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

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Sebastián Filip**
AUTHOR

VEDOUCÍ PRÁCE — **RNDr. Jiří Dvořák, CSc.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Sebastián Filip**
Studijní program:	Strojní inženýrství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**RNDr. Jiří Dvořák, CSc.**
Akademický rok:	2016/17

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 diplomové 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 robota v částečně neznámém prostředí.

Cíle diplomové práce:

1. Vypracovat přehled typů daného problému a existujících metod řešení.
2. Navrhnout a implementovat metody pro vybrané typy tohoto problému.
3. Provést ověřovací a srovnávací experimenty.

Seznam literatury:

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.

EYERICH, P.; KELLER, T.; HELMERT, M. High-Quality Policies for the Canadian Traveler's Problem. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010), 2010, pp. 51-58.

XU, Y. F.; HU, M. L.; SU, B.; ZHU, B. H.; ZHU, Z. J. The Canadian Traveller Problem and Its Competitive Analysis. Journal of Combinatorial Optimization, 2009, vol. 18, no. 2, pp. 195-205.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

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

ABSTRAKT

Tato práce se zabývá problémem kanadského cestujícího (CTP), který se dá definovat jako problém hledání nejkratší cesty ve stochastickém prostředí. V rešeršní části práce je zpracován přehled typů tohoto problému a k nim existujících metod řešení. V dalších částech se práce zaměřuje na stochastickou variantu CTP (SCTP), pro kterou jsou vybrané metody řešení (strategie) probrány více do hloubky. Zároveň jsou prezentovány i originální strategie pojmenované UCTO2 a UCTP. Dále se práce zabývá popisem okenní aplikace implementované v jazyku Java. Ta byla vyvinuta pro ověření a otestování správné funkce vybraných strategií. Nakonec jsou vyhodnoceny provedené experimenty, a z nich plynoucí srovnání vybraných strategií.

ABSTRACT

This thesis deals with Canadian traveller problem (CTP), which can be defined as the shortest path problem in a stochastic environment. The overview of different CTP variants is presented in theoretical part of this thesis, as well as known solutions to these variants. In the next parts, the thesis focuses on the stochastic variation of CTP (SCTP). For this variant chosen solutions (strategies) are discussed more in depth. At the same time, the original strategies named UCTO and UCTP are presented. Further, the thesis deals with the description of a window application implemented in Java, which has been developed to validate and test the functionality of selected strategies. The final part contains experiments and comparison of selected strategies.

KLÍČOVÁ SLOVA

problém kanadského cestujícího (CTP), adaptivní strategie, stochastická optimalizace

KEYWORDS

Canadian traveller probelm (CTP), adaptive strategy, stochastic optimization

BIBLIOGRAFICKÁ CITACE

FILIP, S. *Řešení problému kanadského cestujícího.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 73 s. Vedoucí diplomové práce RNDr. Jiří Dvořák, CSc.

PODĚKOVÁNÍ

Touto cestou bych rád poděkoval rodičům a přítelkyni za jejich vytrvalou podporu v průběhu celého studia. Zároveň bych rád poděkoval vedoucímu mé diplomové práce panu RNDr. Jiřímu Dvořákovi, CSc. a panu Ing. Petru Šoustkovi za jejich odbornou pomoc a cenné připomínky.

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 1. 3. 2017 ……

Bc. Sebastián Filip

OBSAH

1 Úvod

1.1 Úvod do problému kanadského cestujícího

Problém kanadského cestujícího, dále značen CTP (*Canadian Traveller Problem*), byl poprvé publikován roku 1991 v článku „*Shortest paths without map*“ [1]. Jeho autoři Papadimitrijou a Yannakakis položili základy pro CTP, když se zabývali problémem hledání nejkratší trasy ve známém prostředí, ale s nekompletními informacemi o jeho stavu. CTP může být charakterizován následovně. Cestující má v plánu dojet z výchozího do cílového města a má s sebou kompletní mapu všech cest, avšak není známý stav průchodnosti jednotlivých úseků silnic. Ty mohou být neprůjezdné, například kvůli sněhovým závějím, nebo spadlým stromům znemožňující cestujícímu průjezd. Stav cest je možný vypozorovat pouze při dojezdu do výchozího místa příslušného úseku. Problém je navrhnout strategii, která by zaručovala dobrou cenu cesty z výchozího do cílového města za předpokladu výše popsané nejistoty[1].

Problém je možno popsat grafem $G(V,E)$, kde hrany E představují úseky silnic a vrcholy V představují místa, z kterých je možno vypozorovat stav přilehlých úseků silnic (tj. hran). Cena nebo čas, který je potřeba vynaložit na průchod danou hranou představuje ohodnocení dané hrany grafu $w(e)$. V rámci této práce jsou v následujícím textu použita tato označení: hrany pro úseky silnic, vrcholy nebo uzly pro hranice těchto úseků a agenta pro cestujícího.

Jak je výše naznačeno, tak jediným zdrojem nejistoty je nekompletní informace o stavu hran. Agent buď ví, je-li daná hrana blokována nebo průjezdná, protože to vypozoroval dojetím do uzlu s ní incidentním, nebo vůbec neví o jejím stavu. Proto je stavový prostor CTP omezen výrazem $|V| \cdot 3^{|E|}$, kde $|V|$ je počet vrcholů a $|E|$ počet hran grafu. Stavový prostor tedy roste exponenciálně s přibývajícím počtem hran s neznámým stavem (tzv. stochastické hrany). Nebude-li řečeno jinak, tak všechny hrany v grafu CTP jsou považovány za stochastické.

Následujícím popisem je ukázáno, jak vypadá pohyb agenta: Na začátku si prostředí náhodně vygeneruje instanci I grafu $G(V,E)$, ve které budou některé hrany označeny jako blokované. Agent zná graf $G(V,E)$, ale stav hran instance I je mu ukazován postupně a to tak, že vidí stavy všech hran incidentních k vrcholu jeho momentální pozice. Agent provede tah, tj. vybere si neblokovanou hranu e incidentní k vrcholu jeho momentální polohy a tím se dostane do vrcholu, ve kterém vybraná hrana končí. Za tento tah agent zaplatí cenu $w(e)$. Cesta agenta je posloupnost tahů z počátečního do cílového vrcholu. Cena cesty je suma všech ohodnocení hran na cestě a zápis $C(x,y)$ značí nejmenší cenu cesty z vrcholu x do vrcholu y. Způsob vybrání hrany je dán strategií agenta. Typ strategie, která musí reagovat na informace získané za běhu, se označuje jako adaptivní (online). Většina strategií řešících CTP je adaptivních.

[1] Za předpokladu znalosti cestujícího o stavu cest by bylo řešením najít nejkratší trasu mezi výchozím a cílovým městem. Tento problém představuje typický problém známý z teorie grafů jako SPP (*Shortest Path Problem*), na který již existuje mnoho exaktních metod zaručujících nalezení optimální cesty. Mezi nejznámější z nich patří Dijkstrův algoritmus, Bellman-Fordův algoritmus a A* algoritmus.

Kritérium pro hodnocení kvality online strategie je obvykle konkurenční poměr (*competitive ratio*), který může být definován následovně: pro jakoukoli instanci *I* grafu je celková cena cesty online strategie maximálně *c*-krát větší než offline optimum instance *I*, kde *c* je konkurenční poměr. Alternativním kritériem je nejhorší případ (*worst case*), kde, jak napovídá název, je kvalita strategie dána exaktně vyjádřenou nejhorší možnou cenou pro daný graf *G*(*V*,*E*). Demonstrace rozdílu mezi těmito dvěma kritérii může být uvedena následujícím příkladem:

„*Nechť jsou vygenerovány dvě instance grafu I_1 a I_2. Optimální cena offline cesty I_1 je 10 dnů a optimální cena I_2 je 2 dny. Při uvážení kritéria konkurenční poměru by agent zvolil online strategii, která by vedla k cestě dlouhé 20 a 4 dny postupně pro I_1 a I_2. Následováním kritéria nejhoršího případu by agent preferoval strategii, která by vždy vedla k cestě dlouhé 15 dnů. Zdá se, že obě tato kritéria mohou být užitečná za určitých okolností…*“ [3] (překlad vlastní)

Z teoretického pohledu je problém CTP definován jako určení výpočetní složitosti navrhnutí výše zmíněné strategie. Papadimitrijou a Yannakakis dokázali, že když počet možných blokací není fixní, tak navržení strategie, která by garantovala daný konkurenční poměr, je PSPACE-úplný problém, zatímco optimalizace očekávaného poměru je #P-těžký problém. [1]

CTP může být reprezentován jako speciální případ MDP (*Markov Decision Processes*), a to sice jako deterministický POMDP(*Partially Observed MDP*), dále označený jako DPOMDP. V MDP je dán stavový prostor, prostor akcí a přechodová funkce pravděpodobností přechodů z jednoho stavu do druhého za použití dané akce. Dále je dána funkce cen (odměn), která přiděluje očekávanou cenu, kterou agent zaplatí (dostane) při použití dané akce v daném stavu. Řešením MDP je najít strategii, tj. jaké akce použít v jakém stavu, která by minimalizovala (maximalizovala) očekávanou cenu (odměnu) při cestě z výchozího do konečného stavu. POMDP je navíc rozšířený o množinu pozorování, které agent může udělat pokaždé, když provede akci, a dále je známá funkce pozorování, která určuje pravděpodobnostní rozdělení nad všemi pozorováními. Je dokázáno, že optimální strategie může být nalezena v polynomiálním čase k velikosti MDP (tj. počtu stavů a akcí), např. lineárním programováním. Velikost MDP ale často expanduje do nekonečna a exaktní řešení jsou proto často exponenciálního charakteru (někdy lze MDP reformulovat tak, aby se zredukovala velikost stavového prostoru). CTP může být vyjádřen jako MDP s konečným počtem stavů, kde stavy jsou vrcholy a akce jsou přechodové funkce zprostředkované hranou na sousední vrchol. Jak bylo uvedeno dříve, stavový prostor CTP roste exponenciálně s přibývajícím počtem hran a exaktní řešení MDP pro problém kanadského cestujícího je proto exponenciální v čase. I přes tuto skutečnost má smysl v některých případech vyjádřit CTP jako DPOMDP, jednak pro speciální typy grafů a jednak v situacích, kde se nějakým „chytrým“ způsobem dá zredukovat stavový prostor. Obecně je CTP s fixními stavy hran složitější problém, než kdyby docházelo k opětovnému vyhodnocení stochastických hran po každé agentově akci. Možná paradoxně, jelikož je agent lépe informován o stavu vypozorovaných hran, ale z hlediska MDP potřebuje tato verze mnohem větší stavový prostor, protože každý stav je závislý na všech předchozích pozorováních.

1.2 Shrnutí obsahu jednotlivých kapitol práce

Ve druhé kapitole je popsán přehled typů daného problému CTP na základě různých parametrů, jako je obnovitelnost blokací nebo znalost cestujícího o pravděpodobnostech blokací. Společně s popisem jednotlivých typů je shrnuta základní literatura s nimi spojená a metody řešení v ní se vyskytující. V poslední podkapitole jsou krátce shrnuty další „speciální“ typy CTP.

Ve třetí kapitole jsou vybrané strategie probrány více do hloubky a následně jsou jejich nežádoucí vlastnosti demonstrovány na příkladu. Dále jsou představeny dvě originální strategie pojmenované UCTO2 a UCTP.

Čtvrtá kapitola je věnována popisu implementace vybraných strategií, včetně popisu okenní aplikace, která byla vytvořena pro ověření správné funkce.

Pátá kapitola se podrobněji zabývá nastavením parametrů vybraných strategií. Jsou udělány experimenty, které mají za cíl nastínit vliv těchto parametrů na kvalitu řešení a na časovou náročnost výpočtů. Dále je zde popis a výsledky navrhnutých testů, které byly udělány pro srovnání vybraných strategií.

2 Typy CTP

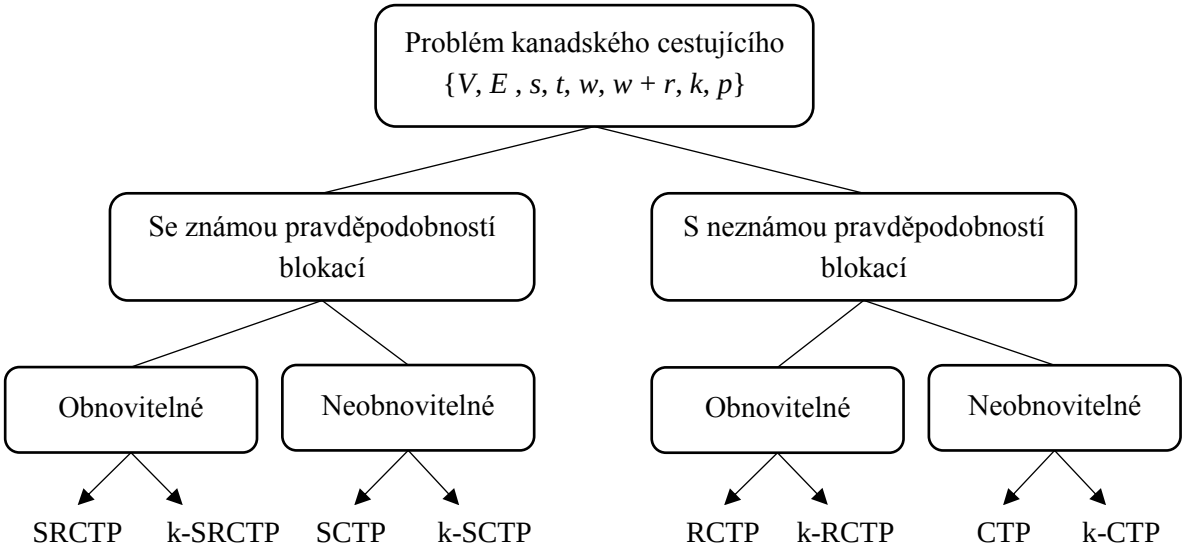
Od prvního představení problému již uběhlo mnoho let, během kterých vyšla řada článků a studií týkajících se problému kanadského cestujícího a jeho modifikací. Řešení problému může být přímo aplikováno v reálných situacích, jako jsou navigace v dopravě, navigace robota v dynamickém nebo částečně známém prostředí, adaptivní transportní systémy a jiné další. Zejména rychlý rozvoj moderních technologií, jako je GPS a mobilní komunikace, přispívá k rozvoji dynamické navigace založené na informacích sbíraných v reálném čase. Proto se v nedávné době dostalo problému kanadského cestujícího relativně hodně pozornosti a byly studovány různé jeho typy. V této kapitole jsou vyjmenovány modifikace, jež byly studovány, a k nim krátce shrnuty příslušné existující metody řešení.

Problém může být klasifikován do dvou hlavních skupin dle znalosti agenta o pravděpodobnosti blokací. První skupinou jsou problémy, ve kterých pravděpodobnost není vůbec známá. Strategie jsou zde proto obvykle heuristického charakteru. Druhou skupinu tvoří problémy stochastické (jsou známy pravděpodobnosti blokace) a strategie tedy s těmito pravděpodobnostmi obvykle exaktně počítají.

Další klasifikace problému je možná dle charakteru hran a to na grafy neobnovitelné, kde jednou blokovaná hrana zůstává neprůjezdná napořád, a obnovitelné, přičemž je známá cena (tj. penalizace), která je zapotřebí k znovuotevření hrany. V článku [3] jsou tyto penalizace asociovány s vrcholy grafu, kdežto v novějších pracích [6, 7] se tyto penalizace vážou ke hranám. Obecně se tato modifikace považuje za zobecnění CTP a může být aplikována na všechny typy CTP a to tak, že pro neobnovitelné platí, že čas do obnovy je nekonečný. Nadmnožinou k této klasifikaci jsou tzv. dvojitě ohodnocené grafy, pro které platí, že každá hrana má dvě ohodnocení a není známo, která z nich bude aktuální (tj. ohodnocení hran má charakter diskrétních náhodných proměnných). Tyto grafy jsou modelem například pro řešení navigace v městské dopravě, při možnosti výskytu hustějšího provozu nebo snížení kapacit cest (například kvůli dopravní nehodě). Obnovitelné CTP jsou podmnožinou dvojitě ohodnocených grafů, což vyplývá ze skutečnosti, že každý obnovitelný CTP lze převést na dvojitě ohodnocený graf a to tak, že druhé ohodnocení hrany z obnovitelných CTP vznikne přičtením času obnovy k ohodnocení dané hrany.

Při snaze o exaktní vyjádření konkurenčního poměru se vyskytuje modifikace problému, kde je maximální počet možných blokací dán parametrem *k*. Modifikace je spíše určena pro teoretické bádání. Je důležité poznamenat, že pro *k* rovno počtu hran grafu je problém k-CTP ekvivalentní klasickému CTP. Každá strategie, která zná počet maximálních blokací hran, se může rozšířit o podmínku, která by při zjištění nemožnosti blokování žádné další hrany, změnila strategii agenta tak, aby šel nejkratší cestou z momentální pozice do cílové.

Formální rámec pro typy CTP uvedené v podkapitolách 2.1 a 2.2 může být vyjádřen jako osmice $\{V, E, s, t, w, w + r, k, p\}$, kde V je množina vrcholů, E je množina hran, s je výchozí vrchol, t je cílový vrchol, w je množina ohodnocení hran, $w + r$ je množina ohodnocení pro zablokované hrany (pro neobnovitelné je to ∞), k je parametr udávající maximální počet blokací a p je množina pravděpodobností blokace hran.



Obr. 1 Přehled typů CTP

Diagram s přehledem typů CTP probíraných v této kapitole je uveden na obr. 1. Existují však i jiné varianty CTP, vybrané z nich jsou krátce shrnuty v podkapitole 2.3.

2.1 CTP s neznámou pravděpodobností blokace

V této variantě agent zná graf, ale některé hrany mohou být neprůjezdné. Průjezdnost dané hrany je vypozorována až při dojezdu do vrcholu k ní incidentní. Problém je tedy plně deterministický, až na počáteční nejistotu stavu hran. Formální rámec pro všechny podtypy CTP s neznámou pravděpodobností blokace je dán následovně jako sedmice $\{V, E , s, t, w, w + r, k\}$. Významy jednotlivých symbolů je možno najít v úvodu této kapitoly.

2.1.1 Problém kanadského cestujícího (CTP)

CTP představuje původní variantu, jak ji Papadimitrijou a Yannakakis formulovali. Problém je v podstatě hra dvou hráčů mezi agentem a prostředím, které nastavuje podmínky prostředí. Jednou blokovaná hrana zůstává zablokovaná napořád. Vstupními parametry jsou graf $G(V,E)$, ohodnocení hran w, výchozí vrchol s a cílový vrchol t. Předpokládá se, že graf $G(V,E)$ je propojený i po odstranění všech blokovaných hran (vždy existuje cesta z s do t). Problém CTP je tedy možno vyjádřit dle výš uvedené sedmice následovně jako: $\{V, E , s, t, w, \infty, k = |E|\}$. Existující strategie řešící CTP jsou následovné:

Greedy Strategy (GS)
Tato strategie se v literatuře občas označuje názvem Optimism (OMT), například v [2]. Pro svoji jednoduchost a popularitu může být použitá jako základní měřítko pro ostatní strategie. Agent přepokládá, že všechny hrany jsou průjezdné a vydává se po nejkratší cestě do cílového vrcholu. V případě, že v místě s_i zjistí, že hrana e_i je zablokovaná, vydává se nejkratší cestou z s_i do t bez použití hrany e_i. Strategie se opakuje tak dlouho, dokud není

dosaženo cíle *t*. Pro GS byl dokázán v [20] konkurenční poměr ($2^{k+1} - 1$), kde *k* je počet objevených blokací. [4]

Reposition Strategy (RS)

Agent přepokládá, že všechny hrany jsou průjezdné a vydává se po nejkratší cestě do cílového vrcholu. V případě, že v místě s_i zjistí, že hrana e_i je zablokovaná, vydává se zpět na začátek *s*, odstraní z grafu blokovanou hranu e_i, zjistí novou nejkratší cestu a po ní se vydává. Strategie se opakuje tak dlouho, dokud není dosaženo cíle *t*. Pro RS byl dokázán v [21] konkurenční poměr ($2 \cdot k+1$), kde *k* je počet objevených blokací. [4, 5]

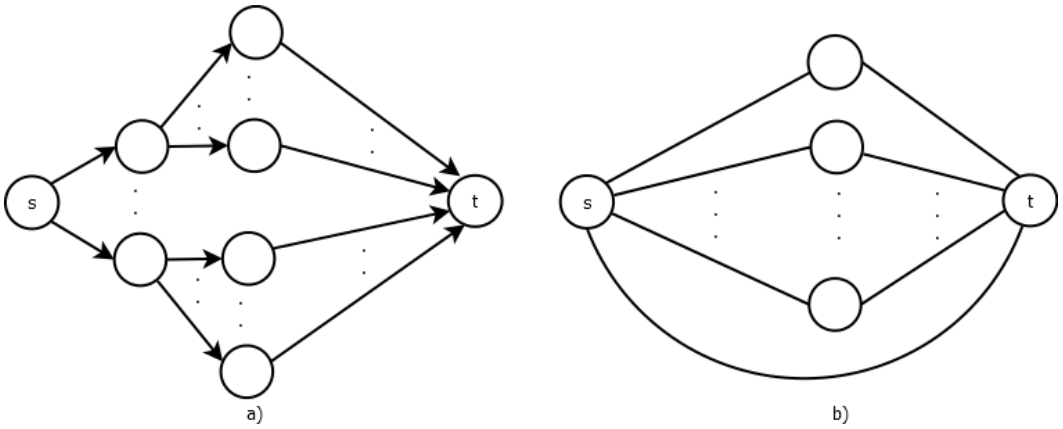
Comparison Strategy (CS)

Strategie kombinuje RS a GS. Cestující přepokládá, že všechny cesty jsou průjezdné a vydává se po nejkratší cestě do cílového vrcholu. V případě, že se v místě s_i zjistí, že hrana e_i je zablokovaná, tak je použita strategie GS nebo RS podle následující podmínky:

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

kde $C(s_i,t \mid E_i)$ vyjadřuje nejmenší cenu cesty z s_i do t po odstranění všech dosud známých zablokovaných hran E_i. Podobně $C(s,t \mid E_i)$ vyjadřuje nejmenší cenu cesty z s do t bez všech dosud známých zablokovaných hran. Při platnosti podmínky použije cestující GS, jinak použije RS. [4]

Z dalších publikací stojí za zmínku Nikolova a Karger [10], kteří se zabývali CTP definovaným na dvou speciálních typech grafů. Prvním z nich je orientovaný acyklický graf, dále značený jako DAG (*Directed Acyclic Graph*) a druhým je graf skládající se z několika vrcholově oddělených cest spojujících výchozí a cílový vrchol, dále značený jako DPG (*Disjoined-Path Graph*). Pro názornost jsou příklady obou typů grafů znázorněny na obr. 2. V jejich práci byla pro DAG prezentována strategie řešící CTP v polynomiálním čase. Výhodou DAG je fakt, že se agent nemůže vrátit do přechozího vrcholu, což výrazně zmenšuje stavový prostor (stavy nejsou závislé na přechozím pozorování). Jejich řešení je založeno na dynamickém programování. Strategie je analogická s GS a pro DAG zaručuje optimalitu (důkaz může být proveden matematickou indukcí začínající cílovým vrcholem). Pro DPG je ve stejném článku uvedena heuristika s názvem *Expected minimum distance*, která je optimální na grafech ohodnocených {0,1}, ale i v obecnějším případě (ohodnocení (0,1)) bylo dosaženo výsledků blízkým optimálním. Přesto zůstává otázkou, jak by si heuristika vedla ve více obecných grafech.



