Michal Kolář
2. NKARTG
Praha 2021

Řešení problému obchodního cestujícího

Geoinformatika, úkol 1

PŘÍRODOVĚDECKÁ FAKULTA
Univerzita Karlova

1 Zadaní úlohy

Vstup: množina uzlů U reprezentujících body.
Výstup: nalezení nejkratší Hamiltonovské kružnice mezi těmito uzly.

Nad množinou U nalezněte nejkratší cestu, která vychází z libovolného uzlu, každý z uzlů navštíví pouze jedenkát, a vrací se do uzlu výchozího. Využijte níže uvedené metody konstrukčních heuristik:

- Nearest Neighbor,
- Best Insertion.

Výsledky porovnejte s výstupem poskytovaným nástrojem Network Analyt v SW ArcMap.

otestování proveďte nad dvěma zvolenými datasety, které by měly obsahovat alespoň 100 uzlů. jako vstup použijte esxistující geografická data (např. města v ČR s více než 10 000 obyvateli, evropská letiště, ...), ohodnocení hran bude představovat vzdálenost mezi uzly (popř. vzdálenost měřenou po silnici); pro tyto účely použijte vhodný GIS.

Výsledky s uvedením hodnot W,k, uspořádejte do přehledné tabulky (metodu Best Insertion nechte proběhnout alespoň 10x), a zhodnoťte je.

Pro implementaci obou konstrukčních heuristik použijte programovací jazyk Python, vizualizaci výstupů proveďte ve vhodné knihovně, např. matplotlib.

2 Popis a rozbor problému

2.1 Základní pojmy

2.1.1 Neorientovaný graf

Neorientovaný graf je dán uspořádanou trojicí disjunktních množin:

$$G = \langle U, H, \rho \rangle,$$

kde U je množinou uzlů, H množinou hran a ρ incidencí grafu. Incidence grafu, definovaná jako

$$\rho : H \to U \otimes U,$$

přiřazuje každé hraně dvojici uzlů h, kde

$$h = \{(u,v)|u,v \in U(G)\}, h \in H.$$

Jednotlivé hrany nemají orientaci, lze jimi procházet oběma směry a uzly lze projít vícekrát.

2.1.2 Cesta grafem

Cesta grafen je posloupností uzlů

$$C = \langle u_0, h_1, u_1, h_2, ..., u_{k-1}, h_k \rangle,$$

kde platí, že pro každou hranu $h_i \in H$, kde $h_i = (u_{i-1}, u_i)$, je každý uzel u_i v posloupnosti pouze jednou.

2.1.3 Kružnice grafu

Kružnicí K grafu G je cesta obsahující alespoň tři různé uzly, přičemž její počáteční uzel u_0 je i jejím koncovým uzlem u_k.

2.1.4 Hamiltonovská cesta

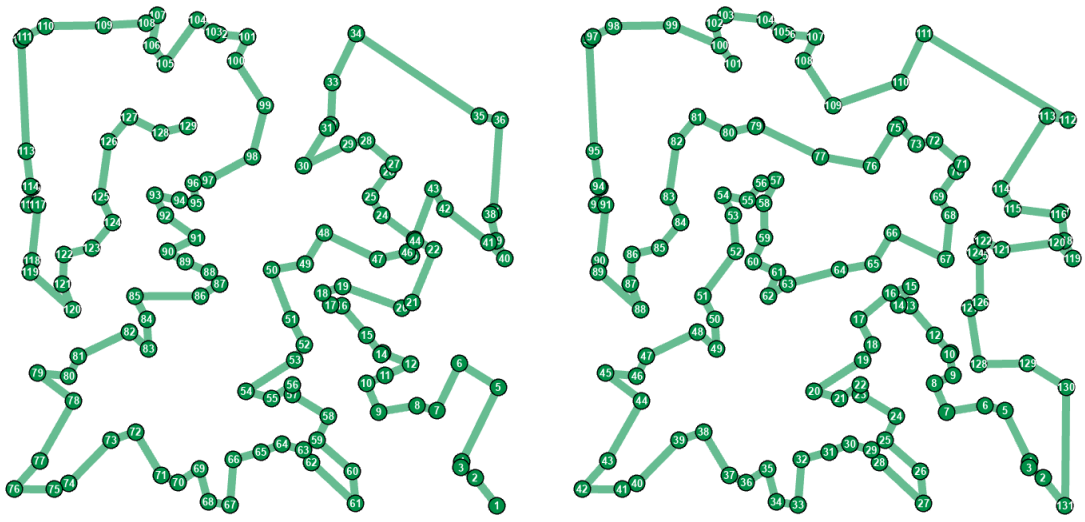
Hamiltonovská cesta C_h v grafu G je cesta obsahující všechny uzly grafu G, tedy

$$\forall u \in U : u \in C_h.$$

2.1.5 Hamiltonovská kružnice

Hamiltonovská kružnice K_h v grafu G je kružnice obsahující všechny uzly grafu G, tedy

$$\forall u \in U : u \in K_h.$$



Obrázek 1: Rozdíl mezi Hamiltonovskou cestou (vlevo) a Hamiltonovskou kružnicí (vpravo), zdroj: autor

2.1.6 Trasa obchodního cestujícího

Trasa obchodního cestujícího K_{TSP} v grafu G je libovolná Hamiltonovská kružnice tvořená uzly $\langle u_0, ..., u_k \rangle$.

2.1.7 Hamiltonovský graf

Hamiltonovský graf G_h je takový graf, ve kterém lze konstruovat minimálně jednu Hamiltonovskou kružnici K_h.

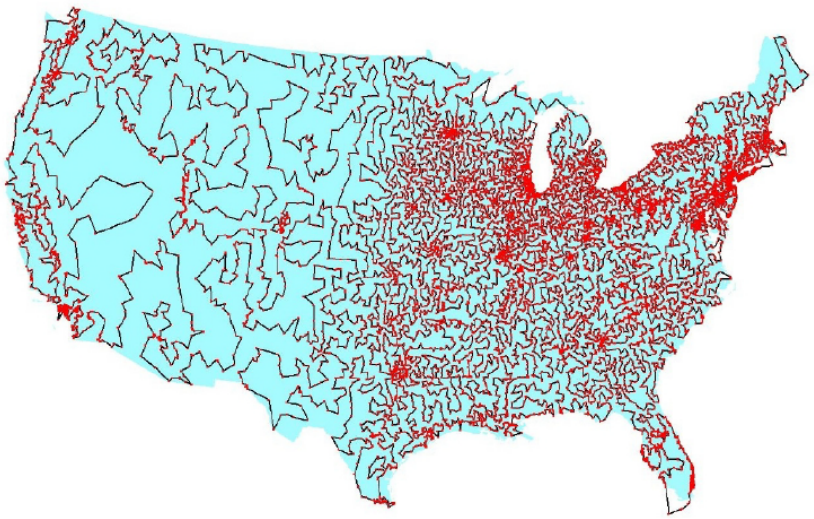
2.2 Úvod do problematiky TSP

Problém obchodního cestujícího (v angličtině *Travelling Salesman Problem*, dále TSP) patří mezi tzv. NP–úplné úlohy, tedy úlohy, u kterých nejsme schopni efektivně nalézt exaktní řešení. Řešení, které nalézáme, je často pouze přibližné a jehož kvalitu považujeme za akceptovatelnou (Bayer 2021). Během řešení TSP z hlediska terminologie teorie grafů hledáme co nejkratší hamiltonovskou kružnici K_h pro Hamiltonovský graf G_h. Jako praktická motivace se uvádí dilema obchodníka (dle toho název celého problému), jakým způsobem navštívit všechna města ve svém regionu, vrátit se do startovního místa a nacestovat během toho co nejkratší vzdálenost (případně nejrychlejší, nejlevnější cestu...) (Kovář 2016).

