# **Zadanie**

**Úloha: Řešení problému obchodního cestujícího**

*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 u uzlů navštíví pouze jedenkrát, a vrací se do uzlu výchozího. Využijte níže uvedené metody konstrukčních heuristik:

* Nearest Neighbor,
* Best Insertion.

Výsledky porovnejte s výstupem poskytovaným nástrojem Network Analyst v SW ArcMap. Otestování proveďte nad dvěma zvolenými datasety, které by měly obsahovat alespoň 100 uzlů. Jako vstup použijte existující 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.

# **Problém obchodného cestujúceho**

* 1. **Stručný opis problému a motivácia pre jeho riešenie**

Problém obchodného cestujúceho (ang. Travelling Salesman Problem) spadá do kategórie kombinatorických optimalizačných úloh. Tento problém ďalej patrí do množiny tzv. NP-ťažkých problémov. Tieto (pravdepodobne) nie je možné vyriešiť exaktne a efektívne v čase polynomiálnom (ang. *problem of* *polynomial time complexity*) (Arora, 2003). Problém obchodného cestujúceho je veľmi známy kvôli svojmu využitiu a prepojeniu s problémami reálneho sveta ako napríklad v logistike (plánovanie zásobovacích trás), pri dizajnovaní mikročipov alebo ako vedľajší problém spojený so sekvenovaním genómu (Bergel, 2020; Borovska, 2022).

* 1. **Stručný popis riešenia v rámci grafovej teórie**

Problém obchodného cestujúceho spočíva v nájdení najkratšej cesty spájajúcej všetkých *n* miest množiny *M* tak, že každé mesto je navštívený presne jeden raz a cesta začína a končí v tom istom bode. V grafovej teórii predstavuje optimálne riešenie problému najkratšia Hamiltonovská kružnica *Kh* úplného neorientovaného a ohodnoteného grafu *G*, ktorého uzly reprezentujú mestá a hrany naopak cesty medzi nimi. Vzdialenosti medzi jednotlivými mestami predstavujú ohodnotenie hrán (Bayer, 2021). Vzhľadom na aplikovanie tohto problému v konkrétnom odbore môžu byť pojmy mesto a vzdialenosť nahradené inými pojmami ako napr. zákazníci alebo náklady časové/finančné.

* + 1. **Prístupy k riešeniu problému**

K problému obchodného cestujúceho je možné pristupovať exaktne rovnako ako aj heuristicky:

* Exaktný prístup vedie k dokázateľne najkratšej Hamiltonovskej kružnice *Kh* (globálnemu minimu riešenia úlohy) predpokladajúc ľubovoľne dlhý čas pre výpočet. Jeho riešením by bol výpočet dĺžok všetkých možných ciest a následný výber tej najkratšej. Pri tomto postupe narastá čas výpočtu exponenciálne už pri minimálnych prírastkoch vstupných bodov. Doposiaľ neexistuje časovo efektívny algoritmus pre exaktné riešenie úlohy (‘Problem des Handlungsreisenden’, 2021).
* Heuristické postupy riešenia problému cestujúceho obchodného spočívajú v nájdení čo najlepšej aproximácie k  optimálnemu riešeniu v čo najkratšom čase. Vo všeobecnosti avšak nie je možné výsledné riešenie a jeho kvalitu nijako overiť a teda toto riešenie môže byť ľubovoľne zlé. Existuje viacero metód vedúcich k prijateľnému výsledku. V rámci zadania sme sa stretli s dvoma konkrétnymi a to metódami *Nearest Neighbour* a *Best Insertion*.
  + 1. **Popis metódy *Nearest Neighbour***

Vo všeobecnosti spočíva algoritmus *Nearest Neighbour* v nasledujúcich jednoduchých krokoch (Bayer, 2021):