Obr. 2 a) Orientovaný acyklický graf, DAG
b) Graf skládající se z několika vrcholově oddělených cest, DPG

2.1.2 k-Problém kanadského cestujícího (k-CTP)

Jedinou modifikací oproti předchozí variantě je horní hranice zablokovaných cest *k*, která je dána jako parametr. Tento typ problému je podmnožinou k CTP, kde $k = |E|$. Důvodem vzniku k-CTP byl fakt, že obvykle bývá počet zablokovaných hran relativně malý k celkovému počtu hran a také fakt, že parametrizovat selhání je poměrně běžné u problémů pracujících s neurčitostí (např. problém byzantských generálů). [3]

Westphal (2008) publikoval důkaz, že neexistují žádné deterministické online algoritmy, které by dosáhly nižšího konkurenčního poměru než $(2 \cdot k+1)$. Rovněž navrhl jednoduchý algoritmus[2], který dosahuje spodní meze, a také dokázal, že pro rando-mizované algoritmy je hranice $(k+1)$. [5]

Problém k-CTP je možno vyjádřit dle sedmice uvedené v začátku podkapitoly 2.1 následovně jako: $\{V, E, s, t, w, \infty, k\}$. Existující strategie řešící k-CTP je:

Randomized Reposition Strategy (RRS)

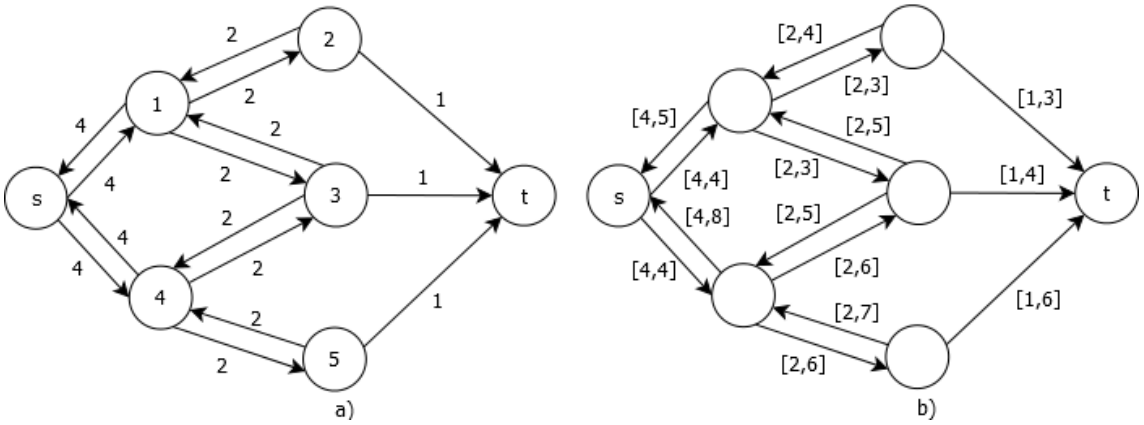
Je to modifikace RS s přidaným prvkem náhody. Počítá s grafem, který se skládá z *n* vrcholově oddělených *s-t* cest $P_1 \ldots, P_n$ (tj. jak dříve uvedeno DPG viz Obr. 2). Nechť cena cesty P_i je C_i a cesty *P* jsou uspořádány dle ceny vzestupně tj. $C_1 \leq C_2 \leq \ldots \leq C_n$. Vybere se $k+1$ cest *P* z grafu $G(V,E)$. Je jisté, že jedna z nich bude offline optimum (ale není jasné která). Dále se definuje příslušná distribuce pravděpodobností pro *P* a vybere se cesta P_i, po které agent zkusí projít. Jestli se na P_i vyskytne blokace, tak se agent vrací do *s* a strategie se opakuje s menším seznamem cest $\{P\} - P_i$. Konkurenční poměr RRS garantuje navrhovanou spodní hranici $(k+1)$.

RRS se skládá ze dvou dílčích algoritmů. První z nich je určen pouze pro grafy splňující podmínku podobných cen cest. Druhý algoritmus je pak navržen takovým způsobem, že dokáže dělit cesty do tříd, tak aby v každé třídě byla splněna podmínka podobných cen cest. RRS kombinuje výše zmíněné algoritmy a díky toho může být aplikován na DPG s libovolně ohodnocenými hranami. [9]

[2] Algoritmus pojmenoval *backtrack* (návratový), později na Westphala navázal článek *The Canadian Traveller Problem and its competitive analysis* [5], kde z *backtrack* vznikla Reposition strategie.

2.1.3 Obnovitelný CTP (RCTP)

V této variantě, poprvé zavedené v [3], je každý vrchol asociován s dobou obnovení (penalizací), která je zapotřebí k znovuotevření k němu incidentních hran v případě blokace. V této práci se ve shodě s [6, 7] uvažuje obecnější případ, kde každá hrana *e* je asociována s penalizací *r*(*e*). Předpokladem pro toto zobecnění je práce s orientovaným grafem[3]. Na obr 3 a) je znázorněn ohodnocený orientovaný graf, ve kterém jsou ve vrcholech vyjádřeny doby penalizací *r*(*e*), k němu ekvivalentní dvojitě ohodnocený graf je na obr. 3 b), kde je nad každou hranou *e* uspořádaná dvojice [*w*(*e*), *w*(*e*) + *r*(*e*)] (*w*(*e*) = cena hrany, *r*(*e*) = cena obnovení). Z obrázku je zřejmé, že RCTP lze reprezentovat grafem ohodnoceným diskrétními náhodnými proměnnými.



Obr. 3 a) Ohodnocený orientovaný graf s penalizačními časy ve vrcholech
b) Ekvivalentní dvoj-ohodnocený graf k *a*) (ve vrcholu *s* je *r*(*e*) = 0)

Problém RCTP je možno vyjádřit dle sedmice uvedené v podkapitole 2.1 následovně jako: {*V*, *E* , *s*, *t*, *w*, *w* + *r*, *k* =|*E*|}, varianta {*V*, *E* , *s*, *t*, *w*, *w* + *r*, *k*} je dále značena k-RCPT, kde *k* je dán jako parametr určující maximální počet blokací. Existující strategie řešící RCTP jsou následovné:

Waiting Strategy (WS)
Agent se s výchozího uzlu vydává nejkratší cestou do cílového uzlu. Při zjištění neprůjezdné hrany *e* čeká na obnovení, zaplatí penalizaci *r*(*e*) a pokračuje dál. Toto se opakuje, dokud agent nedojde do cílového uzlu. [6]

Recovery Greedy Strategy (RGS)
Agent se z výchozího uzlu vydává nejkratší cestou do cílového uzlu. Při zjištění neprůjezdné hrany *e* je první možností agenta čekat a zaplatit penalizaci *r*(*e*), nebo pokračovat novou

[3] Každý neorientovaný graf lze převést na orientovaný.

nejkratší cestou do cílového uzlu. Agent si vybere tu variantu, která je výhodnější (má menší očekávanou cenu cesty do cílového uzlu). [6]

Recovery Greedy & Reposition Strategy (GR)
Tato strategie byla původně navrhnuta k řešení dvojitě ohodnoceného CTP, kde počet maximálních blokací je dán parametrem *k*. Strategie lze tedy aplikovat na k-RCTP. Je založena na třech dílčích strategiích. První je GS, která nerespektuje fakt, jestli agent našel nějakou novou zablokovanou hranu, druhá je rovněž GS, která vede agenta směrem do cílového uzlu až do doby, dokud neobjeví blokovanou hranu. Poslední dílčí strategií je RS, která vede agenta do doby, dokud neobjeví blokovanou hranu. Hlavní součástí je algoritmus, který pomocí podmínek řídí, jaké dílčí strategie má být použito. [8]

Bar-Noy & Schieber RCTP strategie
Strategie je optimální z hlediska nejhoršího případu za předpokladu, že ceny obnovení nejsou relativně velké oproti cenám hran[4], a také že počet blokací *k* je dopředu znám (varianta k-RCTP). Pro graf $G(V,E)$, kde n je počet uzlů a m je počet hran lze přiřadit $k = |E| = m$ a řešit problém RCTP, ale problém může nastat u větších grafů, protože časová složitost hlavního algoritmu je $O(k^2 \cdot m + k \cdot n \cdot log(n))$.

Základním kamenem této strategie je algoritmus, který se volá rekurzivně pro *k* a je složen ze dvou kroků – označovacího a aktualizujícího. Označovací krok je podobný Dijkstrovu algoritmu a začíná od cílového vrcholu *t* značit vzdálenosti do ostatních vrcholů, přičemž inicializační hodnota je nekonečno tj. $C(t, x) = \infty$. Najde se nejmenší vzdálenost $C(t, x)$ a přidá se do seznamu L (t je první, který se přidá do L). V aktualizujícím kroku se pro všechny incidentní vrcholy y k vrcholu x zkontroluje orientační cena vrcholů y. Jestli se zlepší, tak se zaktualizuje. Optimalita z hlediska nejhoršího případu spočívá v tom, že jsou deterministicky určené všechny ceny cest v případě všech možných selhání (tj. je určená přesná hodnota nejhoršího případu). [3]

2.2 CTP se známou pravděpodobností blokace

Následující typy CTP jsou označovány jako stochastické, proto jsou dále značeny jako SCTP. SCTP představuje variantu, kde agent zná dopředu graf (mapu) a pravděpodobnosti všech hran, že budou blokovány. Cílem je navrhnout kvalitní strategii, která by měla nízkou očekávanou cenu cesty z výchozího do cílového uzlu. Průjezdnost dané hrany je vypozorována při dojezdu do uzlu k ní incidentního. Formální rámec pro všechny podtypy SCTP se známou pravděpodobností blokace je dán následovně jako osmice $\{V, E, s, t, w, w + r, k, p\}$, kde p je množina pravděpodobností. Pravděpodobnost blokace hrany e je $p(e)$. Jinak řečeno, hrana e je průchozí s pravděpodobností $1 - p(e)$.

[4] Při zjištění, že n hran bylo blokováno a agent pokračuje jinou nezablokovanou hranou e, se předpokládá, že po doputování agenta do vrcholu, ve kterém končila e (a zaplacení $w(e)$), jsou všechny předchozí zjištěné blokované hrany n průchozí. (tj. např. pro dříve zjištěnou hranu e_i je cena $w(e_i)$ namísto $w(e_i) + r(e_i)$)

2.2.1 Stochastický CTP (SCTP, k-SCT)

V této variantě platí, že jednou blokovaná hrana zůstává zablokovaná napořád. Vstupními parametry jsou graf $G(V,E)$ a ohodnocení hran w. Problém je tedy plně deterministický, až na počáteční nejistotu stavu hran. Předpokládá se, že Graf $G(V,E)$ je propojený i po odstranění všech blokovaných hran (vždy existuje cesta z s do t). Problém SCTP je tedy možno vyjádřit dle osmice uvedené v úvody podkapitoly 2.2 následovně jako: $\{V, E , s, t, w, \infty, k = |E|, p\}$. Existující strategie řešící SCTP jsou následovné:

Hindsight Optimization (HOP)
V HOP se řeší sekvence zdeterminizovaných problémů, ze kterých je určena strategie agenta ve stochastickém prostředí. V každé pozici agenta se provede sekvence iterací zvaných *rollouts*[5]. Pomocí informace o rozdělení pravděpodobností blokací se v každém *rolloutu* vygeneruje instance daného grafu (již objevené stavy hran se nemění), ve které se řeší deterministický SPP (počítá se s „jasnozřivostí“ agenta) k odhadnutí očekávané ceny, která je dána průměrem všech cen z jednotlivých *rolloutů*. Strategie vybere agentovi akci, která má minimální očekávanou cenu. Počet *rolloutů* je dán jako parametr *N*. Více *rolloutů* vyžaduje více výpočetního času, ale odhady cen mají tendenci být více stabilní. Slabina této strategie je, že se zvedajícím počtem *rolloutů* dochází ke konvergenci k suboptimálnímu řešení. [2]

Optimistic Rollout (ORO)
Tato strategie je modifikace předchozí a také pracuje s *N rollouty*, které se provádí v každé pozici agenta. Hlavním rozdílem je způsob, jakým jsou *rollouty* provedeny a tím reaguje na problém konvergence k suboptimálnímu řešení HOP (kde příčina je „jasnozřivost“ cestujícího). Při výpočtu očekávané ceny cesty z agentovy pozice s_i do t se použije strategie analogická s GS popsanou v podkapitole 2.1.1, tj. cena *rolloutu* je suma cen všech prošlých hran, kde se vždy začíná nejkratší cestou do cíle a při případné blokaci se znova najde nová nejkratší cesta, kterou se pokračuje a toto se opakuje až do dosažení cíle. Strategie vybere agentovi akci, která má minimální očekávanou cenu. [2]

Blind UCT (UCTB)
Strategie UCT (UCT je akronym vytvořený z *Upper Confidence bounds applied to Trees*), poprvé představená v [15], je nejmodernější algoritmus pro mnoho problémů pracujících s nejistotou. Tato strategie je podobně jako HOP a ORO založena na *rolloutech,* ze kterých se určí očekáváná cena. Hlavním rozdílem mezi nimi je způsob, jakým jsou *rollouty* provedeny. V UCT mají *rollouty* na sobě jistou závislost. Díky této závislosti jsou *rollouty*, které se v minulosti jevily neperspektivně, vybírány s přibývajícím počtem *rolloutů* méně často. Naopak neprozkoumané oblasti stavového prostoru jsou zvýhodňovány.

[5] Z důvodů neexistujícího výstižného českého ekvivalentu je zde použito původní anglické slovo. Tento termín se do češtiny překládá mj. jako dojezd letadla po přistání. V souvislosti s CTP se vlastně jedná o pokusný dojezd do cíle.

Jméno UCTB je vytvořeno spojením UCT a anglického slova pro slepost (*Blind*) a to kvůli faktu, že varianta strategie nebere v úvahu specifické informace týkající se CTP. Z tohoto důvodu je často vyžadováno neúnosně mnoho *rolloutů*, aby UCTB konvergovala ke kvalitní strategii. [2]

Optimistic UCT (UCTO)
Úzce navazuje na UCTB, ve snaze dát hlavnímu algoritmu směr k řešení a tím zapříčinit rychlejší konvergenci ke kvalitní strategii. Více informací je možné najít v podkapitole 3.3.1. [2]

2.2.2 Stochastický obnovitelný CTP (SRCTP)

V této variantě je každá hrana asociována s cenou obnovení (penalizací) r, která je zapotřebí ke znovuotevření v případě blokace. Jak zmíněno dříve, je možné tuto variantu reprezentovat jako graf ohodnocený náhodnými diskrétními proměnnými. Problém SRCTP je tedy možno vyjádřit dle osmice uvedené v úvodu podkapitoly 2.2 následovně jako: $\{V, E, s, t, w, w + r, k = |E|, p\}$. Existující strategie řešící SRCTP jsou uvedeny dále:

Metoda hrubé síly
Vychází z faktu, že SRCTP lze reprezentovat grafem ohodnoceným diskrétními náhodnými proměnnými (NDP). Stochastická hrana e bude ohodnocena jednou z hodnot množiny $\{r(e), r(e)+w(e)\}$. Pravděpodobnost, že bude mít hrana e hodnotu $r(e)$ je $p(e)$, naopak platí, že pravděpodobnost ohodnocení hodnotou $r(e) + w(e)$ je $1 – p(e)$.

Jsou-li hrany ohodnocené NDP, znamená to, že cena libovolné cesty je také NDP a je definována jako suma náhodných diskrétních proměnných hran na dané cestě. Lze tedy vyjádřit pro danou cestu všechny její možné ceny. Skládá-li se cesta C z n hran, je možných 2^n hodnot NDP cesty C. Tyto hodnoty lze seřadit od nejlepší po nejhorší a spočítat pro každou hodnotu její příslušnou pravděpodobnost. Při použití výše uvedeného postupu na všechny možné cesty z s do t lze zobrazit kumulativní distribuční funkci přiřazující každé ceně pravděpodobnost, se kterou může být cesta o dané ceně dosažena.

Metoda zaručuje nalezení optimálního řešení, ale nemá praktické využití, jelikož počet NDP cen cest roste exponenciálně. Nicméně metoda slouží jako dobrá demonstrace problému a jeho náročnosti, rovněž může být použita pro malé grafy k nalezení optimálního řešení. Podrobnější popis i s příkladem je možno najít v [12].

Bar-Noy & Schieber SRCTP strategie
Kritériem strategie je hledisko nejhoršího případu za předpokladu, že ceny obnovení nejsou relativně velké oproti cenám hran. V každém kroku si agent vybere z prioritního seznamu $x_1, x_2 \ldots x_j$ sousedů vrcholu x, ve kterém se nachází. Agent se snaží projet hranou (x,x_1), pokud je blokována volí hranu (x,x_2) a tak dále. Jestli všechny hrany selžou, tak čeká ve vrcholu x a zaplatí cenu obnovení $r(x)$.

Tvorba prioritních seznamů je udělána v označovacím algoritmu, který je analogický s Dijsktrovým algoritmem. Je udržována množina L označených vrcholů. Pro každé x ležící v L je znám prioritní seznam a orientační cena cesty z x do cílového vrcholu t dále značená jako $E(x)$. Inicializační hodnota $E(x)$ pro všechny vrcholy mimo t je dána jako nekonečno. Označovací krok najde vrchol x s minimální orientační cenou a přidá jej do L (orientační cena x je finální). Poté následuje aktualizující krok, ve kterém se pro každého souseda y vrcholu x zkontroluje, zdali se přidáním hrany (y,x) do jeho prioritního seznamu zlepší jeho očekáváná orientační cena. Pokud ano, dochází k aktualizaci, tj. vloží se x k řazenému seznamu sousedů y podle cen $w(e(x,y)) + E(x)$. Poté se najde prefix tohoto seznamu, který dává nejlepší očekávanou cenu. [3]

Strategie založené na mravenčích algoritmech

V publikacích [11,12] bylo pro řešení RCTP a CTP využito metaheuristiky známé jako *Ant Colony Optimization*, dále značené jako ACO. Heuristiky založené na ACO řeší problémy podobně, jak své problémy řeší mravenci v přírodě (např. transport potravy do mraveniště). ACO je členem mravenčích algoritmů, reprezentuje metaheuristické optimalizační metody a může být použit např. pro řešení těžkých kombinatorických optimalizačních problémů. Mravenci v koloniích mají heuristické informace o problému a vytváří řešení iterativně přidáváním komponentů do částečných řešení. Zkušenosti jsou reprezentovány tzv. feromonovou stopou, která se mění dynamicky během jejich cest. Kratší cesty znamenají silnější stopu, kvůli které více dalších mravenců tíhne jít stejnou cestou. Několik různých strategií mravenčích systému bylo studováno pro řešení RCTP a CTP. Nejlepších z nich se jevily *Rank-based Ant System* [13] a *Max-Min Ant System* [14]. Výhodou použití ACO je robustnost řešení, stejná strategie může řešit různé typy CTP.

2.2.3 Stochastický CTP se závislými pravděpodobnostmi (SCTP-dep)

Jedná se o variantu SCTP, kde na sobě pravděpodobnosti blokací mohou být závislé. Tedy místo pravděpodobnostního ohodnocení p je dáno pravděpodobností rozdělení popsané Bayesovskou sítí $B = (E,A,P)$, kde A je množina náhodně orientovaných hran mezi náhodnými proměnnými z E a P jsou podmíněné pravděpodobnosti. Tato varianta je spíše zajímavá z teoretického hlediska. Více informací je možno najít v publikaci [22] s názvem *Complexity of Canadian traveler problem variants*.

2.3 Speciální typy CTP

V této podkapitole jsou stručně popsány speciální typy CTP, které není možno vyjádřit formálním rámcem, který je uveden v úvodu kapitoly. Společně s popisem je krátce shrnuto, jakého pokroku se v daných oblastech dosáhlo.

2.3.1 CTP s opakováním (CTP-rep)

Jde o modifikaci CTP, kde figuruje *n* agentů, kteří procházejí grafem postupně (tj. druhý vyjíždí až poté, co první dojde do cílového vrcholu). Jakmile jeden agent dokončí cestu, může sdělit informace o stavech stochastických hran ostatním, poté se stává neaktivním a už nemůže dělat žádné akce nebo pozorování. Úkolem je navrhnout strategii, která by minimalizovala souhrnnou cenu cest. Pro CTP s opakováním byla uvedena v článku *Repeated-Task Canadian Traveler Problem* [18] generalizace UCT, která se stala jádrem strategie následovatelů (*followers policy*), pro kterou byla optimalita na speciálních grafech (DPG) dokázána. Strategie také vykazovala dobré výsledky pro obecnější grafy.

2.3.2 CTP s komunikací (CTP-com)

Jde o modifikaci CTP, kde zároveň grafem cestuje více agentů *n*, kteří si mezi sebou sdílejí informace o stavu stochastických hran. Komunikace mezi agenty probíhá v reálném čase, pro představu například pomocí rádiové stanice nebo telefonicky. Podle charakteru může komunikace být buď úplná, tj. všichni agenti mohou vysílat i přijímat informace, nebo částečná, kde pouze někteří agenti mohou informace vysílat. Cílem je navrhnout strategii, která by minimalizovala souhrnnou cenu cest všech agentů z výchozího do cílového uzlu.

Multi-agentový problém s komunikací aplikovaný na k-CTP byl studován v [16], kde možnost úplné (problém P1) i částečné komunikace (problém P2) byla teoreticky i prakticky prozkoumána, a byla představena rovněž strategie nazvaná *Retrace-Alternating Strategy* (RAS) pro řešení P1 i P2. Z teoretického hlediska byla spodní hranice konkurenčního poměru pro deterministické strategie určena následovně: pro P1 $(2\cdot(k/n)+1)$, a pro P2 $(2\cdot([k-1]/n_i)+1)$, kde n_i je počet agentů s úplnou komunikací. Ověřovací experiment pro RAS byl proveden na příkladu dopravy v městském prostředí. Výsledky ukázaly, že zastoupení většího počtu úplně komunikujících agentů ne vždy zlepšuje konkurenční poměr, neboť ten je ve skutečnosti vázán ke struktuře sítě (grafu).

2.3.3 CTP se vzdáleným snímáním (CTP-remote)

Jde o modifikaci CTP, kde agent má k dispozici akci snímání, která dokáže zjistit stav vzdálené stochastické hrany za určitou cenu. Cílem je minimalizovat souhrnnou cenu cesty a snímání. Modifikace má reálnou aplikaci například v situaci, kdy má řidič (agent) má k dispozici mapu města (graf) a má možnost si zavolat na informace, kde mu budou poskytnuty údaje o stavu dopravy (akce snímaní).