Minimální délka Hamiltonovské kružnice K_h je definována jako

$$W = \sum_{i=1}^{k-1} w(u_i, u_{i+1}) = min.$$

Jedná se tedy o sumu ohodnocení jednotlivých hran $w(u_i, u_{i+1})$ Hamiltonovského grafu G_h. Jak je výše zmíněno, exaktní řešení a ideální algoritmus k jeho nalezení zatím neexistuje. Musíme se spolehnout na metody hledající řešení přibližné. Mezi metody, kterými lze nalézt co nejlepší řešení, patří např. metody založené na prohledávání grafu s ořezáváním, metody konstrukčních heuristik, metody diskrétní optimalizace či stochastické heuristické metody (Bayer 2021, Šeda 2003).



Obrázek 2: Optimální řešení problému obchodního cestujícího pro 13 509 měst v USA, zdroj: Kovář 2016

2.3 Metody konstrukčních heuristik

Blíže budou přiblíženy dvě metody konstrukčních heuristik, a to *Nearest Neighbor* a *Best Insertion*. Obě tyto heuristiky

2.3.1 Nearest Neighbor

Nejjednodušší konstrukční heuristikou pro řešení TSP je *Nearest Neighbor*. Princip je takový, že po volbě libovolného počátečního uzlu u_0 se snažíme nalézt uzel u, který je nejbližším sousedem uzlu u_0. Algoritmus se následně přesouvá na tohoto nejbližšího souseda a opět hledá nejbližší uzel. Princip se opakuje, dokud není vytvořena Hamiltonovská kružnice K_h.

Algoritmus lze tak rozložit na 4 kroky:

1. Všechny uzly $u \in U$ jsou označené jako nezpracované, je zavedena délka W Hamiltonovské kružnice K_h.
2. Vybere se náhodný (nebo předem určený) aktuální uzel u_i a označí se za uzel zpracovaný.
3. Dokud nejsou všechny uzly označeny jako zpracované, tak se:
   (a) vybere nejbližší nezpracovaný uzel u k aktuálnímu uzlu u_i a minimalizuje se přírůstek délky
   $$\Delta w = \min_{u \notin K_h} w(u_i, u),$$
   (b) uzel u se přidá do Hamiltonovské kružnice aktualizací její délky
   $$W = W + \Delta w,$$
   (c) uzel u se označí jako uzel aktuální ($u_i = u$) a přidá se do seznamu uzlů zpracovaných.
4. Spočítá se přírůstek délky mezi posledním uzlem u_k a prvním uzlem u_0 a přičte se k celkové délce W.

2.3.2 Best Insertion

Sofistikovanější kosntrukční heuristikou je *Best Insertion*. Ta zavádí počáteční kružnice o 3 bodech a následně se snaží přidávat další uzly na co nejvhodnější pozici kružnice tak, aby se minimalizoval přírůstek délky cesty Δw, který se počítá na základě trojúhelníkové nerovnosti.

Algoritmus lze rozložit na 3 kroky:

1. Všechny uzly $u \in U$ jsou označené jako nezpracované, je zavedena délka W Hamiltonovské kružnice K_h.
2. Z množiny uzlů U se vyberou 3 náhodné uzly, které vytvoří počáteční kružnici. Body jsou označené jako zpracované, určí se délka W této kružnice.
3. Dokud nejsou všechny uzly označeny jako zpracované, tak se:
   (a) vybere náhodný uzel u z množiny nezpracovaných bodů,
   (b) z dosavadní kružnice se vybere taková hrana $w(u_i, u_{i+1})$, aby cesta u_i, u, u_{i+1} přes uzel u co nejméně prodloužila dosavadní kružnici a minimalizovala
   $$\Delta = \min_{u \notin K_h} w(u_i,u) + w(u,u_{i+1}) - w(u_i,u_{i+1}),$$
   (c) uzel u se přidá do kružnice mezi uzly u_i, u_{i+1} (čímž se označí za uzel zpracovaný) a aktualizuje se její délka
   $$W = W + \Delta w.$$

2.4 Testování výsledků

Efektivita algoritmu pro nalezení co nejlepší Hamiltonovské kružnice lze testovat porovnáním nalezeného výsledku s optimálním řešením (či nalezených výsledků navzájem). Může k tomu sloužit například

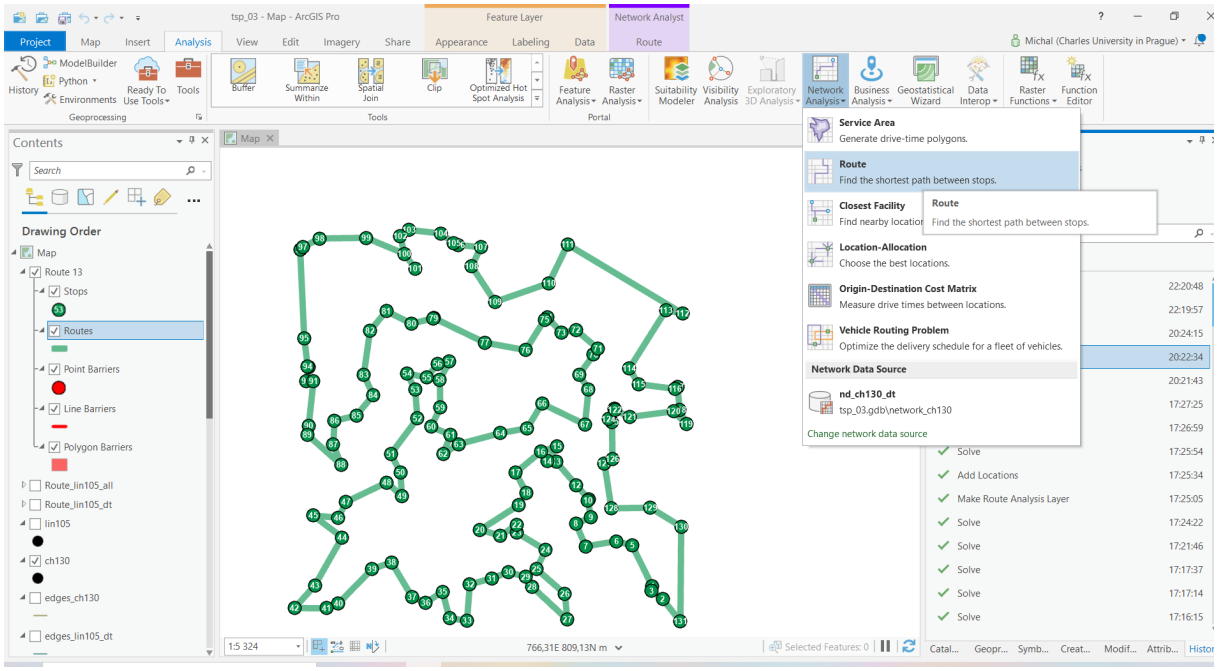
$$k = \frac{W}{W_o},$$

což je poměr nalezené délky W a optima W_o. Čím blíže je k blíže 1, tím lepší výsledek byl nalezen.

