VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ŘEŠENÍ PROBLÉMU KANADSKÉHO CESTUJÍCÍHO

SOLVING CANADIAN TRAVELLER PROBLEM

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Dávid Pavlovič

AUTHOR

VEDOUCÍ PRÁCE — RNDr. Jiří Dvořák, CSc.

SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Dávid Pavlovič**
Studijní program:	Strojírenství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**RNDr. Jiří Dvořák, CSc.**
Akademický rok:	2018/19

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Řešení problému kanadského cestujícího

Stručná charakteristika problematiky úkolu:

Problém kanadského cestujícího (CTP) lze charakterizovat následovně. Cestující hledá cestu z výchozí do cílové pozice, přičemž má k dispozici mapu silniční sítě. Problém spočívá v tom, že některé silnice mohou být neprůjezdné (v Kanadě např. kvůli sněhové bouři), ale to cestující zjistí teprve tehdy, když dorazí na začátek takového úseku. Tento problém lze chápat jako problém navigace robotu v částečně neznámém prostředí.

Cíle bakalářské práce:

Vypracovat přehled typů daného problému a existujících metod řešení.
Implementovat vybrané metody.
Provést a vyhodnotit ověřovací a srovnávací experimenty.

Seznam doporučené literatury:

BAR-NOY, A., SCHIEBER, B. The Canadian Traveller Problem. Proceedings of Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 91), USA, pp. 261-270, 1991.

HUANG, Y., LIAO, C.-S. The Canadian Traveller Problem Revisited. Proceedings of the 23rd International Symposium on Algorithms and Computation (ISAAC 2012), K.-M. Chao, T.-S. Hsu, and D.-T. Lee (Eds.): ISAAC 2012, LNCS 7676, pp. 352-361, 2012.

XU, Y. F. et al. The Canadian Traveller Problem and Its Competitive Analysis. Journal of Combinatorial Optimization, vol. 18, no. 2, pp. 195-205, 2009.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2018/19

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Táto práca sa zaoberá riešením problému kanadského cestujúceho. Predstavme si, že cestujúci má mapu, na ktorej je každá cesta spojená s časom potrebným na jej prejdenie. Táto mapa však nemusí byť úplne spoľahlivá a na niektorých úsekoch môže byť tamojšími podmienkami cesty spôsobené, že čas na prejdenie daného úseku bude dlhší, alebo že priechod bude nemožný. Práca sa zaoberá prehľadom typov tohto problému a ich riešeniami. Ďalej sa zaoberá popisom dvoch aplikácií implementovaných v jazyku Python, ktoré slúžia na overenie daných stratégií riešenia problému. Na záver sú vyhodnotené dané experimenty a porovnaná efektivita daných stratégií.

ABSTRACT

This thesis deals with Canadian traveller problem. Imagine a traveller that have a map on which every road is associated with time that is needed to get through this road. Hovewer, this map may not be totally reliable, and the time needed to pass through on some of the roads may be different due to bad road conditions, or the pass will be impossible. This thesis deals with type overview of this problem and the solutions. Further, the thesis deals with the description of two applications implemented in Python, which serves on verification of the strategies. The final part contains experiments and comparison of effectiveness of selected strategies.

KLÍČOVÁ SLOVA

Problém kanadského cestujúceho, stochastická optimalizácia, adaptívna stratégia

KEYWORDS

Canadian traveller problem, stochastic optimization, adaptive strategy

BIBLIOGRAFICKÁ CITACE

PAVLOVIČ, D. *Řešení problému kanadského cestujícího,* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí diplomové práce RNDr. Jiří Dvořák, CSc.

PODĚKOVÁNÍ

Týmto by som rád poďakoval mojim rodičom za ich neustálu podporu a môjmu kamarátovi Ondrejovi. Zároveň by som rád poďakoval vedúcemu mojej bakalárskej práce pánovi RNDr. Jiřímu Dvořákovi, CSc. za jeho odborné pripomienky a trpezlivosť.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením RNDr. Jiřího Dvořáka, CSc. a s použitím literatury uvedené v seznamu literatury.

V Brně dne 20. 5. 2019 ……

Dávid Pavlovič

Obsah

1 ÚVOD

1.1 Úvod do problému kanadského cestujúceho

Problém kanadského cestujúceho (CTP) bol predstavený Papadimitriom a Yannakisom v roku 1991 [2]. Autori sa zaoberali problémom hľadania najkratšej trasy v známom prostredí s nekompletnými informáciami o jeho stave. Tento problém možno charakterizovať následovne: cestujúci sa snaží dostať z počiatočného bodu *A* do cieľa *B*. Má so sebou kompletnú mapu ciest, ktoré tieto dva body spájajú, aj s ich cenami. Vopred však nie je známe, či sú všetky tieto cesty prejazdné (môžu byť uzatvorené kvôli vplyvom počasia, alebo spadnutým stromom). To, či je daná cesta priechodná, cestujúci spozoruje až keď sa dostane k danej ceste. Problémom teda zostáva navrhnúť časovo čo najefektívnejšiu stratégiu volby trasy z počiatočného bodu *A* do cieľového bodu *B*.

Daný problém je tiež možné demonštrovať na grafe. Predstavme si, že cestujúci pozná graf $G(V,E)$, v ktorom uzly V korešpondujú s určitými miestami na mape (napríklad mestami) a hrany E značia cesty, ktoré sa medzi uzlami nachádzajú. Navyše každá hrana E je asociovaná s konkrétnou reálnou nezápornou dĺžkou $l(e)$, interpretovanou ako čas potrebný na prejdenie tejto hrany (cesty) e. Ako bolo už predtým spomenuté, cestujúci disponuje mapou, ktorá však nie je úplne spoľahlivá, pretože niektoré cesty môžu byť v stave, že priechod nimi nieje možný a tento stav cesty bude odhalený až keď cestujúci príde k tejto ceste. Problémom zostáva navrhnúť optimálne riešenie tejto situácie pri cestovaní z určitého počiatočného bodu s do cieľového bodu t.

Hlavný problém pri určovaní dobrej cestovnej stratégie pramení z on-line stavu problému: rozhodnutia musia byt založené na neúplných informáciach, bez znalosti o blokovaných cestách. Všimnime si, že vymyslieť off-line cestovnú stratégiu je jednoduché: ak máme informácie o všetkých neprejazdných cestách v predstihu, optimálne riešenie je dané najkratšou prejazdnou cestou z počiatočného bodu s do cieľového bodu t.

Dizajnom algoritmov pre on-line verziu tohto problému bolo v poslednej dobe venovanej veľa pozornosti. Kvalita on-line algoritmov je prevažne meraná dvoma kritériami. Prvým kritériom je konkurenčný pomer (competitive ratio) algoritmu. Konkurenčný pomer môžeme definovať ako pomer výkonnosti, resp. efektivity on-line algoritmu k výkonnosti, resp. efektivite off-line algoritmu pre tú istú inštanciu problému. Druhým kritériom je výkonnosť, alebo efektivita pri najhoršom možnom prípade riešenia problému.

Papadimitrijou a Yannakis zistili, že ak počet možných blokácií nie je fixný, môžeme povedať, že navrhnutie ideálnej stratégie, ktorá by garantovala daný konkurenčný pomer, je PSPACE-úplny problém. Naopak, optimalizácia očakávaného pomeru je #P-ťažký problém. Mnoho rokov nebol zaznamenaný takmer žiadny progres

v približovaní algoritmov pre tento problém. Bar-Noy a Schieber [2] objavili niekoľko variácií CTP, použitím kritéria najhoršieho možného prípadu, kde úlohou bolo nájsť statický algoritmus, ktorý minimalizuje maximálnu dĺžku cesty. Pracovali s typom problému známym ako k-CTP, pri ktorom je počet blokád ohraničený zhora veličinou *k*. Treba si uvedomiť, že pre ľubovoľne zvolené *k* zostáva problém nájdenia stratégie, ktorá garantuje konkrétnu dĺžku trasy PSPACE-úplnym problémom. Ďalej Bar-Noy a Schieber pracovali s obnoviteľným k-CTP, kde každá blokovaná cesta bola asociovaná s časom potrebným na odstránenie blokády. Tiež sa zaoberali stochastickým modelom, kde nezávislá pravdepodobnosť blokácie pre každú hranu bola daná a dopredu známa. Tento model, ktorý sa snaží optimalizovať očakávaný pomer, je, ako už bolo vyššie spomenuté, #P-ťažký problém.

Následne, Karger a Nikolova [11] pracovali na stochastickom CTP so špeciálnymi grafovými triedami a vyvinuli presné algoritmy tým, že aplikovali techniky z teórie Markových rozhodovacích procesov (Markov Decision Processes). Eyerich, Keller a Helmert [6] prišli so zaujímavými algoritmami, ktoré spadajú do tzv. optimistic rodiny. Tieto algoritmy predpokladajú dopredu akýsi optimistický variant situácie, kedy je cestujúci vopred oboznámený (alebo správne predpovedá) s podmienkami na cestách. Tieto algoritmy sa nazývajú HOP, ORO a UCT, pričom tretí menovaný sa líši tým, že pri odhadovaní ideálnej trasy počíta aj s predchádzajúcimi situáciami, z ktorých sa môže „poučiť“. UCT sa ďalej delí na blind a optimistic. V posledných rokoch sa s CTP zaoberali mnohí vedci a prišli s mnohými zaujímavými riešeniami.

1.2 Zhrnutie obsahu jednotlivých kapitol práce

V druhej kapitole je popísaný prehľad typov CTP na základe parametrov, ktorými sú známosť cestujúceho o pravdepodobnosti blokácií ciest a obnoviteľnosť týchto blokácií. V závere tejto kapitoly sú stručne rozobrané aj určité takzvané „špeciálne“ typy CTP.

V tretej kapitole sú podrobnejšie popísané vybrané stratégie riešiace jednotlivé typy problémov CTP.

V štvrtej kapitole sa práca venuje implementácii vybraných stratégií riešenia CTP a tiež popisuje dve aplikácie napísané v programovacom jazyku Python, ktoré slúžia na otestovanie daných stratégií a ich efektivity.

V piatej kapitole sa práca zaoberá vyhodnotením testovania, výsledkom a porovnaniu efektivity daných stratégií.

Šiesta kapitola obsahuje záver a zhrnutie danej práce.