Problém byl představen v [17], kde byl aplikován na CTP. Optimální strategie na speciálních typech grafů byly představeny v [17,19].

3 Vybrané strategie řešící CTP

Tato kapitola je určená k detailnějšímu popisu vybraných strategií řešících CTP. Rozebírány jsou strategie, které již byly zmíněny v předchozí kapitole a to buď z důvodu, že jejich složitost by narušovala plynulost textu, nebo z důvodu plánované implementace. Dále jsou v této kapitole představeny originální strategie UCTO2 a UCTP.

3.1 Recovery Greedy & Reposition Strategy (GR)

Tato strategie je aplikovatelná na k-RCTP. Je založena na třech dílčích strategiích. Hlavní součástí je algoritmus, který pomocí podmínek řídí, jaké dílčí strategie má být použito.

Nechť $C^A(x, y \mid E_i)$ je cena cesty získána při použití adaptivní strategie *A* při cestě z uzlu *x* do uzlu *y* za použití informace blokovaných uzlů E_i a nechť $C(x, y \mid E_i)$ je cena nejkratší cesty z uzlu *x* do uzlu *y* za použití informace blokovaných uzlů E_i. Nechť *i* je počet zjištěných blokací a platí $1 \leq i \leq k$. Parametr q je dán vztahem (3.1),

$$q = \frac{C(s_i, t\ , k-i)}{C(s, t \mid E_i)} \tag{3.1}$$

kde $C(s_i, t\ , k-i)$ je cena nejhoršího případu při cestě z s_i do *t* při blokovaných $k-i$ hran. Ze vztahu je zřejmé, že proměnná q se updatuje pokaždé, když agent objeví novou zablokovanou hranu, kterou chtěl projet. Při platnosti podmínky dané rovnicí (3.2) se použije adaptivní heuristika GS (viz podkapitola 2.1.1), která vede agenta do doby, než se na zvolené cestě neobjeví blokace.

$$q\ \leq 2(k-i) \tag{3.2}$$

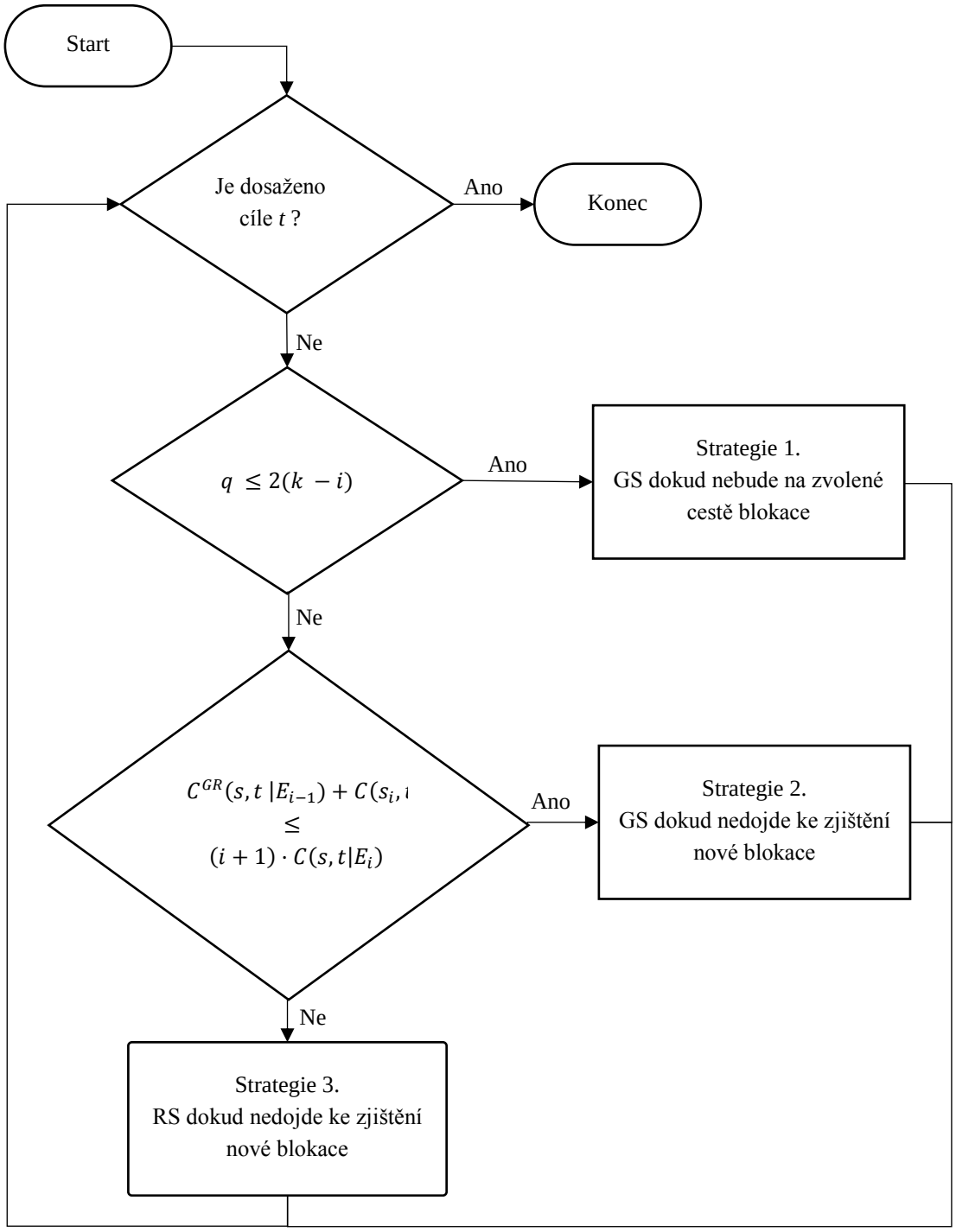
Je důležité poznamenat, že $2(k-i)+1$ se zmenšuje v průběhu cesty, což má za následek platnost vztahu (3.3), jakmile je první strategie použita. [8]

$$q\ \leq \frac{C(s, t\ , k)}{C(s\ , t)} \tag{3.3}$$

Při neplatnosti rovnice (3.2) se vyhodnotí rovnice (3.4) a jestli je splněna podmínka, tak se použije taktéž GS, ale s tím rozdílem, že jen do doby, dokud agent nespatří novou zablokovanou hranu. Při neplatnosti se použije RS taktéž do doby objevení další blokace.

$$C^{GR}(s, t\ |E_{i-1}) + C(s_i, t\ |E_i)\ \leq (i+1) \cdot C(s, t|E_i) \tag{3.4}$$

Kdykoliv když dojde k přerušení používání dílčí adaptivní strategie, tak se inkrementuje *i* a označí se nová blokovaná hrana e_i. Tok algoritmu je pro lepší představu znázorněn v diagramu na obr. 4.

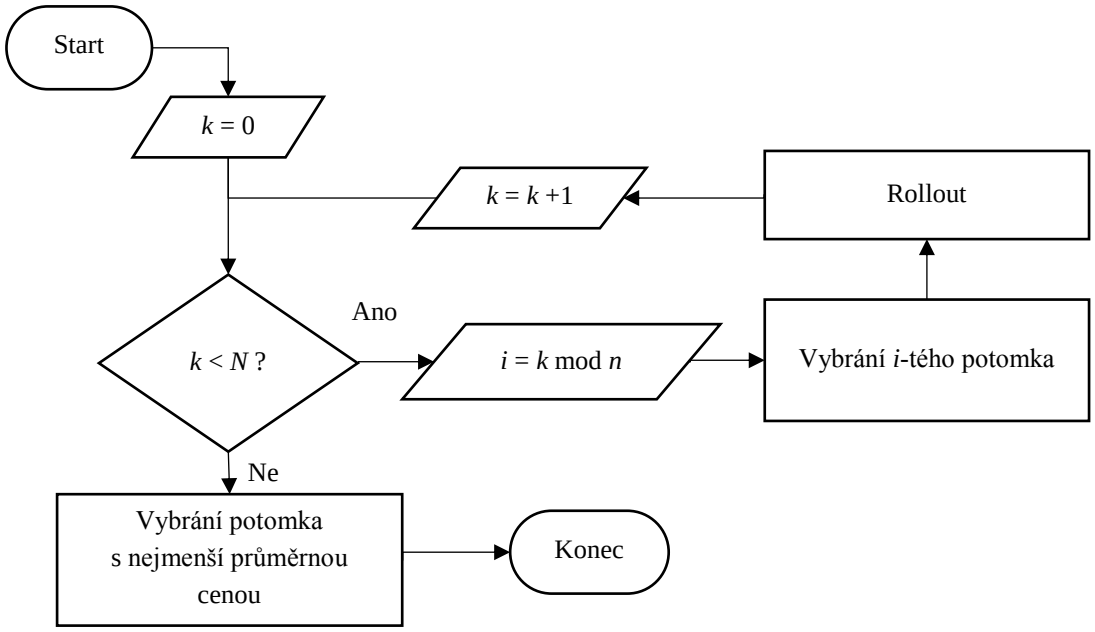


Obr. 4 Diagram znázorňující tok algoritmu GR

3.2 HOP & ORO

Toto jsou strategie řešící SCTP založené na *rolloutech* (viz podkap. 2.2.1). *Rollout* představuje jistou „simulaci“ řešení problému takovým způsobem, že se vygeneruje graf, kde jsou některé hrany dle svých pravděpodobnostních ohodnocení označeny jako blokované. Každý *rollout* tedy odpovídá jedné možné podobě grafu. V HOP i ORO je pro každého potomka (tj. sousední

vrchol) agentovy momentální pozice dána jedna simulační instance, pro kterou je vypočítán odhad k dokončení. HOP pro odhad využívá jednoduše nejkratší vzdálenost, zatímco ORO používá GS (viz podkapitola 2.1.1). Pro každého potomka je zachovávána informace o počtu provedených *rolloutů*, a také souhrnná cena k dokončení. Z těchto dat se poté spočítá průměrná cena k dokončení a agent si vybere jako další akci potomka s nejmenší průměrnou cenou. Pro lepší představu je algoritmus pro výběr akce znázorněn v diagramu na obr. 5. Parametr N je celkový počet *rolloutů*. Se zvyšujícím N jsou odhady cen stabilnější. Algoritmus se provádí tak dlouho, dokud není dosaženo cílového vrcholu t.



Obr. 5 Diagram algoritmu pro výběr akce HOP & ORO, n je celkový počet potomků momentální pozice

3.3 UCT algoritmy

Strategie, které jsou implementovány UCT algoritmy dokáží řešit SCTP. Název UCT pochází z anglického *Upper Confidence bounds applied to Trees*, což v doslovném překladu znamená: „horní hranice důvěry aplikované na stromy“. Jak překlad napovídá, UCT je spjato se stromovými strukturami, kde kořen stromu tvoří vrchol, který odpovídá momentální pozici agenta v grafu. Hlavní princip UCT spočívá ve výběru následníka z daného uzlu, který se jeví perspektivně, nebo byl v předchozích iteracích zkoumán méně často.

V této podkapitole je dále používáno označení uzel stromu a vrchol grafu k rozlišení stromu od grafu $G(V,E)$, který představuje instanci CTP problému. Uzel stromu má v sobě asociovanou informaci o vrcholu grafu, který představuje a zároveň i další informace, jako je celkový počet návštěv a suma odhadovaných cen. Je zřejmé, že na rozdíl od grafu, kde je každý vrchol unikátní, je u stromu možné mít více uzlů reprezentující stejný vrchol.

Nechť ρ je stavová sekvence pro (k+1)-ní *rollout*, který končí ve stavu s_i a začíná kořenem stromu tj. $\rho = (s,s_1,s_2...s_i)$. Nechť uzel s_i má n potomků $s'_1,\ s'_2,\ s'_3\ \dots\ s'_n$. Hlavním pilířem UCT algoritmů je způsob, jakým je uzlu s_i vybrán potomek s'_i ze všech alternativních potomků. Je vybrán kandidát s'_i, který maximalizuje UCT formuli popsanou vzorcem (3.5),

$$B\left(\frac{\log R^k(\rho)}{R^k(\rho_i)}\right)^{\frac{1}{2}} - C(\rho,\rho_i) - C^k(\rho_i) \qquad (3.5)$$

kde

- ρ_i je sekvence (ρ, s'_i) tj. ρ prodloužené o s'_i.
- $C(\rho, \rho_i)$ je cena cesty z ρ do ρ_i, tj. cena hrany z posledního stavu sekvence ρ do navrhovaného následníka s'_i.
- $R^k(\rho)$ představuje počet *rolloutů* začínajících sekvencí ρ mezi prvními *k-rollouty*.
- Podobně, $R^k(\rho_i)$ představuje počet *rolloutů* začínající sekvencí ρ_i mezi prvními *k-rollouty*.
- $C^k(\rho_i)$ je průměrná cena k dokončení $R^k(\rho_i)$ *rolloutů*.
- $B > 0$ balancuje váhy mezi využíváním a objevováním. Více o jeho vlastnostech je v textu níže.

Jestli $R^k(\rho_i) = 0$, tak je hodnota formule (3.5) považována za ∞, algoritmus proto začíná m prvních *rolloutů* tím, že vybere každého následníka ρ jednou. UCT formule je navržena tak, aby vybrala každého nástupce libovolně a dávala dostatečný počet využití ρ, ale následníci, kteří se v minulosti jevili neperspektivně, jsou vybíráni s přibývajícím časem méně a méně často. Parametr B balancuje váhy využívání a objevování následujícím způsobem: nižší hodnota podporuje využívání, zatímco vyšší objevování. Vhodné zvolení B je takové, aby lineárně rostlo s optimální cenou cesty. Ta však v CTP není známá, proto je použito odhadu, kde B je pro k-tý *rollout* vypočítán jako průměrná cena předešlých k *rolloutů*. Toto je také žádoucí, protože po aplikování škálovací konstanty na ohodnocení hran zůstává UCT algoritmus invariantní. Pro první *rollouty* není průměrná cena známá, ale to nevadí, protože B neovlivňuje výběr prvních *rolloutů* tak jako tak.

Doba trvání algoritmu může být dána buď omezením výpočetního času, anebo počtem *rolloutů*, tj. průchodů stromem. Výhodou UCT je, že se v jakémkoli okamžiku dá výpočet zastavit a vybrat momentálně nejslibnějšího potomka kořene stromu (tj. uzel s minimální průměrnou očekávanou cenou). Vrchol, který vybraný uzel představuje, je pak zvolen jako agentova další akce.

3.3.1 UCT s rozvojem do hloubky (UCTB, UCTO)

V této podkapitole budou popsán UCT algoritmus, z kterého vychází UCTB a UCTO. Podobně jako u HOP a ORO pracuje UCT s N *rollouty*, které jsou předány algoritmu jako parametr. Hlavním rozdílem je způsob, jakým jsou *rollouty* provedeny, ty jsou v UCT na sobě závislé. Algoritmus může být rozdělen na čtyři základní části:

1. rollout,
2. vytvoření sekvence stavů (pomocí UCT formule),
3. výpočet očekávané ceny
4. zpětná propagace výsledků.

Ad 1. Rollout
Podobně jako v HOP a ORO se vygeneruje instance grafu, kde jsou některé hrany označeny jako blokované dle příslušné pravděpodobnosti. Je důležité poznamenat, že již vypozorované hrany, zůstávají ve stejném stavu (tj. jednou objevená blokované hrana bude v každém *rolloutu* pořád zablokovaná, a stejně to platí i pro objevené průchozí hrany).

Ad 2. Vytvoření sekvence stavů
Pro danou vygenerovanou instanci se přidává jeden stav za druhým do sekvence, která začíná momentální pozicí agenta. Přidává se tak dlouho, dokud se nenajde cílový stav, který se přidá na konec vytvářené sekvence. Každý prvek sekvence představuje uzel prohledávaného stromu, proto se dá považovat tento UCT algoritmus za hledání do hloubky. Výběr stavu je proveden pomocí UCT formule. Pro každý vybraný uzel je provedena expanze, která přidá do stromu všechny jeho potomky. Při přidávání potomků do sekvence je třeba zabránit zacyklení, která mohou vznikat. To může být zrealizováno buď podmíněním výběru stavu, který je již v sekvenci, nebo omezením hloubky stromu, do které se může expandovat. Velikost hloubky by měla být úměrná velikosti grafu.

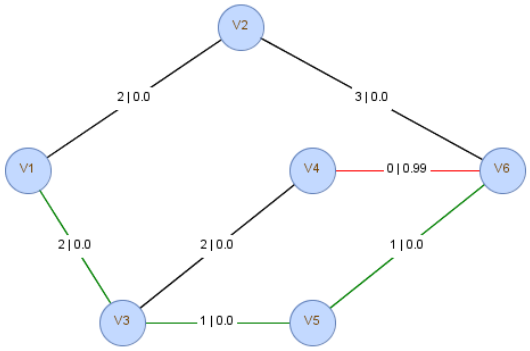
V implementaci této práce je vybrána první varianta, ve které je zabráněno přidávání stavů, které již v sekvenci jsou. Dojde-li přesto k situaci, že už nemůže být dosaženo toho, aby cílový stav byl posledním v sekvenci (to může být zapříčiněno např. „slepou uličkou“), tak končí sekvence stavem, který byl přidán jako poslední.

Ad 3. Výpočet očekávané ceny
V tomto kroku se jednoduše sečtou délky hran mezi stavy vybrané sekvence. Nekončí-li sekvence v cílovém stavu, zbytek cesty z koncového stavu sekvence do cílového stavu je odhadnut pomocí strategie GS aplikované na danou vygenerovanou instanci grafu.

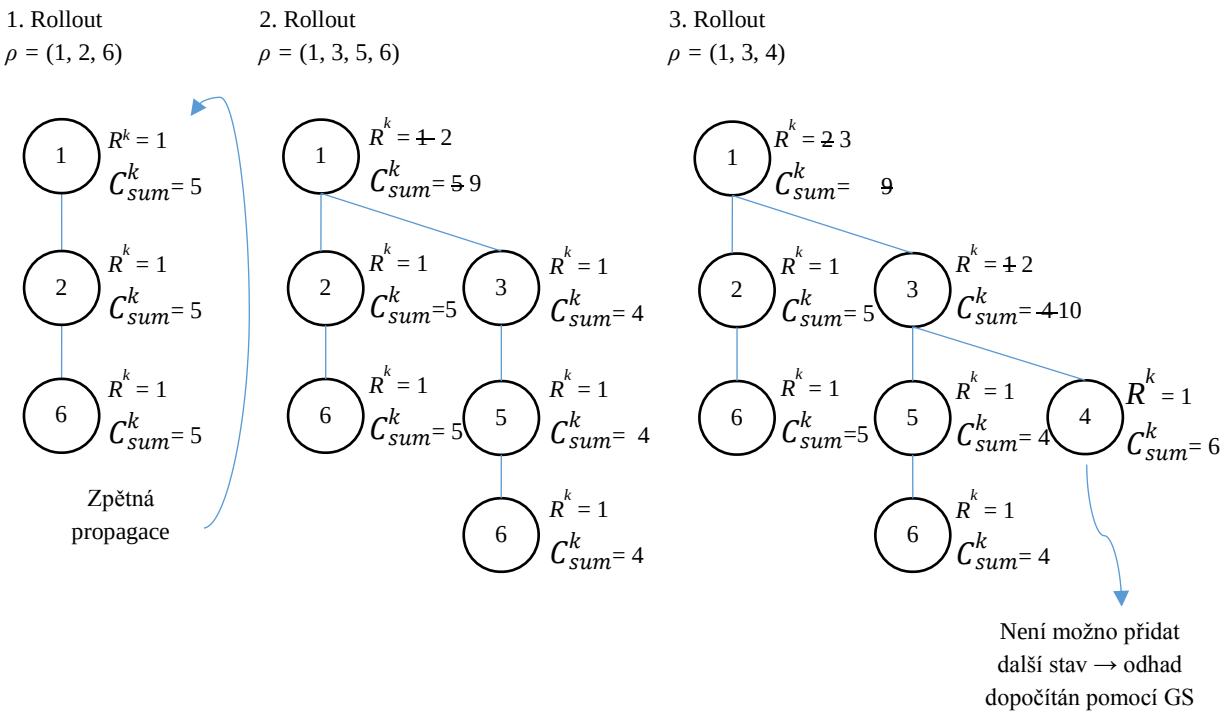
Ad 4. Zpětná propagace
Zpětná propagace slouží k „uložení“ informací získaných během *rolloutu*, tj. očekávaná cena se asociuje s posledním stavem sekvence a postupně se stejná hodnota připíše i k rodičovským uzlům stromu až ke kořenu. Společně s tím se u každého uzlu, kterým sekvence procházela, inkrementuje celkový počet návštěv. Obě tyto hodnoty se pak totiž dále využívají ve vyhodnocování UCT formule v kroku dva.

Pro lepší představu je na obr. 7 uveden triviální příklad, kde jsou vyhodnoceny graficky první tři *rollouty* společně s obrázkem stromu. V každém uzlu je asociována informace o celkové ceně a o počtu provedených návštěv daného uzlu. Pro jednoduchost nejsou ve stromu

zobrazeny doposud nenavštívené uzly. Graf pro daný příklad je uveden na obr. 6 a pro jednoduchost má pouze jednu stochastickou hranu s téměř jistou pravděpodobností blokace.



Obr. 6 Instance grafu pro ukázkový příklad. Červeně je zvýrazněná blokovaná hrana, zeleně nejkratší cesta.



Obr. 7 Grafické znázornění prvních tří *rolloutů*, které vychází z grafu na obr.6, kde *s* = V1 a *t* = V6.

Je důležité poznamenat, že ceny asociované s uzly UCT stromu jsou sumací všech odhadů a tudíž je třeba dosazovat do UCT formule (3.5) za $C^k(\rho_i)$ hodnotu $C^k_{sum}(s_i)/R^k(s_i)$.

Blind UCT (UCTB)

UCT algoritmus tak, jak byl zatím popsán, nebere v úvahu jakékoliv specifické informace týkající se CTP problému, a proto se tato varianta nazývá slepá (*blind*) UCTB. Kvůli této vlastnosti je často vyžadováno neúnosně mnoho *rolloutů*, aby UCTB konvergovala ke kvalitní strategii. [2]

Optimistic UCT (UCTO)

Tato strategie úzce navazuje na UCTB ve snaze dát algoritmu UCT s formulí (3.5) směr k řešení a tím zajistit rychlejší konvergenci ke kvalitní strategii. Nechť $C^{GS}(s'_i)$ je cena cesty z uzlu s'_i do cílového určená strategií GS. Jsou provedeny následující tři úpravy.

- Během *rolloutu* jsou shody pro nenavštívené následníky vyřešeny tím, že se použije následník s'_i s menší očekávanou cenou cesty do cíle $C^{GS}(s'_i)$.
- Při vyhodnocování UCT formule jsou $R^k(\sigma)$ a $C^k(\sigma)$ definovány, jako kdyby měly M *rolloutů* navíc pro každého následníka ρ_i, každý s očekávanou cenou $C^{GS}(s'_i)$. M je předán algoritmu jako další parametr.
- Parametr B v UCT formuli je dále podle Eyericha násoben konstantou 0,1[6], aby bylo více podporováno využívání.