2.5 Obchodní cestující v ArcGIS Pro

Funkce řešící problém obchodního cestujícího jsou často součástí geoinformačních SW. V ArcGIS Pro se tyto funkce dají nalézt v Network Analyst. Extenze Network Analyst umožňuje provádět síťové analýzy, jako jsou hledání cesty do nejbližšího zařízení, alokace zdrojů, a právě také TSP.

TSP se v ArcGIS Pro Network Analyst řeší vytvořením vrstvy okružní analýzy (*Route analysis layer*). Metoda, kterou tato vrstva využívá, je Dijkstrův algoritmus. Výsledkem není ovšem Hamiltonova kružnice, nýbrž Hamiltonova cesta, je potřeba proto brát na tento fakt zřetel.



Obrázek 3: Síťové analýzy v ArcGIS Pro, zdroj: autor

3 Řešení problému

Problém obchodního cestujícího je řešen v jazyce Python a v SW ArcGIS Network Analyst. Výsledky řešení, použité datasety jsou následně v kapitole 4.

3.1 Implementace v Pythonu

Pro řešení problému obchodního cestujícího byly použity dvě metody konstrukčních heuristik: Nearest Neighbor a Best Insertion. Pro každou z těchto metod byla sestrojena individuální funkce. Obě metody lze optimalizovat, byly proto dále sestrojeny funkce procházející všechna možná řešení Nearest Neighbor (podle počátečního uzlu) a funkce vybírající co nejlepší možné řešení Best Insertion na základě počtu opakování. Pro zajímavost byla také sestrojena funkce, která testuje, kolikrát je potřeba opakovat Best Insertion, než bude nalezeno optimální řešení. Níže popsané funkce psudokódem byly následně použity k vygenerování výsledků a vizualizaci knihovnou *matplotlib*.

3.1.1 Nalezení Hamiltonovské kružnice podle zadané sekvence

Tato funkce ze zadaného seznamu souřadnic a sekvence sestrojí Hamiltonovu kružnici a vypočítá její sumu vah. V mém řešení je použita na konstrukci optimálního řešení a řešení ze SW ArcGIS Pro.

Algorithm 1 tour_solution(seznam uzlů, sekvence):

Přeskup seznam uzlů podle sekvence
a *převeď* uzly na numpy vektory.

Inicializuj sumu vah,
odeber první uzel ze seznamu,
který se stává aktuálním uzlem u.

Dokud je v seznamu nějaký uzel:
    vem další uzel ze seznamu, stává se uzlem u_i,
    vypočítej vzdálenost mezi uzly u a u_i,
    přičti ho k sumě vah
    a *přesuň* se na další uzel ($u_i = u$).
Vrať celkovou sumu vah a seznam uzlů.

3.1.2 Nearest Neighbor

Tato funkce na základě seznamu uzlů a počátečního uzlu vypočítá Hamiltonovskou kružnici metodou konstrukčních heuristik Nearest Neighbor. Výsledkem je suma vah a seznam uzlů tvořící Hamiltonovskou kružnici.

Algorithm 2 nearest_neighbor(seznam uzlů, počáteční uzel):

Inicializuj seznam nezpracovaných a zpracovaných uzlů a sumu vah.

Pokud index zadaného počátečního bodu je v seznamu uzlů,
    tak *vem* ze seznamu uzel daného indexu, stává se aktuálním uzlem u_i.
Jinak
    Vyber náhodný uzel ze seznamu.

Ulož startovní pozici a
označ aktuální uzel jako zpracovaný uzel.

Dokud existují nezpracované uzly:
    inicializuj seznam vzdáleností mezi aktuálním uzlem u_i a zbylých nezpracovaných uzlů.
    Pro nezpracovaný uzel u:
        spočítej vzdálenost mezi u_i a u a
        přidej ho do seznamu vzdáleností.

    Najdi nejkratší vzdálenost
    a *zjisti* index daného uzlu.

19:
20: *Přičti* nejkratší vzdálenost k sumě vah.
21: *Odeber* daný uzel ze seznamu nezpracovaných uzlů, stává se uzlem aktuálním,
22: a *označ* ho jako zpracovaný uzel.
23:
24: *Přičti* vzdálenost mezi prvním a posledním zpracovaným uzlem k sumě vah.
25: *Přidej* startovní uzel na konec seznamu zpracovaných uzlů.
26:
27: **Vrať** sumu vah, seznam zpracovaných uzlů.

3.1.3 Best Insertion

Tato funkce na základě seznamu uzlů vypočítá Hamiltonovskou kružnici metodou konstrukčních heuristik Best Insertion. Výsledkem je suma vah a seznam uzlů tvořící Hamiltonovskou kružnici.

Algorithm 3 best_insertion(seznam uzlů):

1: *Inicializuj* seznam nezpracovaných a zpracovaných uzlů a sumu vah.
2:
3: **Pro** 3 opakování:
4: *Vyber* náhodný nezpracovaný uzel
5: a *označ* ho jako zpracovaný uzel.
6:
7: *Ulož* startovní pozici.
8:
9: **Pro** každý zpracovaný uzel:
10: *spočítej* vzdálenost mezi aktuálním uzlem u_1 a uzlem následujícím u_2,
11: *přičti* vzdálenost k sumě vah.
12:
13: **Dokud** existují nezpracované uzly:
14: *Vytvoř* aktuální Hamiltonovskou kružnici,
15: *Vyber* náhodný nezpracovaný uzel, *stává se* uzlem aktuálním.
16: *Inicializuj* seznam možných přírůstků sumy vah.
17:
18: **Pro** každé možné zařazení uzlu do Hamiltonovské kružnice:
19: *vypočítej* možný přírůstek délky hrany,
20: a *přidej* ho do seznamu možných přírůstků sumy vah.
21:
22: *Najdi* nejmenší možný přírůstek
23: a *zjisti* index daného uzlu.
24:
25: *Zařaď* aktuální uzel na danou pozici,
26: *přičti* daný přírůstek k sumě vah.
27:
28: *Přidej* startovní uzel na konec seznamu zpracovaných uzlů
29:
30: **Vrať** sumu vah, seznam zpracovaných uzlů.

3.1.4 Nalezení nejlepšího řešení Nearest Neighbor

Tato funkce na základě seznamu uzlů projde veškeré možné Hamiltonovské kružnice zkonstruované metodou Nearest Neighbor a vybere to nejlepší (tedy s nejnižší sumou vah).

