VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA

Hornicko – geologická fakulta

Katedra geoinformatiky

PROSTOROVÁ ANALÝZA DAT

Síťové analýzy v Pgrouting – část 1.

Lucie Orlíková

Ostrava, 2020

ÚVOD DO SÍŤOVÝCH ANALÝZ

Aparát teorie grafů je důležitým prostředkem pro zachycení a analýzu struktury systému, algoritmy optimalizace na grafech slouží k řešení rozsáhlé třídy matematických modelů operačního výzkumu. Jednou z nejčastěji vyhledávaných skupin algoritmů s širokým uplatněním zejména v oblasti silniční dopravy jsou algoritmy pro hledání optimálních cest na grafech. Také další oblasti teorie grafů (konstrukční úlohy na grafech, metody hledání kritické cesty, problematika okružních jízd apod.) mohou sloužit jako kvalitní teoretická základna řešení problémů z oblasti dopravy. Většina algoritmů má heuristický charakter.

Rozeznáváme tři základní typy úloh o hledání optimálních cest. Pro všechny typy úloh předpokládáme neorientovaný, souvislý, hranově ohodnocený graf, který představuje schematické znázornění dopravní sítě.

Úlohy o hledání nejkratší cesty (někdy je nejkratší cesta označována také jako minimální cesta) mohou být dále rozčleněny takto:

- hledání nejkratší cesty z daného počátečního vrcholu do daného koncového vrcholu
- hledání nejkratší cesty z daného počátečního vrcholu do všech ostatních vrcholů grafu (popř. hledání nejkratší cesty ze všech ostatních vrcholů do daného koncového vrcholu)
- hledání minimální cesty mezi libovolnými dvěma vrcholy grafu

K řešení prvních dvou typů úloh se používají Dijkstrovy algoritmy. Pro hledání minimální cesty mezi libovolnými dvěma vrcholy se užívá Floydův algoritmus. Jeho výsledkem je matice vzdáleností mezi vrcholy (distanční matice), s využitím této matice a matice přímých vzdáleností lze pak snadno určit i minimální cestu mezi vybranými dvěma vrcholy. Algoritmy pro hledání nejkratší cesty mají v silniční dopravě rozsáhlé použití. Podle charakteru úlohy může být graf hranově ohodnocen vzdáleností mezi vrcholy, náklady na přepravu apod.

Hledání nejspolehlivější cesty

Pro hledání nejspolehlivější cesty se využívá algoritmus hledání nejkratší cesty z počátečního do koncového vrcholu. Hrany grafu jsou ohodnoceny pravděpodobnostmi úspěšného nebo neúspěšného průchodu příslušnou hranou. Při ohodnocení pravděpodobností neúspěšného průchodu hranou je třeba ji přepočítat na pravděpodobnost úspěšného průchodu hranou, algoritmus uvažuje právě tuto pravděpodobnost úspěchu. V silniční dopravě se může jednat např. o pravděpodobnost, s jakou na daném úseku komunikace nedojde k nehodě, pravděpodobnost, že nenastane krizová situace apod.

Hledání cesty s maximální kapacitou

Pro hledání cesty s maximální kapacitou se používá známý algoritmus využívající řezových množin a krácení hran. Algoritmus pro hledání cesty s maximální kapacitou lze v silniční dopravě použít především při hledání trasy pro přepravu nadrozměrných nákladů.

Předchozí typy úloh lze také navzájem kombinovat. Příkladem může být přeprava nadrozměrného nákladu. Nejdříve je třeba určit všechny cesty ze zdrojového do cílového vrcholu, po nichž je možné náklad přepravovat. Je-li těchto cest několik, vybere se z nich taková, pro kterou jsou náklady na přepravu (popř. vzdálenost) minimální. V tomto případě bude tedy nejdříve použit algoritmus pro hledání cesty s maximální kapacitou, jestliže bude nalezeno více možných cest, vybere se z nich optimální podle algoritmu pro hledání nejkratší cesty. Podobně lze kombinovat také algoritmus hledání nejspolehlivější a nejkratší cesty.

Síťová analýza je disciplína teorie grafů, která je zaměřena na analýzu projektů. Projektem se rozumí soubor časově vymezených činností, nutných k dosažení určitého cíle. Činnosti jsou mezi sebou navzájem propojeny technologickými a organizačními návaznostmi. Modelem projektu je síťový graf (speciální typ orientovaného grafu).