Tyto modifikace směřují první *rollouty* směrem k slibným částem stavového prostoru[7]. Dodatečný počet M *rolloutů* je určen empiricky. [2]

3.3.2 UCT s rozvojem do šířky

Toto je originální modifikace UCT algoritmů. Hlavní rozdíl této varianty UCT oproti variantě předchozí spočívá v tom, že se nesnaží přidávat stavy do sekvence, která by končila v cílovém stavu, a namísto toho postupně odhaduje ceny všem potomkům, podobně jako HOP a ORO. Teprve jakmile je uzel stromu už jednou prozkoumán, tak při další návštěvě (po vybrání UCT formulí), je tento uzel expandován a jeho potomci jsou zkoumáni. Ze začátku tedy prozkoumává stavový prostor do šířky, ale čím dál více prozkoumává slibné nástupce i do hloubky. Počet iterací algoritmu je dán parametrem N, podobně jako u UCT s prohledáváním do hloubky. Je na místě poznamenat, že tam zároveň tento parametr udává počet *rolloutů*, ale ve skutečnosti se také jedná o počet iterací algoritmu (tj. počet vytvořených sekvencí stavů). Algoritmus může být rozdělen na čtyři základní části:

1. průchod stromu (pomocí UCT formule),
2. expanze uzlu a výběr uzlu,
3. odhad ceny pro vybraný uzel,
4. zpětná propagace výsledků.

Ad 1. Průchod stromu

Průchod stromem je zprostředkován UCT formulí, která je dána vzorcem (3.6). Význam jednotlivých parametrů je stejný jako v UCT formuli (3.5), uvedené na začátku podkapitoly 3.3. Pokud ještě nebyli někteří následníci ještě navštíveni, tak je preferován kterýkoliv z nich.

[6] V praktických experimentech s vlastní implementací UCTO bylo zjištěno, že prakticky nedocházelo k objevování, proto bylo namísto konstanty 0,1 použito konstanty s hodnotou 5.

[7] Podobné modifikace UCT algoritmu byly použity s velkým úspěchem ve hře GO.

Průchod stromem končí výběrem nenavštíveného uzlu, nebo výběrem cílového uzlu (tj. uzlu reprezentujícího cílový vrchol *t*).

$$B\left(\frac{\log R^k(\rho)}{R^k(\rho_i)}\right)^{\frac{1}{2}} - C^k(\rho_i) \tag{3.6}$$

Ad 2. Expanze uzlu a výběr uzlu

Uzel vybraný v předchozím bodě je expandován, tj. jsou do stromu přidání všichni jeho potomci. V rámci této práce nejsou znovu přidávány rodičovské uzly, pokud nedošlo k objevení nové blokované hrany, a tím se zabraňuje prozkoumávání neperspektivních částí stavového prostoru. Při objevení nové blokace je nutno znovu zvážit všechny možnosti, a proto je rodič přidán.

Pokud byl rodič v přechozím běhu algoritmu (před minulou akcí agenta) navštíven alespoň desetkrát (tato hranice byla volena empiricky), je rodič přidán do stromu až po skončení algoritmu. Je mu přiřazena očekávaná průměrná hodnota, která odpovídala druhé nejlepší možnosti v přechozí volbě akce. Hodnota je držena v paměti spolu s informací o vrcholu, kterému odpovídá. Toto opatření pomáhá eliminovat repetitivní výpočty.

Ad 3. Odhad ceny pro vybraný uzel

Odhad ceny z vybraného uzlu může být proveden více způsoby, např. pomocí GS, HOP, ORO, nebo nějakým jiným způsobem. Je důležité poznamenat, že pro každý vybraný uzel stromu je očekáváná cena odhadnuta právě jednou, proto musí mít odhad, pokud možno, co nejlepší vypovídající hodnotu.

V rámci této práce byla zvolena metoda odhadu následovně. Pro každý vybraný uzel je provedena série *rolloutů*, kde jejich počet N_r je dán algoritmu jako parametr. Nechť je v bodu jedna vybrán uzel reprezentující stav s_i a nechť rodič tohoto uzlu je s_g. Nechť N_t označuje množství *rolloutů*, ve kterých hrana mezi rodičem s_g a potomkem s_i byla průjezdná a nechť N_b označuje množství *rolloutů*, ve kterých byla hrana blokována. Odhad, neboli průměrná cena $C^r(s_i)$, je pak dána rovnicí (3.7), která průměruje výsledky zaznamenané v jednotlivých *rolloutech*.

$$C^r(s_i) = \frac{\sum_1^{N_t}[C(s_g, s_i) + C(s_i, t)] + \sum_1^{N_b} C^{GS}(s_g, t)}{N_t + N_b} \tag{3.7}$$

kde

- $C(s_g, s_i)$ je cena hrany z rodičovského uzlu do vybraného
- $C(s_i, t)$ je cena nejkratší cesty z s_i do t určená Dijkstrovým algoritmem
- $C^{GS}(s_g, t)$ je cena cesty z rodičovského uzlu do cílového za použití GS

Pokud v jakémkoli *rolloutu* nastane situace, že nelze cestu do cíle dokončit je tento *rollout* jednoduše přeskočen a proto platí vztah (3.8):

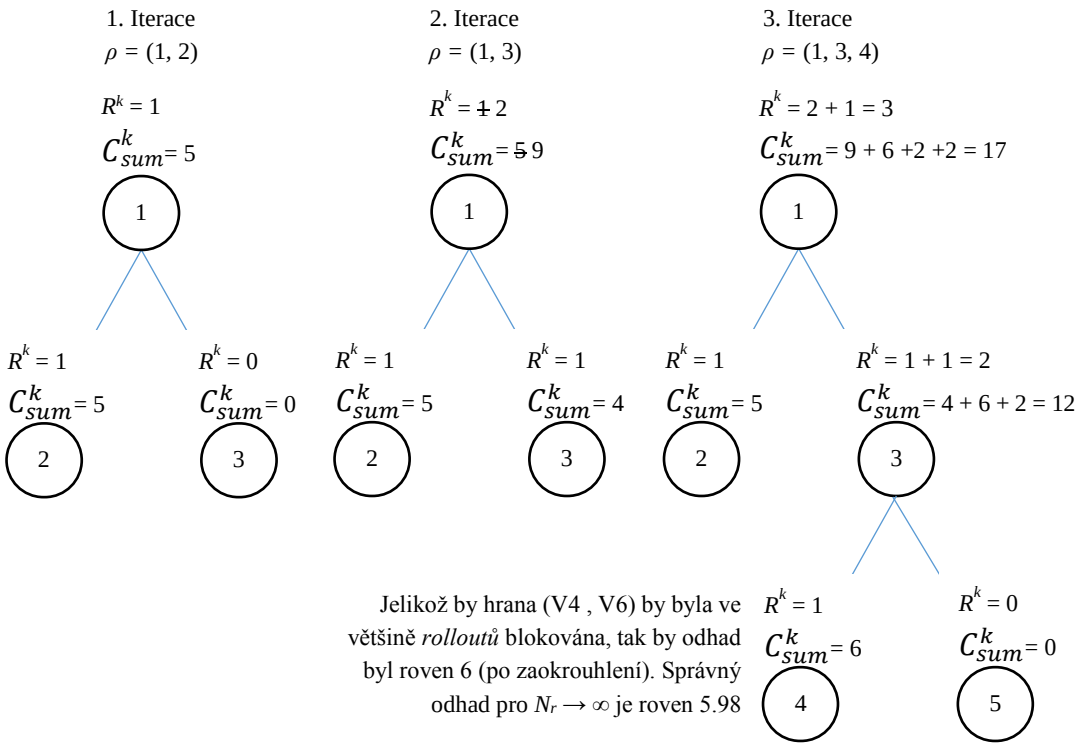
$$N_t + N_b \leq N_r \tag{3.8}$$

Odhad *rolloutu* při průjezdné hraně využívá nejkratší cesty a přiřazuje optimistickou cenu vybranému uzlu, což zapříčiní možné následné vybrání uzlu do budoucnosti a prozkoumání v dalších iteracích algoritmu. Při blokované hraně je naopak odhad proveden GS, a tím dojde ke zhoršení průměrné ceny oproti odhadu při průjezdné hraně. Je důležité poznamenat, že odhady vypovídají o tom, jaká je průměrná cena k dokončení při volbě daného potomka s_i z daného rodiče s_g.

Ad 4. Zpětná propagace výsledků

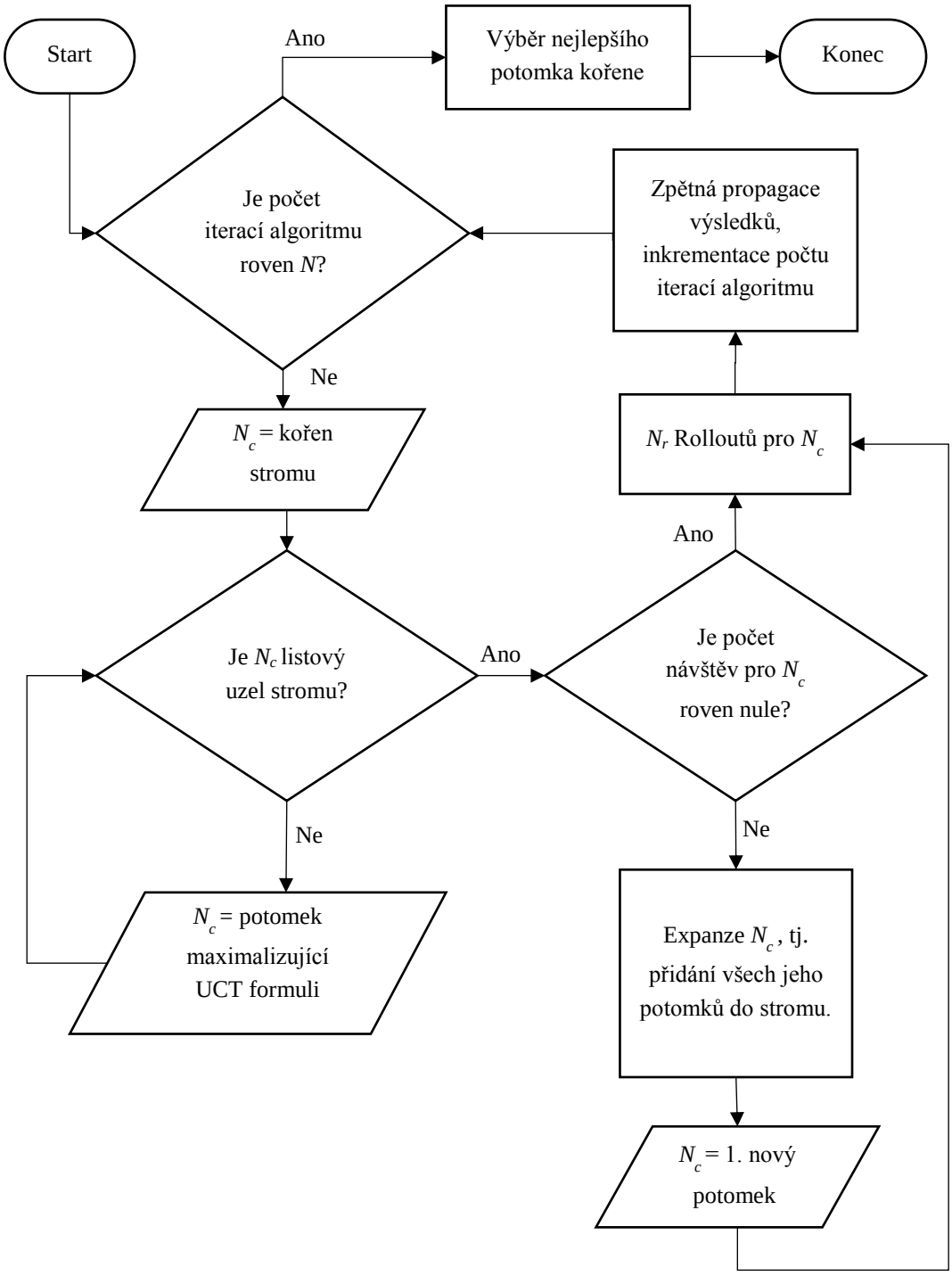
Tento krok slouží k „uložení“ informací získaných během iterace algoritmu, tj. očekávaná cena $C^r(s_i)$ se asociuje s vybraným uzlem s_i. Rodiči s_g vybraného uzlu se k jeho stávajícímu odhadu $C^r(s_g)$ přičte stejná hodnota $C^r(s_i)$ navýšena o hodnotu hrany, která spojuje rodiče s prarodičem. Postupně se toto opakuje až ke kořenu UCT stromu (kořenovému uzlu a jeho potomkům se připisuje stejná částka, protože kořenový uzel již nemá rodiče, jehož hodnota hrany by se měla přičíst). Společně s tím se u každého uzlu, u kterého se aktualizuje odhad, inkrementuje celkový počet návštěv. Obě tyto hodnoty se pak totiž dále využívají ve vyhodnocování UCT formule v kroku jedna.

Pro lepší představu je na obr. 8 uveden triviální příklad, kde jsou vyhodnoceny graficky první tři iterace algoritmu společně s obrázkem stromu (záměrně jsou vynecháni potomci navštívených uzlů, jsou-li všichni doposud nenavštívení). V každém uzlu je asociována informace o celkové ceně a o počtu provedených návštěv daného uzlu. Graf pro daný příklad je uveden na obr. 6 a pro jednoduchost má pouze jednu stochastickou hranu s téměř jistou pravděpodobností blokace.



Obr. 8 Grafické znázornění prvních tří iterací, které vychází z grafu na obr. 6, kde s = V1 a t = V6.

Pro úplnost je diagram UCT algoritmu s rozvojem do šířky znázorněn na obr. 9, kde N_c je vybraný uzel (*current node*).



Obr. 9 Diagram popisující výběr akce UCT algoritmem s rozvojem do šířky.

Optimistic UCT 2 (UCTO2)

Toto je originální strategie, která úzce navazuje na UCTO, tedy má všechny modifikace pro UCT formuli, které má UCTO. Jediným rozdílem je to, že je implementován pro UCT algoritmus s rozvojem do šířky. Do algoritmu tedy navíc vstupuje parametr udávající počet *rolloutů* dělaných pro odhad N_r a namísto počtu *rolloutů* N se v UCTO2 mluví o celkovém počtu iterací N.

Pruning UCT (UCTP)

Toto je originální strategie navazující na UCTO2. Hlavní myšlenkou je manipulace s UCT stromem, kde vhodné prořezávání málo perspektivních větví umožňuje UCT algoritmu rychlejší konvergenci ke kvalitní strategii. Prořezávání je běžná technika u algoritmů pracující se stromovými strukturami, např. α–β prořezávání u AO stromů. Název UCTP vznikl z anglického slova pro prořezávání (*Pruning*).

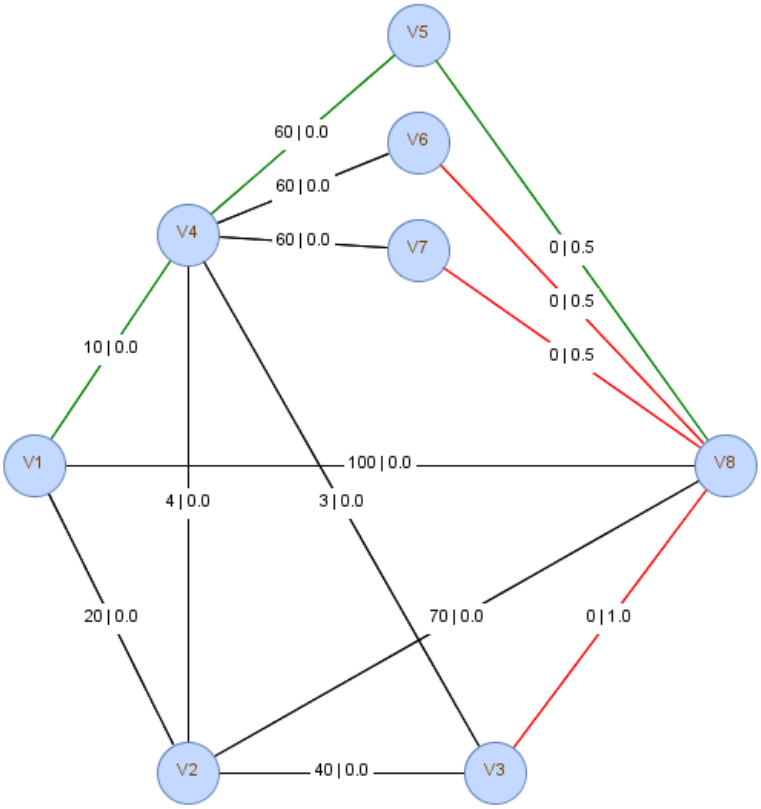
Je udržována množina M všech již navštívených vrcholů a ke každému vrcholu v množině je asociována informace o rodiči vrcholu spolu s odhadovanou cenou $C^r(s_i)$, viz rovnice (3.7). Po každé první návštěvě vrcholu je do M přidán navštívený vrchol, a pokud přidávaný vrchol v M již je, musí se rozhodnout, který z nich se zachová. Dále je vysvětleno, jak funguje prořezávání.

Nechť s_i, s_j jsou dva uzly UCT stromu. Oba mají stejného potomka s_k. Nechť s_k s rodičem s_i je už jednou navštíven. Při expanzi a následnému *rolloutu* vrcholu s_k s rodičem s_j musí být rozhodnuto, který rodič uzlu s_k je perspektivnější. Potomek s_k horšího rodiče je odřezán. Rozhodovacím kritériem je průměr odhadů z prvních *rolloutů* začínajíce od s_k (tj. $C^r(s_k)$) a končíce v jednom z přímých potomků kořenového uzlu. Je důležité poznamenat, že odhad $C^r(s_k)$ vrcholu je různý pro každého z rodičů s_i, s_j, viz rovnice (3.7).

Pokud odřezaný uzel s_k má již potomky, na kterých byly provedeny výpočty, jsou tito potomci předáni novému rodičovskému uzlu a informace získané z předešlých *rolloutů* jsou zpětnou propagací postupně vneseny do všech rodičovských uzlů až ke kořenu. Je důležité poznamenat, že kvůli výše popsaného prořezávání nejsou v UCT stromu nikdy dva stejné, již navštívené uzly.

3.4 Příklad demonstrující nežádoucí vlastnosti vybraných strategií

V následující podkapitole je uveden příklad vysvětlující chování GS, HOP, ORO a UCT algoritmů. Cílem je demonstrovat „pasti“, ve kterých se jednotlivé strategie chovají suboptimálně.



Obr. 10 Instance grafu představující CTP problém. Popisky hrany představují ohodnocení: $w(e)|p(e)$. Červeně jsou vybarveny zablokované hrany, zeleně nejkratší cesta.

Pro příklad je uvažován graf na obr.10, kde by kvalitní strategie měla navrhnout cestu V1→V4→V2→V8 za cenu 84. Nejkratší cesta má cenu 70, ale prochází hranou s 50% šancí na blokaci. Ve vrcholu V4 agent nemůže vědět, která z cest bude průchozí a v případné vrácení z některého z uzlů V5, V6, V7 skončí agent s celkovou cenou přibližně 190, nebo dokonce vyšší.

S využitím strategie GS je agent veden teoreticky nejkratší cestou V1→V4→V3→V8, i přes skutečnost, že hrana (V3,V8) má vysokou pravděpodobnost, že bude blokována. Proto GS skončí následováním cesty V1→V4→V3→V4→V(5,6,7)…→V8 za celkovou cenu přinejlepším 76 a přinejhorším 450. Při zvyšování počtu pokusů by průměrná cena konvergovala přibližně ke 181.

Průměrná cena je ve strategii ORO vypočítána GS, která přiřadí vrcholům V2,V4 a V8 očekávané ceny přibližně 195, 181 a 100 postupně. Agent tedy následuje tedy cestu V1→V8 za cenu 100 (pozn.: hodnoty byly vypočteny pro N = 10 000).

Strategie HOP ve vrcholu V1 přiřadí nejmenší průměrnou cenu cesty vrcholu V4, protože všechny možné nejkratší cesty vždy vedou přes V4. Ve vrcholu V4 by vrcholům V5,V6,V7 přiřadila cenu přibližně 120 a vrcholu V2 cenu blízkou 68,75 (většinou by totiž nejkratší cesta vedla zpátky do V4 a pak jedním z vrcholů V5, V6, V7 za cenu 68, ale v případě blokací všech tří vrcholů existuje slibná varianta jít do V8 za 70). Ještě menší hodnotu by však HOP přiřadila vrcholu V3. Z podobného principu jako pro V4 by vrcholu V3 přiřadila cenu 67,75. Ve V3 by

si však strategie „uvědomila“ suboptimalitu svého rozhodnutí a vrátila by se zpět[8]. Celkově by tedy agent pomocí HOP byl následován cestou V1→V4→V3→V4→V2→V8 za cenu 90.

UCTB a UCTO by i pro vyšší počet *rolloutů* N = 10 000 vedly agenta skrze V1→V2→V8 za celkovou cenu 90, zatímco UCTP už i při relativně nízkých nastaveních vstupních parametrů (N_i = 20, N_r = 20) je schopna zjistit, že do uzlu V2 je nejkratší cesta skrze V1→V4→V2, potomek V2 s rodičem V1 je z UCT stromu úplně odstraněn (zachován je V2 z rodičem V4). UCTP tedy vede agenta skrze V1→V4→V2→V8 za cenu 84, což je navrhovaná cena kvalitní strategie počítající s pravděpodobností výskytu blokací.

[8] Toto chování je třeba v HOP ještě ošetřit, protože by se strategie zacyklila ve smyčce V1→ V4→ V1→ V4… K překonání toho problému bylo v implementaci využito cachovacího mechanismu, který přiřazuje předchozímu navštívenému vrcholu druhou nejlepší průměrnou cenu (první byla pro vybraný vrchol, tj. pro momentální polohu agenta). Podobný mechanismus je nutné použít u všech algoritmů pracující s odhadem cen na základě nejkratší cesty (UCTO2 a UCTP).

4 Implementace vybraných strategií řešící CTP

V této kapitole je popsáno, jakým způsobem byly implementovány vybrané strategie řešící CTP. Byla implementována okenní aplikace, která byla vytvořena s cílem ověřit správnou funkci strategií řešících CTP a zároveň umožnit uživateli snadno testovat různé strategie, popř. jednoduše aplikací rozšířit o další strategie. Aplikace umožňuje řešit problém vícekrát a pro srovnání nabízí přehled výsledků z jednotlivých běhů strategií s jejich průměrnou cenou k dokončení, a také průměrným časem, který byl vynaložen na řešení. Ze strategií řešících klasickou variantu CTP byly implementovány strategie GS, RS a CS. Za stochastickou variantu CTP byly implementovány strategie HOP, ORO, UCTB, UCTO, UCTO2 a UCTP.