Algorithm 4 nn_best(seznam uzlů):

1: *Inicializuj* seznam možných sum vah a seznam možných seznamů zpracovaných uzlů (dále Hamiltonovskou kružnici).
2:
3: **Pro** každý možný startovní uzel:
4: *Vypočítej* sumu vah a Hamiltonovskou kružnici pomocí algoritmu Nearest Neighbor,
5:
6: *přidej* spočítanou sumu vah a Hamiltonovskou kružnici do příslušných seznamů.
7: *Najdi* nejmenší možnou sumu vah
8: *zjisti* index daného sumy vah,
9: a *najdi* Hamiltonovskou kružnici s nejmenší sumou vah
10:
11: **Vrať** sumu vah a Hamiltonovskou kružnici.

3.1.5 Nalezení co nejlepšího řešení Best Insertion

Tato funkce na základě seznamu uzlů a počtu opakování projde Hamiltonovské kružnice zkonstruované metodou Best Insertion a vybere to nejlepší (tedy s nejnižší sumou vah).

Algorithm 5 nn_best(seznam uzlů, počet opakování):

1: *Inicializuj* seznam možných sum vah a seznam možných seznamů zpracovaných uzlů (dále Hamiltonovskou kružnici).
2:
3: **Pro** zadaný počet opakování:
4: *Vypočítej* sumu vah a Hamiltonovskou kružnici pomocí algoritmu Best Insertion,
5: *přidej* spočítanou sumu vah a Hamiltonovskou kružnici do příslušných seznamů.
6:
7: *Najdi* nejmenší možnou sumu vah
8: *zjisti* index daného sumy vah,
9: a *najdi* Hamiltonovskou kružnici s nejmenší sumou vah
10:
11: **Vrať** sumu vah a Hamiltonovskou kružnici.

3.1.6 Ladění metody Best Insertion

Tato funkce se na základě seznamu uzlů, optimální sumy vah a hledaného koeficientu k snaží nalézt odpovídající řešení metody Best Insertion. Výsledkem je počet opakování a seznam vývoje minimálního koeficientu k.

Algorithm 6 bi_tuning(seznam uzlů, optimální suma vah, hledaný koeficient k):

Inicializuj seznam koeficientů k, seznam vývoje minimálního koeficientu k, a počet opakování.

Dokud je podmínka pravdivá:
aktualizuj počet opakování,
vypočítej sumu vah a Hamiltonovskou kružnici algoritmem Best Insertion,

vypočítej koeficient k,
na základě předchozích výsledků *najdi* minimální hodnotu k
a tu *přidej* do seznamu vývoje minimálního koeficientu k.

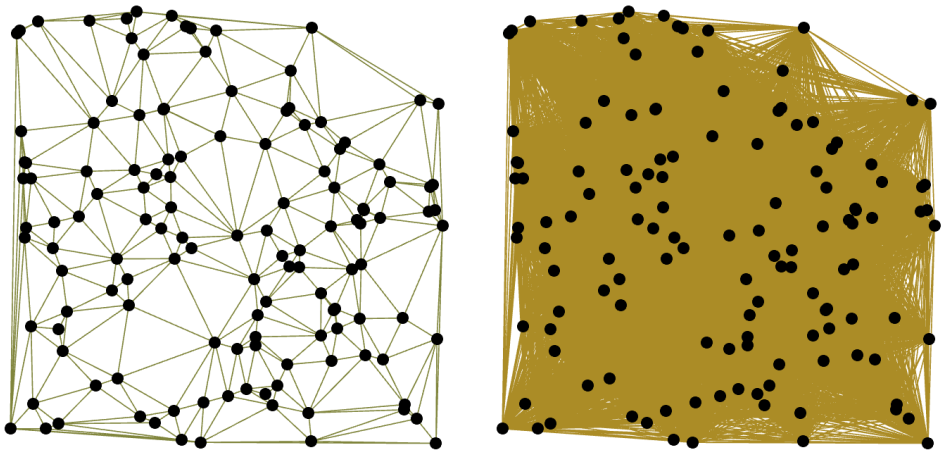
Pokud je koeficien k menší nebo roven hledanému koeficientu k:
Vrať počet opakování, seznam vývoje koeficientu k, aktuální Hamiltonovskou kružnici.

3.2 Network Analyst

Nejdříve je potřeba načíst *.csv soubor se souřadnicemi do ArcGIS Pro. Funkcí *Display XY* se uzly grafu vizualizují a převedou na *Feature class*. Dále je potřeba vygenerovat hrany mezi uzly, které poslouží jako síťový dataset *Network Dataset*, nad kterým budou probíhat výpočty. Byly otestovány dva síťové datasety:

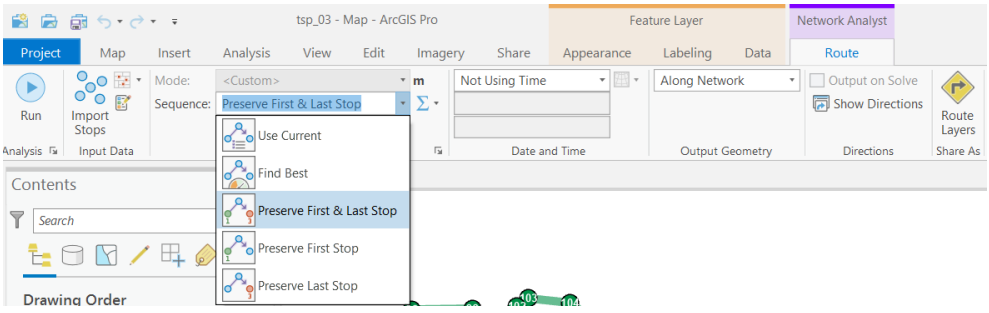
1. hrany vzniklé Delaunayho triangulací,
2. veškeré existující hrany mezi uzly.

Porovnání obou výstupů je na obrázku 4. Pro sestrojení prvního síťového datasetu byla použita funkce *Delaunay Triangulation* (dostupná v SW QGIS). Výsledkem jsou trojúhelníkové polygony, které je potřeba převést na liniové prvky funkcí *Polygon to Line* a následně je rozdělit v místech uzlů (*Split Line At Vertices*). Pro sestrojení druhé zmíněné varianty je nejdříve potřeba vytvořit tabulku souřadnic všech dvojic uzlů funkcí *Generate Near Table*. Následně funkcí *XY to Line* jsou na základě tabulky vytvořeny liniové prvky spojující všechny uzly. K vytvoření síťového datasetu je potřeba nejdříve vložit *Feature Class* hran do samostatného *Feature Datasetu* a poté použít funkci *Create Network Dataset*. Funkcí *Build Network* se vybuduje síť, nad kterou budou probíhat výpočty a vypočtou se váhy hran (v našem případě euklidovská vzdálenost).



Obrázek 4: Delaunayho triangulace (vlevo) a veškeré existující hrany mezi uzly (vpravo), zdroj: autor

Pro každý síťový dataset byla vytvořena vlastní vrstva okružní analýzy (*Network Analysis* > *Route*) Je nutné zvolit správný zdroj síťový dataset, nad kterým bude analýza probíhat. Do vrstvy okružní analýzy je také potřeba načíst vrstvu uzlů, skrz které bude potřeba najít nejkratší cestu (*Import Stops*. Parametry okružní analýzy jsou na obrázku 5.