1. Náhodne je vybraný ľubovoľný počiatočný uzol *us* hľadanej kružnice *Kh*.
2. Z počiatočného uzlu je nájdená hrana medzi ním a ďalším uzlom *ui* – tzv. najbližším susedom uzlu *us* – s najnižším ohodnotením *w(us, ui).*
3. Uzol *us* sa označuje ako uzatvorený, nájdený uzol *ui* sa stáva výstupným uzlom *us* a ohodnotenie cesty medzi nimi *w(us, ui)* je prirátané k celkovej dĺžke *W* kružnice *Kh.*
4. Krok 2 a 3 sa opakuje pokiaľ nie sú všetky uzly označené za spracované.
5. Hrana medzu uzlom priradeným do množiny uzlov Hamiltonovskej kružnice *Kh* ako posledný a počiatočným uzlom *us* sa pripočíta k dĺžke *W.*

Obrázky 1-3 zobrazujú 3 z možných riešení pre čo najlepšiu Hamiltonovskú kružnicu na skúšobnom datasete C, pričom C je definovaný nasledovne.

﻿C = [[12, 3], [4, 15], [0, 1], [5, 10], [7, 7], [15, 4], [20, 19], [-3, 0], [30, 37]].

Z obrázkov je možno pozorovať nedostatok algoritmu v podobe pretínania hrán.

* + 1. **Popis metódy *Best Insertion***

Algoritmus *Best Insertion* spočíva v nasledujúcich krokoch (Bayer, 2021):

1. Prvotne je vybraná trojica ľubovoľných uzlov, ktoré tvoria začiatočnú Hamiltonovskú kružnicu *Kh.* (Obr. 4)

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedChart, line chart

Description automatically generated

*Obrázok 1-3 Grafy Hamiltonovskej kružnice vygenerovanej pomocou Nearest Neighbour algoritmu v Pythone*

1. Inicializačná dĺžka Hamiltonovskej kružnice je daná súčtom hrán medzi bodmi.
2. Počiatočné body sú označené za spracované a z množiny doposiaľ nespracovaných bodov je náhodne vyberaný bod *u.*
3. Z bodov doposiaľ vytvorenej kružnice hľadáme takú hranu *ui, ui+1*, aby hrana tvorená z uzlov *ui, u*a *ui+1* bola čo najmenším predĺžením počiatočnej kružnice. Pri tomto kroku je používaný princíp trojuholníkovej nerovnosti.
4. Uzol *u* sa označuje za spracovaný a je pridaný do Hamiltonovskej kružnice na správnu pozíciu a dĺžka kružnice je aktualizovaná.
5. Krok 3-5 sa opakuje pokiaľ existujú v množine bodov nespracované body.

Obr. 4 zobrazuje prvý krok algoritmu *Best Insertion* pre vyššie definovaný dataset C*.* Obrázky 5-6 ďalej zobrazujú 2 z možných riešení pre čo najlepšiu Hamiltonovskú kružnicu pre dataset C. Práve aplikovanie princípu trojuholníkovej nerovnosti eliminuje nežiadúci efekt pretínania hrán, aj keď nie úplne a vyskytnú sa aj riešenia s pretnutými hranami.

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated*Obr. 4 Inicializácia počiatočnej Hamiltonovskej kružnice*

Chart, line chart

Description automatically generated

*Obr. 5-6 Grafy Hamiltonovskej kružnice vygenerovanej pomocou Best Insertion algoritmu v Pythone*

# **Výsledky experimentov**

* 1. **Riešenie Hamiltonovskej kružnice**

Úloha bola riešená v programovacom jazyku Python. Podľa zadania boli vytvorené 2 skriptá, jeden vypočítanie dĺžky Hamiltonovskej kružnice za použitia *Nearest Neighbour* algoritmu a jeden pomocou algoritmu *Best Insertion*. Obe skriptá obsahujú nasledovné funkcie –

* Funkcia ﻿**openTXT\_exportCoordinates** pre otvorenie a extrahovanie dát z textového súboru
* Samotný algoritmus *Nearest Neighbour/ Best Insertion*
* Funkcia **﻿path** pre získanie pozícií (súradníc) uzlov množiny Hamiltonovskej kružnice v správnej postupnosti
* Funkcia ﻿**visualize\_hamiltionianP** zobrazujúca graf nájdenej Hamiltonovskej kružnice
* Funkcia **﻿iteration\_NearestNeighbour/ ﻿iteration\_BestInsertion** umožňujúca nastavenie ľubovoľného maximálneho počtu iterácii daných algoritmov a ukladanie dosiahnutých výsledkov.
  1. **Použité datasety**