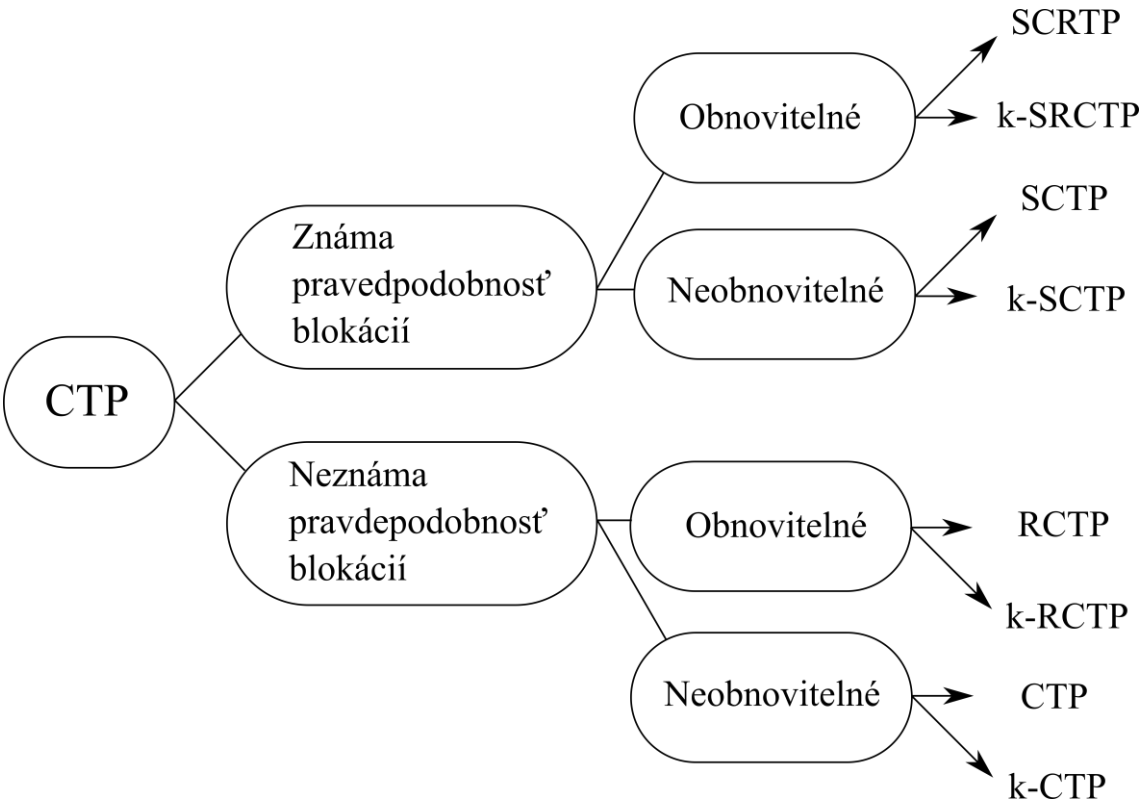
2 TYPY CTP

Problému CTP sa v poslednej dobe venovalo veľa pozornosti, a to hlavne z dôvodu, že riešenie tohto problému má vplyv na aplikovanie v reálnych situáciach, ako napríklad navigácia robota v známom či čiastočne známom prostredí, navigácia v automobilovej doprave, adaptívne trasportné systémy, atď. Môže tiež pomôcť pri riešení takzvaných dynamických situácií, kedy sa môžu podmienky meniť za behu. Ide napríklad o navigačný systém GPS či mobilnú komunikáciu, kde sa údaje zbierajú v reálnom čase.

V tejto kapitole sú vymenované rôzne typy daného problému, ktoré už predtým boli študované a skúmané a stručne popísané stratégie, ktorými možno daný typ problému riešiť.

Problém možno rozdeliť do dvoch hlavných skupín na základe znalosti agenta o blokáciach jednotlivých hrán. Prvú skupinu tvoria typy problémov, pri ktorých cestujúci nemá dopredu informácie o blokovaných hranách. Druhú skupinu tvoria typy, ktoré možno nazvať ako stochastické. Pravdepodobnosť blokácií hrán je dopredu známa, čo sa značne podpisuje aj na stratégiach, ktorými sú problémy spadajúce do tejto skupiny riešené. Ďalej sa problémy môžu členiť podľa toho, či hrany v danom grafe sú neobnoviteľné, to znamená že raz blokovaná hrana bude neprejazdná navždy, alebo obnoviteľné, pri ktorých je známy aj čas potrebný k odstráneniu blokácie hrany. Tu možno ešte poznamenať, že pri niektorých typoch sa čas potrebný k odstráneniu blokácie (cena) spája s hranou, a pri niektorých s uzlom. Toto členenie možno zjednodušiť tým, že všetky hrany budeme považovať za obnoviteľné s tým, že tie v skutočnosti neobnoviteľné budú mať nekonečnú cenu obnovy.

Špecifickým typom problému je takzvaný k-CTP. Ide o problém, v ktorom sa vyskytuje modifikácia, kde k je parameter, ktorý určuje maximálny počet blokácií. Táto modifikácia je však použitá iba kvôli teoretickým štúdiám. Ak je k rovné počtu všetkých hrán v grafe, ide o klasické CTP. Výhodou tejto metódy je, že ak stratégia zistí cez podmienku, že už bol dosiahnutý maximálny počet blokácií ciest, môže cestujúceho poslať rovno do cieľa najkratšou možnou cestou.



Obr. 1 Prehľad typov CTP

2.1 CTP s neznámou pravdepodobnosťou blokácie

Ide o typ problému, kde cestujúci disponuje mapou (grafom), ale niektoré hrany môžu byť blokované. To, či je hrana blokovaná alebo nie, sa cestujúci dozvedá, až keď príde k uzlu, za ktorým daná hrana nasleduje.

2.1.1 Problém kanadského cestujúceho (CTP)

Jedná sa o úplne základný variant tohto problému, z ktorého ostatné varianty vychádzajú. Tento variant položil základ pre daný problém, keď ho prvýkrát publikovali Christos Papadimitriou a Mihalis Yannakakis v roku 1989 [12]. Je to najjednoduchší typ CTP, v ktorom je základom vygenerovaný graf $G(V,E)$, v ktorom má každá hrana svoje vlastné ohodnotenie. Predpokladáme, že blokácie na daných hranách sa nedajú odstrániť, a že aj po odstránení všetkých problematických hrán vždy existuje cesta z počiatočného bodu s do cieľového bodu t.

K tomuto problému existuje hneď niekoľko stratégií riešenia:

Greedy Strategy (GS)

Táto metóda je považovaná za najjednoduchšiu, je možné s ňou ostatné metódy porovnávať a zistiť tak ich efektivitu. Cestujúci sa vyberá po najkratšej ceste z *s* do *t*, pričom ak narazí na hranu e_i ktorá je blokovaná a nedá sa cez ňu pokračovať do cieľa t, vymaže cestujúci túto hranu z grafu a z nového pozmeneného grafu vyráta najefektívnejšiu cestu, ktorou sa vydá. Tento proces sa opakuje až pokiaľ cestujúci nepríde do cieľa *t*. [2] Táto stratégia bude podrobnejšie rozobraná v tretej kapitole.

Reposition Strategy (RS)

Tak ako pri GS, cestujúci znova na začiatku predpokladá, že všetky hrany sú prejazdné a vyberie si najefektívnejšiu trasu z *s* do *t*. Vždy keď narazí na blokovanú hranu e_i, vráti sa znovu na začiatok grafu *s* a vydá sa novou optimálnou trasou do cieľa (s tým, že zistenú blokovanú hranu z grafu odstráni). Zistilo sa, že žiaden deterministický on-line algoritmus v tejto stratégii nemôže mať konkurenčný pomer menší ako $2k+1$ (k = objavený počet blokácií). [2], [4]

Comparison Strategy (CS)

Ak cestujúci dosiahne hrany e_i, ktorá je blokovaná, od tohto bodu si zvolí Greedy Strategy (GS) alebo Reposition Strategy (RS), v závislosti na nasledujúcich podmienkach. Ak platí:

$$C(s_i,t \mid e_i) \leq C(s,t \mid e_i),$$

kde $C(s_i,t \mid e_i)$ je najkratšia možná cesta z s_i do t po odstránení doposiaľ známych zablokovaných hrán e_i, a $C(s,t \mid e_i)$ je najkratšia cesta z *s* do *t* bez všetkých známych blokovaných hrán, cestujúci si zvolí Greedy Strategy (GS). Ak naopak platí podmienka:

$$C(s_i,t \mid e_i) > C(s,t \mid e_i),$$

cestujúci si zvolí Reposition Strategy (RS). [2] Tejto stratégii sa bude podrobnejšie venovať tretia kapitola.

2.1.2 k-Problém kanadského cestujúceho (k-CTP)

Problém veľmi podobný klasickému CTP problému s tým, že máme dopredu daný maximálny počet blokovaných hrán ohraničený veličinou *k*. Tento fakt má vplyv aj na stratégie riešenia tohto problému, hlavne vďaka tomu, že ak cestujúci objaví *k* počet blokácií, vie vopred, že už sa v grafe žiadna nenachádza a môže sa tak vybrať najkratšou cestou do cieľa *t*.

Bolo zistené, že maximálny konkurenčný pomer pre deterministické online algoritmy je $2k+1$. Tento objav učinil a publikoval Westphal v roku 2008. Krátko na to

objavil aj algoritmus zvaný *backtrack*, pomocou ktorého sa dá zistiť spodná medza a z ktorého neskôr vznikla Reposition Strategy (RS). [1]

Randomized Reposition Strategy (RRS)

Uvažujeme graf, ktorý pozostáva z n vrcholovo oddelených s-t ciest $P_1,\ldots,P_n$. Cenu cesty P_i vyjadríme nasledujúcim vzťahom:

$$c_i = \sum_{e \in P_i} w_e$$

Nech cesty P sú usporiadané podľa ich cien vzostupne $c_1 \leq \cdots \leq c_n$. Tento algoritmus môže byť vnímaný ako randomizovaná verzia algoritmu *backtrack*, a funguje následovne: uvažujeme k+1 najkratších možných vrcholovo oddelených s-t ciest v grafe. V tomto momente nevieme, ktoré z ciest sú blokované. S istotou vieme len to, že aspoň jedna z nich bude optimálna cesta. Ďalej zadefinujeme vhodné priemerné rozdelenie pravdepodobnosti a zvolíme si cestu podľa tohto rozdelenia, ktorú sa pokúsime prejsť. Ak je táto cesta priechodná, sme hotoví. Ak naopak priechodná nie je, vrátime sa na začiatok s a opakujeme procedúru pre menší súbor ciest. Bolo tiež zistené, že vždy platí konkurenčný pomer k+1. [4]

2.1.3 Obnoviteľný CTP (RCTP)

Problém zvaný RCTP sa vyznačuje tým, že raz zablokovaná hrana nie je zablokovaná navždy, ale po nejakom čase môže byť znova odblokovaná. Ak chce cestujúci touto hranou prejsť, musí zaplatiť penalizáciu. Budeme uvažovať jednoduchší variant tohto problému a to taký, že s penalizáciou $r(e)$ budú späté hrany, nie uzly.

Waiting Strategy (WS)

Keď cestujúci príde k inkriminovanej hrane e_i a zistí, že daná hrana je blokovaná a nedá sa cez ňu prejsť, zaplatí danú cenu za odblokovanie tejto hrany – penalizáciu a prejde ďalej.[3] Úplna cena trasy z s do t pomocou WS je nasledovná:

$$Cw(R) = S(s,t) + \sum_{i=1}^{k} r(e_i),$$

pričom $S(s,t)$ je cena najkratšej cesty z s do t a $r(e_i)$ je cena penalizácie hrany e_i.

