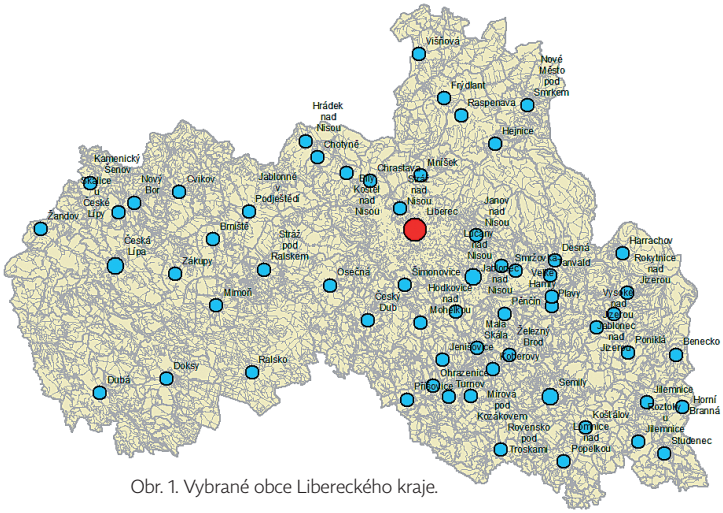
Trasa obchodního cestujícího v GIS (algoritmy řešení)

Antonín Svoboda, Krajský úřad Libereckého kraje

Jeden z úkolů, které jsem v poslední době řešil, bylo nalezení optimální okružní trasy po obcích Libereckého kraje – obce je třeba spojit nejkratší možnou trasou a vrátit se do výchozího místa. Je to klasická optimalizační úloha „problému obchodního cestujícího“, někdy nazývaná „problém čínského listonoše“. Výchozím bodem je město Liberec a podle zadání potřebuji v každé obci vyložit zboží a vrátit se zpět do Liberce. Jaká je tedy nejkratší možná trasa? Liberecký kraj má celkem 215 obcí, já ale budu řešit rozvoz zboží pouze do obcí nad 1000 obyvatel. Takových obcí je 62.

Při řešení problému obchodního cestujícího je třeba zdůraznit, že existují dva druhy tohoto problému. Je to metrický a nemetrický problém obchodního cestujícího. Metrický problém znamená, že vzdálenost z místa A do místa B je stejně dlouhá, jako vzdálenost z místa B do místa A. U nemetrického problému toto neplatí, protože jsou na komunikacích jednosměrky, objížďky a uzavírky. V GIS řešíme především nemetrický problém obchodního cestujícího.



Obr. 1. Vybrané obce Libereckého kraje.

Pro řešení tohoto úkolu potřebujeme tabulku vstupních dat, což je tabulka vzdáleností mezi jednotlivými obcemi (viz tabulku 1).

JAK PROBLÉM ŘEŠIT?

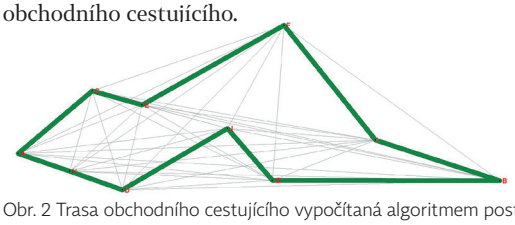
Existuje několik algoritmů, jak hledat a nalézt nejkratší trasu. Nejjednodušší metodou je nalezení všech možných kombinací tras a výběr trasy s nejmenší délkou. Tento algoritmus se nazývá algoritmem postupných permutací.

Id	A	B	C	D	E	F	G	H	I	J
A	0	186	643	296	630	475	438	157	338	557
B	186	0	466	110	455	575	252	224	235	401
C	643	466	0	358	26	1013	240	679	543	449
D	296	110	358	0	349	665	142	322	250	342
E	630	455	26	349	0	1008	239	672	544	462
F	475	575	1013	665	1008	0	773	352	487	706
G	438	252	240	142	239	773	0	447	305	279
H	157	224	679	322	672	352	447	0	237	481
I	338	235	543	250	544	487	305	237	0	245
J	557	401	449	342	462	706	279	481	245	0

Tabulka 1. Příklad tabulky vstupních dat – tabulka vzdáleností.

Algoritmus postupných permutací (APP)

Algoritmus je založen na principu **vyhledání všech možných tras** a výběru trasy **s nejkratší délkou**. Za pomocí rekurze může být algoritmus velmi krátký a poměrně efektivní. Hlavní výhodou tohoto algoritmu je skutečnost, že **vždy najde nejkratší trasu** ze všech možných tras. Tato trasa se nazývá trasou obchodního cestujícího. Další výhodou tohoto algoritmu je, že umí řešit metrický i nemetrický problém obchodního cestujícího.