Za pomoci vytvorených skrípt boli vygenerované Hamiltonovské kružnice pre datasety so –

1. súradnicami 110 autobusových zastávok v oblasti Vysokých Tatier.
2. súradnicami 142 reštaurácii v tej istej oblasti.
   1. **Výsledky algoritmov pre dataset A)**
      1. **Metóda *Nearest Neighbour***

Keďže je táto metóda deterministická vzhľadom na výber počiatočného bodu, je možné vygenerovať celkovo *n* počet možných riešení, kde *n* predstavuje počet vstupných bodov. Vzhľadom na náhodný výber inicializačného uzlu a teda celkového náhodného charakteru vygenerovaných riešení bol algoritmus iterovaný 200-krát. V prvých 3 riadkoch Tabuľky 1 sú zobrazené hodnoty najkratších nájdených ciest zaokrúhlené na 3 desatinné miesta. Bod s indexom 15 sa javí byť jedným z optimálnych adeptov pre inicializačný uzol. Posledné 3 riadky predstavujú najhoršie dosiahnuté výsledky.

* + 1. **Metóda *Best Insertion***

Ako inicializačné body prvotnej Hamiltonovskej kružnice boli zvolené náhodné 3 body datasetu. Algoritmus bol spustený 200-krát. Pri tomto množstve iterácií narástol výpočtový čas približne 2.5-násobne oproti výpočtovému času algoritmu Nearest Neighbour (Tabuľka 1). Tabuľka 1 opäť sumarizuje 3 najlepšie a najhoršie výsledky.

*Tabuľka 1 Výsledky oboch algoritmov pre dataset A) pri celkovom počte 200 iterácií*

|  |  |
| --- | --- |
| **Počet iterácií** | 200 |
|  |  |
| **Dĺžka Hamiltonovskej kružnice [km]** | |
| *Nearest Neighbour* | *Best Insertion* |
| ﻿148,178 | ﻿138,867 |
| ﻿148,407 | ﻿139,014 |
| ﻿148,418 | ﻿154,514 |
|  |  |
| ﻿184,218 | ﻿160,528 |
| ﻿182,887 | ﻿164,950 |
| ﻿182,495 | ﻿174,826 |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Výpočtový čas [sec]** | 0,992 | ﻿2,634 |

* 1. **Výsledky algoritmov pre dataset B)**

Tabuľka 2 sumarizuje dosiahnuté výsledky pre dataset B pre obe metódy. Výpočtový čas narástol opäť o približne 2.8-násobok.

*Tabuľka 2 Výsledky oboch algoritmov pre dataset B) pri celkovom počte 200 iterácií*

|  |  |
| --- | --- |
| **Počet iterácií** | 200 |
|  |  |
| **Dĺžka Hamiltonovskej kružnice [km]** | |
| *Nearest Neighbour* | *Best Insertion* |
| ﻿ ﻿192.226 | ﻿ ﻿178.259 |
| ﻿ ﻿192.996 | ﻿ ﻿178.621 |
| ﻿ ﻿193.321 | ﻿ ﻿180.806 |
|  |  |
| ﻿ ﻿﻿ ﻿236.053 | ﻿ ﻿207.408 |
| ﻿ ﻿234.709 | ﻿ ﻿206.026 |
| ﻿ ﻿234.682 | ﻿ ﻿203.038 |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Výpočtový čas [sec]** | ﻿1.741 | ﻿ ﻿ ﻿4.822 |

# **Porovnanie so softwarovým riešením**

Dosiahnuté výsledky boli porovnané s dĺžkami uzavretej cesty pre oba datasety vypočítanej v SW ArcGIS PRO. V ArcGIS bolo nutné určiť počiatočnú zastávku, ktorou bol prvý bod datasetu. Preto boli z dosiahnutých výsledkov vybrané výsledky, kde pre metódu *Nearest Neighbour* bol počiatočný bod s indexom 0 a pre metódu *Best Insertion* počiatočné body s indexmi 0, 1 a 2 pre dosiahnutie čo najlepšej možnej porovnateľnosti. (Tabuľka 3). Tabuľka 4 zobrazuje hodnoty kritéria *k*pri vzájomnom porovnávaní metód.