Pro implementaci bylo využito programovacího jazyku Java a objektově orientovaného přístupu. Přestože je Java interpretovaný jazyk a je tedy zákonitě pomalejší než jazyky nativní, jako je C a C++, má i určité výhody. Jedna z nich je, že návrh grafického uživatelského rozhraní je o poznání jednodušší. Další z výhod může být fakt, že Java je multiplatformní a tedy je bez problému spustitelná na nejběžnějších operačních systémech jako jsou Windows, Mac OS X, *nix a další. Jediné, co je potřeba ke spuštění, je JRE (*Java runtime environment*), který je pro všechny tyto systémy volně stažitelný a dokonce bývá i běžnou základní výbavou.

Aplikace je kvůli OOP návrhu jednoduchá na rozšíření. Celá práce je vytvořená jako Open Source projekt. Zdrojové kódy jsou dostupné na serveru GitHub [23], kde je možno si celý projekt naklonovat a vyzkoušet. Typy problému, které je možné řešit, jsou CTP a SCTP. Jediný rozdíl mezi nimi je, zdali strategie pracuje exaktně s pravděpodobností blokací hran nebo ne. Pravděpodobnosti musí být totiž aplikaci dodány tak jako tak, aby se z nich mohla vygenerovat instance řešeného problému.

Aplikace byla vyvinuta v programovacím prostředí Eclipse Kepler [24] a využívá tří externích knihoven. První z nich je jgrapht [25], dalším je jgraphx [26] a poslední je log4j [27]. Pro sestavení aplikace bylo použito frameworku Maven. Pro návrh grafického rozhraní bylo využito pluginu vývojového prostředí nesoucího název WindowBuilder [28]. Pro tvorbu UML diagramů bylo použito pluginu vývojového prostředí ObjectAid [29].

4.1 Vstupní data

Vstupem se rozumí data, která reprezentují graf. Ta jsou vyřešeny textovými soubory[9], kde je uveden seznam sousedů pro daný uzel a k tomu odpovídající seznam vah a seznam pravděpodobností blokací hran. Je použito klíčových slov k rozlišení, kde každá sekce začíná. Seznam používaných klíčových slov spolu s jejich popisem a formátem hodnot je patrný z tab. 1.

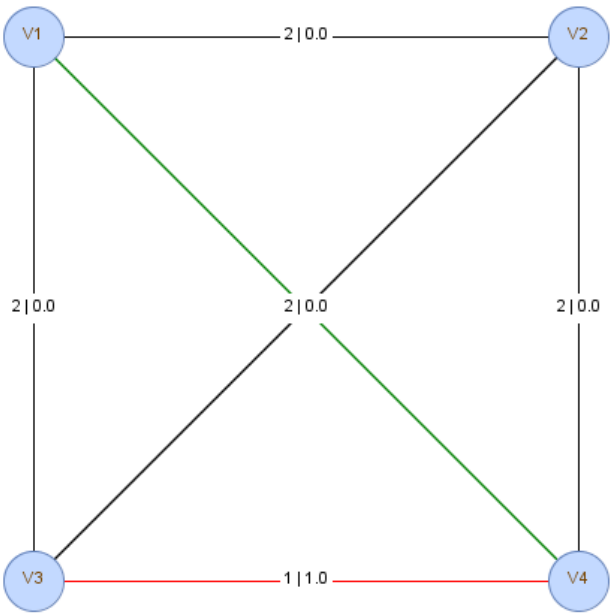
[9] Textový formát byl zvolen po inspiraci knihovnou TSP-lib [30], která obsahuje balíček typických testovacích grafů pro problém cestovního obchodního (TSP). Inspirace pro názvy klíčových slov pochází rovněž odtud.

Tab. 1 Seznam a popis klíčových slov vstupního dokumentu

Klíčové slovo	**Popis**	**Formát hodnot** **Příklad řádku**
DIMENSION :	Určuje počet vrcholů	Kladné celé číslo DIMENSION : 44
NODE_COORD_SECTION	Začátek seznamu souřadnic vrcholů [x,y]	První je pořadí vrcholu začínající od 1 do počtu vrcholů, mezerou jsou odděleny souřadnice x a y. Ty jsou zadané jako kladná reálná čísla. 4 11.1 66.6
EDGE_DIRECTIONS_SECTION	Začátek seznamu sousedů.	Kladná celá čísla oddělená mezerami. 5 6 7 8
EDGE_WEIGHT_SECTION	Začátek seznamu vah.	Kladná reálná čísla oddělená mezerami. 1.1 2.4 5.8 6
PROBABILITY_SECTION	Začátek seznamu pravděpodobností blokací hran.	Kladná reálná čísla v intervalu <0,1> oddělená mezerami. 0.5 0.5 0 1
EOF	Označuje konec souboru	

Na obr. 11 je možno vidět podobu textového souboru i s vizualizací instance grafu, která je souborem definována.

```
DIMENSION : 4
NODE_COORD_SECTION
1 500 500
2 1000 500
3 500 1000
4 1000 1000
EDGE_DIRECTIONS_SECTION
2 3 4
1 3 4
1 2 4
1 2 3
EDGE_WEIGHT_SECTION
2 2 3
2 2 3
2 2 1
2 2 1
PROBABILITY_SECTION
0 0 0
0 0 0
0 0 1
0 0 1
EOF
```



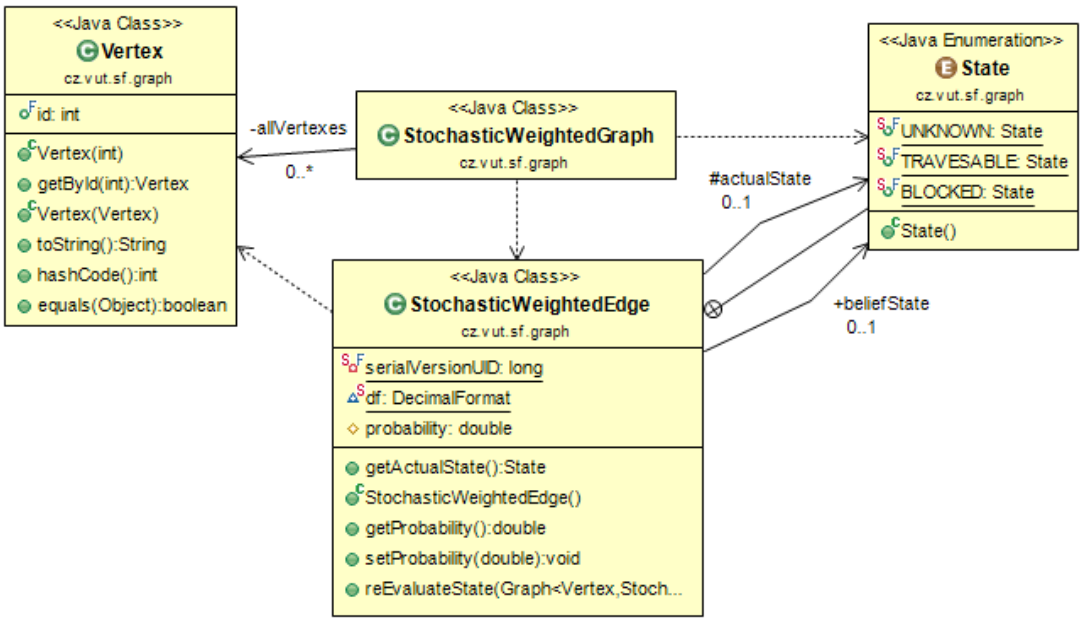
Obr. 11 Ukázka textového souboru spolu s grafem, který reprezentuje.

4.2 Popis hlavních tříd

V této podkapitole bude stručně shrnuta podoba hlavních tříd a vztahy mezi nimi. Cílem této podkapitoly je ukázat čtenářovi OOP návrh a v případě nahlédnutí do zdrojových kódů urychlit orientaci v jednotlivých třídách, kterých je přes čtyřicet a jsou rozděleny do šesti balíčků (v Java označovaných jako *package*).

4.2.1 Třída reprezentující graf

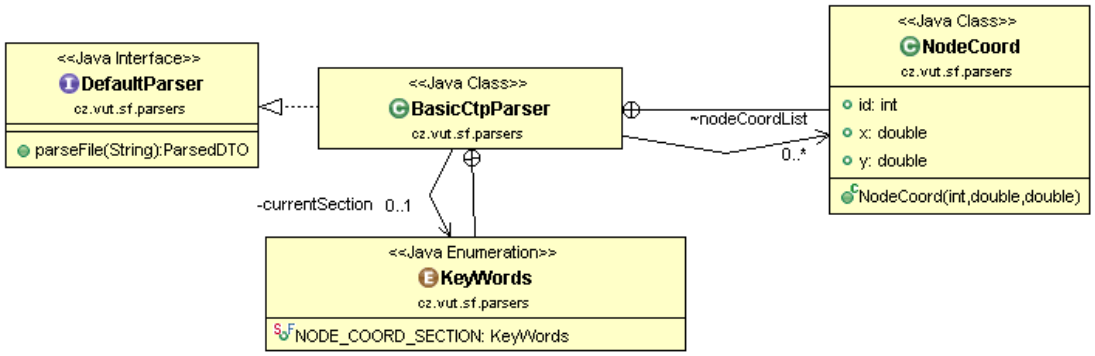
Implementace grafů je velice důležitá pro řešení CTP, proto bylo využito externí knihovny jgrapht, kde je implementace grafů vyřešena velice pěkně. Hlavní třída reprezentující stochastický graf dědí z třídy `SimpleDirectedWeightedGraph<T, T>` výše zmíněné knihovny. Dvě generiky jsou určeny pro vrcholy a hrany, jejich implementace je udělána dle potřeb CTP problému. Každá hrana grafu má dva stavy – přesvědčení (*beliefState*) a skutečný (*actualState*). Při tvorbě grafu jsou skutečné stavy vyhodnocovány dle pravděpodobnostního rozložení a stavy přesvědčení jsou nastaveny na „neznámé“ a jsou při agentově průchodu grafem aktualizovány na hodnotu skutečných stavů. UML diagram tříd reprezentující graf, jeho hrany a vrcholy je na obr. 12.



Obr. 12 UML diagram tříd reprezentujících graf, pro přehlednost nejsou vyobrazeny všechny pole a metody.

4.2.2 Třída pro analýzu vstupních dat

Třída pro analýzu dat je programovaná proti rozhraní, takže v případě jinak zadaných dat je možné si dopsat vlastní syntaktický analyzátor. Ten bude implementovat stejné rozhraní, ve kterém je hlavička pouze jedné metody, která vrací datový objekt (ten je nezbytný ke tvorbě grafu). Na obr. 13 je UML diagram znázorňující implementovaný analyzátor, schopný číst data ve formátu popsaném v podkapitole 4.1.



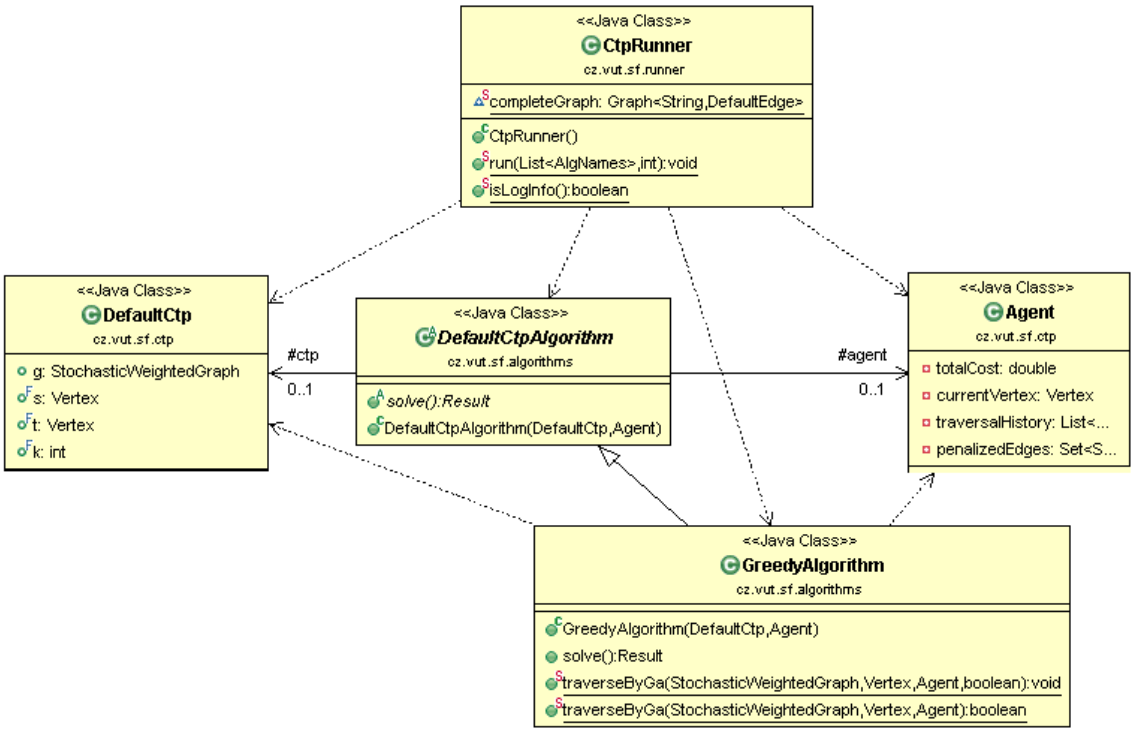
Obr. 13 UML diagram hlavních tříd spojených s analýzou vstupních dat.

4.2.3 Třída pro spouštění algoritmů

Třída starající se o spuštění jednotlivých algoritmů se jmenuje `CtpRunner` a pomocí dat získaných z globálních proměnných spouští jednotlivě vybrané algoritmy. Globální proměnné jsou drženy v asociativním poli, které je načítáno ze souboru *config.properties* a měněno skrze grafické rozhraní.

Klíčovou třídou pro algoritmy je abstraktní třída `DefaultCtpAlgorithm`, ze které dědí všechny ostatní. Tato třída obsahuje dvě pole, kterými jsou třída reprezentující CTP problém a

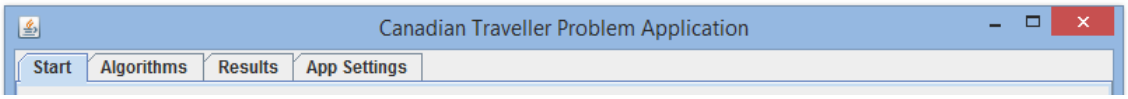
třída reprezentující agenta. UML diagram znázorňující vztahy mezi výše popsanými třídami a jedním implementovaným algoritmem je na obr. 14.



Obr. 14 UML diagram znázorňující třídu agenta, CTP problému, greedy algoritmu a jejich vztahů ke třídě, která algoritmy spouští.

4.3 Uživatelské rozhraní

Tato podkapitola je věnována popisu jednotlivých částí grafického rozhraní implementované okenní aplikace pojmenované *Canadian Traveller Problem Application*. Základem aplikace jsou čtyři panely, jejichž záložky (viz obr.15) jsou umístěny v horní části aplikace. Uživatel mezi nimi stránkuje jednoduše po kliknutí na hlavičky jednotlivých záložek.



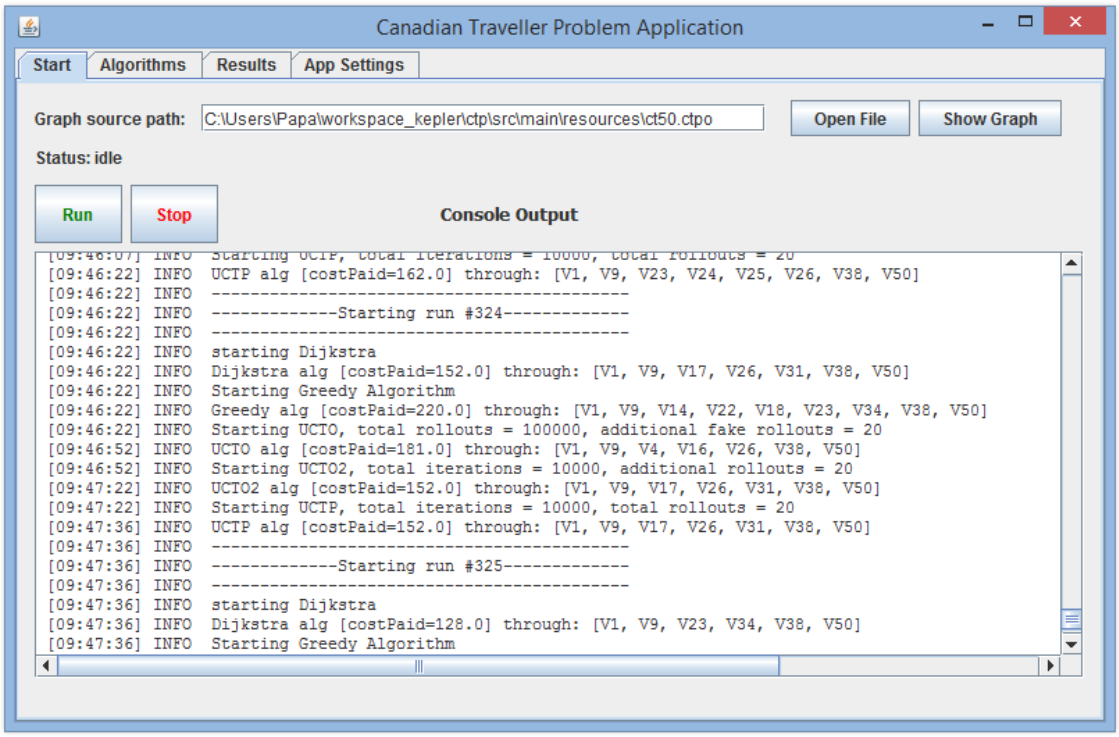
Obr. 15 Záložky hlavních panelů aplikace.

4.3.1 Startovací panel

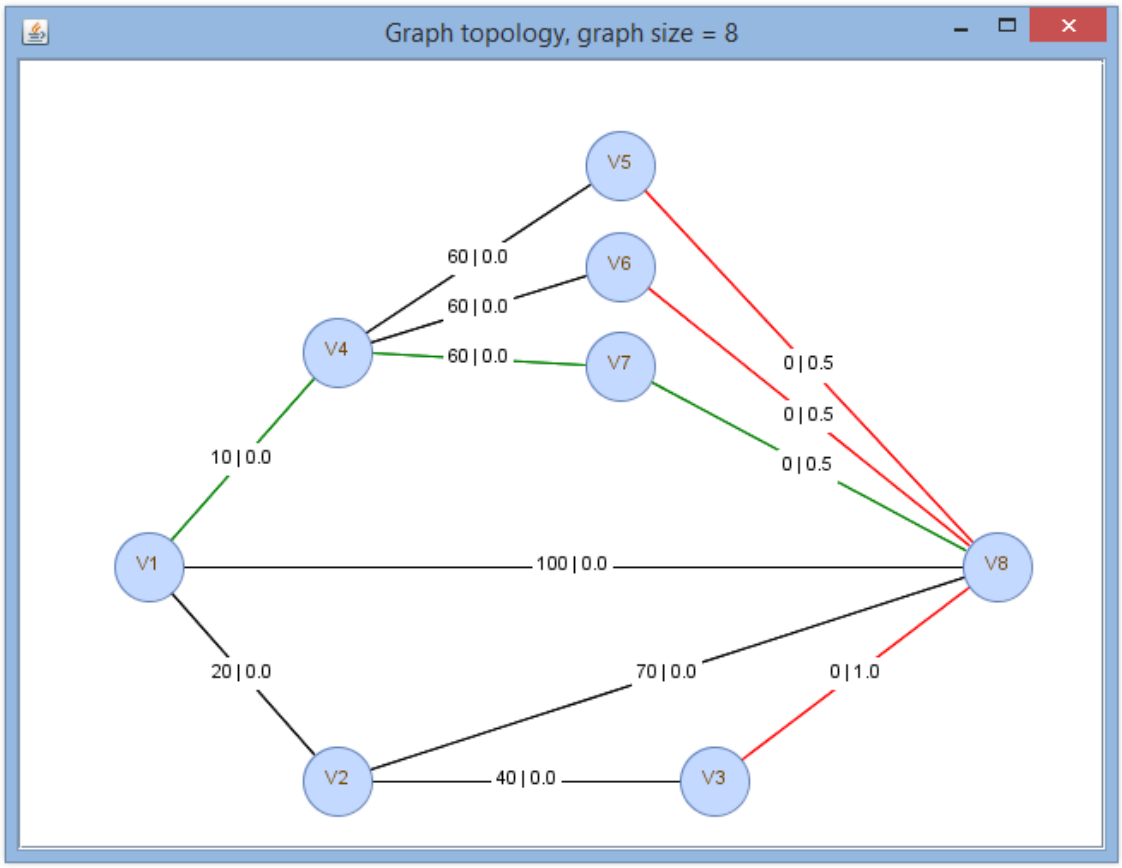
Startovací panel je možno vidět na obr. 16. Jednou z hlavních funkcí panelu je načítání vstupních dat ze souboru (viz podkap. 4.1). Pro validaci dat slouží tlačítko *Show Graph*, které pro načtený soubor zobrazí instanci grafu, tj. některé hrany jsou označeny za blokované dle příslušných pravděpodobností. Příklad grafu je na obr. 17. Pokud se při syntaktické analýze zjistí chyba (například špatný formáte dat), je chyba vypsána do konzolového výstupu společně s informací, ve které sekci došlo chybě a ve kterém řádku. Pokud není ve vstupním dokumentu

sekce popisující souřadnice vrcholů, tak jsou vrcholy grafu náhodně rozmístěny po celém prostoru otevřeného okna. V takových případech jsou pak grafy nečitelné a nepřehledné, ale pro samotné řešení CTP problému to samozřejmě nemá vliv. Větší grafy stejně není možno přehledně zobrazit. Dojde-li po stisknutí tlačítka *Show Graph* k otevření okna (viz obr. 17), tak potom je ověřeno, že načtený soubor je v pořádku a tedy výpočty na daném grafu mohou být prováděny.

Další důležitou funkcí startovacího panelu je spouštění výpočtů nastavených v panelu algoritmů, o kterém je více řečeno v následující podkapitole. Spouštění je zrealizováno stisknutím tlačítka *Start*, při kterém se založí fronta úkolů, ve které jsou jednotlivé uživatelem vybrané algoritmy řešící CTP (při nastavení více iterací výpočtů je stejná fronta po vyprázdnění znovu vytvořena). Zároveň při startu výpočtů se změní status nad tlačítkem *Start* z hodnoty *idle* na hodnotu *busy*. Pro případ potřeby zastavení výpočtů je určeno tlačítko *Stop*. Důvody zastavení mohou být například špatné nastavení, nebo špatný odhad časové náročnosti výpočtů. Okamžitě, po stisknutí tlačítka *Stop*, jsou výsledky již provedených výpočtů dostupné v panelu výsledků. Úplné zastavení však probíhá pouze po skončení výpočtu algoritmu, který byl jako poslední odebrán z fronty úkolů. V případě nutnosti okamžitého vypnutí nebo zacyklení některého z algoritmů je nutno celou aplikaci ukončit. Výstupy z aplikace je možno vidět v reálném čase v textovém poli, které je umístěno ve střední části panelu (viz obr.16).