Recovery Greedy Strategy (RGS)

Akonáhle sa cestujúci na ceste z s do t dostane k hrane e_i, ktorá je blokovaná, rozhodne sa, či bude čakať, resp. zaplatí penalizáciu $r(e_i)$, alebo sa vydá novou najkratšou cestou

do cieľa. Táto voľba závisí od toho, ktorý variant je cenovo výhodnejší. Tento proces sa opakuje pri každej blokovanej hrane, až dokým cestujúci nepríde do cieľa. [3]

Recovery Greedy & Reposition Strategy (GR)

Táto stratégia bola vyvinutá pre prípad k-CTP. Skladá sa z troch častí: prvá časť sa riadi pomocou Greedy Strategy (GS). Druhá časť sa tiež riadi pomocou GS a vedie cestujúceho do cieľa *t* do doby, pokiaľ nenarazí na blokovanú hranu. Tretia časť sa riadi pomocou Reposition Strategy (RS), ktorá tiež riadi agenta až do doby styku s blokovanou hranou. O tom, ktorá z týchto častí stratégie bude použitá, rozhoduje hlavný algoritmus, ktorý je súčasťou tejto stratégie. [5]

Bar-Noy & Schieber RCTP stratégia

Nech n je počet uzlov a m počet hrán v grafe $G(V,E)$. Veličina k značí hornú hranicu možných blokovaných hrán. Časová zložitosť hlavného algoritmu, ktorý určí cestovaciu stratégiu príchodu cestovateľa k fixnému cieľu a ktorý zároveň garantuje najmenší čas pre stratégiu „najhoršieho prípadu“, bude vyzerať nasledovne: $O(k^2 \cdot m + k \cdot n \cdot log(n))$. Tento algoritmus je podobný Dijkstrovmu algoritmu, skladá sa z dvoch častí: označujúcej a aktualizačnej. Princíp je ten, že algoritmus začína od konca, teda od cieľa *t*, a postupne analyzuje vzdialenosti od ostatných vrcholov. Najmenšiu z týchto vzdialeností uloží do zásobníka *L*. Postupne takto prechádza od vrcholu k vrcholu a analyzuje dĺžky ich naväzujúcich hrán. Následne sa v aktualizačnej časti tieto vzdialenosti skontrolujú a ak sa nájde nejaká kratšia ako tá, ktorá už v zásobníku *L* je (vždy medzi dvoma konkrétnymi uzlami *a* a *b*), tak sa prepíše. [1]

2.2 CTP so známou pravdepodobnosťou blokácie

Tento typ CTP, na rozdiel od predošlého, radíme medzi stochastické problémy. V tomto variante uvažujeme tak, že cestujúci, ktorý má k dispozícii mapu $G(V,E)$, pozná tiež pravdepodobnosť blokácie všetkých hrán v predstihu. Či je naozaj daná hrana blokovaná, sa cestujúci dozvedá pri príchode k uzlu, na ktorý daná hrana naväzuje. Úlohou riešiacich stratégií zostáva nájdenie časovo najmenej náročnej trasy z *s* do *t*. V tomto prípade, na rozdiel od predchádzajúceho, budú stratégie pracovať inak, keďže môžu dopredu pracovať s informáciou pravdepodobnosti blokácie daných hrán.

2.2.1 Neobnoviteľný CTP (SCTP)

Ide o stochastický variant, v ktorom vychádzame z faktu, že raz blokovaná hrana bude blokovaná navždy. Cestujúci má k dispozícii mapu $G(V,E)$ s tým, že každá hrana v grafe má svoju cenu. Aby bolo isté, že problém má vždy riešenie, predpokladáme, že cesta z *s* do *t* bude existovať aj po odstránení všetkých nepriechodzích hrán.

Hindsight Optimization (HOP)

Táto stratégia pristupuje k akýmsi sekvenciám, resp. iteráciám zvanáným *rollouts*. Počet týchto sekvencií je parameter značený písmenom *N*. Viac týchto sekvencií vyžaduje aj viacej času, ale o to sú potom odhady cien trás stabilnejšie. V každej sekvencii sa najprv vygenerujú určité podmienky na ceste, ktoré závisia na pravdepodobnosti blokácií jednotlivých ciest. Inými slovami, náhodne sa určí stav neznámych ciest podľa pravdepodobností blokácií. Ak sú výsledky cien ciest nedostačujúce, daná sekvencia sa počíta ako neúspešná. V opačnom prípade, ak pôjde o úspešnú sekvenciu, vyráta sa vzdialenosť od pozície cestujúceho do cieľa v náhradnom grafe, ktorý je priechodný v danej sekvencii. Cena odhadu HOP pre *N* sekvencií je priemerom vyrátaných vzdialeností z *s* do *t* zo všetkých úspešných sekvencií.

Alternatívou k tejto stratégii môže byť stratégia „*averaging over clairvoyance"* (táto stratégia bola vynájdená v roku 1995 Russelom a Norvigom) [13]. Pre každý prípad predpokladáme, že cestujúci je „jasnovidec (clairvoyant)" a vie dopredu, ktoré cesty sú priechodzie, a preto sa vyberie po najkratšej možnej. Keďže podmienky na ceste sa nedajú takto dopredu jednoznačne vedieť, spriemerujeme niekoľko možných stochastických variantov. HOP v poslednej dobe zaujalo komunitu stochastického plánovania, kde poslúžilo ako základ pre mnohé veľmi účinné plánovacie systémy. Taktiež bola táto stratégia úspešne využitá pri zaobchádzaní so skrytými informáciami v kartových hrách, napríklad pri hre pre jedného hráča *Klondike Solitaire,* alebo pri hrách pre viac hráčov ako *Bridge* a *Skat*. Napriek úspechom má tento prístup veľa známych nedostatkov: často konverguje k suboptimálnej taktike ak sa počet sekvencií blíži k nekonečnu. Frank a Basin prišli (v roku 2001) s ukážkou pre hru *Bridge* a Russel a Norvig veľmi jednoducho opísali (v roku 1995), kde môže HOP zlyhať. [6]

Optimistic Rollout (ORO)

Stratégia je naväzujúca a v mnohom podobná stratégii HOP. Taktiež pracuje so sekvenciami zvanými *rollouts*. Pre každú pozíciu, v ktorej sa cestujúci nachádza (pre každý uzol) sa prevádzajú sekvencie. Rozdiel oproti HOP je vo výpočte ceny jednotlivých *rolloutov*: pri úspešnej sekvencii *W* sa nepredpokladá, že cestujúci vie dopredu stav ciest, ale simuluje sa optimálna taktika na danú sekvenciu *W* tak, že sa použije cena trasy z *s* do *t* ako cena danej sekvencie. Pri výpočte dĺžky trasy z ľubovoľného uzlu do cieľa sa používa už skôr zmieňovaná stratégia GS (cestujúci sa vždy vyberie najkratšou možnou cestou, po narazení na blokovanú hranu sa vyráta nová najkratšia cesta). [6]

Optimistic UCT (UCTO)

Poslednou stratégiou z rodiny tzv. optimistických stratégií je UCTO. UCT je skratka pre anglický názov *Upper Confidence bounds applied to trees.* Je to algoritmus riešiaci situácie, pri ktorých sa pracuje s neistotami, napríklad pri kartových hrách (podobne ako HOP). Podobne ako predošlé algoritmy pracuje UCT so sekvenciami – *rolloutami,* ktorých počet značíme písmenom *N*. Podobne ako pri ORO sa sekvencie prevádzajú v každom mieste, ktorým cestujúci prejde (v každom uzle), bez použitia cestujúcemu skrytých informácií. Rozdielom medzi ORO a UCT je spôsob, ktorým sa stanovujú jednotlivé pohyby cestujúceho po mape. V stratégii ORO je každá jedna sekvencia nezávislá na ostatných, v UCT majú na seba sekvencie akúsi závislosť. Sekvencie, ktoré sa už raz ukázali ako neúspešné, sa pri ďalších sekvenciách vyberajú menej často a začnú sa preferovať úseky ciest, ktoré sa pri predošlých sekvenciách neobjavovali. Táto stratégia má tiež alternatívu s názvom *blind UCT.* [6]

2.2.2 Obnoviteľný CTP (SRCTP)

Prípad, v ktorom má cestujúci k dispozící mapu $G(V,E)$. Blokovaná hrana môže byť opäť prejazdná, avšak aby sa cez ňu cestujúci dostal, musí zaplatiť penalizáciu $r(e)$. Ide o plne stochastický druh CTP.

Ant Colony Optimization Metaheuristic

Táto stratégia je jedna z možných stratégií, ako možno riešiť SRCTP. Vychádza zo živého príkladu v prírode, zo spôsobu práce mravcov v prírode. Je to metóda založená na populácií mravcov, pri ktorej mravce iteratívne konštruujú riešenia. Konštrukcia riešenia je pravdepodobnostne ovplyvnená heuristickými informáciami vo feromónovej stope a čiastočne aj pamäťou jednotlivých mravcov. Stopy feromónov sa počas procesu vyhľadávania upravujú tak, aby odrážali kolektívne skúsenosti kolónie mravcov ako celku. Mravce si tak medzi sebou vymieňajú informácie o cestách (napríklad o ich dĺžkach). V každom cykle si každý mravec vytvorí vlastné riešenie za použitia konštruktívnej vyhľadávacej procedúry. Mravci budujú tieto riešenia náhodným chodením po konštrukčnom $G(V,E)$. [7]

Bar-Noy & Schieber stratégia

Predpokladajme, že každá hrana e je asociovaná s cenou $l(e)$ a pravdepodobnosťou blokácie hrany $p(e)$. Vždy, keď sa cestujúci pokúsi prejsť z bodu x cez hranu e, pravdepodobnosť, že hrana e nie je blokovaná je $1 - p(e)$, Ak je hrana e blokovaná, cestujúci si môže buď zvoliť nejakú inú neblokovanú hranu, alebo zaplatí cenu $l(x,x)$ za odblokovanie hrany e. Riešenie je založené na predpoklade, že platí $l(x,x) < l(x,y)$ pre všetkých y susedov x. Vo všeobecnosti si môžeme predstaviť, že operácie vykonané

cestujúcim v bode x sa skladajú z viacero fáz. V každej fáze použije cestujúci zoznam priorít $x_1,x_2,...x_j$ z x susedov. Pri počiatku fázy sa cestujúci pokúša prejsť cez hranu (x,x_1), ak je (x,x_1) blokovaná, cestujúci sa pokúsi prejsť hranou (x,x_2) a tak ďalej. Ak každý z týchto pokusov zlyhá, cestujúci zostáva na mieste x a začína ďalšia fáza. Všimnime si, že pri všetkých prioritných zoznamoch, ak máme známy predpokladaný čas potrebný k prejdeniu z x_i do t, tak pre x_i v každom zozname môže byť predpokladaný čas potrebný na cesu z x do t vypočítaný. [1]

2.2.3 CTP s pravdepodonosťami na sebe závislými (CTP-dep)

Ide o PSPACE – úplny problém. CTP-dep môžeme považovať za zovšeobecnenie klasického CTP, kde pravdepodobnosti blokovania jednotlivých hrán môžu byť na sebe závislé, teda namiesto funkcie pravdepodobnosti p ako tomu je v definícií CTP, máme v tomto prípade všeobecné rozdelenie pravdepodobnosti. Rovnako ako pri CTP, problémom je nájsť optimálnu cestu z s do t. Predpokladáme, že B je špecifikované ako *Bayesovská sieť* náhodných premenných E. Každá náhodná premenná predstavuje stav hrany (blokovaná, odblokovaná). *Bayesovská sieť* $B = (E,A,P)$ pozostáva zo sady náhodne orientovaných hrán A medzi náhodnými premennými E, takže (E,A) je orientovaný acyklický graf. P predstavuje podmienené pravdepodobnosti, jednu pre každú $e \in E$. Explicitná konštrukcia *Bayesovskej siete* nie je ako dôkaz potrebná, a preto sa táto práca o nej nezmieňuje. [8]

2.3 Ostatné typy CTP

V tejto podkapitole sú stručne opísané dva typy CTP, ktoré je len ťažko možné zaradiť do niektorého z predošlých variantov.