Obrázek 5: Parametry okružní analýzy v SW ArcGIS Pro Network Analyst, zdroj: autor

V kapitole 2.5 je zmíněno, že výsledkem řešení problému obchodního cestujícího v SW ArcGIS je Hamiltonovská cesta, nikoliv kružnice. Je potřeba proto nejdříve vypočítat Hamiltonovskou cestu volbou parametru sekvence *Find Best*, díky které je nalezen vhodný startovní uzel, který se pro další krok zduplikuje a zařadí se na konec sekvence. Hledání nejkratší cesty je poté spuštěno znovu, ale tentokrát je jako parametr sekvence zvolena volba *Preserve First Last Stop*. Tímto krokem končí nalezená Hamiltonovská cesta na stejném místě a lze ji považovat za kružnici.

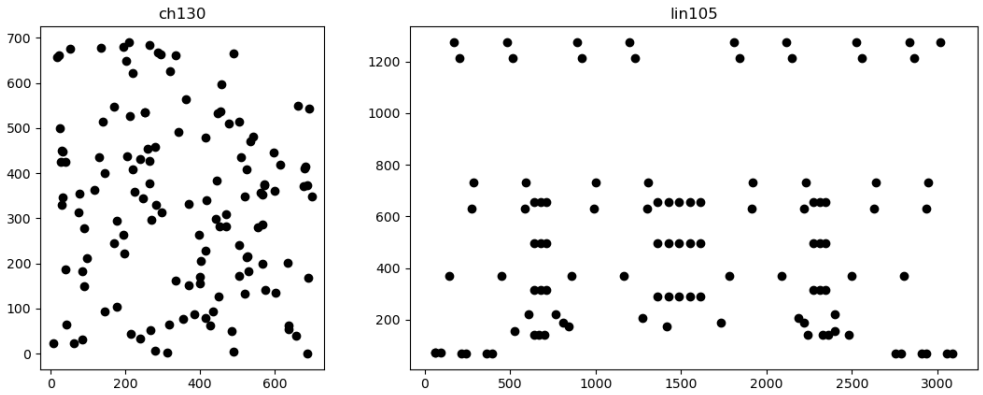
4 Experiments and results

4.1 Vstupní datasety

Jako vstupní datasety byly použity datasety z knihovny TSPLIB vytvořené Gerhardem Reineltem z Universität Heidelberg (Reinelt 1995). TSPLIB je knihovna vzorových dat pro řešení TSP a jeho variací. Celkově je v ní 111 problémů, které mají 14 až 85900 uzlů. Každý problém přestavuje soubor dat z geografie, elektrických obvodů nebo uměle vytvořených dat (Walker 2018). Z této knihovny byly vybrány dva datasety (obrázek 6), které mají více jak 100 uzlů, jsou použitelné pro symetrický TSP, jako váha hran je považována euklidovská vzdálenost ve 2D a bylo pro ně nalezeno optimální řešení:

1. ch130,
2. lin105.

Oba datasety jsou s velkou pravděpodobností synteticky vytvořeny. První, "ch130", se skládá ze 130 uzlů, přičemž je u něho poznámka "Churritz"(což může být tvůrce tohoto problému, ale bohužel nejsou dostupné žádné záznamy o tomto faktu). Druhý, "lin105", je zase problém 105 uzlů a je subsetem problému "lin318", který byl vytvořen dvojicí Lin-Kernighan. Ta má na svědomí vytvoření jedné z nejúčinnějších heuristik řešících TSP (Lin, Kernighan 1973).



Obrázek 6: Vstupní datasety: ch130 (vlevo) a lin150 (vpravo), zdroj: autor

4.2 Výsledky

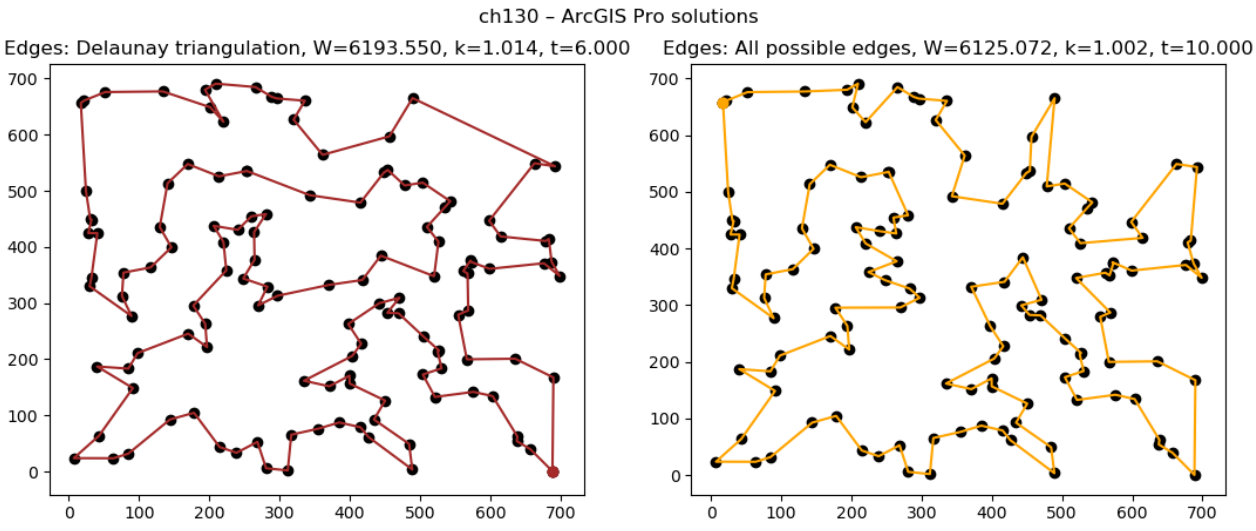
Řešení úlohy TSP v jazyce Python a v SW ArcGIS, popsaná v kapitole 3, byla otestována na datasetech z kapitoly 4.1. Nejdříve byly porovnány výsledky z ArcGIS Pro pro dva různé síťové datasety (obrázky 7 a 9). Následně byly porovnány výsledky z řešení v jazyce Python s optimálním řešením a s lepším řešením ze SW ArcGIS (obrázky 8 a 10). Výsledky jsou uspořádány v tabulkách 1 a 2. Zhodnocení výsledků je v kapitole 5.

