VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA

Hornicko – geologická fakulta

Katedra geoinformatiky

PROSTOROVÁ ANALÝZA DAT

Síťové analýzy v Pgrouting – část 2.

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.

Servisní oblasti

Service area je nad sítí definovaná oblast zahrnující všechny úseky sítě dosažitelné ze zkoumaného centra za stanovených podmínek (např. oblast dosažitelná za méně než 5 minut od stanice záchranné služby). Výsledkem analýzy jsou soustředné oblasti tvořené zkoumanými intervaly (časovými). Vymezené oblasti lze použít pro zjištění rozlohy regionu, počtu obyvatel a podobných ukazatelů.

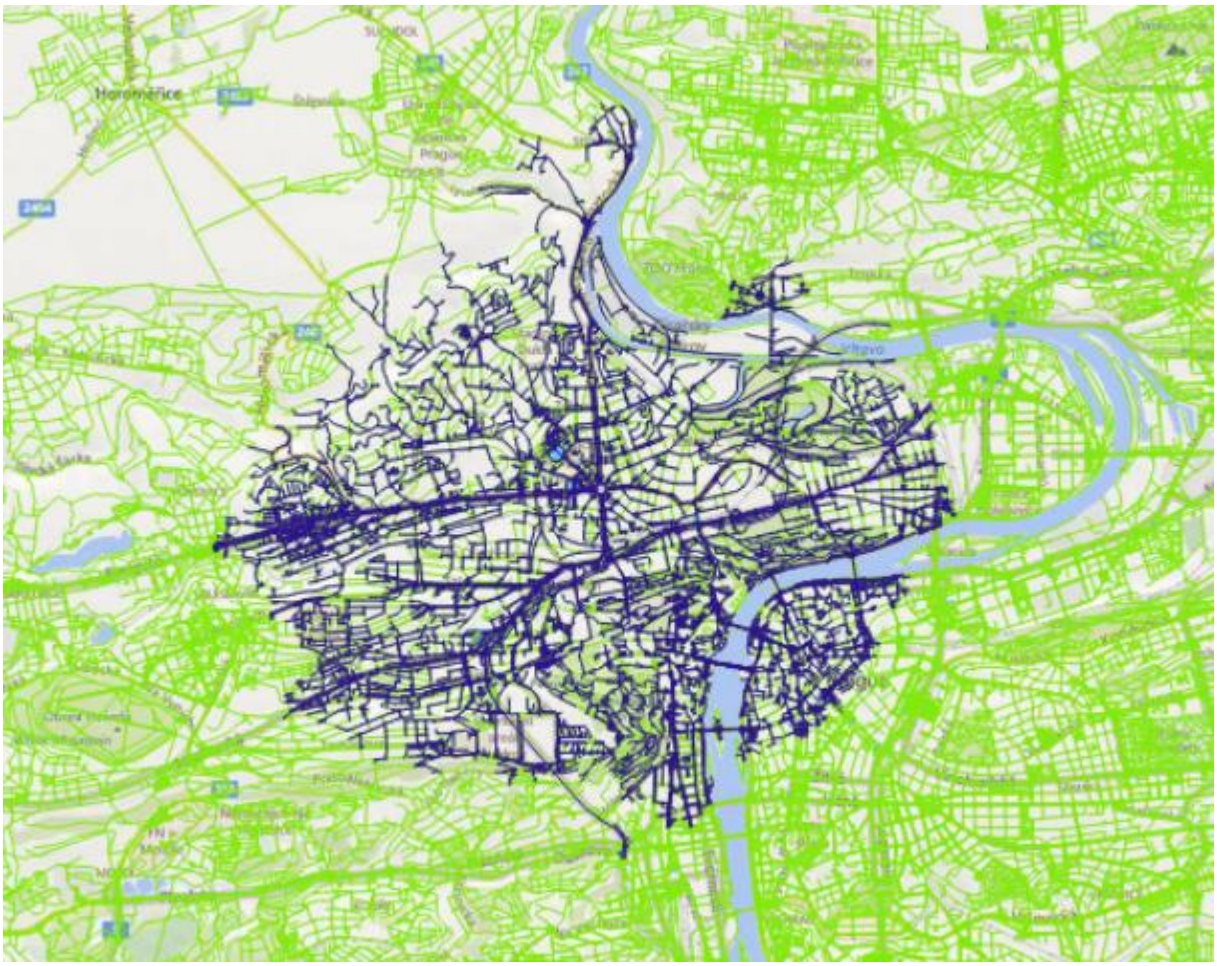
Service Area neboli obslužné zóny představují hrany (ulice), které spadají do oblasti dané odporem. Odpor omezuje dosažení obslužné oblasti a přístupnost zařízení se pak odlišuje podle něj. Zařízení, kolem jsou dány lokalizací na síti a vždy do analýzy musí vstupovat alespoň jedno. Je možné také vytvářet složené obslužné zóny, např. ve vzdálenosti 5, 10 a 15 km. Obslužné zóny mají podobnou funkčnost jako Buffer. Také generují „obalové“ zóny kolem bodových prvků a představují tak zóny obsluhované danými body. Na rozdíl od Buffer se jedná o síťovou analýzu, tedy analýzu po síti. Vygenerované obslužné polygony představují konkrétní obslužné zóny pro dané zařízení. Na začátku jsou prázdné a teprve po vyřešení analýzy se vytvoří. V jejich atributech se nachází informace o tom, ke kterému zařízení náleží. Při tvorbě obslužných zón je nejdůležitější nastavení odporu (Impedance), kterým může být vzdálenost, čas, náklady nebo jiný atribut, podle kterého se zóny vytvářejí.