2.3.1 CTP so vzdialeným snímaním ciest (CTP-remote)

Ide o navigačný CTP, kde je graf spočiatku známy, ale niektoré hrany môžu byť blokované na základe danej známej pravdepodobnosti. Úlohou je optimalizovať cenu cesty z s do t. Môžeme zovšeobecniť tento CTP tak, že sa snažíme nájsť čo najmenšiu cenu cesty a diaľkového snímania. Toto diaľkové snímanie pozostáva z toho, že cestujúci môže za akúsi cenu dopredu vypozorovať stav ciest (napríklad pomocou platenej funkcie na internete, ktorá mu to umožní). [9]

2.3.2 CTP s opakovaním (CTP-rep)

V tomto variante do grafu vstupuje viacero agentov, ale vždy je aktívny iba jeden. Agent sa stane neaktívnym (dostane sa „mimo hry“), ak dosiahne cieľ t a nemôže už vykonávať žiadne akcie ani pozorovania. Po tom, čo sa agent dostane do cieľa,

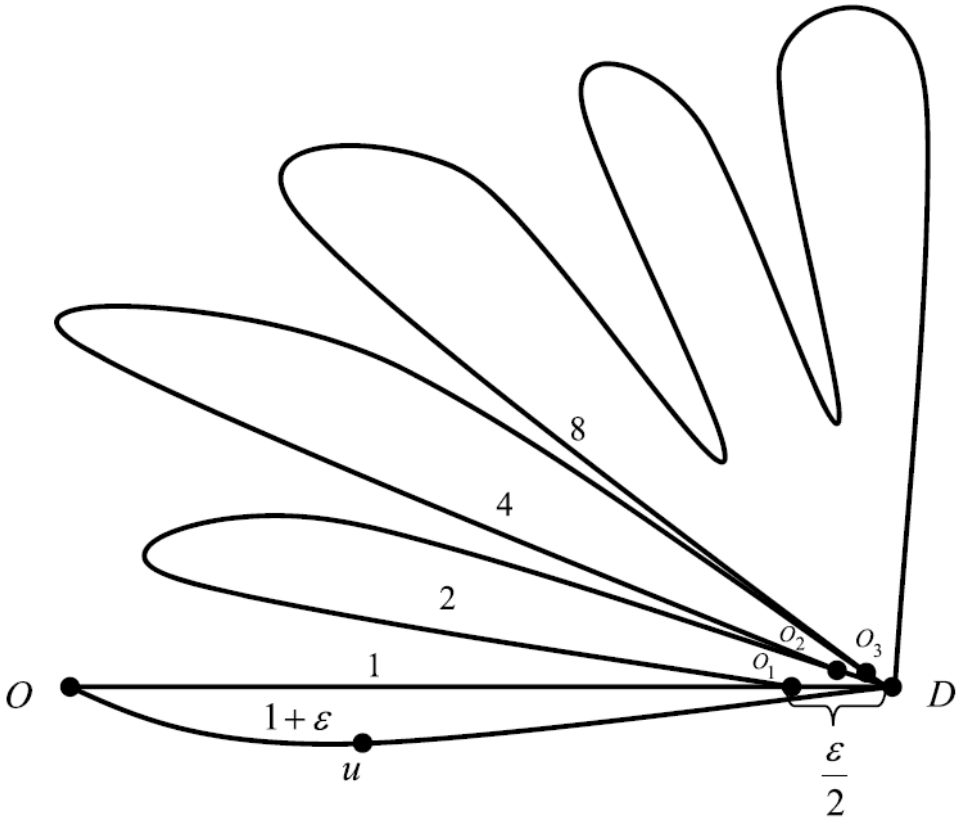
ostatným agentom sprostredkuje informácie, ktoré po ceste nadobudol (o dĺžke a stave jednotlivých hrán). Cieľom je optimalizácia súhrnnej ceny ciest všetkých agentov. [10]

3 VYBRANÉ STRATÉGIE RIEŠIACE CTP

V tejto kapitole si bližšie rozoberieme niektoré stratégie, ktoré už boli spomínané v minulej kapitole, avšak viac dopodrobna. Venovať sa budeme tým stratégiam, ktoré budeme implementovať a neskôr testovať a porovnávať.

3.1 Greedy Strategy (GS)

Táto stratégia je založená na nasledujúcom princípe : keď sa cestujúci dostane na miesto O_i a vie, že hrana $e_i = (O_i,Y_i)$ je blokovaná, vydá sa najkratšou možnou trasou z s_i do t, pričom blokovaná hrana e_i zostáva nevyužitá. Pre CTP so sekvenciou blokovania E_k je konkurenčný pomer Greedy Strategy $2^{k+1} - 1$.



Obr. 2 Najhorší možný prípad GS

Poznámka k obrázku: O a D označujú počiatok a koniec cesty, O_i označuje miesto, kde došlo k zisteniu blokácie, čísla pri hranách sú ceny jednotlivých hrán a ε je kladná konštanta. [2]

3.2 Reposition Strategy (RS)

Táto stratégia sa v mnohom podobá GS. V mnohých literatúrach ju môžeme nájsť pod názvom „*backtrack*“. Na začiatku si stratégia vypočíta pomocou Dijkstrovho algorimu najkratšiu možnú cestu z *s* do *t* (neberúc v úvahu blokované hrany, o ktorých cestujúci dopredu nevie). Keď sa cestujúci dostane na miesto O_i a vie, že hrana $e_i = (O_i,Y_i)$ je blokovaná, vráti sa naspäť na začiatok *s* a vypočíta si novú najkratšiu trasu, tentokrát už bez použitia e_i. Westphal (2008) dokázal, že konkurenčný pomer pre túto stratégiu je vždy väčší ako $2k + 1$ [4]. RS môže byť v niektorých prípadoch veľmi neefektívna, keďže k celkovej cene cesty sa musí prirátať aj cena, ktorú cestujúci zaplatí za to, že sa po každom objavení blokovanej hrany musí vrátiť naspäť na začiatok grafu. Z tohto dôvodu sa v mnohých prípadoch pri výpočtoch dáva prednosť Greedy alebo Reposition Strategy. RS je z týchto troch stratégií považovaná za najmenej efektívnu.

3.3 Comparison Strategy (CS)

Ak cestujúci príde k O_i a zistí, že hrana $e_i = (O_i,Y_i)$ je blokovaná, rozhodne sa pre Greedy Strategy alebo Reposition Strategy podľa nasledujúcich podmienok. Ak platí:

$$C_{GS}(O_iD|E_i) \leq C_{opt}(OD|E_i) \ (i = 1, 2, \ldots, k),$$

potom sa cestujúci rozhodne pre Greedy Strategy. Ak naopak platí:

$$C_{GS}(O_iD|E_i) > C_{opt}(OD|E_i) \ (i = 1, 2, \ldots, k),$$

potom cestujúci použije Reposition Strategy. Pre CTP so sekvenciou blokácií E_k je konkurenčný pomer pre Comparison Strategy $2k+1$. Ďalej už sa o Comparison Strategy budeme zmieňovať ako o CS.

Všimnime si, že existujú dva typy tzv. čiastkových ciest v prejdenej trase $(O,O_1,O_2, \ldots, D)$. Prvý typ je $(O,O_1,O_2, \ldots ,O_j)$, kde cestujúci používa Greedy Strategy vždy, keď sa dostane k blokovanej hrane (okrem objavenia poslednej blokovanej hrany). V tomto prípade pre každé $i \in \{1, 2, \ldots, j\}$ platí:

$$C_{GS}(O_iD|E_i) \leq C_{opt}(OD|E_i).$$

Druhým typom je (O_j,O), t.j. po zistení zablokovanej hrany v bode O_j sa cestujúci vráti na začiatok *O*. Po príchode do počiatočného bodu *O* sa cestujúci vyberie novou najkratšou cestou z *O* do *D*, pričom predošlá nájdená hrana bola z grafu odstránená.

3.4 Waiting Strategy (WS)

Táto stratégia patrí do spôsobov riešení obnoviteľného CTP. Ako už z názvu vyplýva, cestujúci bude v určitej situácií čakať. Na začiatku si cestujúci pomocou Dijkstrovho algoritmu vypočíta najkratšiu cestu z s do t a vyberie sa po nej. Akonáhle dôjde k blokovanej hrane e_i, tak zostane čakať v uzle x_i, na ktorý blokovaná hrana naväzuje. Cestujúci zaplatí penalizáciu, s ktorou je daná hrana spojená. Táto penalizácia je akási daň, ktorú treba zaplatiť za odstránenie blokácie. Po zaplatení penalizácie sa cestujúci znova vydá po vyrátanej trase až do cieľa. Tento postup sa opakuje pri každom objavení novej blokovanej hrany e_i. Táto stratégia môže byť v mnohých prípadoch neefektívna, najmä pri väčšom počte blokovaných hrán, pretože vôbec neberie do úvahy veľkosť penalizácií na vyrátanej najkratšej trase.

3.5 Recovery Greedy Strategy (RGS)

Stratégia je podobná Waiting Strategy. Na začiatku sa znova vypočíta najkratšia cesta pomocou Dijkstrovho algoritmu a cestujúci sa po nej vydá. Ak príde k blokovanej hrane e_i, rozhodne sa, či zaplatí penalizáciu a počká, alebo či sa vydá novou najkratšou cestou, ktorá už nebude zahŕňať blokovanú hranu e_i. Cestujúci sa rozhodne na základe toho, čo je pre neho výhodnejšie. Tento postup sa opakuje pri nájdení každej novej blokovanej hrany. RGS je efektívnejšia ako WS, pretože tu už sa zohľadňuje aj veľkosť penalizácií a cestujúci si tak môže vybrať najlepšiu trasu.

