- v sea el nodo más próximo a un nodo que ya está en el circuito.
- v sea el nodo más lejano a un nodo que ya está en el circuito.
- v sea el nodo *más barato*, o sea el que hace crecer menos la longitud del circuito.
- v se elige al azar.

Heurísticas y algoritmos aproximados para el TSP

Heurísticas de inserción

En el caso de grafos euclideanos (por ejemplo grafos en el plano $\mathbb{R}^2$), se puede implementar un algoritmo de inserción:

- Usando la cápsula convexa de los nodos como circuito inicial.
- Insertando en cada paso un nodo v tal que el ángulo formado por las aristas (w, v) y (v, z), con w y z consecutivos en el circuito ya construido, sea máximo.

Hay muchas variantes sobre estas ideas.

Heurísticas y algoritmos aproximados para el TSP

Heurística del árbol generador mínimo (G grafo completo)

`encontrar un árbol generador mínimo` T `de` G

`construir` T' `duplicando las aristas de` T

`recorrer` T' `usando` **DFS** `y armar un circuito hamiltoniano de` G

¿Cuál es la complejidad de este algoritmo?

Heurísticas y algoritmos aproximados para el TSP

Heurística del árbol generador mínimo

Teorema: Si las distancias del grafo G cumplen la desigualdad triangular, la heurística del árbol generador mínimo es un algoritmo aproximado con una perfomance en el peor caso dada por

$$I(C^H)/I(C^*) = X^H(G)/X^*(G) \leq 2$$

O sea, si las distancias son euclideanas hay algoritmos polinomiales para el problema del TSP aproximado.

Heurísticas y algoritmos aproximados para el TSP

Perfomance de otros algoritmos aproximados en el peor caso

Si las distancias de G son euclideanas se puede probar que valen las siguientes cotas para la perfomance en el peor caso:

Vecino más próximo	$X^H(G)/X^*(G) \leq 1/2(\lceil \log n \rceil + 1)$
Inserción del más próximo	$X^H(G)/X^*(G) \leq 2$
Inserción del más lejano	$X^H(G)/X^*(G) \leq 2 \log n + 0{,}16$
Inserción del más barato	$X^H(G)/X^*(G) \leq 2$

Heurísticas y algoritmos aproximados para el TSP

Heurísticas de mejoramiento - Algoritmos de búsqueda local

¿Cómo podemos mejorar la solución obtenida por alguna heurística constructiva como las anteriores?

Heurística 2-opt de Lin y Kernighan

obtener una solución inicial H
por ejemplo con alguna de las heurísticas anteriores
mientras sea posible **hacer**
elegir (u_i, u_{i+1}) y $(u_k, u_{k+1}) \in H$ tal que
$c_{u_i u_{i+1}} + c_{u_k u_{k+1}} > c_{u_i u_k} + c_{u_{i+1} u_{k+1}}$
$H := H \setminus \{(u_i, u_{i+1}), (u_k, u_{k+1})\} \cup \{(u_i, u_k), (u_{i+1}, u_{k+1})\}$
fin mientras

¿Cuándo para este algoritmo? ¿Se obtiene la solución óptima del TSP de este modo?

Heurísticas y algoritmos aproximados para el TSP

Heurísticas de mejoramiento - Algoritmos de búsqueda local

- En vez de elegir para sacar de H un par de aristas cualquiera que nos lleve a obtener un circuito de menor longitud podemos elegir, entre todos los pares posibles, el par que nos hace obtener el menor circuito (más trabajo computacional).
- Esta idea se extiende en las heurísticas k-opt donde se hacen intercambios de k aristas. Es decir, en vez de sacar dos aristas, sacamos k aristas de H y vemos cual es la mejor forma de reconstruir el circuito. En la práctica se usa sólo 2-opt o 3-opt.

Algoritmos de descenso o búsqueda local

Esquema general

S = conjunto de soluciones
$N(s)$ = soluciones "vecinas" de la solución s
$f(s)$ = valor de la solución s

`elegir una solución inicial` $s^* \in S$
repetir
    `elegir` $s \in N(s^*)$ `tal que` $f(s) < f(s^*)$
    `reemplazar` s^* `por` s
hasta que $f(s) > f(s^*)$ `para todos los` $s \in N(s^*)$

Algoritmos de descenso o búsqueda local

- ¿Cómo determinar las soluciones vecinas de una solución s dada?
- ¿Qué se obtiene con este procedimiento? ¿Sirve?
- Óptimos locales y globales
- Espacio de búsqueda

Tabu Search

- Objetivo: minimizar una función f sobre un conjunto de soluciones S.
- Metaheurística que guía una heurística de búsqueda local para explorar el espacio de soluciones evitando los óptimos locales.
- Iterativamente se mueve de una solución a otra hasta que se cumple algún criterio de terminación.
- Cada $s \in S$ tiene asociada una vecindad $N(s) \subseteq S$ y cada solución $s' \in N(s)$ es alcanzada desde s realizando **movimientos**.