Obr. 2 Trasa obchodního cestujícího vypočítaná algoritmem postupných permutací.

Nevýhodou tohoto algoritmu je extrémní náročnost na výpočetní výkon. To je způsobeno exponenciálním nárůstem permutací při zvyšujícím se počtu bodů. Počet permutací

odpovídá funkci N! (n-faktoriál). Funkce faktoriál velmi prudce exponenciálně roste a algoritmem postupných permutací lze řešit úlohy do maximálně 15 bodů. Například pro 25 bodů bychom pro výpočet takové trasy potřebovali více času, než je doba existence celého vesmíru! Je to ale jediný algoritmus, který vždy najde trasu obchodního cestujícího, tedy nejkratší možnou trasu.

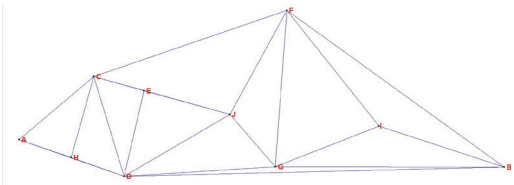
Bodů	Počet permutací	Doba výpočtu [hodiny]	Doba výpočtu [roky]
1	1		
2	2		
3	6		
4	24		
5	120		
6	720		
7	5 040		
8	40 320		
9	362 880		
10	3 628 800		
11	39 916 800		
12	479 001 600		
13	6 227 020 800		
14	87 178 291 200	2,8	
15	1 307 674 368 000	41,3	
16	20 922 789 888 000	660	
17	355 687 428 096 000	11 220	31
18	6 402 373 705 728 000	201 960	553
19	121 645 100 408 832 000	3 837 240	10 513
20	2 432 902 008 176 640 000	76 744 800	210 260
21	51 090 942 171 709 400 000	1 611 640 800	4 415 454
22	1 124 000 727 777 610 000 000	35 456 097 600	97 139 993
23	25 852 016 738 885 000 000 000	815 490 244 800	2 234 219 849
24	620 448 401 733 239 000 000 000	19 571 765 875 200	53 621 276 370
25	15 511 210 043 331 000 000 000 000	489 294 146 880 000	1 340 531 909 260

Tabulka 2. Výpočetní doba metodou postupných permutací.

Výsledkem nemusí být pouze jedna jediná trasa. Může nastat situace, kdy existuje více různých tras, ale všechny mají stejnou nejkratší délku. U metrického problému obchodního cestujícího může být výsledkem jedna trasa, po které můžeme cestovat buď po směru hodinových ručiček, nebo proti směru hodinových ručiček. U nemetrického problému může být výsledkem pouze jedna jediná **orientovaná** trasa.

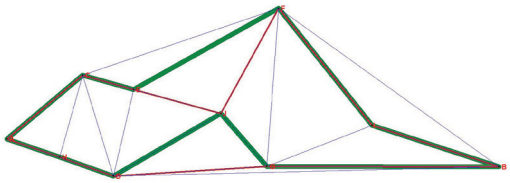
Algoritmus 3NET

Při zvyšujícím se počtu bodů velmi narůstá počet spojnic mezi jednotlivými body trasy. Pokud chceme snížit náročnost na výpočetní výkon, je třeba snížit počet spojnic mezi jednotlivými body trasy. To lze vyřešit tak, že body začleníme do trojúhelníkové sítě. Počet spojnic bodů trasy je v trojúhelníkové síti výrazně nižší než celkový počet spojnic mezi všemi body. Hledání trasy na trojúhelníkové síti je výrazně rychlejší, než u algoritmu postupných permutací.



Obr. 3. Spojnice trojúhelníkové sítě.

U tohoto algoritmu je velmi důležitá konstrukce trojúhelníkové sítě. Pokud je síť konstruována špatně, najdeme sice nejkratší trasu na trojúhelníkové síti, ale nemusí to být trasa obchodního cestujícího, tedy ta nejkratší možná. Abychom našli trasu obchodního cestujícího, musíme tento algoritmus doplnit vertexovou optimalizací.



Obr. 4. Trasa 3NET (červená), trasa APP (zelená).