Kostra grafu

Kostra grafu je vlastně vzájemné propojení všech míst na síti, které nesmí obsahovat kružnici. Pro hranově ohodnocené grafy lze sestrojit minimální, popř. maximální kostru grafu. Úlohu o hledání minimální kostry grafu lze aplikovat např. při hledání nejlevnějšího propojení dané oblasti telefonním kabelem, dopravní sítí apod. Také v určitých krizových situacích je algoritmus hledání kostry grafu využitelný. Příkladem může být krizová situace, při níž dojde k částečnému nebo úplnému zneprůchodnění dopravních komunikací (sněhová kalamita, zasypání, zatopení apod). Pro zajištění nejnutnějšího spojení mezi určenými strategicky významnými body je třeba alespoň částečně obnovit komunikační síť tak, aby každý ze strategických bodů byl propojen s ostatními. Známe vzdálenosti mezi vrcholy sítě, rychlost každého typu vozidla, který bude v dané situaci použit, a předpokládáme, že potřebný počet odklízecích mechanismů bude předem rozmístěn na určená stanoviště. Ohodnocení hran udává vzdálenosti mezi krajními vrcholy hrany, a protože při krizové situaci je nejdůležitějším hlediskem čas, bude v tomto případě nutné vzdálenost přepočítat na čas potřebný k zprůjezdnění hrany (pomocí rychlostí jednotlivých typů vozidel). Nalezením minimální kostry tohoto grafu se určí komunikace, které mají být přednostně zprůjezdněny, aby byly vrcholy sítě navzájem propojeny co nejrychleji.

Eulerovský tah

Eulerovský tah je tvořen posloupností vrcholů a hran grafu, každá hrana grafu se v eulerovském tahu vyskytuje právě jednou. Eulerovský tah může být uzavřený nebo otevřený. Uzavřený eulerovský tah lze sestrojit pomocí Fleuryho algoritmu na grafu, který má všechny vrcholy sudého stupně. Tah projde každou hranou grafu právě jednou, začíná i končí ve stejném vrcholu. Otevřený eulerovský tah se konstruuje na grafu, který má právě dva vrcholy lichého stupně. Začíná v jednom z vrcholů lichého stupně, projde každou hranou grafu právě jednou a vrátí se do druhého z vrcholů lichého stupně. Také při konstrukci otevřeného eulerovského tahu v grafu se dvěma vrcholy lichého stupně lze použít upravený Fleuryho algoritmus. Pro konstrukci uzavřeného eulerovského sledu v grafu se sudým počtem vrcholů lichého stupně se používá Edmonsnův algoritmus. V tomto případě nelze sestrojit eulerovský tah (posloupnost vrcholů a hran, v níž se každá hrana vyskytuje právě jednou), ale pouze tzv. eulerovský sled, ve kterém se budou některé hrany vyskytovat dvakrát. Je-li graf hranově ohodnocený, má význam hovořit o minimálním (popř. maximálním) eulerovském sledu (sled, který má minimální, popř. maximální součet ohodnocení hran). Praktický význam má především tento typ úloh, kde je důležitý správný výběr hran, po kterých bude nutno projít dvakrát, aby se minimalizovala nákladová funkce. Úloha hledání uzavřeného eulerovského tahu je také známa jako úloha čínského pošťáka. Už z tohoto označení je zřejmá jedna z možných praktických aplikací problému. Pošťák vyjde z jednoho určeného vrcholu (poštovní úřad), projde každou ulicí ve svém rajónu a po skončení roznášky se vrací zpět do výchozího vrcholu. Další možnou praktickou aplikací úlohy je např. hledání optimální trasy čistícího vozu apod.

Hamiltonovská kružnice

Hamiltonovská kružnice je takový podgraf grafu, který je kružnicí a ve kterém se každý vrchol grafu vyskytuje právě jednou. Pro řešení praktických úloh na hranově ohodnocených grafech je opět důležité vyhledávání minimální, popř. maximální hamiltonovské kružnice, pokud existuje. Pro tuto úlohu je známo několik metod řešení, mezi nejznámější patří heuristický algoritmus určení

hamiltonovské kružnice pro kompletní grafy nebo metoda větvení a mezí, ze které vychází známý Littlův algoritmus. Úloha nalezení minimální hamiltonovské kružnice bývá také označována jako úloha obchodního cestujícího. Obchodní cestující má za úkol navštívit všechny vrcholy sítě (každý z nich právě jednou) a vrátit se zpět do výchozího vrcholu. Cílem je stanovit trasu obchodního cestujícího tak, aby celkové náklady byly minimální a aby cestující prošel každým vrcholem právě jednou. Obdobného charakteru jsou i možné praktické aplikace této úlohy.

ZADÁNÍ ÚLOHY A POSTUP V POSTGIS

- vyzkoušejte si stažení a import dat do databáze PostGIS – nejdříve na datech pro Prahu
- otestujte algoritmy pro hledání optimální cesty, dále výpočet servisních oblastí a úlohu obchodního cestujícího
- vše následně aplikujte na data pro oblast vašeho bydliště

Síťové analýzy (tzv. routing) zajišťuje v prostředí PostGIS nadstavba označovaná jako **pgRouting**. Nadstavbu v databázi aktivujeme příkazem:

CREATE EXTENSION pgrouting;

Jako podkladová data použijeme data OpenStreetMap pro území Hlavního města Prahy. Data stáhneme přes tzv. Overpass API. Území je dáno minimálním ohraničujícím obdélníkem (bbox), který můžeme zjistit např. ze stránek http://boundingbox.klokantech.com (formát CSV).