4 IMPLEMENTÁCIA VYBRANÝCH STRATÉGIÍ

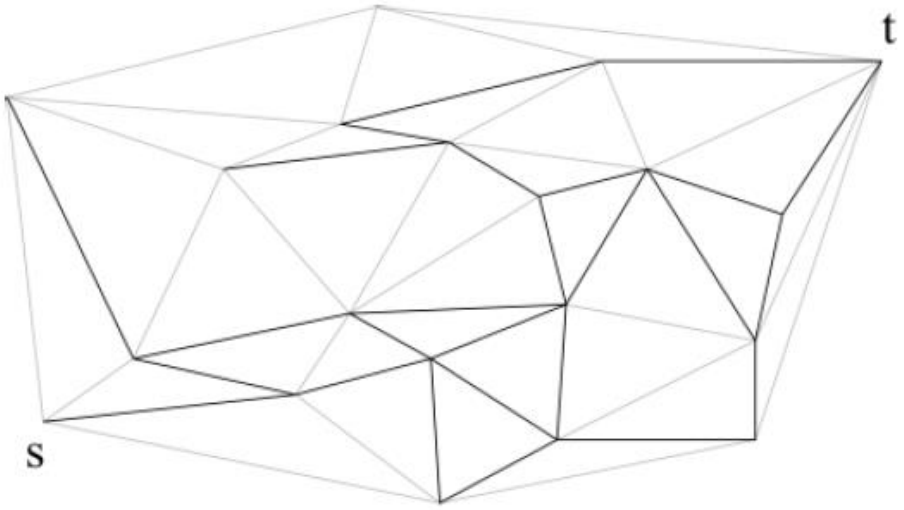
V tejto kapitole sú implementované vybrané stratégie ktoré riešia CTP, stručne popísaný je programovací jazyk Python a Delaunayho graf, s ktorým sme pri implementácií pracovali. Pre účel implementácie vybraných stratégií boli vytvorené dve aplikácie, pomocou ktorých sa testovali jednotlivé stratégie. Prvá aplikácia s názvom *CTP calc* slúži primárne pre výpočty a druhá s názvom *CTP solver* slúži ako vzor toho, ako dané algoritmy fungujú. Výsledky testovania sa porovnávali, aby sa tak mohlo určiť, ktorá zo stratégií bola najefektívnejšia pre daný prípad. Pri porovnaní sme pracovali s priemerným časom, za ktorý boli stratégie schopné prísť s najlepším riešením a tiež s cenou cesty daných riešení. Implementovaných bolo päť stratégií. Všetky tieto stratégie riešia CTP s neznámou pravdepodobnosťou blokácie (to znamená, že agent dopredu nevie aké sú šance, že jednotlivé hrany budú blokované). Tri z nich sú zástupcami typu CTP s neobnoviteľnými hranami (Greedy Strategy, Reposition Strategy, Comparison Strategy) a ďalšie dve sú zástupcami CTP s hranami obnoviteľnými (Waiting Strategy, Recovery Greedy Strategy).

4.1 Programovací jazyk Python

Aplikácia bola programovaná v programovacom jazyku Python. Je to vysokoúrovňový skriptovací programovací jazyk, ktorý v roku 1991 navrhol Guido van Rossun. Ponúka dynamickú kontrolu dátových typov a podporuje rôzne programovacie paradigmy, vrátane objektovo orientovaného, imperatívneho, procedurálneho alebo funkcionálneho prístupu. Je vyvíjaný ako open source projekt, ktorý zadarmo ponúka inštalačné balíky pre väčšinu bežných platforiem (Unix, MS Windows, macOS, Android...).

4.2 Delaunayho graf

Tento typ grafu sa používa v matematike a vo výpočtovej geometrii. Je tiež známy pod názvom Delaunayho triangulácia. Jeho podstatou je triangulácia, ktorá je pre danú množinu P diskrétnych bodov v rovine postavená tak, že žiaden bod P neleží vo vnútri kruhu, ktorý je opísaný akémukoľvek trojuholníku v $DT(P)$. Delaunayho triangulácia maximalizuje minimálny uhol a minimalizuje maximálny uhol zo všetkých uhlov všetkých trojuhloníkov, ktoré sa nachádzajú v triangulácii. Triangulácia je pomenovaná po Borisovi Delaunaym.



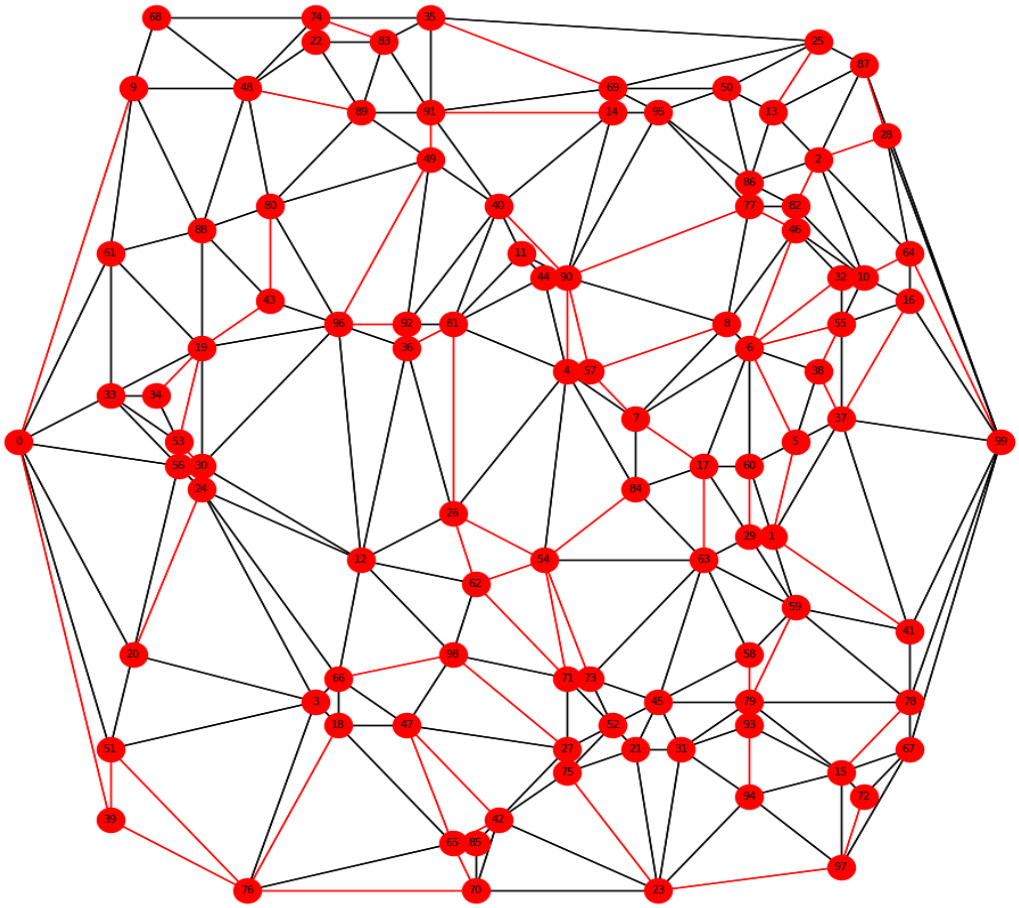
Obr.3 Príklad Delaunayho grafu

4.3 Triedy vytvorené pre testovacie aplikácie

V tejto podkapitole sú stručne opísané základné triedy, ktoré boli vytvorené na mieru pre testovacie aplikácie.

4.3.1 Trieda „graphGen“

Táto trieda má tri funkcie. Prvá funkcia sa nazýva *create*, a slúži na vytvorenie grafu *G*, na ktorom sa neskôr testujú implementované stratégie. Vstupným údajom do funkcie je iba počet uzlov, z ktorých sa graf vytvorí. Následne sa tieto uzly náhodne rozložia a spojí ich náhodný počet hrán. Dĺžky hrán v grafe sú vypočítané pomocou analytickej geometrie a to vždy ako vzdialenosť dvoch bodov v rovine (využívajú sa súradnice počiatočného a konečného uzlu hrany). Druhou funkciou v tejto triede je funkcia *generate_blockades*. Jej vstupný parameter je číslo, ktoré reprezentuje koľko percent hrán v grafe *G* bude blokovaných. Tretia a posledná funkcia *save_fig* slúži na uloženie vykreslenia grafu *G* ako obrázku s formátom .png. Čierne hrany v grafe sú hrany priechodné, červené hrany sú blokované.



Obr.4 Príklad Delaunayho grafu vygenerovaného našou aplikáciou

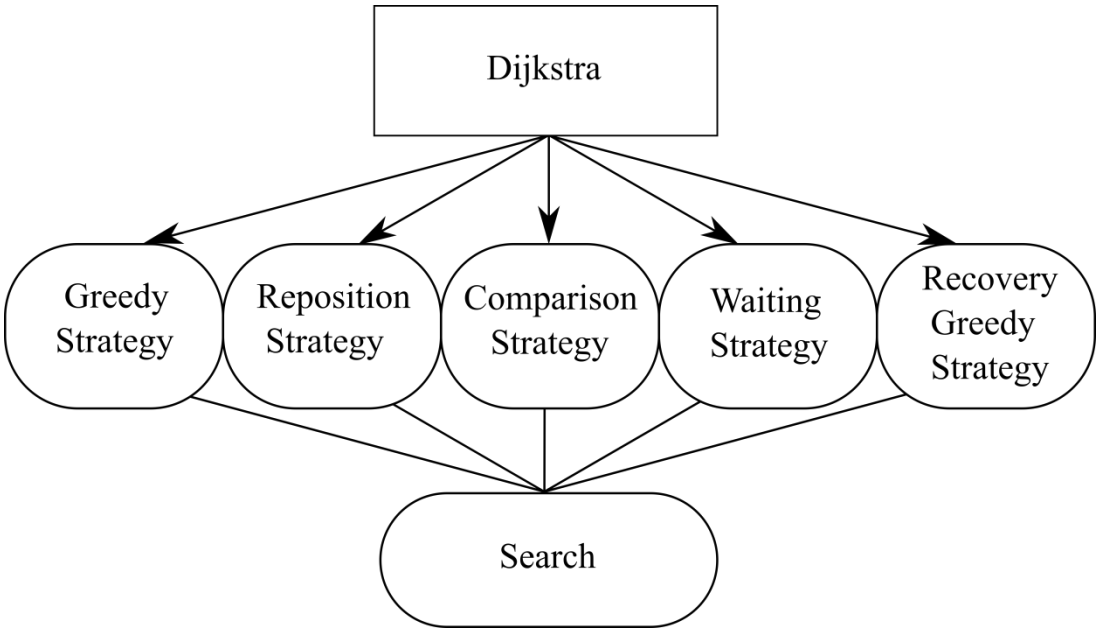
4.3.2 Trieda „dijkstra“

Trieda „dijsktra“ slúži na vyhľadanie najkratšej možnej cesty z počiatočného bodu do cieľa. Súčasťou tejto triedy sú dve „mini“ funkcie: *get_path* a *get_path_length*. Prvá slúži na zoradenie uzlov, cez ktoré vedie najkratšia cesta z počiatočného až po konečný uzol. Druhej funkcii sa na vstup privedie zoznam uzlov, z ktorých cesta pozostáva, a vráti nám dĺžku (cenu) tejto trasy. Hlavnou funkciou tejto triedy je funkcia *shortest_path*, ktorá reprezentuje Dijkstrov algoritmus nájdenia najkratšej cesty.