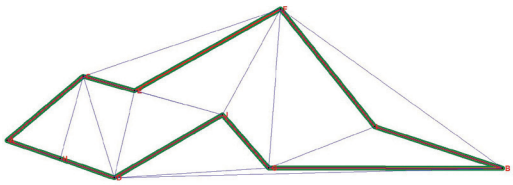
Na trojúhelníkové síti hledáme trasu tak, že z výchozího bodu postupujeme po síti a přidáváme další body do trasy, pokud je již nemáme v trase. Máme-li všechny body v trase, spočítáme délku trasy. Nemůžeme-li pokračovat dále po síti, vrátíme se o krok zpět a pokračujeme jiným směrem. Ze všech tras vybereme trasu s nejmenší délkou.

Tento rekurzivní algoritmus, někdy nazývaný **hladový algoritmus**, najde nejkratší trasu na trojúhelníkové síti, ale nemusí to být trasa obchodního cestujícího. V dalším kroku provedeme vertexovou optimalizaci.

Vertexová optimalizace trasy

Tato optimalizační metoda je založena na jednoduchém principu. Z trasy vyčleníme úsek, u něhož zahustíme vertexy spojnicemi. Na takto zahuštěném úseku hledáme úsek s kratší délkou. Pokud nalezneme kratší úsek trasy, nahradíme jej za původní a tím dostaneme i kratší celou trasu.

Po optimalizaci můžeme dostat trasu, která vede mimo trojúhelníkovou síť. Při větším počtu bodů se to stává celkem běžně, ale zároveň je nová trasa kratší, než vyhledaná trasa na trojúhelníkové síti. Může být trasa obchodního cestujícího, tedy nejkratší možná, ale také to tak být nemusí.



Obr. 5. Trasa po vertexové optimalizaci může být trasou obchodního cestujícího.

Algoritmus hledání na trojúhelníkové síti je rychlejší, než algoritmus postupných permutací. S optimalizací nalezne velmi krátkou trasu, ale přesto nemusí nalézt trasu obchodního cestujícího, tedy tu nejkratší možnou. Algoritmus lze použít až do 25 bodů, při zvyšujícím se počtu bodů se doba výpočtu stává neúnosnou a exponenciálně narůstá.

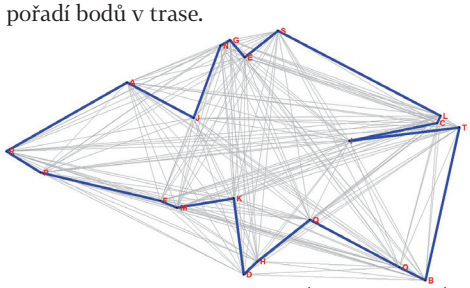
APT2O

Pro vyšší počet bodů by předchozí algoritmy byly nepoužitelné. Algoritmus polární trasy s dvojitou optimalizací je poměrně rychlý a efektivní algoritmus, jak nalézt velmi

krátkou trasu. Kromě vstupní tabulky vzdáleností vyžaduje tento algoritmus i souřadnice bodů.

Algoritmus je rozdělen to tří částí. Nejprve se vypočítá prvotní trasa, poté se provede vertexová a následně úseková optimalizace. Pořadí, v jakém provedeme optimalizace, může vést k různým výsledkům, tedy k nalezení kratší trasy.

Při výpočtu prvotní trasy musíme nejprve vypočítat centrální vztažný bod (**těžiště**) jako aritmetický průměr x-ovych a y-ových souřadnic jednotlivých bodů. Všechny souřadnice bodů přepočítáme na polární souřadnice vzhledem k tomuto těžišti (odtud název Algoritmus polární trasy). Polární souřadnice bodů seřadíme podle velikosti úhlu od 0 do 360 stupňů a vzdálenosti vzestupně a tím máme dáno pořadí bodů v trase.

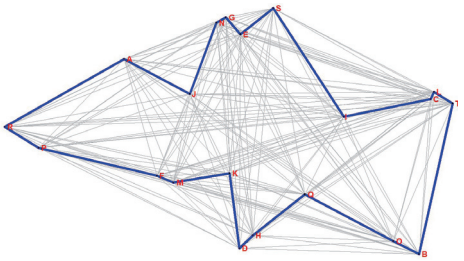


Obr. 6. Polární trasa (bez optimalizace).

Prvotní trasa není sice nijak krátká, ale výpočet je velice rychlý. Tuto prvotní trasu je nutné optimalizovat.

Pro optimalizaci můžeme použít stejnou metodu, jako u algoritmu hledání na trojúhelníkové síti, tj. optimalizaci vertexů trasy.

