VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

HLEDÁNÍ NEJKRATŠÍ CESTY POMOCÍ MRAVENČÍCH KOLONIÍ - JAVA IMPLEMENTACE

ANT COLONY OPTIMIZATION ALGORITHMS FOR SHORTEST PATH PROBLEMS - JAVA IMPLEMENTATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. MAREK DOSTÁL
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. RADOMIL MATOUŠEK, Ph.D.
SUPERVISOR

BRNO 2014

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2013/14

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Marek Dostál

který/která studuje v **magisterském studijním programu**

obor: **Aplikovaná informatika a řízení (3902T001)**

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:

Hledání nejkratší cesty pomocí mravenčích kolonií - Java implementace

v anglickém jazyce:

Ant Colony Optimization Algorithms for Shortest Path Problems - Java implementation

Stručná charakteristika problematiky úkolu:

Cílem práce bude návrh programové aplikace, která bude využitelná v úlohách nalezení nejkratší cesty. Pro zvolené prostředí bude pomocí různých strategií ACO (Ant Colony Optimization, optimalizace pomocí mravenčích kolonií) generována optimální trajektorie. Předpokládá se implementace v pseudo 3D prostředí, kde překážky budou mít nepřekonatelný charakter (zadaný 2D topologií) a terén bude mít 3D charakteristiku.

Cíle diplomové práce:

1. Implementace různých ACO strategií v jazyce Java.
2. Implementace pseudo 3D prostředí a řešení úlohy nalezení nejkratší cesty.
3. Otestování navrženého řešiče na zvolených testovacích úlohách.
4. Popis ACO strategií, popis implementace a diskuze k dosaženým výsledkům vč. diskuze k využití exaktněji pojatých heuristik typu Dijkstrova algoritmu.

Seznam odborné literatury:

A. Colorni, M. Dorigo et V. Maniezzo, Distributed Optimization by Ant Colonies, actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991.

M. Dorigo, Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italie, 1992.

Vedoucí diplomové práce:doc. Ing. Radomil Matoušek, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2013/14.

V Brně, dne 25. 11. 2013

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA STROJNÍHO INŽENÝRSTVÍ DĚKANÁT

Ing. Jan Roupec, Ph.D.
Ředitel ústavu

prof. RNDr. Miroslav Doupovec, CSc., dr. h. c.
Děkan

ABSTRAKT

Tato diplomová práce se zabývá hledáním nejkratší cesty pomocí mravenčích algoritmů. V teoretické části jsou popsány mravenčí algoritmy. V praktické části jsou zvoleny tyto algoritmy pro návrh a implementaci hledání nejkratší cesty v jazyce Java.

ABSTRACT

This diploma thesis deals with ant colony optimization for shortest path problems. In the theoretical part it describes Ant Colony Optimization. In the practical part ant colony optimization algorithms are selected for the design and implementation of shortest path problems in the Java.

KLÍČOVÁ SLOVA

Mravenčí kolonie, Pseudo 3D prostředí, Java, Ant systém, Elitism ant system, Rank – based ant system, Max – min ant systém, Ant colony systém.

KEYWORDS

Ant colony, Pseudo 3D environment, Java, Ant system, Elitism ant system, Rank – Based ant system, Max – Min ant system, Ant colony system.

Prohlášení o originalitě

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, pod vedením vedoucího diplomové práce pana doc. Ing. Radomila Matouška, Ph.D. a s použitím uvedené literatury.

V Brně dne 29.5.2014 Podpis: ……..…

BIBLIOGRAFICKÁ CITACE

DOSTÁL, M. *Hledání nejkratší cesty pomocí mravenčích kolonií – Java implementace.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2014. 69 s.
Vedoucí diplomové práce doc. Ing. Radomil Matoušek, Ph.D.

PODĚKOVÁNÍ

Děkuji rodině za trpělivost a veškerou podporu v průběhu celého studia.

Rovněž vedoucímu diplomové práce doc. Ing. Radomilu Matouškovi, Ph.D. za pomoc a odborné vedení při vypracování této práce.