*Tabuľka 3 Výsledky a porovnanie s ArcGIS PRO pre oba datasety*

|  |  |
| --- | --- |
| **Dataset A** | Dĺžka [km] |
| *ArcGIS PRO* | 161,394 |
| *Nearest Neighbour* | 177,797 |
| *Best Insertion* | 153,647 |

|  |  |
| --- | --- |
| **Dataset B** | Dĺžka [km] |
| *ArcGIS PRO* | 232.510 |
| *Nearest Neighbour* | ﻿223.326 |
| *Best Insertion* | ﻿﻿190.185 |

*Tabuľka 4 Hodnoty kritéria k pre vzájomné porovnanie výsledkov*

|  |  |
| --- | --- |
| **Dataset A** | **k [%]** |
| *NN vs ArcGIS* | 110,16 |
| *BI vs ArcGIS* | 95,2 |
| *NN vs BI* | 115,72 |

|  |  |
| --- | --- |
| **Dataset B** | **k [%]** |
| *NN vs ArcGIS* | 96,05 |
| *BI vs ArcGIS* | 81,8 |
| *NN vs BI* | 117,43 |

# **Záver a možné vylepšenia**

Podľa očakávaní generoval algoritmus *Best Insertion* lepšie výsledky ako algoritmus *Nearest Neighbour*. Algoritmus *Best Insertion* generoval takisto lepší výsledok ako výsledok dosiahnutý v ArcGIS Pro SW. Z tohto porovnania sa ale nedá viesť k záveru, že vytvorený algoritmus *Best Insertion* je lepší ako algoritmus používaný v SW ArcGIS Pro. Nutnosť nastavenia počiatočnej zastávky pre určenie najkratšej možnej cesty v ArcGIS Pro spôsobuje, že výsledok ze deterministicky určený na základe tejto zastávky. Je možné, že existuje spôsob, ako v SW dosiahnuť lepší výsledok ako dosiahnutý.

Možné zlepšenie algoritmu *Best Insertion* by mohla zabezpečiť cielená voľba počiatočných bodov podľa konvexnej obálky datasetu. Funkcia určujúca tieto body konvexnej obálky by mohla byť zaradená do skripta.

Vzhľadom na náhodný výber uzlov a teda celkový náhodný charakter kódu by bolo možné dosiahnuť presnejšieho výsledku pre porovnanie s výsledkom z ArcGIS Pro, ak by oba algoritmy boli iterované aspoň 10-krát (pri *Nearest Neighbour* s identickým počiatočným bodom ako bol v ArcGIS Pro, pri *Best Insertion* s prvými bodmi datasetu ako iniciálnymi) a následné výsledky by boli spriemerované, mohla by byť dosiahnutá lepšie porovnateľnosť s výsledkom z ArcGIS Pro.

Na záver, funkcia, ktorá by otvárala priamo shapefile a nie len textovú verziu s *x-* a *y-*súradnicami by urýchlila proces predpracovania dát k následnému využitiu v algoritmoch.

# **Bibliografia**

Arora, S. (2003) ‘Approximation schemes for NP-hard geometric optimization problems: a survey’, *Mathematical Programming*, 97(1), pp. 43–69. doi:10.1007/s10107-003-0438-y.

Bayer, T. (2021) ‘Problém obchodního cestujícího, konstrukční heuristiky’ Available at: https://web.natur.cuni.cz/~bayertom/index.php/9-teaching/37-geoinformatika (Accessed: 4. Januray 2022)

Bergel, A. (2020) ‘The Traveling Salesman Problem’, in Bergel, A. (ed.) *Agile Artificial Intelligence in Pharo: Implementing Neural Networks, Genetic Algorithms, and Neuroevolution*. Berkeley, CA: Apress, pp. 209–224. doi:10.1007/978-1-4842-5384-7\_10.

Borovska, P. (2022) ‘Solving the Travelling Salesman Problem in Parallel by Genetic Algorithm on Multicomputer Cluster’.

‘Problem des Handlungsreisenden’ (2021) *Wikipedia*. Available at: https://de.wikipedia.org/w/index.php?title=Problem\_des\_Handlungsreisenden&oldid=217972541 (Accessed: 6 January 2022).