4.3.3 Triedy pre jednotlivé implementované stratégie

Každá z daných stratégií má vlastnú triedu. Prvé tri triedy su *greedyStrategy*, *repositionStrategy* a *comparisonStrategy*. Tieto triedy reprezentujú stratégie pre druh CTP s neobnoviteľnými hranami, a teda nepracujú s penalizáciami. Ďalšie dve triedy sú *waitingStrategy* a *recoveryGreedyStrategy*. Tie zase reprezentujú riešenia pre CTP

s hranami obnoviteľnými a pracujú aj s penalizáciami. Tieto triedy sú si veľmi podobné, líšia sa iba v danom prevedení hľadania najkratšej cesty. Spočiatku postupuje každá rovnako - vydá sa po najkratšej trase vypočítanej pomocou Dijkstrovho algoritmu. Ich správanie sa však začne líšiť, akonáhle narazia po ceste na prvú blokovanú hranu. Pre všetky platí, že vstupnými údajmi sú počiatočný a konečný uzol a výstupom je čas, za ktorý daná stratégia našla optimálnu trasu a cena tejto trasy. Všetky tieto triedy dedia od triedy *dijkstra*.



Obr.5 Ukážka dedenia jednotlivých tried v aplikáciach

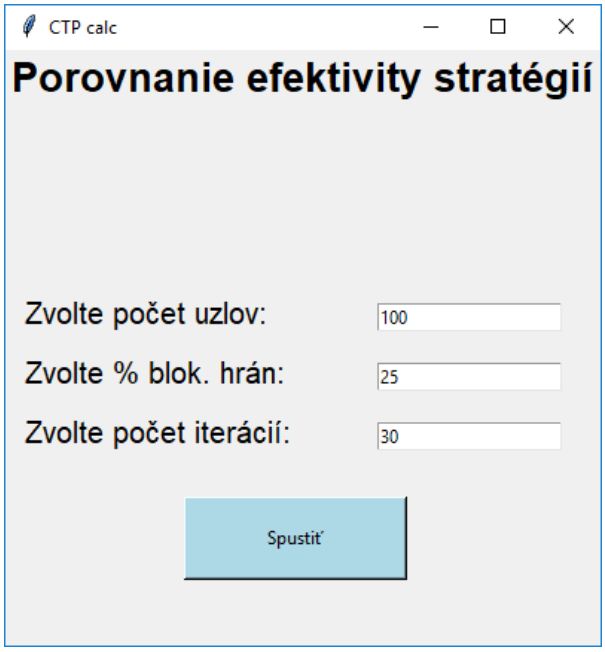
4.4 Výsledky testov z výpočtovej aplikácie CTP calc

Výsledky testov daných stratégií sa zapisujú do súboru *results.xlsx*, čo je súbor vo formáte tabuľkového procesora Microsoft Excel. Na vykreslenie výsledkov bol použitý *boxplot* graf, ktorý je súčasťou importovanej knižnice *matplotlib*. Obrázky Delaunayho grafov s blokovanými hranami pre každú iteráciu sa ukladajú do priečinku *temp*, ako súbory s koncovkou .png. Ukážka danej tabuľky a *boxplot* grafu sa nachádza v nasledujúcej kapitole.

4.5 Uživateľské rozhranie výpočtovej aplikácie CTP calc

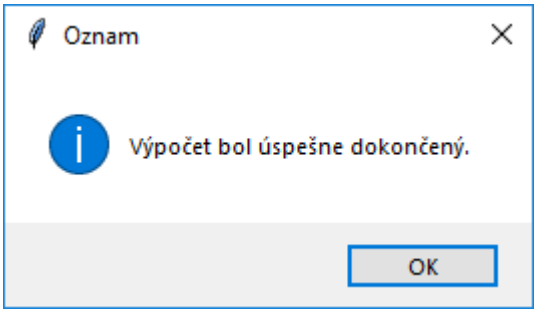
Uživateľské rozhranie je veľmi jednoduché. Bolo vytvorené pomocou vstavanej knižnice pre tvorbu grafického rozhrania *tkinter*. Po spustení aplikácie sa otvorí okno, v ktorom si užívateľ zvolí údaje pre testovanie. Užívateľ môže zadať počet uzlov v grafe,

percentuálny počet blokovaných hrán a počet iterácií testu. Po zadaní údajov užívateľ klikne na tlačidlo s nápisom „Spustiť“, čím odštartuje proces testovania.



Obr.6 Ukážka užívateľského panelu pri spustení aplikácie CTP calc

Ak výpočet prebehne bezchybne, po jeho skončení a zapísaní výsledkov sa objaví textové okno s oznamom „výpočet bol úspešne dokončený“.



Obr.7 Ukážka textového okna aplikácie CTP calc

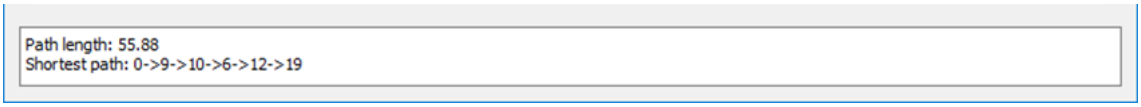
4.6 Užívateľské rozhranie vzorovej aplikácie CTP solver

Táto aplikácia bola vytvorená na vzorovú ukážku toho, ako fungujú jednotlivé algoritmy stratégií. Pri tvorbe bola použitá knižnica pre grafické rozhranie s názvom

PyQt5, ktorá je interaktívnejšia a má oveľa viac možností pri tvorbe, ako knižnica *tkinter*. Jej hlavnou výhodou je možnosť presne si grafické rozhranie upraviť pomocou aplikácie *Qt designer*. Aplikácia *CTP solver* slúži na ukážku algoritmov priamo na vygenerovanom grafe.

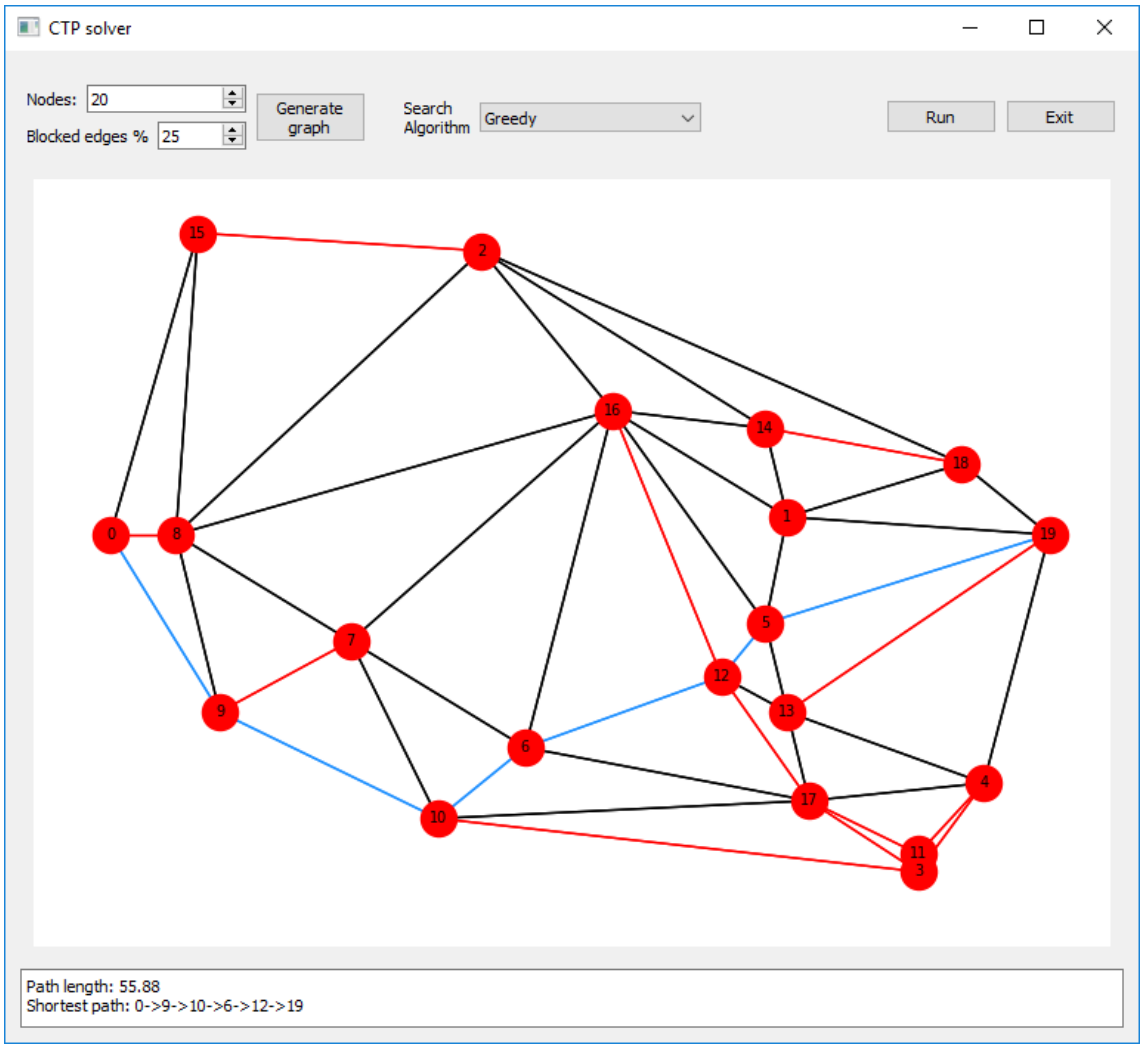
Obr.8 Ukážka užívateľského panelu aplikácie CTP solver

Po spustení aplikácie si užívateľ zvolí počet požadovaných uzlov v grafe v kolónke *Nodes* a percentuálny počet blokovaných hrán v kolónke *Blocked edges %*. Následne klikne na tlačidlo *Generate graph*, ktoré vygeneruje graf. V rolete *Search Algorithm* si užívateľ zvolí ktorú stratégiu chce testovať. Ďalej klikne na tlačidlo *Run*, čím sa spustí výpočet.



Obr.9 Ukážka konzoly s výsledkami aplikácie CTP solver

Po prevedení výpočtou sa v konzoli vypíše dĺžka danej najkratšej trasy vypočítanej vybranou stratégiou a uzly v poradí, v akom nasledujú pri trase z počiatočného uzlu do uzlu konečného.



Obr.10 Ukážka chodu aplikácie CTP solver

Predchádzajúci obrázok ukazuje výslednú podobu aplikácie po výpočte. Čiernymi farbami v grafe sú označené hrany, ktoré nie sú blokované a červenými sú označené blokované hrany. Modré hrany sú tie, ktorými zvolený algoritmus prechádzal pri trase z počiatočného bodu do bodu konečného.

Do budúcna by sa aplikácia mohla „doladiť“, aby mal užívateľ väčšiu škálu možností pri testoch. Pridať by sa napríklad mohli interaktívne okná, kde by si užívateľ mohol sám zvoliť počiatočný a konečný uzol. Taktiež by sa mohla pridať možnosť, aby si užívateľ mohol priamo vybrať, ktoré konkrétne hrany budú blokované. Docielilo by sa tak lepšej využiteľnosti aplikácie v reálnych situáciach.

5 VÝSLEDKY TESTOVANIA

V tejto kapitole sa nachádza popis testovania stratégií a tiež vyhodnotenie ich efektivity. Porovnávali sme medzi sebou stratégie riešiace neobnoviteľný (GS, RS, CS) a obnoviteľný (WS, RGS) typ CTP.