Obsah:

SEZNAM POUŽITÝCH SYMBOLŮ

OS	Operační systém
API	Application Programming Interface
Java ME	Java Micro Edition
Java SE	Java Standard Edition
Java EE	Java Enterprise Edition
JVM	Java Virtual Machine – interpret Javy pro daný operační systém
OOP	Objektově Orientované Programování
GUI	Graphic User Interface
ACO	Ant Colony Optimization
AS	Ant System
EAS	Elitism Ant System
RBAS	Rank–Based Ant System
MMAS	Max–Min Ant System
PSO	Particle Swarm Optimization

1. ÚVOD

V živé přírodě lze nalézt mnoho tvorů s různými strategiemi hledání potravy. Živočichové si po nesčetně let zdokonalovali své strategie do té míry, že řešení problémů pomocí biologicky inspirovaných přístupů se jeví jako velice výkonné i v porovnání s exaktněji pojatými metodami. V posledních 30 letech dochází k intenzivnímu studování těchto strategií a k jejich postupné počítačové implementaci v kontextu výkonných optimalizačních metod. Autory těchto metod jsou především zahraniční výzkumníci a týmy, a proto je logicky technicky zavedené názvosloví převážně anglické. V tomto ohledu je vhodné připomenout, že ne všechny zavedené pojmy jsou z logických důvodů dále interpretovány do českého jazyka. Dále diskutované algoritmy jsou představiteli skupiny algoritmů označované jako tzv. rojová či hejnová inteligence (*Swarm Intelligence*). Tato skupina je dále řazena do oblasti tzv. *Soft Computing*, či ekvivalentně do oblasti tzv. počítačové inteligence (*CA, Computational Intelligence*), což jsou oblasti, které lze obecně zastřešit oborem tzv. umělé inteligence (*AI, Artificial Intelligence*). Jako příklad této třídy optimalizačních meta-heuristik je možné uvést metodu optimalizace včelím rojem, hejnem částic, rojem světlušek a mravenčí kolonií. Dané algoritmy mohou být pochopitelně kombinovány s dalšími principy z oblasti Soft computing.

Diplomová práce je věnována inteligenci hejna, konkrétně mravenčím algoritmům a jejich základním variantám. V první kapitole je stručně uveden popis rojové inteligence, její druhy a využití. Konkrétní popis mravenčích algoritmů a biologická předloha je předmětem následující kapitoly. V další části je prezentována realizace uvedených biologických principů do tzv. umělých mravenčích kolonií. V práci jsou dále popsány konkrétní modifikace mravenčích algoritmů, jako jsou *Ant System, Elitism Ant System, Rank-Based Ant System, Max-Min Ant System a Ant Colony Optimization*. Následující kapitola ukazuje samotnou implementaci uvedených algoritmů. Je zde popsán princip funkcionality programovacího jazyka Java, implementace uvedených algoritmů ve formě vlastních metod, tedy tzv. řešičů, dále vlastní popis programové aplikace zahrnující tvorbu mapy překážek v pseudo 3D prostředí, tvorbu objektů překážek, aj. Poslední kapitola, která nese potenciál vědeckého zkoumání, je věnována testování navržených řešičů, kde se snaží v první části nalézt optimální nastavení jednotlivých parametrů algoritmů a dále je provedeno testování na vybraných mapách, které reprezentují různou složitost problému hledání nejkratší cesty. V závěru jsou porovnány výsledky provedených testů a je vyhodnocena nejúspěšnější strategie k vyhledávání nejkratší cesty.

Nedílnou součástí této práce je vlastní programová aplikace v jazyce Java, která umožňuje návrh vlastních pseudo 3D prostředí a je rovněž určena k simulaci chování daných algoritmů vzhledem k jejich parametrizaci i zadané povaze úlohy. Pomocí této aplikace byly získány předložené výsledky.

2. ROJOVÁ INTELIGENCE

Rojová inteligence se inspiruje v chování určitých druhů, které vykazují tzv. *kolektivní chování*. Jako příklad uveďme roj včel, hejna ptáků. Implementací tohoto chování dostáváme technik umělé inteligence, která je schopna řešit rozličné problémy.

Obecně je rojová inteligence postavena na populacích jednoduchých *agentů*, kteří interagují lokálně s okolím a vzájemně mezi sebou. Komunikace těchto agentů může probíhat přímo mezi sebou nebo nepřímo působením na okolí v místě svého působení [01].

Mezi výhody metod rojové inteligence, patří jednoduchost implementace, relativně nízká výpočetní náročnost z pohledu globálnosti řešení oproti uvažovaným exaktním metodám.