Obr. 16 Startovací panel, hlavní části jsou tlačítka *Run*, *Stop*, *Show Graph*. Výstupy z aplikace je možno v reálném čase sledovat v konzolovém výstupu, který je ve střední části panelu.



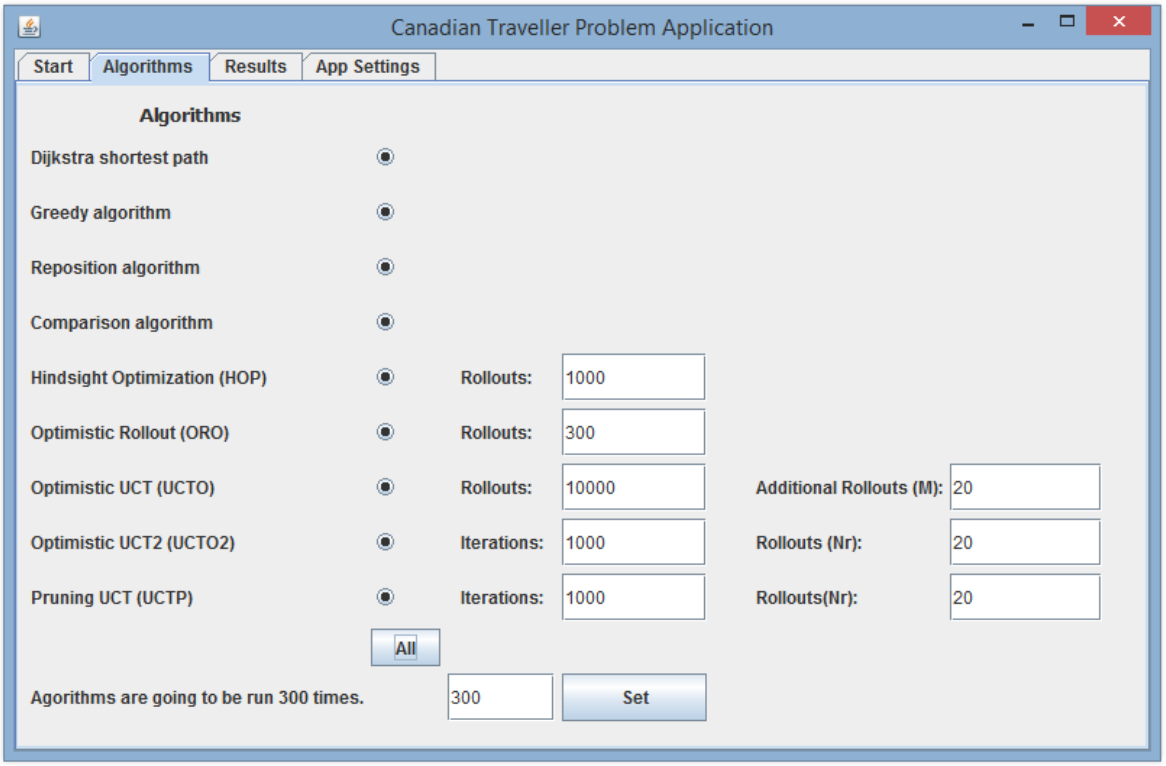
Obr. 17 Otevřené okno po stisku tlačítka *Show Graph* znázorňující instanci grafu. Popisky hrany představují ohodnocení: $w(e)|p(e)$. Červeně jsou vybarveny zablokované hrany, zeleně nejkratší cesta.

4.3.2 Panel algoritmů

Panel algoritmů je možno vidět na obr.18. Hlavní funkcí panelu je výběr a nastavení parametrů jednotlivých algoritmů. Dále tento panel umožňuje nastavit počet běhů vybraných algoritmů. Výběr je realizován tlačítkem on/off. Je možno označit všechny algoritmy (strategie), které byly zmíněny na začátku této kapitoly, kromě UCTB. To bylo uděláno z důvodu jeho „špatné“ funkce[10]. Pro menší počet *rolloutů* totiž docházelo k častým zacyklením, což vedlo na časově náročné výpočty.

Všechny parametry algoritmů se udávají jako celá nezáporná čísla a jsou po potvrzení enterem validována a uložena do paměti aplikace.

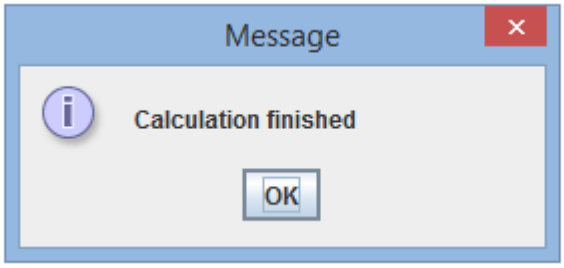
[10] Špatné výsledky UCTB algoritmu nebyly až takové překvapení, i v článku [2] si v testech vedl relativně špatně a pro menší počty *rolloutů* docházelo také k zacyklení.



Obr. 18 Panel s výběrem implementovaných algoritmů.

Po vyplnění požadovaných algoritmů je možno výpočty spustit ze startovacího panelu, kde je uživatel v konzolovém výstupu informován o průběhu výpočtů. Při začátku jednotlivých algoritmů jsou zde vypsány hodnoty vstupních parametrů pro daný algoritmus. Po skončení jednotlivých algoritmů je možno vidět, kudy byl agent veden a také kolik za cestu zaplatil.

Po skončení všech výpočtů je uživatel uvědomen pomocí vyskakovacího okna. Podobu tohoto okna je možno vidět na obr. 19.



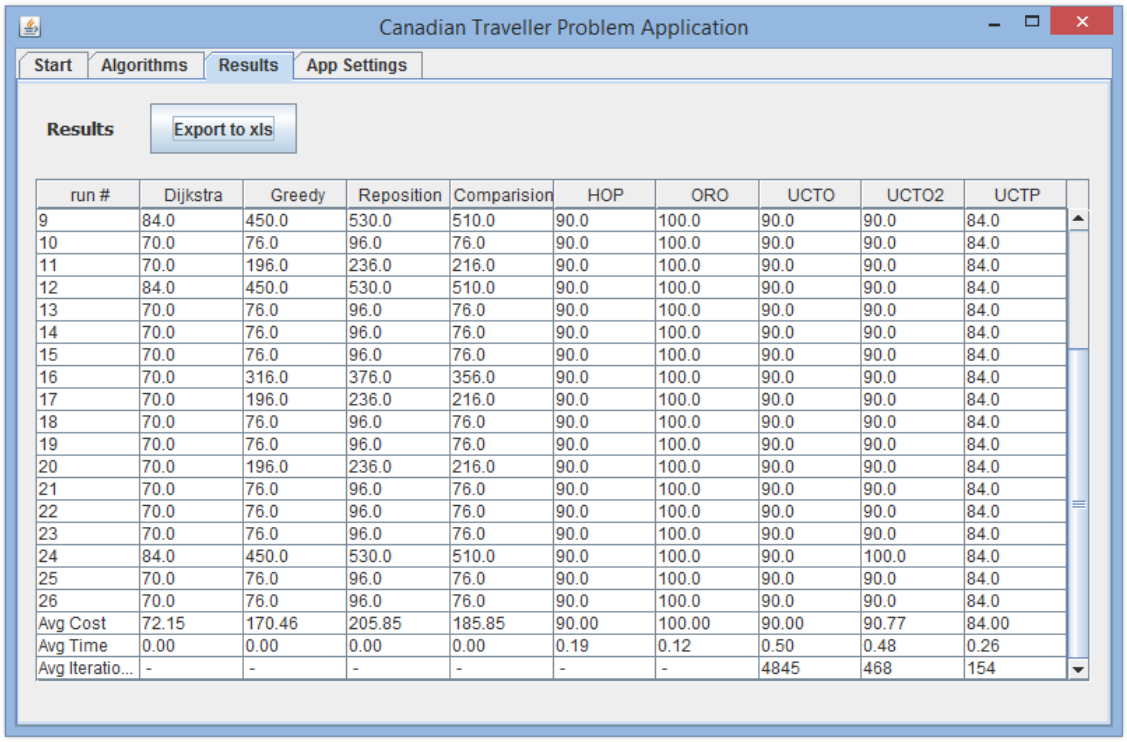
Obr. 19 Zpráva oznamující konec výpočtů.

4.3.3 Panel výsledků

Tento panel je možno vidět na obr. 20. Hlavní funkcí je zobrazení výsledků z jednotlivých výpočtů. Ty jsou pro přehlednost dány v tabulce. Výsledky je možno vidět po skončení výpočtů (tj. po oznámení o ukončení viz obr. 19), nebo po stisknutí tlačítka *Stop*, které je součástí startovacího panelu.

Ve sloupcích jsou vybrané algoritmy a v řádcích ceny pro daný běh. Poslední tři řádky tabulky jsou průměrné hodnoty cen, výpočetního času a počtu iterací UCT algoritmů v rozhodovacím kroku. Poslední řádek je zobrazen, pouze jsou-li UCT algoritmy nastaveny na výpočet s omezeným časem na rozhodnutí o další agentově akci. Toto nastavení je možno provést ve čtvrtém panelu, o němž je řečeno více v následující podkapitole.

Celou tabulku je možno pro další zpracování dat exportovat do xls souboru. Exportování je zrealizováno stisknutím tlačítka, po kterém se otevře navigátor, který je možno nasměřovat dle potřeby do jakéhokoliv adresáře. Informace o nastavení parametrů jednotlivých algoritmů jsou také uloženy do exportovaného souboru, aby je bylo možno později dohledat.

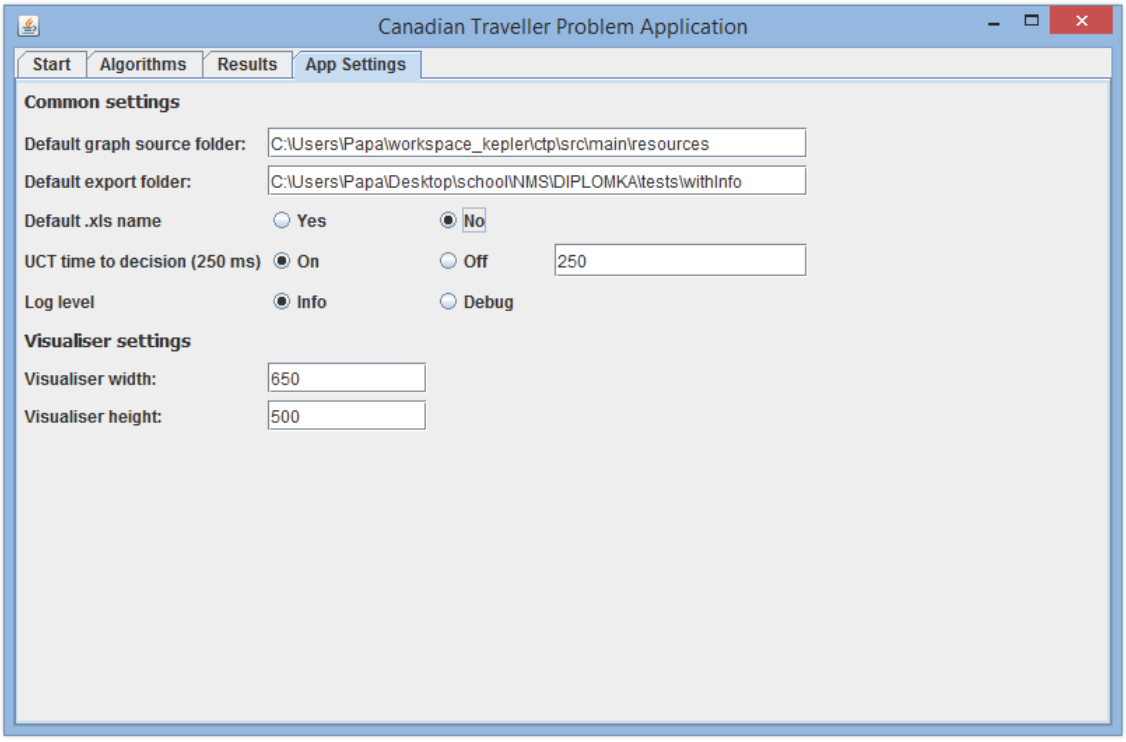


run #	Dijkstra	Greedy	Reposition	Comparision	HOP	ORO	UCTO	UCTO2	UCTP
9	84.0	450.0	530.0	510.0	90.0	100.0	90.0	90.0	84.0
10	70.0	76.0	96.0	76.0	90.0	100.0	90.0	90.0	84.0
11	70.0	196.0	236.0	216.0	90.0	100.0	90.0	90.0	84.0
12	84.0	450.0	530.0	510.0	90.0	100.0	90.0	90.0	84.0
13	70.0	76.0	96.0	76.0	90.0	100.0	90.0	90.0	84.0
14	70.0	76.0	96.0	76.0	90.0	100.0	90.0	90.0	84.0
15	70.0	76.0	96.0	76.0	90.0	100.0	90.0	90.0	84.0
16	70.0	316.0	376.0	356.0	90.0	100.0	90.0	90.0	84.0
17	70.0	196.0	236.0	216.0	90.0	100.0	90.0	90.0	84.0
18	70.0	76.0	96.0	76.0	90.0	100.0	90.0	90.0	84.0
19	70.0	76.0	96.0	76.0	90.0	100.0	90.0	90.0	84.0
20	70.0	196.0	236.0	216.0	90.0	100.0	90.0	90.0	84.0
21	70.0	76.0	96.0	76.0	90.0	100.0	90.0	90.0	84.0
22	70.0	76.0	96.0	76.0	90.0	100.0	90.0	90.0	84.0
23	70.0	76.0	96.0	76.0	90.0	100.0	90.0	90.0	84.0
24	84.0	450.0	530.0	510.0	90.0	100.0	90.0	100.0	84.0
25	70.0	76.0	96.0	76.0	90.0	100.0	90.0	90.0	84.0
26	70.0	76.0	96.0	76.0	90.0	100.0	90.0	90.0	84.0
Avg Cost	72.15	170.46	205.85	185.85	90.00	100.00	90.00	90.77	84.00
Avg Time	0.00	0.00	0.00	0.00	0.19	0.12	0.50	0.48	0.26
Avg Iteratio...	-	-	-	-	-	-	4845	468	154

Obr. 20 Panel výsledků všech algoritmů po 26ti cyklech výpočtů.

4.3.4 Panel nastavení

Panel nastavení je možno vidět na obr. 21. Jeho hlavní funkcí je možnost upravovat nastavení aplikace. Z názvů je celkem patrné, co většina z parametrů dělá. Za zmínku stojí pouze *UCT time to decision*, který umožňuje uživateli vymezit výpočetní čas pro rozhodnutí UCT algoritmů. Větší čas znamená více iterací a tudíž i lepší prohledání stavového prostoru, což většinou vede k lepším výsledkům. Čas se udává v ms. Právě pro toto nastavení je později v panelu výsledků poslední řádek věnovaný průměrnému počtu iterací. U UCT algoritmů s rozvojem do šířky by totiž vždy mělo být provedeno minimálně tolik iterací, kolik mají vrcholy daného grafu maximálně následníků (doporučeno je, aby to bylo přibližně kvadrát této teoretické hodnoty). Při menších hodnotách hrozí riziko výskytu zacyklení.



Obr. 21 Panel s nastavením.

5 Testování

V této kapitole je shrnut postup při testování, společně s výsledky, kterých bylo dosaženo. Dále je zde popis návrhu parametrů vybraných strategií společně s krátkým shrnutím vlivu parametrů na kvalitu řešení a na časové náročnosti výpočtů. Testovány byly strategie GS, HOP, ORO, UCTO, UCTO2 a UCTP pro řešení problému SCTP. GS byl vybrán jako měřítko kvality pro jeho jednoduchost a popularitu. Samozřejmě se u tohoto algoritmu očekávaly horší výsledky pro řešení SCTP, protože GS explicitně nepočítá s pravděpodobností blokací hran.

5.1 Parametry vybraných strategií

Tato podkapitola se zabývá detailnějším popisem vstupních parametrů jednotlivých strategií HOP, ORO, UCTO, UCTO2 a UCTP. Cílem bylo zjistit, pro které hodnoty parametrů je dobrý poměr mezi kvalitou a rychlostí dané strategie. Zároveň byla snaha zjistit, jak je velikost grafu důležitá pro návrh hodnot parametrů. Vzhledem k podobnosti parametrů jsou HOP a ORO probrány společně v nadcházející podkapitole a podobně to platí u strategií založených na UCT algoritmu.

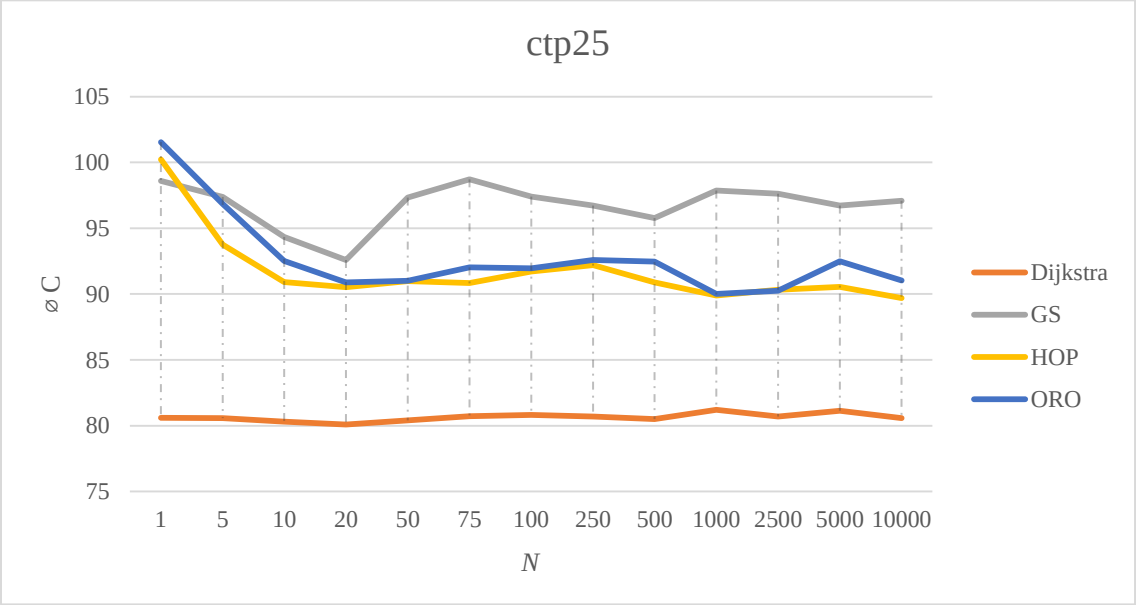
Všechny testy v této podkapitole byly udělány na třech grafech s 25, 50 a 100 vrcholy, které mají postupně 97, 179 a 563 hran a jsou dále označovány jako *ctp25*, *ctp50* a *ctp100*. Pravděpodobnosti blokací hran těchto grafů byly nastaveny v 75 % v intervalu <0.1, 1> zbylých 25 % hran má nulovou pravděpodobnost blokace. Ohodnocení hran byla volena náhodně jako celá čísla z intervalu <10, 50>.

5.1.1 HOP & ORO

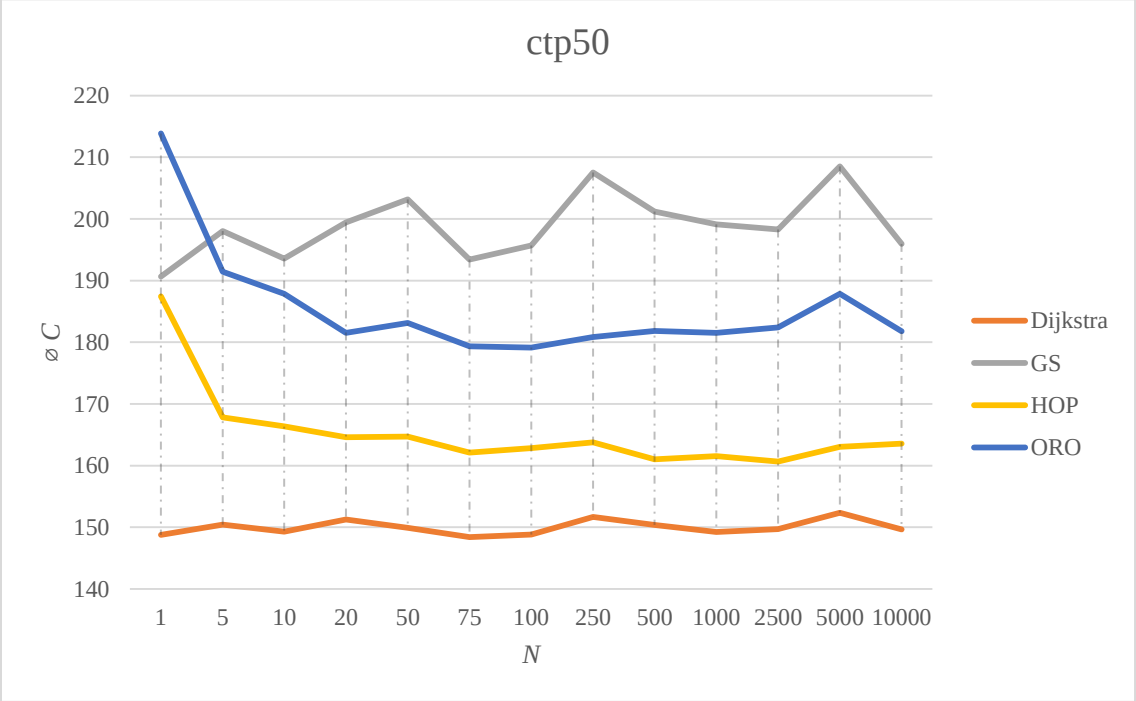
Tyto dvě strategie požadují pro svoji funkci pouze jeden parametr *N*, jenž udává počet *rolloutů*, které budou provedeny pro každého přímého následníka agentovi momentální pozice. Logicky je tedy časová náročnost výpočtu větší s přibývajícím *N*, dále časová náročnost roste s počtem nutných rozhodnutí agenta během cesty z výchozího do cílového vrcholu. Toto je však přímo svázáno s topologií grafu.

HOP počítá odhad cesty v jednom *rolloutu* na základě nejkratší cesty (Dijkstrův algoritmus), a tak je výpočet samozřejmě rychlejší pro stejný počet *N* než u ORO, kde je odhad proveden pomocí GS (zde je provedeno více výpočtů nejkratší cesty pro jeden *rollout*).