5.1 Parametre vybraných stratégií a testovania

Všetkých päť stratégií je výpočetnou aplikáciou spúšťaných naraz, a teda vstupné parametre boli pre všetky stratégie rovnaké. Na začiatku sa vygeneroval 100 uzlový graf a tomuto grafu sa vygenerovali blokované hrany, ktorých percentuálny počet si môže zvoliť užívateľ. Počiatočný uzol bol vždy osamostatnený na ľavej strane a konečný uzol na pravej. Penalizácia bola iniciovaná ku každej hrane náhodne v intervale <5,10>. Testovanie sa prevádzalo vždy na 10 iteráciach a to tak, že v každej iterácií prebehla grafom každá z piatich stratégií, pričom v každej iterácií sa vždy vygenerovalo, ktoré hrany boli blokované. Test prebehol 3-krát. Prvýkrát s celkovým počtom náhodne blokovaných 10 % hrán v grafe, druhý krát s 25 % blokovaných hrán a tretí s 50 % hrán, ktoré boli zablokované.

Všetky testy boli prevedené na notebooku HP Pavilion Power 15-bc411, s nasledovnou hardvérovou výbavou:
- Procesor: Intel® Core™ i5-8300H (2,3 GHz)
- Pamäť: 8192 MB DDR4 (2400 MHz)
- Grafická karta: NVIDIA GeForce GTX 1050 (4GB)

Pri všetkých testoch boli údaje potrebné pre simuláciu uložené na SSD disku, čo urýchlilo celý proces testovania a tiež ovplyvnilo priemerné časy výpočtov aplikácie.

5.2 Výsledky

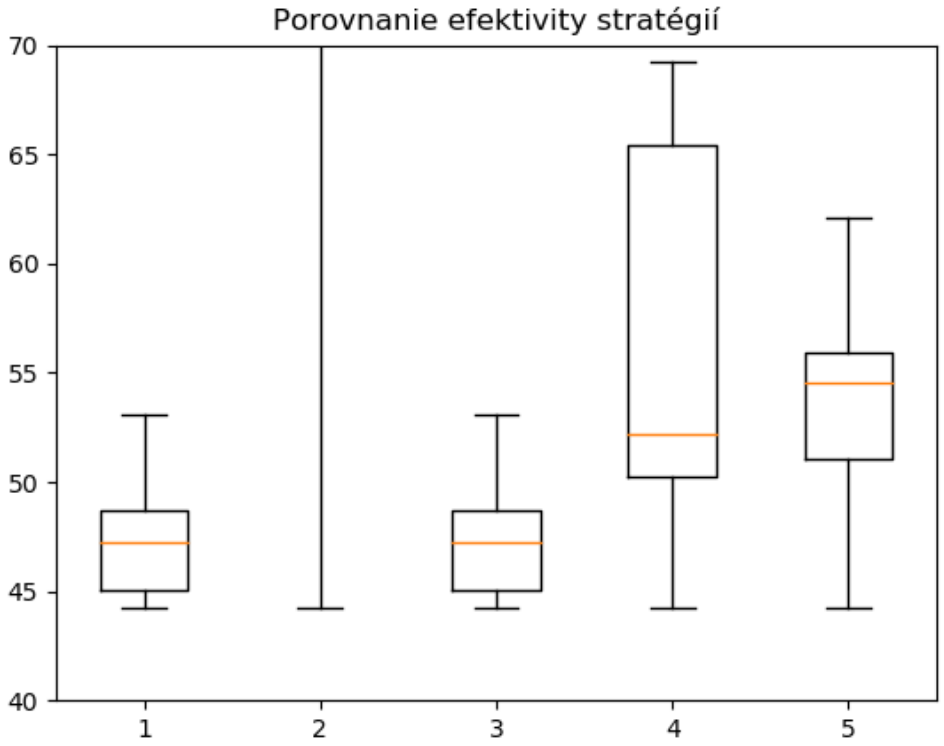
Nasledujú výsledky testovania. V následujúcich tabuľkách sú vždy zľava údaje: názov stratégie, priemerná cena cesty, maximálna cena cesty, minimálna cena cesty, medián, priemerný čas výpočtu najkratšej cesty danou stratégiou. V nasledujúcich grafoch môžeme vidieť rozptyl najčastejších cien ciest jednotlivých stratégií, minimálnu a maximálnu cenu a medián. Na ose Y (vertikálna os) sú vypísané ceny ciest a na ose X (horizontálna os) čísla stratégií. Stratégie sú očíslované nasledovne: 1 - GS, 2 - RS, 3 - CS, 4 - WS, 5 - RGS.

5.2.1 Test č.1

Nasledujúci test bol prevedený na grafe s počtom blokovaných hrán 10 % z celkového počtu hrán:

Strategy	Average Cost	Maximum Cost	Minimum Cost	Median	Average Time
Greedy	47,41800457	53,07432411	44,20842759	47,21961089	0,004139644
Reposition	117,470013	176,0478954	44,20842759	121,3285279	0,004322978
Comparison	47,41800457	53,07432411	44,20842759	47,21961089	0,009641867
Waiting	56,40842759	69,20842759	44,20842759	52,20842759	0,001617867
Recovery Greedy	53,81800457	62,07432411	44,20842759	54,57607167	0,0095932
Dijkstra algorithm	44,20842759				

Obr.11 Tabuľka prvej fázy testovania s 10 % blokovaných hrán



Obr.12 Graf testovania po prvej fáze testu s 10 % blokovaných hrán

Z údajov a grafov po prvom teste bolo zjavné, že v skupine stratégií riešiacich neobnoviteľný CTP na tom bola výrazne najhoršie RS. Fakt, že sa cestujúci po nájdení každej blokovanej hrany vráti na počiatočný bod s, z ktorého vyráta novú najkratšiu trasu, sa javí ako veľmi neefektívny. Pri GS a CS môžeme vidiť rovnaké výsledky, čo sa týka cien ciest, keďže CS čiastočne pozostáva z GS. Z toho bolo zrejmé, že algoritmus zastupujúci RS v CS nebol použitý ani raz. GS však potrebovala na výpočet najkratšej trasy v priemere menej času ako CS, a preto môžeme povedať, že v tomto teste bola najefektívnejšia.

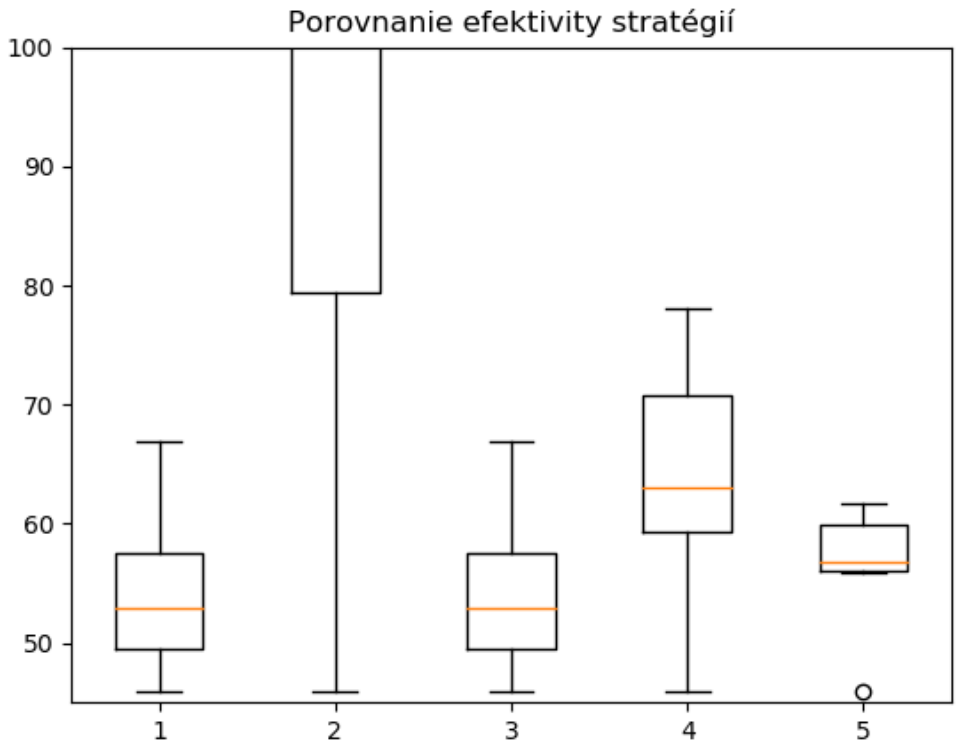
V porovnaní stratégií reprezentujúcich obnoviteľný typ CTP boli výsledky veľmi tesné. Pri WS vyšla väčšia priemerná cena cesty, ale medián vyšiel menší. Priemerný čas však vyšiel výrazne menší u WS. Obe stratégie sa v tomto teste ukázali ako pomerne efektívne, o niečo lepšie výsledky však mala RGS.

5.2.2 Test č.2

Následujúci test bol prevedený na grafe s počtom blokovaných hrán 25 % z celkového počtu hrán:

Strategy	Average Cost	Maximum Cost	Minimum Cost	Median	Average Time
Greedy	54,01048287	66,90198297	45,99646963	52,93466147	0,006338178
Reposition	207,9318182	547,1314264	45,99646963	157,8705615	0,009037378
Comparison	54,01048287	66,90198297	45,99646963	52,93466147	0,016463422
Waiting	63,49646963	77,99646963	45,99646963	62,99646963	0,001708178
Recovery Greedy	56,9400561	61,75063355	45,99646963	56,72537377	0,0155776
Dijkstra algorithm	45,99646963				

Obr.13 Tabuľka druhej fázy testovania s 25 % blokovaných hrán



Obr.14 Graf testovania po druhej fáze testu s 25 % blokovaných hrán

Na prvý pohľad bolo opäť vidieť, že údaje pri RS sú enormne vyššie ako pri ostatných stratégiach. Túto stratégiu teda opäť označíme ako najmenej efektívnu. GS a CS mali opäť rovnaké priemerné ceny ciest a medián. GS bola však pri hľadaní najkratšej cesty približne 2,5-krát rýchlejšia ako CS, a preto je znova najefektívnejšou.