Tabu Search

- Explorar todo $N(s)$ puede ser impracticable computacionalmente.
- Restringe la búsqueda a $V^* \subset N(s)$ con $|V^*| \ll |N(s)|$.
- Usa **memoria** para definir V^*.
- Permite moverse de s a s' aun si $f(s') > f(s)$ para salir de un óptimo local.
- Se pueden generar ciclos.
- **Lista tabú:** memoriza soluciones y vecindades consideradas en iteraciones anteriores.

Tabu Search - Lista Tabú

- Memoriza las $|T|$ últimas soluciones visitadas.
- Memoriza movimientos reversos asocidos con los movimientos hechos.
- Clasifica como tabú ciertos atributos de las soluciones.

- Memoria "a corto plazo".
- Tamaño y permanencia en T.
- Almacenar atributos o movimientos es más efectivo.
- Puede hacer que soluciones no visitadas sean tabú.
- **Función de aspiración:**
  - cuando con un movimiento tabú se obtiene una solución mejor que la mejor hasta ese momento, se permite elegirla.
  - cuando todos los movimientos o vecinos posibles son tabú, se elige alguno de ellos ("el menos tabu").

Tabu Search - Esquema general

elegir una solución inicial $s_0 \in S$
inicializar la lista tabú T
inicializar la función de aspiración A
mientras no se verifique el criterio de parada **hacer**
    generar $V^* \subseteq \{s \in N(s_0) : s$ no es tabu o $A(s) \leq \bar{A}(s^*)\}$
    elegir $s_0 \in V^*$ tal que $f(s_0) \leq f(s)\ \forall s \in V^*$
    actualizar la función de aspiración A
    actualizar la lista tabú T
    si $f(s_0) < f(s^*)$ **entonces**
        $s^* := s_0$
    fin si
fin mientras
retornar s^*

Tabu Search

¿Qué hay que hacer para usar este esquema?

- Determinar el conjunto de soluciones factibles S.
- Determinar la función objetivo f.
- Dar un procedimiento para generar los elementos de $N(s)$.
- Decidir el tamaño del conjunto $V^* \subseteq N(s)$ que será considerado en cada iteración.
- Definir la lista Tabú T y su tamaño.
- De ser posible definir una cota inferior para la función objetivo f.
- Definir la función de aspiración $A(z)$ y el umbral de aceptación.
- Definir criterios de parada.

Tabu Search

Los criterios de parada más simples son:

- Se encontró una solución óptima (si es posible saberlo).
- $\{s \in N(s_0) : s$ no es tabu o $A(s) \leq \bar{A}(s^*)\} = \emptyset$.
- Se alcanzó el número máximo de iteraciones permitidas.
- El número de iteraciones realizadas sin modificar s^* es mayor que un número máximo determinado.
- Cualquier combinación de los anteriores.

Tabu Search - Mejoras

Uso de la memoria "a largo plazo":

- Frecuencia: guarda la frecuencia de ocurrencias de atributos en las soluciones visitadas para penalizar o premiar (según convenga) movimientos que usan atributos muy usados en el pasado.
- Intensificación: intensifica la búsqueda en alguna región de S porque es considerada buena bajo algún criterio.
- Diversificación: explora nuevas regiones de S no exploradas.
- Camino de soluciones entre dos soluciones prometedoras.
- Etc.

Tabu Search - Ejemplo: Viajante de comercio

Solución inicial: dada por alguna heurística o al azar.
Espacio de soluciones: permutaciones de $(1, 2, \ldots, n)$.
Tamaño del espacio: $(n-1)!/2$.
Movimientos: k intercambios, por ejemplo $k = 2$.
Cardinal de $N(s)$, $|N(s)|$: $n(n-1)/2$.

Es fácil generar vecinos al azar y actualizar el costo.

Con estas definiciones se puede usar el esquema básico.