2.1 Příklady algoritmů založených na rojové inteligenci

K typickým zástupcům algoritmů rojové inteligence patří:

- optimalizace mravenčí kolonií (ant colony optimization),
- optimalizace včelím rojem (bees algorithm),
- optimalizace hejnem částic (particle swam optimization),
- optimalizace hejnem světlušek (glowworm swarm optimization).

Optimalizace mravenčích kolonií je předmětem této práce a je jí tudíž věnován větší prostor v následující kapitole 3.

2.2 Optimalizace včelím rojem

Cílem těchto algoritmů je napodobení chování skutečných včel. Dle typu chování, které se snaží tyto algoritmy napodobovat, je lze rozdělit na:

- Optimalizační metody napodobující páření chování včel.

  Tyto algoritmy se inspirují svatebním letem královny. Kde královna je nositelkou dědičné informace, obsahující řešení optimalizačního problému. Celý proces hledání optimálního řešení, probíhá simulovaným svatebním letem, kde královna se setkává s trubci a podle zdatnosti trubce se s ním spáří. Trubci jsou v každém kroku královny nově generováni.

  Po ukončení letu, královna naklade vajíčka obsahující řešení problému. A o tyto vajíčka se starají dělnice, které se snaží za pomocí lokálního prohledávání zlepšit tato řešení. Pokud je řešení problému lepší, než stávající královny, je tato královna nahrazena novou s lepším řešením a opakuje se opět celý cyklus svatebního letu [02].

Některé vybrané varianty dle [02]:

- honeybee mating optimisation algoritm,
- bee system (včelí systém),
- queen-bee evolution.

- Optimalizační metody inspirované hledáním potravy.

Podle článků [03], [04], se včelí roj v základu rozděluje na dělníky, diváky a zvědy. Při hledání řešení se roj dále rozděluje do dvou skupin (včel dělnic a diváků). První skupina včelích dělnic se rozmístí na zdroje potravy. Možné řešení problému představuje zdroj potravy a jeho množství určuje kvalitu řešení. Včelí dělnice mění polohu a testují možné zdroje potravy. Zapamatovávají si pouze ty zdroje, které mají větší množství nektaru (kvalitu řešení) než předešlé zkoumané řešení. Po ukončení hledání zdrojů dělnice předají informace o zdrojích včelám divákům na tzv. tančící ploše. Včely diváci tyto informace vyhodnotí a zvolí hlavní zdroje potravy. Včelí dělníci, kterým byl zdroj potravy odmítnut, se stávají zvědy a vyhledávají nové zdroje, které jsou pak nabídnuty dělníkům.

Využitím těchto metod se zabývala bakalářská práce [12], která prezentovala globální optimalizaci pomocí včelího algoritmu.

Některé vybrané varianty dle [03], [04]:

- artificial bee colony algoritm (algoritmus umělého včelího roje),
- bees algorithm (včelí algoritmus),
- bee colony optimization algoritm (optimalizační algoritmus včelím rojem).

2.3 Optimalizace hejnem částic

Předlohou pro tuto metodu, byla různá živočišná hejna např. ptačí hejna, hejna ryb.

V tomto algoritmu se hejno skládá z jednotlivých částic. Tyto částice mají svoji *polohu*, *rychlost*, *paměť předchozích úspěchů hledání.* Jednotlivé částice, jsou ovlivňovány ostatními úspěšnějšími částicemi. Pohyb hejna, resp. částic probíhá v jednotlivých krocích. V každém kroku se také upravují hodnoty popisující částice [05].

Optimalizace hejnem částic (PSO, Particle Swarm Optimization), byla využita k řešení mnoha optimalizačních úloh. Pomocí PSO byl například úspěšně řešen problém plánování pohybu robota, viz [05].

2.4 Optimalizace hejnem světlušek

Tento algoritmus se inspiruje chováním světlušek. Jeho anglické označení je logicky Firefly algorithm. Autorem algoritmu je Xin She, který tento algoritmus rovněž prezentoval na konferenci MENDEL v roce 2011 (konference byla pořádána na půdě FSI VUT v Brně). Každý člen hejna je vybaven virtuální svítivou látkou, která je nastavována v závislosti na úspěšnosti člena při procházení prohledávaného prostoru. Čím více je člen

roje úspěšný, tím více *září* a více přitahuje ke své pozici další ostatní členy roje. Každý člen roje má omezenou vzdálenost dohledu[06].