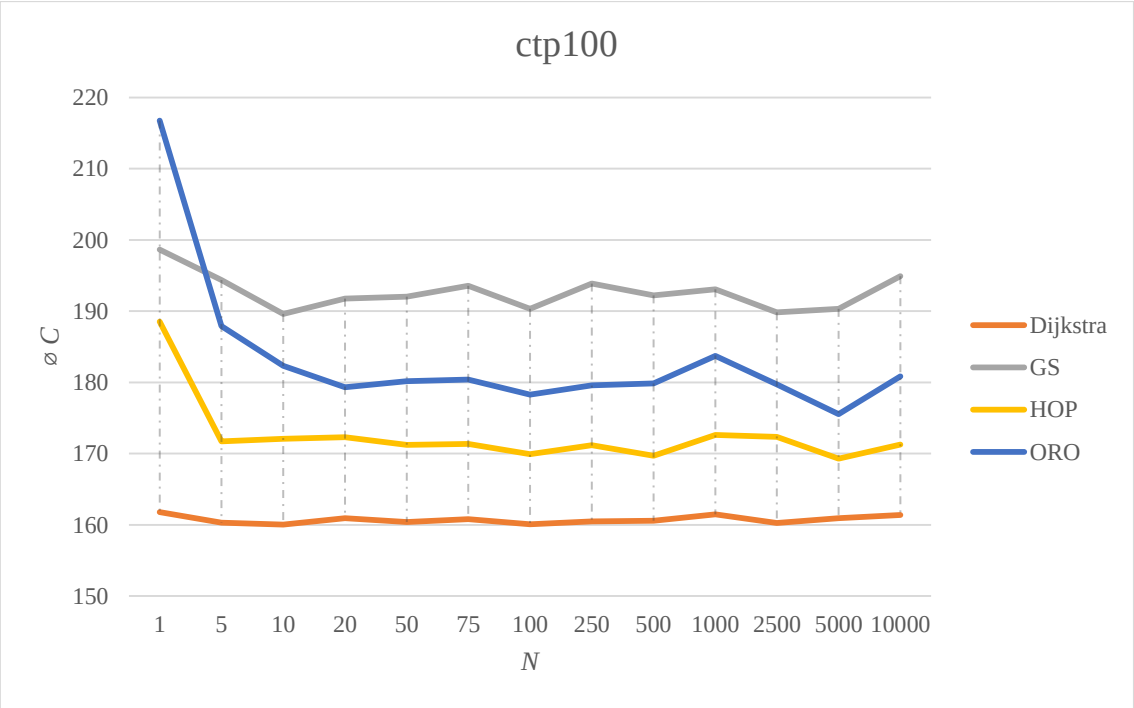
Obě strategie byly testovány na třech výše zmíněných grafech *ctp25*, *ctp50* a *ctp100*. Pro každou testovací hodnotu *N* bylo provedeno 200 běhů, pokaždé s jinou instancí daného grafu. Výsledky z testů jsou pro jednotlivé grafy patrné z grafů na obrázcích obr. 22, obr. 23 a obr. 24, kde je z informativního hlediska rovněž znázorněna průměrná cena GS a Dijkstrova algoritmu provedeného offline, tudíž se jedná o průměr nejlepších možných cen daných instancí (tato hodnota je ovšem pro adaptivní strategie pouze teoretická). Tyto dva algoritmy nejsou samozřejmě závislé na parametru *N*.



Obr. 22 Graf závislosti průměrné ceny cest (⌀ *C*) na počtu *rolloutů* (*N*) pro *ctp25*.



Obr. 23 Graf závislosti průměrné ceny cest (⌀ *C*) na počtu *rolloutů* (*N*) pro *ctp50*.



Obr. 24 Graf závislosti průměrné ceny cest (⌀ *C*) na počtu *rolloutů* (*N*) pro *ctp100*.

Z grafů je patrné, že se zvedajícím počtem *rolloutů* je kvalita řešení HOP i ORO přibližně stejná už od 50 *rolloutů*. Odhady jsou sice stabilnější, ale na testovaných grafech neměly stabilnější odhady statisticky významný dopad na celkovou kvalitu řešení.

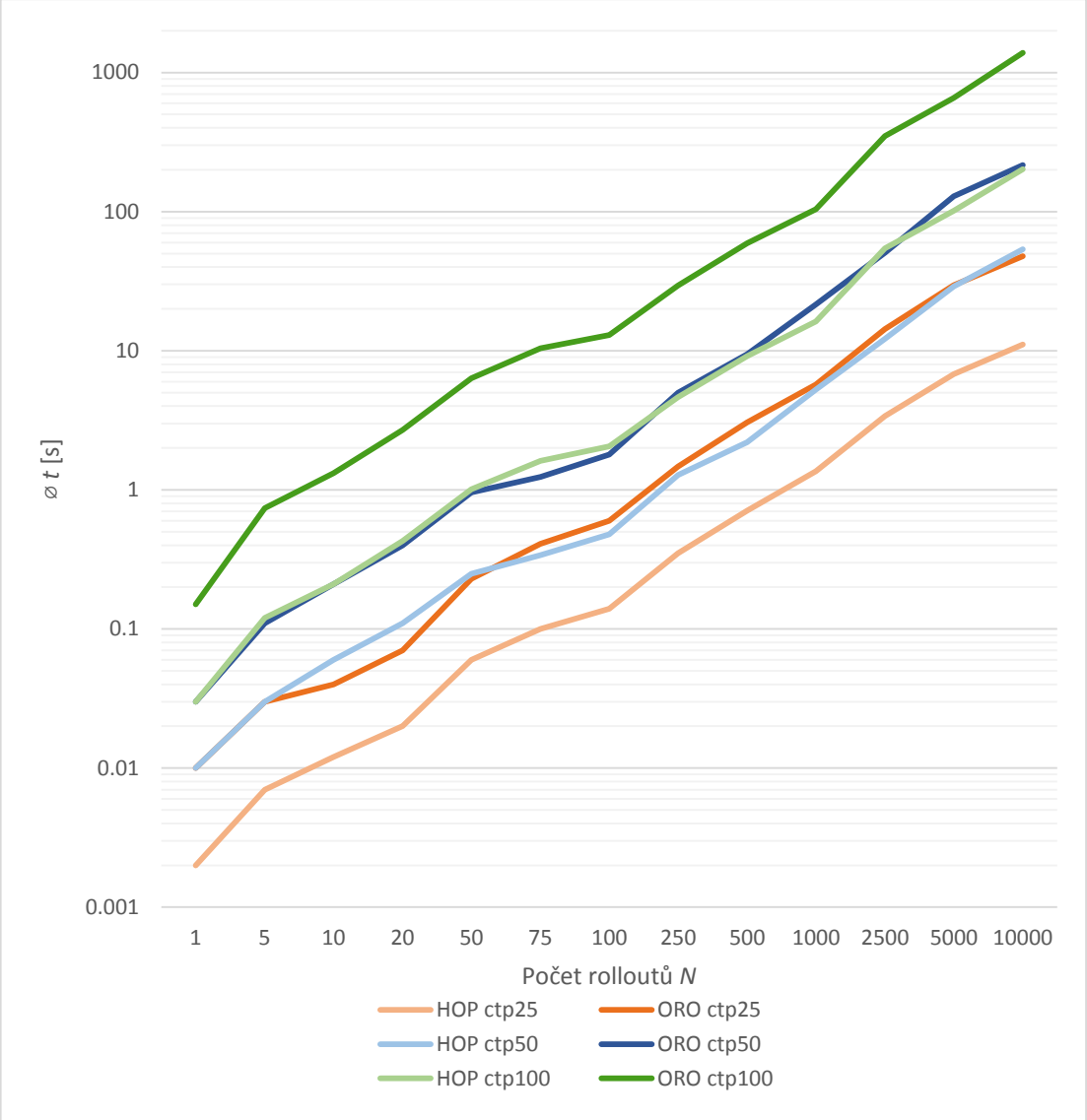
Z obr. 23 je patrné, že i při 200 bězích byly průměrné hodnoty cen (⌀ *C*) strategie GS ještě pořád velmi rozptýlené, jelikož je počet možných instancí grafů úměrný stavovému prostoru, který je exponenciální v závislosti na počtu stochastických hran. Je zřejmé, že by bylo zapotřebí více opakování, aby se průměrné hodnoty ustálily.

V tab. 2 jsou vypsány průměrné hodnoty časů v sekundách pro všechny tři grafy. Jak je z tabulky zřejmé, pro *ctp25* byl HOP přibližně 3 krát rychlejší než ORO se stejným počtem *rolloutů*. Pro *ctp50* byl HOP přibližně 4 krát rychlejší a pro *ctp100* byl HOP přibližně 6 krát rychlejší.

Tab. 2 Průměrné časy (v sekundách) strategií HOP a ORO pro daný počet *rolloutů* a daný graf.

N	HOP ctp25	ORO ctp25	HOP ctp50	ORO ctp50	HOP ctp100	ORO ctp100
1	0.002	0.01	0.01	0.03	0.03	0.15
5	0.007	0.03	0.03	0.11	0.12	0.74
10	0.012	0.04	0.06	0.21	0.21	1.32
20	0.02	0.07	0.11	0.4	0.43	2.69
50	0.06	0.23	0.25	0.96	1.01	6.34
75	0.1	0.41	0.34	1.24	1.62	10.44
100	0.14	0.6	0.48	1.79	2.05	12.98
250	0.35	1.47	1.28	4.99	4.65	29.4
500	0.71	3.06	2.2	9.45	9.13	59.26
1000	1.36	5.72	5.25	21.53	16.27	103.92
2500	3.39	14.36	12.2	50.67	54.51	349.52
5000	6.79	29.66	28.92	129.12	101.38	657.35
10000	11.09	47.83	53.79	215.8	202.51	1385.29

Časová závislost pro počet *rolloutů N* je přibližně lineární, ale v některých případech rostla o trochu rychleji než lineárně a naopak v některých případech rostla i o trochu pomaleji než lineárně. Pro přesnější představu by bylo zapotřebí provést více experimentů a naměřit více dat. Graf znázorňující data z tab. 2 je možno vidět na obr. 25. Co se velikosti grafů týče, tak výpočetní rychlost je přímo úměrná počtu rozhodnutí, které je potřeba v daném běhu strategie vypočítat. Pro testovací grafy (*ctp25*, *ctp50* a *ctp100*) vyšla výpočetní rychlost tak, že HOP pro větší graf byl přibližně stejně rychlý jako ORO pro menší graf, což je také patrné z obr. 25.



Obr. 25 Závislost výpočetního času HOP a ORO na počtu *rolloutů*.

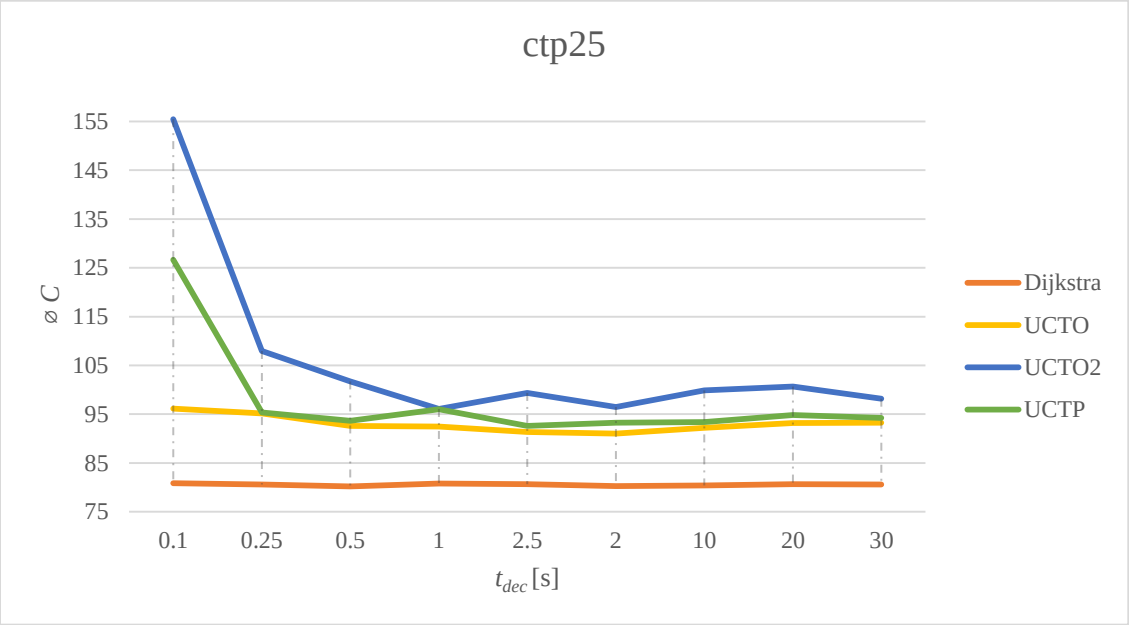
5.1.2 UCT algoritmy

Vstupních parametrů do UCT algoritmů je více, tudíž je odhad nastavení náročnější než u HOP a ORO. V rámci této práce byl parametr M (viz podkap. 3.3.1) nastaven na hodnotu 20, která byla navrhnuta v [2]. Tento parametr má v UCTO důležitou roli ve vyhodnocování UCT formule (viz rovnice (3.5)), ve které je navíc „schovaný“ další parametr B, který balancuje váhy mezi objevováním a využíváním. Po experimentech s tímto parametrem vyšlo najevo, že jeho velikost má rovněž relativně velký dopad na práci UCT algoritmů. Jeho hodnota je volena, jak bylo popsáno dříve, jako průměrný odhad cesty od kořenového uzlu vynásobený 5 pro UCTO. Pro UCTO2 a UCTP je vynásobený 10, aby bylo více podporováno objevování.

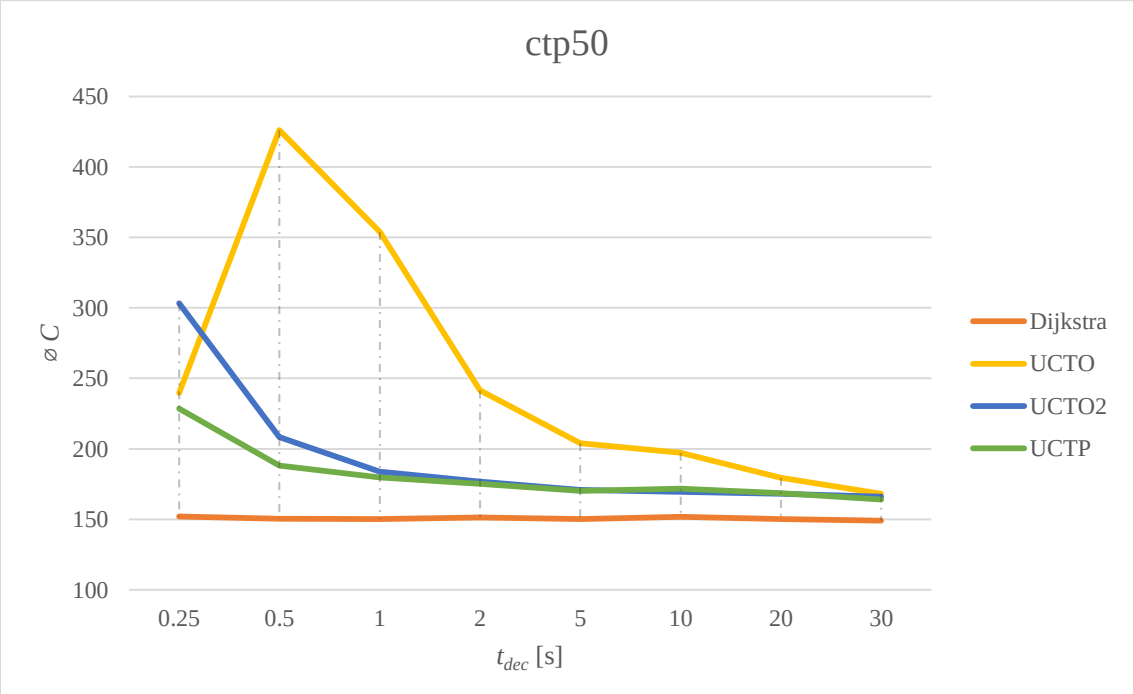
V UCTO2 a v UCTP je navíc parametr N_r udávající počet *rolloutů* provedených v každé iteraci. Ty mají za účel odhadnout každému uzlu UCT stromu očekávanou hodnotu. V rámci této práce je zvolena konstanta 20, která byla experimentálně určena jako dobrý kompromis mezi rychlostí a kvalitou odhadu.

Dále se tato podkapitola bude zabývat pouze závislostí kvality řešení na počtu iterací UCT algoritmu *N* (v UCTO značených taktéž jako *rollouty*). Pro zjištění této závislosti bylo využito stejného postupu jako pro HOP a ORO, tj. pro všechny tři testovací grafy (*ctp25*, *ctp50* a *ctp100*) bylo spočítáno 200 běhů výpočtů.

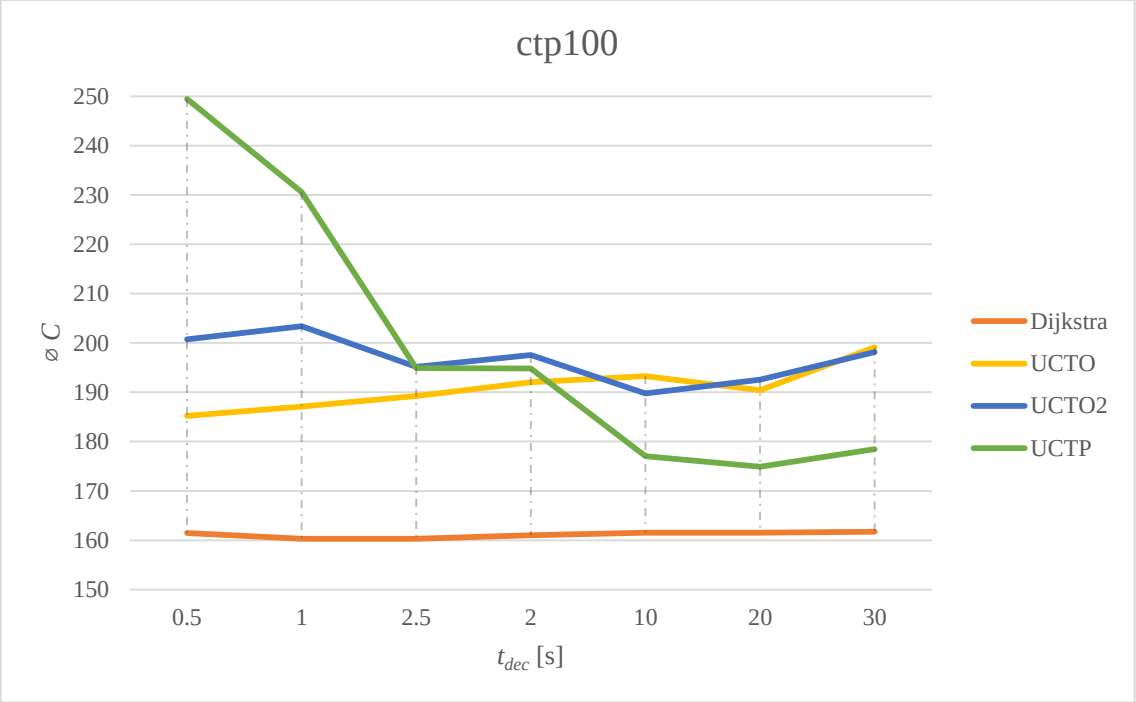
Jelikož je časová závislost na počtu iterací diametrálně rozlišná pro UCT algoritmy s rozvojem do hloubky a UCT algoritmy s rozvojem do šířky, byl v testech nastavován maximální čas na rozhodnutí t_{dec} namísto *N*, neboť tyto dva parametry jsou přímo úměrné (za více času se stihne provést více iterací). Výsledky z testů jsou pro jednotlivé grafy patrné z grafů na obrázcích obr. 26, obr. 27 a obr. 28.



Obr. 26 Graf závislosti průměrné ceny cest (⌀ *C*) na t_{dec} pro *ctp25*.



Obr. 27 Graf závislosti průměrné ceny cest (⌀ *C*) na t_{dec} pro *ctp50*.

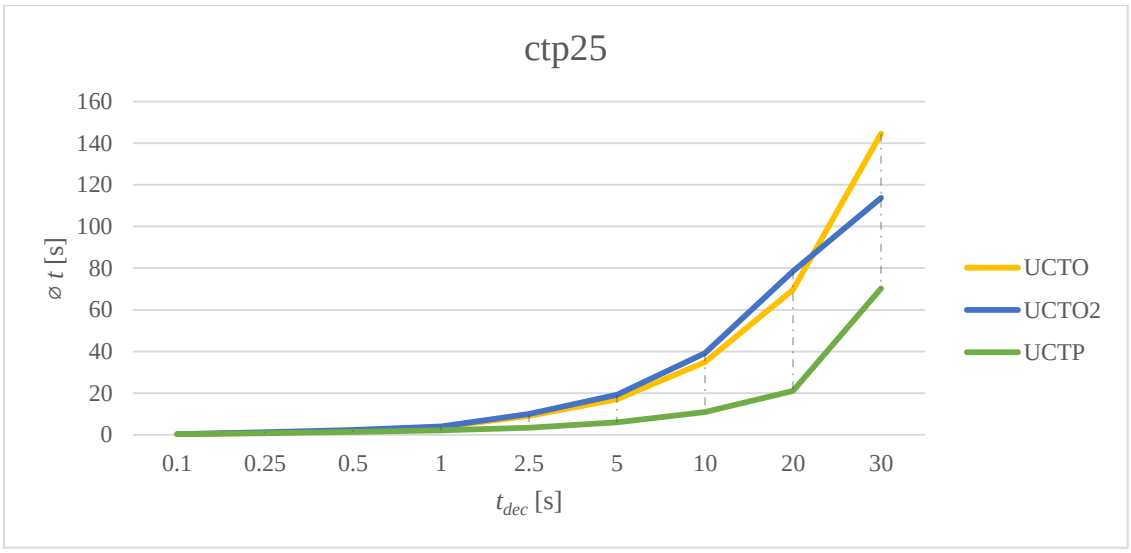


Obr. 28 Graf závislosti průměrné ceny cest (⌀ *C*) na t_{dec} pro *ctp100*.