POSTUP V PROSTŘEDÍ POSTGIS

Častou operací v síťových analýzách je výpočet servisní sítě. Zajímá nás, kam je možné se v rámci sítě dostat do určitého času. V našem případě nastavíme 300 sekund.

Upravíme penalizaci pro průchod, aby se více blížil realitě. Budeme uvažovat, že můžeme jet kdekoli jen o něco málo pomaleji než po hlavních silnicích a zásadně zvýhodníme jen dálnice.

```
UPDATE routing.configuration SET penalty=1.2;

UPDATE routing.configuration SET penalty=1.0 WHERE tag_key = 'highway' AND
 tag_value IN ('secondary', 'secondary_link', 'tertiary', 'tertiary_link');

UPDATE routing.configuration SET penalty=1.0 WHERE tag_key = 'highway' AND
 tag_value IN ('primary','primary_link');

UPDATE routing.configuration SET penalty=1.0 WHERE tag_key = 'highway' AND
 tag_value IN ('trunk','trunk_link');

UPDATE routing.configuration SET penalty=0.8 WHERE tag_key = 'highway' AND
 tag_value IN ('motorway','motorway_junction','motorway_link');


SELECT a.*, b.geom AS geom FROM pgr_drivingDistance('
 SELECT gid AS id,
 source,
 target,
 cost_s * penalty AS cost,
 reverse_cost_s * penalty AS reverse_cost
 FROM routing.ways JOIN routing.configuration
 USING (tag_id)',
 find_node('Thákurova', 2077, 7, 'routing.ways_vertices_pgr'),
 300,
 directed := true) AS a
LEFT JOIN routing.ways AS b
ON (a.edge = b.gid) ORDER BY seq;
```



Vytvoření sítě

Ne vždy je možné pracovat se sítí postavenou nad daty OSM. Pokud máme vlastní síť, můžeme se pokusit vybudovat graf nad ní.

Příprava dat

Pokud nemáme data připravena pro síťové analýzy, např. nám chybí uzly v místech křížení silnic, pak je nutné před vlastním vybudováním grafu realizovat úpravu dat.

K dispozici je funkce **pgr_nodeNetwork**, která dokáže doplnit uzly v místech křížení, případně dotáhnout linie k jiným liniím, v případě nedotahů.

V případě, že funkce selže, jako v následujícím ukázce nad ulicemi Prahy, můžeme zkusit alternativní postup popsaný dále.

```
SELECT pgr_nodeNetwork('ruian_praha.ulice', 1, 'ogc_fid', 'geom');

ERROR:  line_locate_point: 1st arg isn't a line

CONTEXT:  SQL statement "create temp table inter_loc on commit drop as ( select * from (

    (select l1id, l2id, st_linelocatepoint(line,source) as locus from intergeom)

     union

    (select l1id, l2id, st_linelocatepoint(line,target) as locus from intergeom)) as foo
```

where locus<>0 and locus<>1)"

PL/pgSQL function pgr_nodenetwork(text,double precision,text,text,text,text,boolean) line 191 at EXECUTE

Alternativní způsob využívá běžných nástrojů PostGIS a snahu o vytvoření multilinie agregací z existující kolekce linií.

```
CREATE TABLE ulice_noded AS

SELECT d.path[1], geom FROM (

  SELECT ST_Union(geom) g FROM ruian_praha.ulice

) dta

, ST_Dump(g) d;
```

Vytvoření grafu

Před vytvořením grafu, který realizuje funkce **pgr_createTopology**, je nutné přidat sloupce source a target, kam jsou zapsány identifikátory uzlů.

Vhodné je také vytvořit primární klíč a indexovat geometrii.

```
ALTER TABLE ulice_noded ADD PRIMARY KEY (path);

CREATE INDEX ON ulice_noded USING gist(geom);

ALTER TABLE ulice_noded ADD COLUMN "source" integer;

ALTER TABLE ulice_noded ADD COLUMN "target" integer;
```

Graf se vytvoří pomocí funkce **pgr_createTopology**(), kde se zadají názvy sloupců s geometrií, id a sloupce pro zápis id nodů (source, target). Hodnota 1 ve funkci představuje toleranci pro tvorbu grafu (v mapových jednotkách, tj. v našem případě jde o 1 metr).

```
SELECT pgr_createTopology('ulice_noded', 1, 'geom', 'path', 'source', 'target');
```

Na závěr je vhodné ohodnotit graf pomocí např. délky úseků.

```
ALTER TABLE ulice_noded ADD COLUMN length FLOAT;

UPDATE ulice_noded SET length = ST_Length(geom);
```