4.2.1 ch130

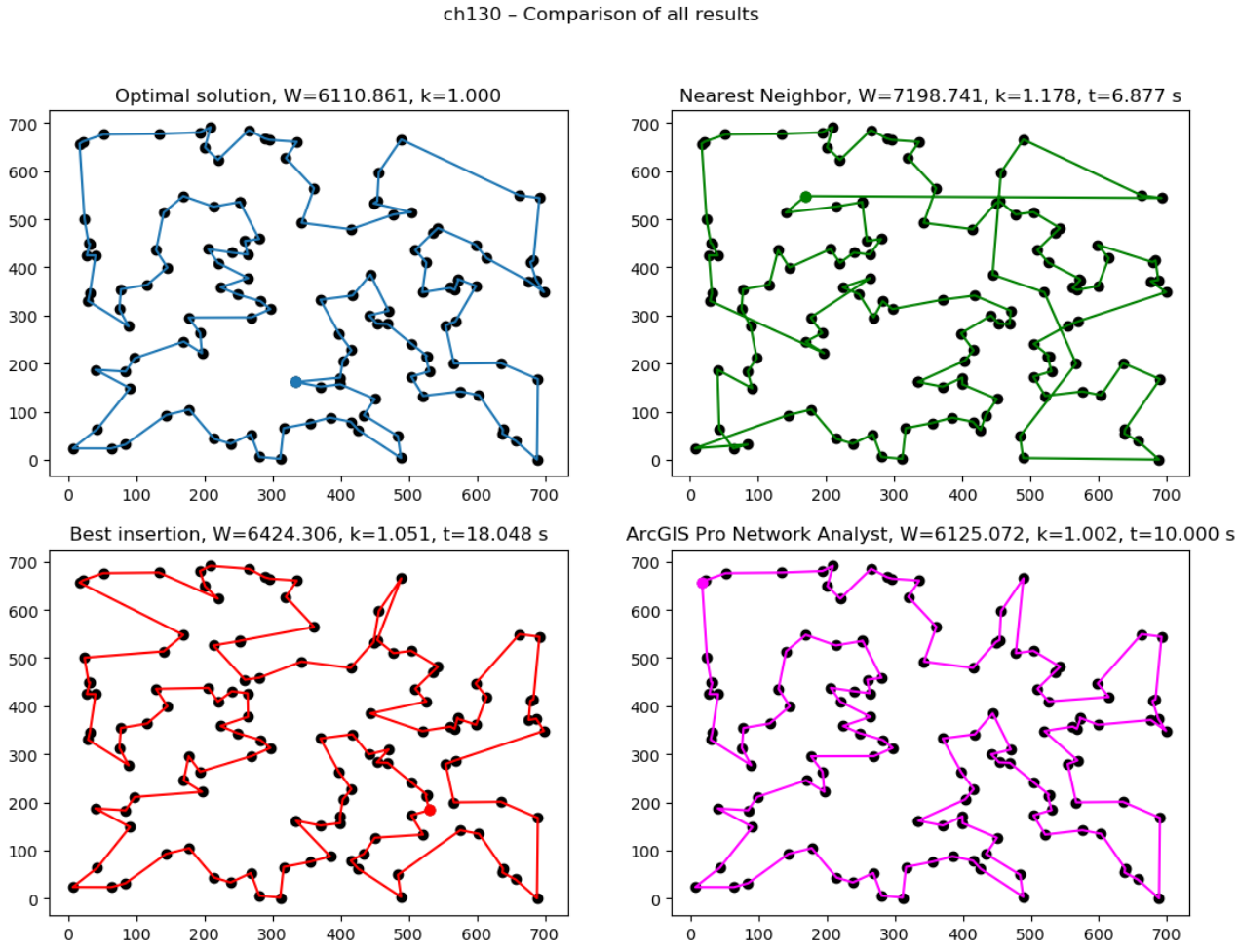
Pro dataset "ch130"byly vygenerována celkem 4 řešení: řešení v ArcGIS Pro pro Delaunayho triangulaci (obrázek 7 vlevo) a veškeré hrany (obrázek 7 vpravo), řešení konstrukční heuristikou Nearest Neighbor, kdy bylo vybráno nejlepší možné řešení (obrázek 8 vpravo nahoře), a řešení konstrukční heuristikou Best Insertion, kdy bylo vybráno nejlepší řešení v n opakování, kdy n je rovno počtu uzlů (obrázek 8 vlevo dole). Veškeré výsledky jsou shrnuty v tabulce 1.

Tabulka 1: Porovnání všech řešení pro dataset ch130, zdroj: autor

Metoda	W	k	$t[s]$
Optimal solution	6110,861	1,000	-
Nearest Neighbor	7198,741	1,178	6,877
Best Insertion (130x)	6424,306	1,051	18,048
ArcGIS Pro (DT)	6193,550	1,014	6,000
ArcGIS Pro (all edges)	6125,072	1,002	10,000



Obrázek 7: Řešení problému "ch130"v SW ArcGIS Pro, zdroj: autor



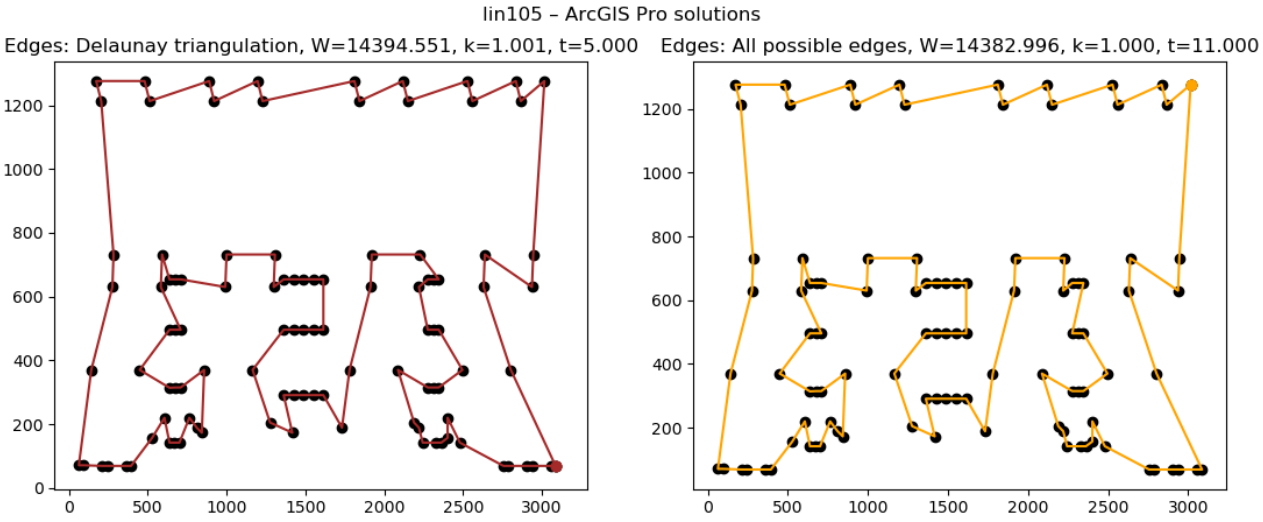
Obrázek 8: Porovnání řešení problému "ch130", zdroj: autor