Pri porovnaní WS a RGS sú výsledky výrazne lepšie pri RGS. Dosahovala menšiu priemernú cenu cesty a medián. Výrazne menší čas potrebný na nájdenie najkratšej cesty mala však opäť WS. Tieto výsledky možno pripísať tomu, že WS si

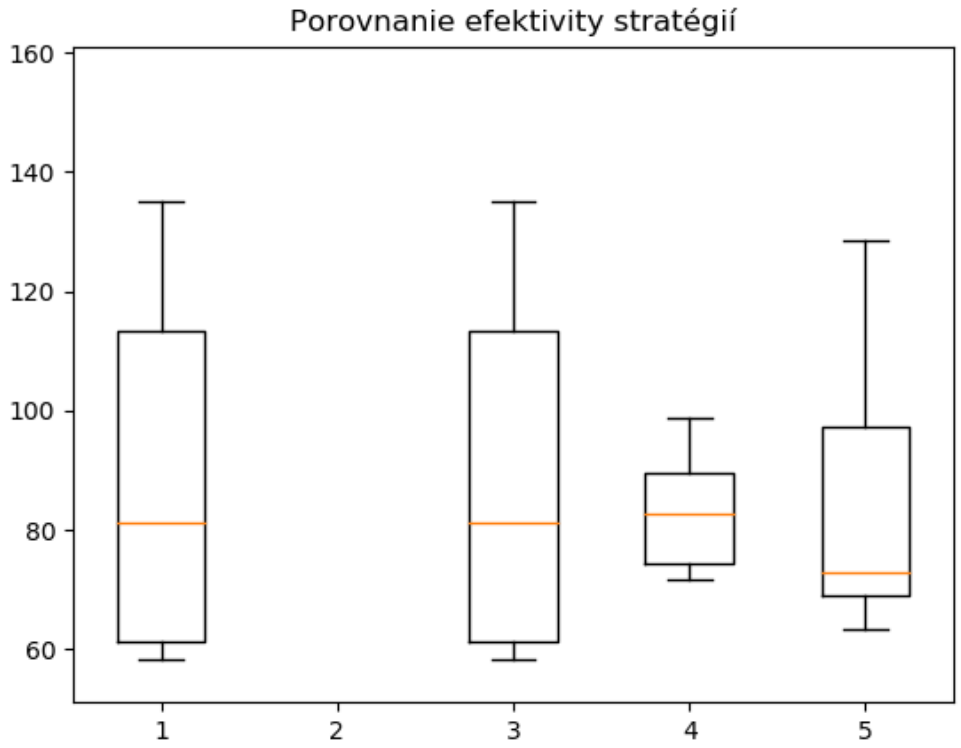
vypočíta najkratšiu cestu iba raz a vydá sa po nej, aj keby všetky hrany mali byť po ceste blokované. Neberie totiž do úvahy iné optimálne riešenie. S pribúdajúcim počtom blokovaných hrán sa tak efektívnosť WS oproti RGS znižuje.

5.2.3 Test č.3

Následujúci test bol prevedený na grafe s počtom blokovaných hrán 50 % z celkového počtu hrán:

Strategy	Average Cost	Maximum Cost	Minimum Cost	Median	Average Time
Greedy	87,23393336	135,0735285	58,26719582	81,08136453	0,024724622
Reposition	1250,310127	3660,218897	319,0686113	791,7406882	0,0367096
Comparison	87,23393336	135,0735285	58,26719582	81,08136453	0,072862311
Waiting	82,95303746	98,65303746	71,65303746	82,65303746	0,0015408
Recovery Greedy	83,31869413	128,602364	63,26719582	72,76805088	0,064506267
Dijkstra algorithm	47,65303746				

Obr.15 Tabuľka tretej fázy testovania s 50 % blokovaných hrán



Obr.16 Graf testovania po tretej fáze testu s 50 % blokovaných hrán

Z grafu nám vypadla RS, pretože jej hodnoty boli príliš veľké. Priemerná cena ciest bola až 1250,3, medián 791,7 a maximálna cena cesty bola až 3660,2. RS sa znovu ukázala ako veľmi neefektívna. GS a CS opäť vykazovali rovnaké výsledky. GS však bola opäť rýchlejšia. Tentokrát takmer 3-násobne.

Pri porovnaní WS a RGS sa efektívnejšie javila opäť RGS, najmä vďaka menšiemu mediánu, ktorý považujeme za najkľúčovejší prvok pri určovaní efektivity danej stratégie.

5.3 Celkové zhodnotenie testov

V každom z troch testov sme mohli vidieť, že najmenej efektívna bola CS. Bolo to spôsobené tým, že do celkovej ceny cesty sa vždy musí započítať aj cena *backtracku* z uzlu e_i, na ktorý naväzovala novonájdená blokovaná hrana do počiatočného bodu *s*. Posledný test, v ktorom bolo v grafe až 50 % hrán blokovaných, ukázal jasnú nevýhodu tejto stratégie a priemerná cena ciest spolu s mediánom boli niekoľkonásobne väčšie ako pri GS a CS. Spomedzi stratégií riešiacich neobnoviteľný typ CTP bola najefektívnejšia GS. Priemernú cenu cesty a medián mali síce s CS vždy rovnakú, ale bola rýchlejšia vo výpočte najkratšej cesty. To bolo spôsobené najmä tým, že CS musela pri každej iterácií pracovať aj s podmienkou, v ktorej vyhodnocovala použitie GS alebo RS. V žiadnom z prípadov vo výpočtoch pre CS nebola použitá RS, čo ešte viac potvrdzuje efektívnosť GS. CS by sme teoreticky mohli z týchto testov vynechať, pretože v skutočnosti sa tu vždy porovnávala iba efektivita GS voči RS.

Pri hodnotení stratégií riešiacich obnoviteľný typ CTP boli výsledky nasledovné. Vo všetkých troch testoch mala lepší priemerný čas hľadania najkratšej cesty WS. Avšak ako smerodatné sme brali hlavne ukazatele priemernej ceny cesty a mediánu, ktoré mala menšie RGS. Nevýhoda WS bola zrejmá s pribúdajúcim počtom blokovaných hrán. WS totiž berie do úvahy iba jednu najkratšiu cestu vypočítanú pomocou Dijktrovho algoritmu a vydá sa po nej, nehľadiac na veľkosti penalizácií. Pri testoch s 10 % a 25 % blokovaných hrán táto nevýhoda nebola taká zjavná, keďže penalizácie jednotlivých hrán boli zadané vždy ako číslo z intervalu <5,10>, čo je pomerne málo, ale pri 50 % blokovaných hrán už bolo vidieť značné rozdiely medzi výsledkami WS a RGS. Z týchto dvoch stratégií teda bola efektívnejšia RGS. Bolo to zapríčinené podmienkou, ktorá zváži do úvahy práve to, či sa najviac v daný moment oplatí zaplatiť penalizáciu, alebo ísť novou najkratšou cestou s tým, že sa nepoužije novonájdená blokovaná hrana. RGS sa teda ukázala ako stratégia, ktorá sa vie lepšie prispôsobiť rozličným podmienkam v grafe.

6 ZÁVER

Táto práca sa zaoberala problémom kanadského cestujúceho (CTP). Môžeme ho definovať ako problém cestujúceho, ktorý hľadá najkratšiu možnú cestu z bodu *A* do bodu *B* vo vopred neznámom prostredí. V druhej kapitole boli predstavené typy tohto problému a ich rozdelenie, spolu s metódami ich riešenia. V ďalšej kapitole boli podrobnejšie opísané stratégie, ktoré sa neskôr implementovali. Boli to stratégie reprezentujúce druh CTP s neznámou pravdepodobnosťou blokácie. Pre poddruh problému, v ktorom sú hrany neobnoviteľné boli implementované stratégie: GS, RS, CS. Pre typ, kde sa hrany dajú obnoviť za cenu penalizácie, boli vybrané stratégie: WS, RGS. Na testovanie efektívnosti týchto stratégií boli vytvorené dve aplikácie v programovacom jazyku Python. Najväčšou výhodou tohoto jazyka je objektovo orientovaný prístup, ktorý nám umožňuje aplikáciu ľahko upraviť, poprípade pridať ďalšie stratégie na testovanie.

Pri testovaní bolo zistené, že zo stratégií GS, RS a CS bola najefektívnejšia GS. Čo sa týka cien ciest, stratégie GS a CS na tom boli rovnako, ale GS bola značne rýchlejšia. RS sa ukázala v každom teste ako neefektívna z dôvodu započítavania ceny *backtracku*. Pri porovnaní stratégií WS a RGS vyšla ako efektívnejšia druhá menovaná, pretože sa dokázala lepšie prispôsobiť danému grafu. Naopak efektivita nám pri WS s narastajúcim počtom blokovaných hrán klesala.

Zoznam použitej literatúry

[1] BAR-NOY, A.; SCHIEBER, B.; The Canadian Traveller Problem. In *Proceedings of Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 91)*, USA, 1991, pp. 261-270.

[2] XU, Y. F.; HU, M. L.; SU, B. H.; ZHU, B. H., ZHU, Z. J. The Canadian Traveller Problem and its Competitive Analysis. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence,* Atlanta, Georgia, 2009, pp 195-205.

[3] SU, B.; XU, Y. F.; HU, M. L.; ZHU, B. H.; ZHU, Z. J. Online Recoverable Canadian Traveller Problem. In *Proceedings of the international conference on management science and engineering*, 2004, pp. 633-639.

[4] WESTPHAL S. A Note on the k-Canadian Traveller Problem. In *Information Processing Letters,* 2008, pp. 87-89.

[5] LIAO, C.; HUANG, Y. Generalized Canadian Traveller Problems. *International Symposium on Algorithms and Computation*, 2012, pp. 701-712.

[6] EYERICH, P.; KELLER, T.; HELMERT, M. High-Quality Polices for the Canadian Traveler's Problem. In *In Proc. The 24th AAAI Conf. On Artificial Intelligence,* Atlanta, Georgia, 2009, pp. 51-58.

[7] CHIARANDINI,M.; Ant Colony Optimization Exercises [online].[cit. 2019-5-20] Dostupné z:
https://pdfs.semanticscholar.org/1ea9/ac618282a12b1b6564fec2dd2d3d0f142a21.pdf

[8] FRIED, D.; SHIMONY, S., E.; BENBASSAT, A.; WENNER, C.; Complexity of Canadian Traveler Problem Variants. In: *Theoretical Computer Science*, 2013, 487(16), pp. 1-16. ISSN 0304-3975.

[9] BNAYA, Z.; FELNER A.; SHIMONY, S. E. Canadian Traveler Problem with Remote Sensing. In: *Proc. IJCAI*, 2009, pp. 437-442.

[10] BNAYA, Z.; FELNER A.; FRIED, D.; MAKSIN, O.; SHIMONY S. E. Repeated-Task Canadian Traveler Problem, Palo Alto, CA, 2011, pp. 24-30.

[11] NIKOLOVA, E.; KARGER, D. R.; Route Planning under Uncertainty: the Canadian Traveller Problem. In: *Proceedings the 23rd AAAI Conference on Artificial Intelligence,* Chicago, Illinois, 2008, pp. 969-974.

[12] PAPADIMITRIOU, C. H; YANNAKAKIS, M. Shortest Paths without Map. In *Theoretical Computer Science,* 1991, pp 127-150.

[13] RUSSEL, S.; NORVIG, P.; Artifical Intelligence: A Modern Approach, *Third Edition*, New Jersey, 2010, pp 184

Zoznam použitých skratiek

CTP	Canadian Traveller Problem
SCTP	Stochastic CTP
RCTP	Recoverable CTP
SRCTP	Stochastic RCTP
GS	Greedy Strategy
RS	Reposition Strategy
CS	Comparison Strategy
HOP	Hindsight Optimization
ORO	Optimistic Rollout
UCT	Upper Confidence bounds applied to Trees
UCTO	Optimistic UCT

Zoznam príloh

Príloha A – Dátový nosič (CD) s elektronickou verziou tejto práce a vytvorenými aplikáciami