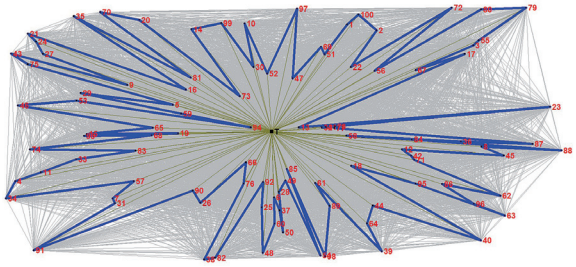
Druhou optimalizační metodou je úseková optimalizace. Ta spočívá v zaměňování úseků trasy, přičemž se hledá jejich optimální umístění.



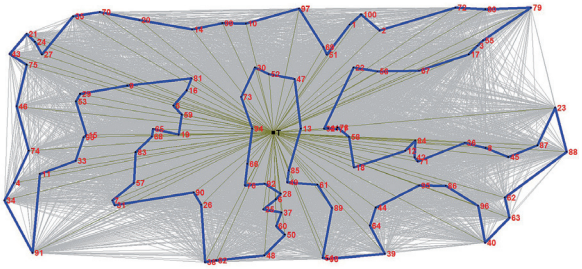
Obr. 7. Polární trasa po optimalizaci.

Výhodou algoritmu polární trasy je velmi rychlý výpočet prvotní trasy i při větším počtu bodů (až do 200 bodů). Za pomocí optimalizací prvotní trasy lze poměrně rychle nalézt krátkou trasu. Nevýhodou je velmi dlouhá prvotní trasa, při větším počtu bodů velmi deformovaná a vyžaduje nutnost optimalizace trasy oběma optimalizačními metodami.

Algoritmus najde poměrně krátkou trasu, nemusí však najít tu nejkratší možnou, tedy trasu obchodního cestujícího. Pro nalezení skutečné trasy obchodního cestujícího je nutná **hluboká optimalizace**, u které se doba výpočtu velmi výrazně prodlouží (až o tři řády).



Obr. 8. Při větším počtu bodů je polární trasa značně deformovaná.

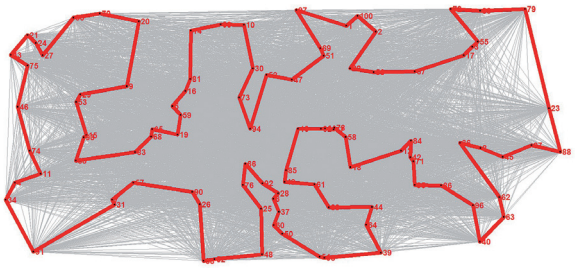


Obr. 9. Po optimalizacích je trasa výrazně kratší (výsledná trasa).

Algoritmus Chiméra

Při hledání trasy obchodního cestujícího na velkém souboru bodů je výhodnější použít algoritmus Chiméra. Tento algoritmus nemá omezení shora, tedy není omezen počtem bodů trasy a zvládne i stovky až tisíce bodů, aniž by příliš rostla náročnost na výpočetní výkon.

Algoritmus sestává ze tří základních částí. Prvním krokem je redukce počtu bodů, ve druhé fázi se stanoví prvotní trasa a ve třetí fázi probíhá addukce bodů do trasy s průběžnou optimalizací trasy. Výhody tohoto algoritmu spočívají především ve vysoké rychlosti výpočtu trasy a neomezeného počtu bodů v trase. Počet bodů je omezen pouze kapacitou paměti počítače. Další výhodou je skutečnost, že pro vlastní výpočet stačí pouze tabulka vzdáleností mezi jednotlivými body a nepotřebuje znát souřadnice bodů. Rovněž umí řešit metrický i nemetrický problém obchodního cestujícího.



Obr. 10. Trasa vypočítaná algoritmem Chiméra.

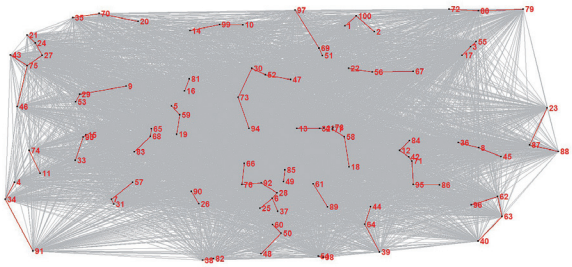
Nevýhodou tohoto algoritmu je složitost jeho naprogramování.

Výsledkem je velmi krátká trasa, ale nemusí to být trasa obchodního cestujícího. Pro nalezení nejkratší možné trasy je nutná **hluboká optimalizace**.