4.2.2 lin105

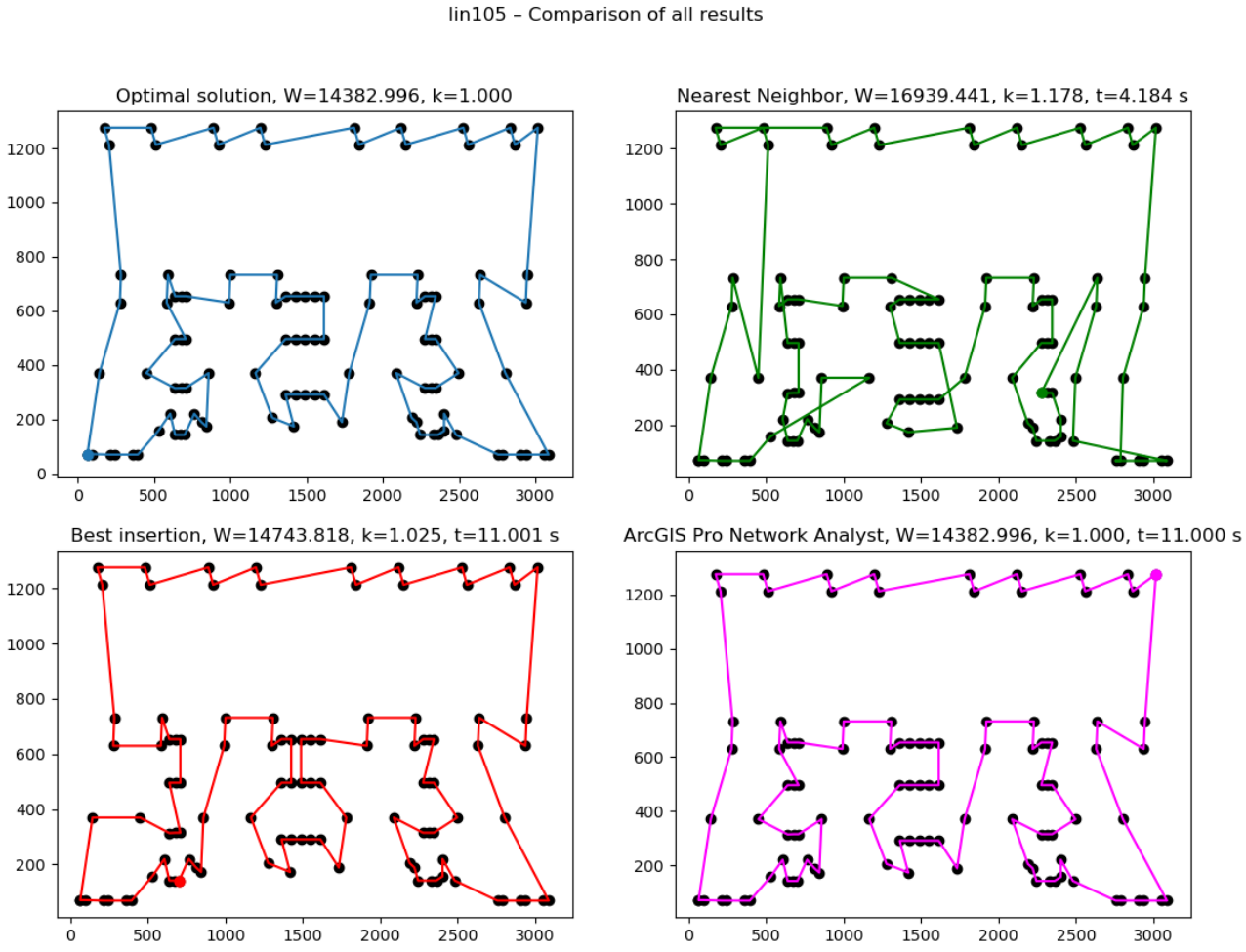
Pro dataset "ch130"byla rovněž vygenerována celkem 4 řešení: řešení v ArcGIS Pro pro Delaunayho triangulaci (obrázek 9 vlevo) a veškeré hrany (obrázek 9 vpravo), řešení konstrukční heuristikou Nearest Neighbor, kdy bylo vybráno nejlepší možné řešení (obrázek 10 vpravo nahoře), a řešení konstrukční heuristikou Best Insertion, kdy bylo vybráno nejlepší řešení v n opakování, kdy n je rovno počtu uzlů (obrázek 10 vlevo dole). Veškeré výsledky jsou shrnuty v tabulce 2.

Tabulka 2: Porovnání všech řešení pro dataset lin105, zdroj: autor

Metoda	W	k	$t[s]$
Optimal solution	14382,996	1,000	-
Nearest Neighbor	16939,441	1,178	4,184
Best Insertion (105x)	14743,818	1.025	11,001
ArcGIS Pro (DT)	14394,551	1,001	5,000
ArcGIS Pro (all edges)	14382,996	1,000	11,000



Obrázek 9: Řešení problému "lin105"v SW ArcGIS Pro, zdroj: autor



Obrázek 10: Porovnání řešení problému "lin105", zdroj: autor

5 Závěr

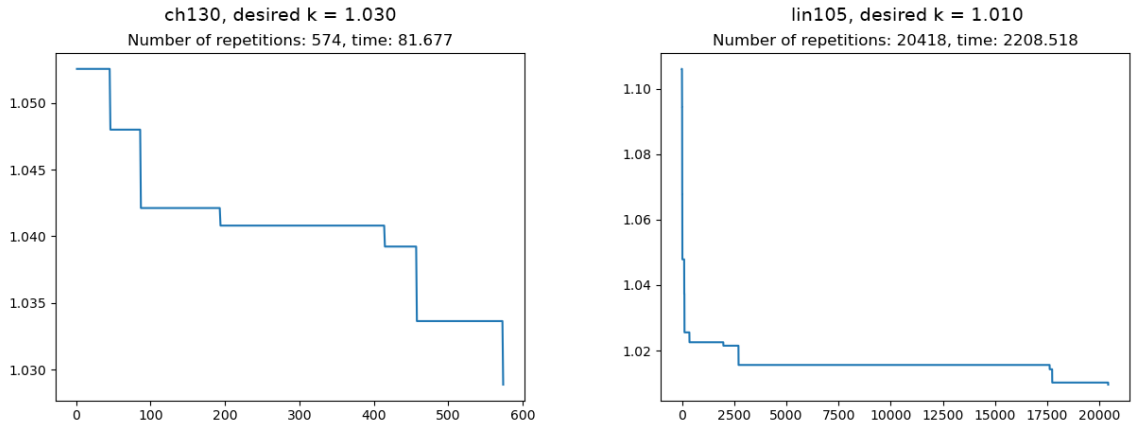
Cílem úlohy bylo pomocí metod konstrukčních heuristik Nearest Neighbor a Best Insertion nalézt nad množinou uzlů nejkratší cestu, která vychází z libovolného uzlu, každý z uzlů navštíví pouze jedenkrát, a vrací se do uzlu výchozího. Jinak řečeno úkolem bylo nalézt Hamiltonovskou kružnici. Úloha byla následně řešena i nástrojem Network Analyst v SW ArcGIS Pro.