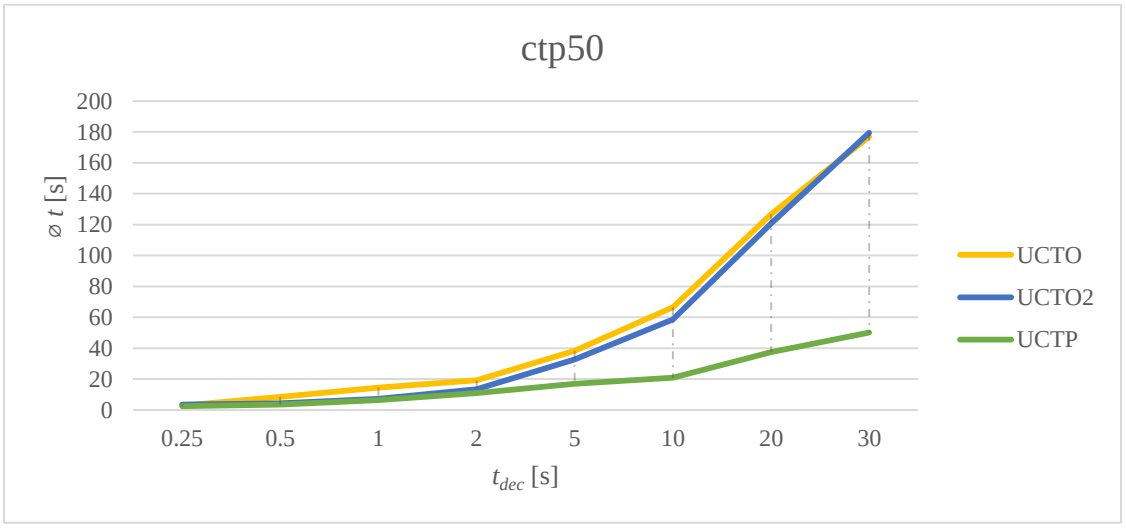
Z výsledků testování není zřejmé, zdali kvalita řešení má obecný trend. Z dalších experimentů se potvrdilo, že závislost kvality řešení je pro dané nastavení ostatních parametrů určitým způsobem spjata s topologií grafu. V některých grafech docházelo k pomalejší konvergenci ke kvalitnější strategii se zvyšujícím se časem na rozhodnutí (např. na *ctp50* viz obr. 27[11]), v jiných se kvalita řešení ustálila v relativně krátkém čase a dále nedocházelo ke zlepšení (viz obr. 26Obr. 26). Pro *ctp100* dokonce docházelo ke zhoršení UCTO a UCTO2 s přibývajícím časem (viz obr. 28). To mohlo být z důvodu relativně malého počtu opakování (*ctp100* má 563 hran a tudíž je obrovské množství možných instancí tohoto grafu). Zhoršení však není tak velké, aby se dalo považovat za statisticky významné. Navíc pro hodnoty t_{dec} = 20 a t_{dec} = 30 došlo i k mírnému nárůstu průměrné ceny cest vypočítaných pomocí Dijkstrova algoritmu (představujícího offline optimum), což může indikovat, že byly generovány „horší“ instance grafu *ctp100*.

Grafy znázorňující závislost průměrných časů výpočtů na času vymezeném pro rozhodnutí t_{dec} jsou pro testované grafy *ctp25*, *ctp50* a *ctp100* postupně na obr. 29, obr. 30 a obr. 31.

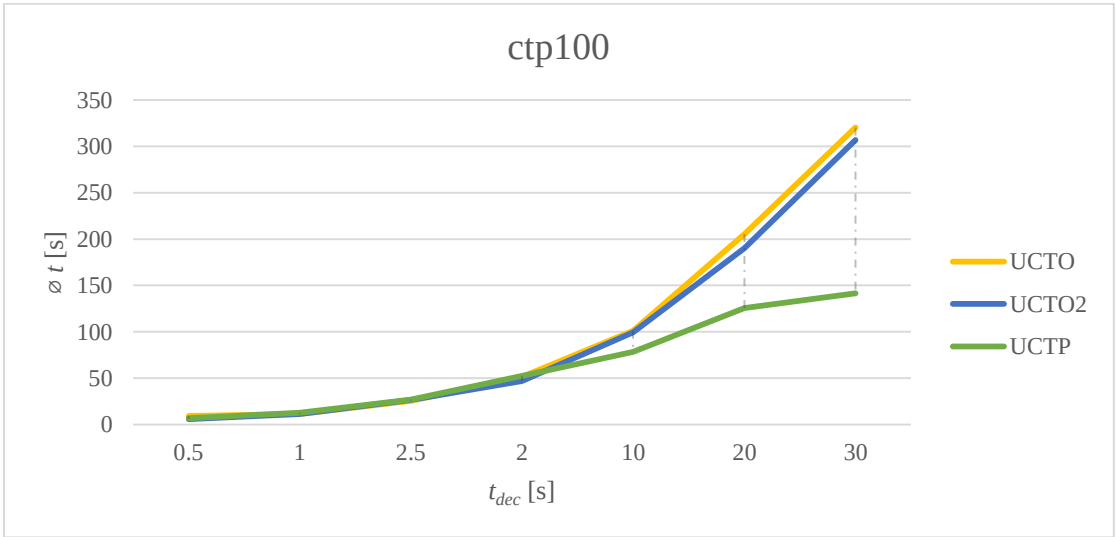
[11] V tomto grafu má navíc UCTO podobnou křivku konvergence, jaká byla zjištěna v [2]. Zhoršení v čase t_{dec} = 0,5 je pravděpodobně způsobeno tím, že je nutno nenavštívené uzly UCT stromu vybrat alespoň jednou. Čím blíže jsou nenavštívené uzly ke kořenovému uzlu, tím více je „rozhozený“ odhad cen, při dalších návštěvách jsou již následníci vybírání UCT formulí a tudíž dochází k jeho opětovnému zlepšení.



Obr. 29 Graf závislosti průměrného času výpočtů na t_{dec} pro *ctp25*.



Obr. 30 Graf závislosti průměrného času výpočtů na t_{dec} pro *ctp50*.



Obr. 31 Graf závislosti průměrného času výpočtů na t_{dec} pro *ctp100*.

Předpoklad rychlosti výpočtů pro UCTO a UCTO2 byl takový, že by měly být přibližně stejné rychlé, protože je exaktně omezen čas t_{dec} (je předpokládáno, že průměrný počet rozhodování by měl být pro všechny UCT algoritmy podobný). Tento předpoklad byl experimentálně potvrzen (viz grafy na obrázcích výše).

Rozdíl však nastává pro UCTP, protože při prohledávání stromu a odřezávání větví UCT stromu, může dojít k situaci, kdy kořen stromu má pouze jednoho přímého potomka. V tomto případě není nutné dál pokračovat v iteracích a označit vrchol, který je asociován s posledním přímým potomkem kořenu jako další agentova akce. Toto „urychlení“ výpočtů je však přímo závislé na topologii grafu. Jak je patrné z grafu na obr. 29 byl UCTP pro *ctp20* s přibývajícím t_{dec} přibližně dvakrát rychlejší než UCTO a UCTO2. Pro *ctp50* byl UCTP dokonce víc než třikrát rychlejší (viz obr. 30) a pro *ctp100* byl UCTP opět dvakrát rychlejší (viz obr. 31).

5.2 Popis testů

Tato podkapitola je věnována popisu podoby dvou testů, které byly provedeny. Každý test měl svoje testovací prostředí, ze kterého se generovaly grafy, na kterých byl problém CTP řešen. Prostředí jsou dále značeny jako A, B. Počet vrcholů byl v obou prostředích zvolen 50[12]. Podrobnější popis jednotlivých prostředí je v textu níže.

V prostředí A byl počet sousedů pro každý vrchol vybrán náhodně mezi 2 až 10. Pravděpodobnosti blokací hran byly nastaveny v 75 % v intervalu <0.1, 1>, zbylých 25 % hran bylo nastaveno na nulovou pravděpodobnost blokace. Ohodnocení hran bylo voleno náhodně jako celá čísla v intervalu <10, 50>.

V prostředí B byl počet sousedů pro každý vrchol vybrán náhodně mezi 2 až 4. Pravděpodobnosti blokací hran byly nastaveny v 75 % v intervalu <0.1, 0.9>, zbylých 25 % hran bylo nastaveno na nulovou pravděpodobnost blokace. Každému vrcholu byly náhodně vybrány 2D souřadnice jako celé čísla z intervalu <0, 99>. Ohodnocení hran bylo vypočítané euklidovskou vzdáleností ze souřadnic pro dané sousední vrcholy.

Hlavním rozdílem mezi prostředím A a B je počet hran. Prostředí A má více hran, což obecně snižuje počet vrcholů, které je nutno překonat, aby bylo dosaženo cílového vrcholu. Na druhou stranu je stavový prostor prostředí A obecně větší.

Pro první test bylo náhodně vygenerováno deset grafů z prostředí A, na kterých byly vybrané strategie testovány každá 500x. Podobně bylo pro druhý test generováno deset grafů z prostředí B, na kterých byly strategie testovány rovněž 500x.

Nastavení parametrů algoritmů bylo zvoleno v souladu z předchozí podkapitolou tak, aby bylo zaručen dobrý poměr mezi kvalitou řešení a dobou výpočtů. Hodnoty byly voleny následovně. Pro HOP bylo nastaveno $N = 1000$. Pro ORO bylo nastaveno $N = 300$, což je pořád v oblasti ustálené kvality řešení (viz obr.23). Tato hodnota byla volena z důvodu, aby HOP

[12] Tato velikost reprezentuje svým způsobem „střední“ velikost grafů, pro které už není řešení CTP triviální.

i ORO měly přibližně stejně výpočetního času, což vyplynulo z výsledků uvedených v podkap. 5.1.1. Pro UCT algoritmy byl vymezen čas na rozhodnutí t_{dec} = 10 s. Tento čas dával možnost pro UCTO vypočítat přibližně 10 000 *rolloutů* (iterací), což bylo v [2] označeno za dobrý kompromis mezi rychlostí a kvalitou strategie.

Všechny testy byly provedeny na notebooku Lenovo IdeaPad G510, jehož hardware je následovný:

- Procesor: Intel® Core™ i5-4210M (2,6 GHz)
- Paměť: 4GB (1x 4GB) DDR3L 1600 MHz
- Grafická karta: AMD Radeon R7 M265 - 2GB

5.3 Výsledky

V této podkapitole jsou prezentovány výsledky z testů, které jsou popsány v předcházející podkapitole. Výsledky prvního testu (prostředí A) jsou uvedeny v tab. 3 a výsledky druhého testu (prostředí B) jsou v tab. 4. Jednotlivé strategie jsou ve sloupcích a jsou pro ně vypsány průměrné ceny cest s 95% intervalem spolehlivosti. V posledních dvou řádcích je celková průměrná cena ($\varnothing C$) a průměrná hodnota času ($\varnothing t$), který bylo zapotřebí k výpočtu řešení danou strategií. Z informativního hlediska je v prvním sloupci tabulek uvedena průměrná cena offline optima (určeno pomocí Dijkstrova algoritmu). Výsledek adaptivní strategie, která si pro daný graf vedla nejlépe, je pro rychlejší orientaci vyznačen tučně.

Tab. 3 Výsledky testu č. 1

Graf	Dijkstra	GS	HOP	ORO	UCTO	UCTO2	UCTP
A50-0	176.3 ± 2	276.7 ± 8	256.1 ± 10	**220.3 ± 5**	238.9 ± 5	243.6 ± 7	239.8 ± 9
A50-1	151.7 ± 2	201.8 ± 6	**161.6 ± 4**	181.5 ± 5	181.2 ± 5	166.4 ± 4	169.5 ± 4
A50-2	116.9 ± 2	161.8 ± 6	**149.5 ± 6**	150.2 ± 5	155.0 ± 6	161.9 ± 7	157.6 ± 6
A50-3	110.1 ± 1	150.3 ± 4	130.7 ± 2	135.8 ± 3	**124.2 ± 3**	127.9 ± 3	132.5 ± 3
A50-4	150.8 ± 2	194.5 ± 7	**160.3 ± 3**	161.0 ± 3	179.1 ± 5	194.0 ± 6	161.6 ± 3
A50-5	200.7 ± 3	279.0 ± 9	243.4 ± 5	**241.9 ± 6**	251.0 ± 5	256.5 ± 6	254.9 ± 5
A50-6	176.2 ± 2	229.8 ± 4	**186.0 ± 2**	198.8 ± 3	217.6 ± 4	228.5 ± 4	187.0 ± 2
A50-7	175.4 ± 2	225.4 ± 5	196.4 ± 4	**196.5 ± 4**	200.4 ± 5	208.5 ± 5	197.2 ± 4
A50-8	192.6 ± 3	262.9 ± 6	**224.6 ± 4**	225.1 ± 4	235.5 ± 5	261.5 ± 5	228.5 ± 5
A50-9	165.6 ± 1	198.7 ± 4	**183.5 ± 3**	183.6 ± 3	194.2 ± 4	199.8 ± 3	193.6 ± 3
⌀ *C*	161.6	218.1	189.2	189.5	197.7	204.9	192.2
⌀ *t* [s]	0	0	5.7	8	72.3	75.1	23.2

Tab. 4 Výsledky testu č. 2

Graf	Dijkstra	GS	HOP	ORO	UCTO	UCTO2	UCTP
B50-0	477.1 ± 6	597.9 ± 12	**550.2 ± 9**	559.2 ± 10	595.9 ± 11	604.4 ± 12	565.1 ± 9
B50-1	607.3 ± 6	834.7 ± 15	727.3 ± 12	**706.2 ± 11**	812.3 ± 17	754.8 ± 14	746.3 ± 12
B50-2	606.5 ± 6	847.7 ± 19	728.7 ± 12	**712.6 ± 12**	775.2 ± 15	812.3 ± 16	744.0 ± 12
B50-3	498.0 ± 6	663.2 ± 14	**568.9 ± 11**	579.8 ± 10	756.4 ± 17	648.9 ± 12	584.2 ± 11
B50-4	430.5 ± 4	522.9 ± 12	**477.9 ± 10**	492.8 ± 8	491.8 ± 10	508.4 ± 13	484.4 ± 9
B50-5	533.0 ± 5	690.8 ± 14	625.4 ± 11	**619.6 ± 10**	670.7 ± 11	646.7 ± 12	660.3 ± 13
B50-6	467.9 ± 6	606.7 ± 15	553.8 ± 12	**548.5 ± 12**	604.4 ± 13	642.1 ± 25	585.5 ± 14
B50-7	535.2 ± 7	700.0 ± 15	616.7 ± 13	**613.5 ± 13**	676.0 ± 15	667.4 ± 14	647.8 ± 14
B50-8	531.7 ± 6	722.2 ± 16	647.7 ± 10	**643.7 ± 12**	671.4 ± 12	678.8 ± 13	654.8 ± 11
B50-9	472.3 ± 4	580.7 ± 10	542.2 ± 8	**536.5 ± 8**	604.1 ± 9	620.5 ± 10	538.9 ± 7
⌀ *C*	516.0	676.7	603.9	601.2	665.8	658.4	621.1
⌀ *t* [s]	0	0	6.8	7.9	108.1	117.3	19.2

Výsledky potvrdily, že strategie přímo pracující s pravděpodobnostmi blokací jsou pro řešení CTP lepší než GS. Nejlépe si vedly HOP a ORO, jejichž výsledky byly pro oba testy velmi podobné. Strategie HOP překonala v 1. testu GS o 13,3 % a v 2. testu o 11,2 %.

UCT algoritmy si nevedly tak dobře, jak bylo předpokládáno a to pravděpodobně kvůli neoptimálnímu nastavení vstupních parametrů, kterých je relativně hodně a jejich nastavení není triviální záležitostí. Dalším faktorem mohly býti grafy testovacích prostředí, které pravděpodobně neobsahovali časté výskyty známých „pastí“, ve kterých se HOP a ORO chovají suboptimálně (viz podkap. 3.4), výjimkou byl pouze graf A50-3 (viz tab. 3), kde UCTO překonal GS o 17,4 %, což přibližně odpovídá výsledkům publikovaných v [2]. Originální strategie UCTP si vedla z UCT algoritmů nejlépe a překonala v 1. testu GS o 11,9 % a v 2. testu o 8,2 %, zároveň výpočetní náročnost oproti dalším UCT algoritmům byla v 1. testu třikrát lepší a v 2. testu dokonce pětkrát lepší. To bylo způsobeno rychlejším rozhodováním, když nastala situace, kdy v UCT stromu byl pouze jeden přímý potomek.

Strategie HOP byla zároveň i nejrychlejší, ale to záleží na vstupních parametrech. V publikaci [2] byl počet *rolloutů* pro HOP a ORO nastaven na 10 000, což zapříčinilo jejich relativně vysokou časovou náročnost, kde ORO vyšel dokonce pomalejší než UCTO. Takto vysoký počet *rolloutů* se zdá být podle výsledků z podkap. 5.1.1 zbytečný. Je však nutno dodat, že pro jistější závěry by bylo zapotřebí provést více testů na různých grafech a pro větší počet opakování.

6 Závěr

Tato práce se zabývala problémem kanadského cestujícího (CTP), který se dá definovat jako problém hledání nejkratší cesty ve stochastickém prostředí. V druhé kapitole byly popsány typy CTP problému a k nim stručně shrnuty existující metody řešení. V dalších kapitolách se práce zaměřila na strategie řešící variantu SCTP, to je problém, kde cestující zná dopředu pravděpodobnosti blokací daných úseků silnic. Cílem je bylo navrhnout adaptivní strategii, která by končila pokud možno, co nejmenší cenou cesty z výchozí do cílové pozice.

Podrobně popsány byly strategie HOP, ORO, UCTB, UCTO poprvé publikovány v [2]. Dále jsou prezentovány originální strategie UCTO2 a UCTP, které sdílejí spolu s výše jmenovanými strategiemi podobný přístup k řešení. Ten spočívá v simulacích, ve kterých jsou pomocí příslušných pravděpodobností blokací generovány možné instance problému. Ty pak slouží k určení odhadů cen cest, pomocí kterých je rozhodnuto, kudy se cestující má vydat.

Strategie byly implementovány v programovacím jazyku Java. Největší výhodou je OOP návrh, který umožňuje snadné rozšíření o další strategie. Pro ověřovací a srovnávací experimenty byla následně vyvinuta okenní aplikace, jejímuž popisu je věnována čtvrtá kapitola této práce.

Pro srovnání bylo využito heuristiky značené GS (*Greedy Strategy*). Při testování bylo ověřeno, že strategie přímo pracující s pravděpodobnostmi blokací jsou pro řešení SCTP lepší než GS. V testech nejlépe dopadla strategie HOP, která dosahovala výsledků lepších než GS v průměru o 12 %. I přes skutečnost, že HOP i ORO mají známé suboptimální chování (viz podkap. 3.4), dosáhly lepších výsledků než UCTO, UCTO2 a UCTP, které toto chování eliminují. Tyto strategie jsou totiž velice komplexní a mají více parametrů, jejichž optimální nastavení není triviální. Nejlépe si ze strategií postavených na UCT algoritmu vedla originální strategie UCTP, která předčila GS v průměru o 10%.

Do budoucna je možné „doladit“ UCT algoritmy tak, aby bylo plně dosaženo jejich potenciálu. Dále je možno strategie modifikovat, aby byly použitelné i pro jiné varianty CTP problému jako například SRCTP, který lépe modeluje reálnou situaci.

Seznam použité literatury

[1] PAPADIMITRIOU, C. H; YANNAKAKIS, M. Shortest paths without map. In *Theoretical Computer Science,* 1991, pp 127-150.

[2] 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.

[3] 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.

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

[5] WESTPHAL S. A note on the k-Canadian traveller problém. In *Information Processing Letters,* 2008, pp. 87-89.

[6] SU, B.; XU, Y. F.; HU, M. L.; ZHU, B. H.; ZHU, Z. J. Online recoverable canadian traveller problem. In *Proceedings of the international konference on managment science and engineering,* 2004, pp. 633-639.

[7] SU, B.; XU, Y. F.; XIAO, P.; TIAN, L.; A risk-reward competetive analysis for the recoverable Canadian traveller problem. In *Proceedings of the 2nd konference on combinatorial otimization and applications,* 2008, pp. 417-426.

[8] LIAO, C.; HUANG, Y. Generalized Canadian traveller problems. *International Symposium on Algorithms and Computation*, 2012, pp. 701-712.

[9] BENDER, M.; WESTPHAL S.; An optimal randomized online algorithm for the k-Canadian Traveller Problem on node-disjoint paths, 2013,

[10] 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.

[11] MATOUSEK, R.; SOUSTEK, P.; DVORAK, J.; BEDNAR, J.; ACO in Task of Canadian Traveller Problem. In: *Proceedings of 18th International Conference on Soft Computing – MENDEL 2012*, Brno, 2012, pp. 600-603.

[12] SOUSTEK, P.; MATOUSEK, R.; DVORAK, J.; BEDNAR, J.; Canadian Traveller Problem: A Solution using Ant Colony Optimization. In: *Proceedings of 19th International Conference on Soft Computing – MENDEL 2013*, Brno, 2013, pp. 439-444.

[13] BULLNHEIMER, B., HARTL, R.; STRAUSS, C.; A new ranked based version of the Ant System A computational Study. In: *Technical report*, Vienna, 1997.

[14] STÜTZLE, T.; HOOS, H. H.; MAX-MIN Ant System. In: *Generation Computer Science*, No. 16, 2000, pp. 889-914.

[15] KOCSIS L.; SZEPESVARI C.; Bandit based Monte-Carlo planning. In: *Proc. ECML*, New York, 2006, pp. 282-293.

[16] ZHANG, H.; Y. XU. The k-Canadian Travelers Problem with Communication. In: *FAW-AAIM*. Berlin, 2011, pp. 17-28.

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

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

[19] FRIED, D.; SHIMONY, E. S.; FELNER, A. Optimal Policies for Special Cases of the Canadian Traveler Problem, Germany, 2010

[20] MANASEE, M. S.; McGEOCH, L. A.; SLEATOR, D. D. Competitive algorithms for server problems. In: *Journal of Algorithms*, 1990, pp. 208-230.

[21] DAVID, S. B.; BORODIN, A.; A new measure for the study of the on-line algorithm. *Algorithmica*, 1994, pp. 73-91.

[22] 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.

[23] Ctp. *GitHub* [online]. 2017 [cit. 2017-05-14]. Dostupné z: https://github.com/SebastianFilip/ctp

[24] *Eclipse* [online]. [cit. 2017-05-14]. Dostupné z: https://eclipse.org/

[25] *JGraphT* [online]. Barak Naveh and Contributors, 2003 [cit. 2017-05-14]. Dostupné z: http://jgrapht.org/

[26] Jgraphx. *GitHub* [online]. [cit. 2017-05-14]. Dostupné z: https://github.com/jgraph/jgraphx

[27] Log4j: Logging Services. *Apache* [online]. [cit. 2017-05-14]. Dostupné z: https://logging.apache.org/log4j/2.x/

[28] WindowBuilder. *Eclipse* [online]. [cit. 2017-05-14]. Dostupné z: https://eclipse.org/windowbuilder/

[29] *ObjectAid* [online]. [cit. 2017-05-14]. Dostupné z: http://www.objectaid.com/

[30] *Ruprecht-Karls-Universität Heidelberg* [online]. Germany [cit. 2017-05-14]. Dostupné z: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95

Seznam použitých zkratek

CTP	Canadian Traveller Problem
SCTP	Stochastic CTP
RCTP	Recoverable CTP
SRCTP	Stochastic RCTP
SPP	Shortest Path Problem
DAG	Directed Acyclic Graph
DPG	Disjoined-Path Graph
GS	Greedy Strategy (viz podkap. 2.12.1.1)
RS	Reposition Strategy (viz podkap. 2.1.1)
CS	Comparison Strategy (viz podkap. 2.1.1)
NDP	Náhodná Diskrétní Proměnná
HOP	Hindsight Optimization (viz podkap. 3.2)
ORO	Optimistic Rollout (viz podkap. 3.2)
UCT	Upper Confidence bounds applied to Trees
UCTB	Blind UCT (viz podkap. 3.3.1)
UCTO	Optimistic UCT (viz podkap. 3.3.1)
UCTP	Pruning UCT (viz podkap. 3.3.2)
OOP	Objektově Orientované Programování

Seznam příloh

Příloha A – Datový nosič s elektronickou verzí této práce a vytvořenou okenní aplikací