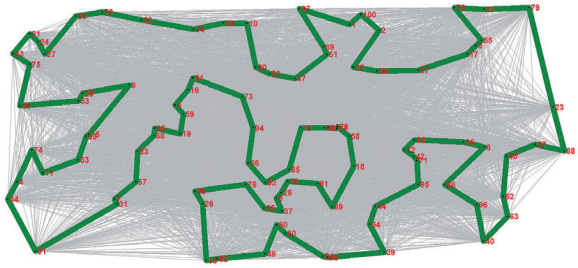
Trasa nalezená algoritmem Chiméra je skutečně velmi krátká, a pokud to není přímo trasa obchodního cestujícího, tak se od nejkratší možné trasy liší minimálně, nejvýše jednotky promile.

Fragmentační algoritmus

Tento algoritmus je vylepšená Chiméra. Také není omezen počtem bodů a umí řešit velké soubory bodů. Vychází z principu skládání trasy z fragmentů trasy obchodního cestujícího do imaginární trasy s následnou optimalizací. Je to velmi rychlý algoritmus, avšak je velmi náročný na naprogramování. Pro nalezení nejkratší možné trasy je nutná hluboká optimalizace.



Obr. 11. Fragmenty trasy.



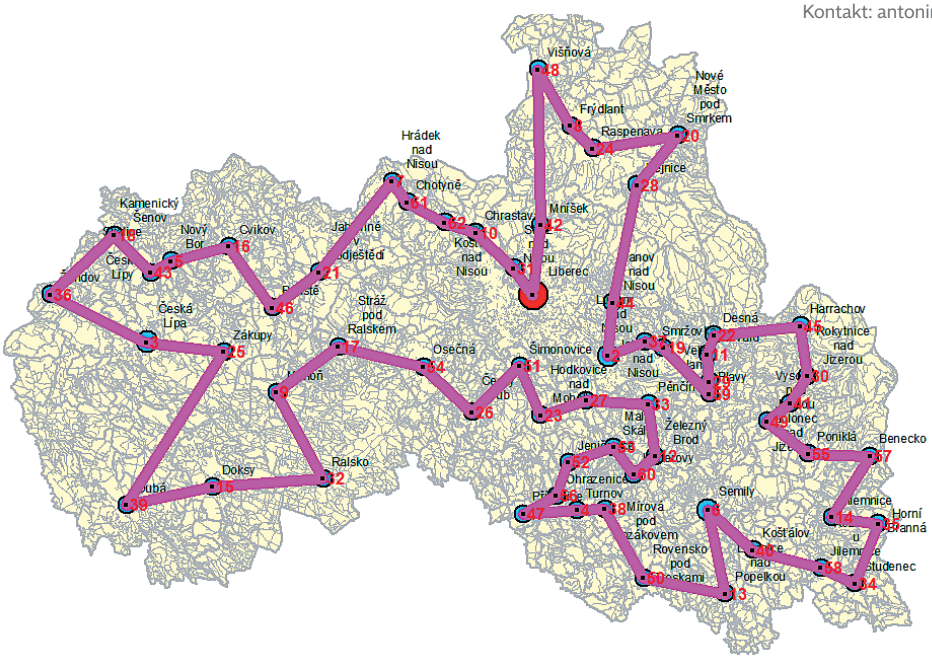
Obr. 12. Výsledná trasa vypočtená fragmentačním algoritmem.

JAK NALÉZT TRASU OBCHODNÍHO CESTUJÍCÍHO?

Pro výpočet trasy obchodního cestujícího je třeba zkusit vícero algoritmů. Pokud se trasy shodují, lze předpokládat, že se jedná o trasu obchodního cestujícího. Pro skutečný výpočet trasy obchodního cestujícího nebo její ověření je nutné použít hlubokou optimalizaci, která je však enormně časově náročná. Hluboká optimalizace může nalézt kratší trasu, ale také může potvrdit již existující trasu v případě, že kratší trasa neexistuje.

Pro mnoho aplikací je rozhodujícím faktorem doba výpočtu trasy než nalezení trasy obchodního cestujícího. V praxi jsou ale situace, kdy nalezení nejkratší možné trasy je nutností. Toho lze docílit za použití hluboké optimalizace, avšak za cenu značného prodloužení doby výpočtu. Například pro 50 bodů trvá výpočet trasy 30 sekund, s optimalizací 90 sekund, ale při použití hluboké optimalizace necelých 23 hodin a 50 minut. «

Ing. Antonín Svoboda, Krajský úřad Libereckého kraje
Kontakt: antonin.svoboda@kraj-lbc.cz



Obr. 12. Výsledná trasa po obcích nad 1000 obyvatel v Libereckém kraji.