Za tímto účelem bylo napsáno 6 funkcí v jazyce Python a byl navržen postup v SW ArcGIS Pro. V kapitole 2 byl čtenář seznámen se základními pojmy teorie grafů, byl zasvěcen do problému obchodního cestujícího a byly představeny dvě metody konstrukčních heuristik: Nearest Neighbor a Best Insertion. V kapitole 3 byla navržena implementace v jazyce Python a v SW ArcGIS Pro. V kapitole 4 byly představeny testovací datasety a byly publikovány výsledky na 4 obrázcích a ve 2 tabulkách.

U datasetu "ch130"metoda Nearest Neighbor dosáhla délky kružnice $W = 7198{,}741$ což je o 17,8% delší než optimální řešení (to činí $W = 6110{,}861$). Metoda Best Insertion si vedla o něco lépe, kružnice byla dlouhá $W = 6424{,}306$, tedy o 5,1% delší než optimum. Nearest Neighbor je zase oproti Best insertion delší o 12,1%. U datasetu "lin105"byla kružnice vygenerovaná metodou Nearest Neighbor ($W = 16939{,}441$) také delší o 17,8% oproti optimálnímu řešení ($W = 14382{,}996$). Best Insertion si vedla tentokrát ještě lépe, kdy oproti optimu zaostávala pouze o 2,5% ($W = 14743{,}818$). Best Insertion je pak oproti Nearest Neighbor kratší o 13,0%. Výpočetní náročnost je však na straně metody NN, jelikož BI zpravidla trvalo 3x kratší dobu. Doba výpočtu by se u BI dala zkrátit počtem opakování.

Řešení v ArcGIS Pro ovšem stále mělo navrch oproti vlastní implementaci v jazyce Python. Volba Delaunayho triangulace jakožto síťovým datasetem vedla ke kratší době výpočtu, jelikož byl omezen počet hran, ale rovněž vedla k nepatrně horším výsledkům, kdy délka kružnice u prvního datasetu ($W = 6193{,}550$) byla o 1,4% delší než optimum. Výpočet nad všemi možnými hranami se prodloužil přibližně dvojnásobně oproti výpočtu nad Delaunayho triangulací, ale dosáhl téměř perfektního výsledku, přičemž u druhého datasetu bylo dokonce dosáhnuto optimálního řešení.

Obě metody se řídí tzv. greedy strategií, neboli ani jedna z metod nezaručuje nalezení globálního minima, pouze minima lokálního. Metody tak generují řešení přibližné, nikoliv optimální, jak je už diskutováno v kapitole 2.3. Výpočetní náročnost u obou metod vychází cca. na $O(N^2)$. Nejnáročnější krok metody Nearest Neighbor představuje opakované hledání nejbližšího uzlu. Výpočet by se mohl urychlit, například využitím přibližného nejbližšího souseda (= *Aproximate Nearest Neighbor, ANN*), která je implementována například v knihovně Flann. Další možnost je nejdříve zkonstruovat Delaunayho triangulaci, čímž se sníží počet sousedních uzlů. Výsledek ovšem vylepšit nelze, jelikož má algoritmus deterministické chování, pokud se opakuje stejná volba počátečního bodu. Počet možných řešení je tedy rovno počtu uzlů v grafu. Proto bylo do mé implementace zahrnut krok, který testuje všechna možná řešení a vybírá to s minimální hodnotou W, aby se bral v potaz ten nejlepší možný výsledek pro takto nedokonalou metodu. Nevýhodou této metody je určitě malá účinnost a také "nedokonalá kružnice", kdy může docházet k protínání cesty. K tomu došlo i u nejlepších řešení u obou datasetů.



Obrázek 11: Minimalizace koeficientu k opakováním algoritmu Best Insertion, zdroj autor

Best Insertion je metoda nedeterministická, jelikož náhodným výběrem počátečních uzlů vzniká pokaždé jiné řešení. Lepšího výsledku tak může být dosáhnuto opakovaným spouštěním algoritmu. To bylo bráno v potaz při mé implementaci, kdy algoritmus proběhne tolikrát, kolik je zadán počet opakování (v tomto případě je počet opakování roven počtu uzlů, tedy stejný počet opakování jako u Nearest Neighbor). Důkaz budiž na obrázku 11, kdy byl algoritmus opakovaně spouštěn, dokud nebyla dosáhnuta požadovaná hodnota k. Požadované hodnoty k skutečně lze dosáhnout, ale délka trvání je náhodná, a pokud je požadována co největší minimalizace hodnoty k, tak délka trvání je i dost dlouhá. Derandomizování metody lze dosáhnout například tím, že se v každém kroku bude přidávat do kružnice takový uzel, který minimalizuje hodnotu Δw. Za cenu vyšší výpočetní náročnosti však nemusí být docíleno lepšího výsledku. Dalším vylepšením je například aproximace startovní kružnice konvexní obálkou. V tomto případě dojde ke zkrácení cesty o dalších zhruba 5 %.

Jednoduché metody konstrukčních heuristik Nearest Neighbor a Best Insertion se ukázaly jako velice snadno implementovatelné algoritmy, které ovšem sužují určité limitace. Ani jedna z metod nedosahuje výsledků jako dosahuje nástroj Network Analyst v SW ArcGIS Pro. Vhodnější by proto bylo využití komplexnějších heuristik, jako je například Lin-Kerninghan heuristika, která je zmíněná v kapitole 4.1.

6 Seznam literatury

Použité vzorce a teorie jsou převzaté z přednášek předmětu *Geoinformatika* a z BAYER, T. (2022).

BAYER, T. (2021): Úloha: Řešení problému obchodního cestující, Výukový materiál k předmětu Geoinformatika, https://web.natur.cuni.cz/ bayertom/images/courses/Geoinf/tsp_zadani.pdf.

BAYER, T. (2022): Problém obchodního cestujícího, konstrukční heuristiky: stručný návod na cvičení, Výukový materiál k předmětu Geoinformatika, https://web.natur.cuni.cz/ bayertom/images/courses/Geoinf/tsp_uloha.pdf.

KOVÁŘ, P. (2016): Úvod do Teorie grafů, Matematika pro inženýry 21. století, https://homel.vsb.cz/ kov16/files/uvod_do_teorie_grafu.pdf.

LIN, S., KERNIGHAN, B. (1973): An Effective Heuristic Algorithm for the Traveling-Salesman Problem. Operations Research, 2, 21, 498–516.

REINELT, G. (1995): TSPLIB 95, Institut für Angewandte Mathematik, Universität Heidelberg, Germany, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

ŠEDA, M. (2003): Teorie grafů, Výukový materiál k předmětu Teorie grafů, http://www.uai.fme.vutbr.cz/ mseda/TG03_MS.pdf.

WALKER, J. (2018): Simulated Annealing: The Travelling Salesman Problem, https://www.fourmilab.ch/documents/travelling/anneal/.

7 Přílohy

- Skripty v jazyce Python
  - `main.py`
  - `tsp_funkce.py`
- Spočítané kružnice
  - `ch130_tour.csv`
  - `ch130_tour_gis_all.csv`
  - `ch130_tour_gis_dt.csv`
  - `lin105_tour.csv`
  - `lin105_tour_gis_all.csv`
  - `lin105_tour_gis_dt.csv`
- Testovací datasety
  - `ch130.csv`
  - `lin105.csv`