Vzorový příklad pro stažení dat:

wget --progress=dot:mega -O praha.osm \

http://www.overpass-api.de/api/xapi?*[bbox=14.224435,49.941898,14.706787,50.177433][@meta]

Data je nutné naimportovat do databáze PostGIS specializovaným nástrojem **osm2routing**.

osm2pgrouting -f praha.osm --schema routing -d nazevdatabaze -U postgres

Po importu se ve výstupním schématu objeví následující tabulky. Dejte si pozor, data jsou transformována do WGS84, geometrie je uložena ve sloupci the_geom.

SELECT f_table_name,f_geometry_column,coord_dimension,srid,type

FROM geometry_columns WHERE f_table_schema = 'routing';

Algoritmus nalezení optimální cesty je implementován v pgRouting ve dvou variantách:

- pgr_dijkstra, viz. Dijkstra's algorithm
- pgr_aStar, viz A* search algorithm

Nejkratší trasa (jeden chodec)

Hledáme nejkratší trasu, nákladem tedy bude délka segmentů trasy. Chodec se může pohybovat v obou směrech (budeme pracovat s neorientovaným grafem).

Nastavíme si cestu ke schématům.

```
SET search_path TO public,routing,ruian_praha;
```

Výchozí a cílový bod můžeme najít s využitím adresních míst RÚIAN. Dojde k vyhledání všech OSM bodů do vzdálenosti 10 m od zadané adresy.

```
SELECT o.osm_id, o.id, a.gml_id FROM

ruian_praha.adresnimista a,

ruian_praha.ulice u,

routing.ways_vertices_pgr o

WHERE a.cislodomovni = 2077 AND a.cisloorientacni = 7 AND u.nazev = 'Thákurova'

AND a.ulicekod = u.kod

AND ST_DWithin(ST_Transform(o.geom, 5514), a.geom, 20);
```

Nalezení relevantního uzlu zpomaluje transformace uzlů ze souřadnicového systému EPSG:4326 do EPSG:5514. Pro urychlení výpočtu si geometrii uzlů v EPSG:5514 přepočítáme.

```
ALTER TABLE routing.ways_vertices_pgr ADD COLUMN geom5514 geometry(point, 5514);

UPDATE routing.ways_vertices_pgr set geom5514 = st_transform(geom, 5514);

CREATE INDEX on routing.ways_vertices_pgr using gist (geom5514);
```

Nejkratší trasu nalezneme voláním funkce pgr_dijkstra. Dijkstrův algoritmus vyžaduje definovat celkem čtyři atributy:

- id - identifikátor hrany
- source - identifikátor počátečního uzlu
- target - identifikátor koncového uzlu
- cost - atribut nákladů

```
SELECT * FROM pgr_dijkstra('

 SELECT gid AS id,

 source,

 target,

 length AS cost
```

```
 FROM routing.ways',
find_node('Thákurova', 2077, 7),
find_node('Václavkova', 169, 1),
directed := false);
```

Náklady jsou počítány v mapových jednotkách souřadnicového systému, v tomto případě stupních. Délku v metrech je uložena v atributu length_m. Příklad výpočtu celkové délky nalezené trasy:

```
SELECT sum(cost) FROM (SELECT * FROM pgr_dijkstra('
 SELECT gid AS id,
 source,
 target,
 length_m AS cost
 FROM routing.ways',
find_node('Thákurova', 2077, 7),
find_node('Václavkova', 169, 1),
directed := false)) AS foo;
```

Geometrii trasy získáte spojením výsledku hledání optimální trasy s původní tabulkou:

```
SELECT a.*, ST_AsText(b.geom) FROM pgr_dijkstra('
 SELECT gid AS id,
 source,
 target,
 length_m AS cost
 FROM routing.ways',
 find_node('Thákurova', 2077, 7),
 find_node('Václavkova', 169, 1),
 directed := false) AS a
LEFT JOIN routing.ways AS b
ON (a.edge = b.gid) ORDER BY seq;
```

Pro hledání optimální trasy lze použít funkci pgr_astar, která pracuje s geografickou informací uzlů hran grafu. To umožňuje ve výpočtu preferovat hrany, které jsou blíže cíle trasy.

```
SELECT * FROM pgr_astar('
 SELECT gid AS id,
 source,
 target,
 length_m AS cost,
 x1, y1, x2, y2
 FROM routing.ways',
 find_node('Thákurova', 2077, 7),
 find_node('Václavkova', 169, 1),
 directed := false);
```