2.5 Využití rojové intelligence

Aplikace založené na rojové inteligenci jsou využívány v mnoha oborech. Hojně je využívána v robotice pro navigaci robotů a vozidel. Směrování toku v sítích. Aplikace rojové inteligence v logistice pro optimalizování cest, také pro optimalizaci cest na tištěných spojích. Využití také našla v oblasti data miningu.

Příkladem využití mravenčích algoritmů jsou různé roboty pro transport předmětů a materiálů. Fraunhofer Institute for Material Flow and Logistics se zabývá využitím těchto robotů v logistickém sektoru. Snaží se za pomocí těchto robotů, navrhnout lepší skladovací systém založený na autonomních robotech [13].

Obr. 1 Ukázka experimentálního automatizovaného skladiště.

3. MRAVENČÍ ALGORITMY

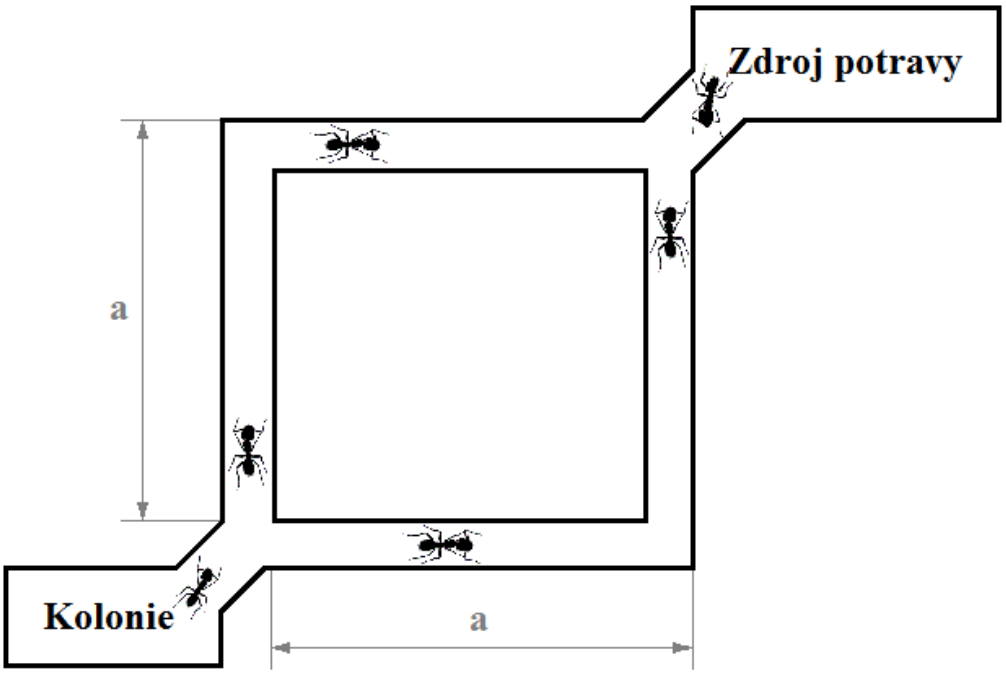
Mravenčí algoritmy využívají samoorganizační vlastnosti svých agentů v kolonii. Inspirací pro tyto algoritmy byly mravenčí kolonie, které dokáží vyvinout koordinovanou činnost při hledání cesty ke zdroji potravy, přestože nejsou nijak centrálně řízeny.

První, kdo tyto algoritmy popsal, byl *M. Dorigo* ve své doktorské práci. Mezi jejich první využití patřilo hledání specifické cesty grafem v úloze tzv. obchodního cestujícího (Travelling Salesman problém) [07], [08].

3.1 Biologická inspirace mravenčích algoritmů

Při pozorování přírodních mravenčích kolonií biologové sledovali schopnosti mravenců nacházet nejkratší cestu mezi hnízdem a zdrojem potravy. Tyto poznatky vedly k pokusům s mravenci, jako je experiment s dvojím mostem ilustrovaným na obr. 2. V experimentu byli pozorování jednotlivý mravenci při hledání cesty od kolonie ke zdroji potravy, kde jim byly nabídnuty dvě stejně dlouhé cesty. Cesta jednotlivých mravenců se ustálila na jedné ze dvou možných variant. Z experimentů vyplynulo, že volba mezi dvěmi stejně dlouhými cestami záleží pouze na náhodné volbě mravenců.

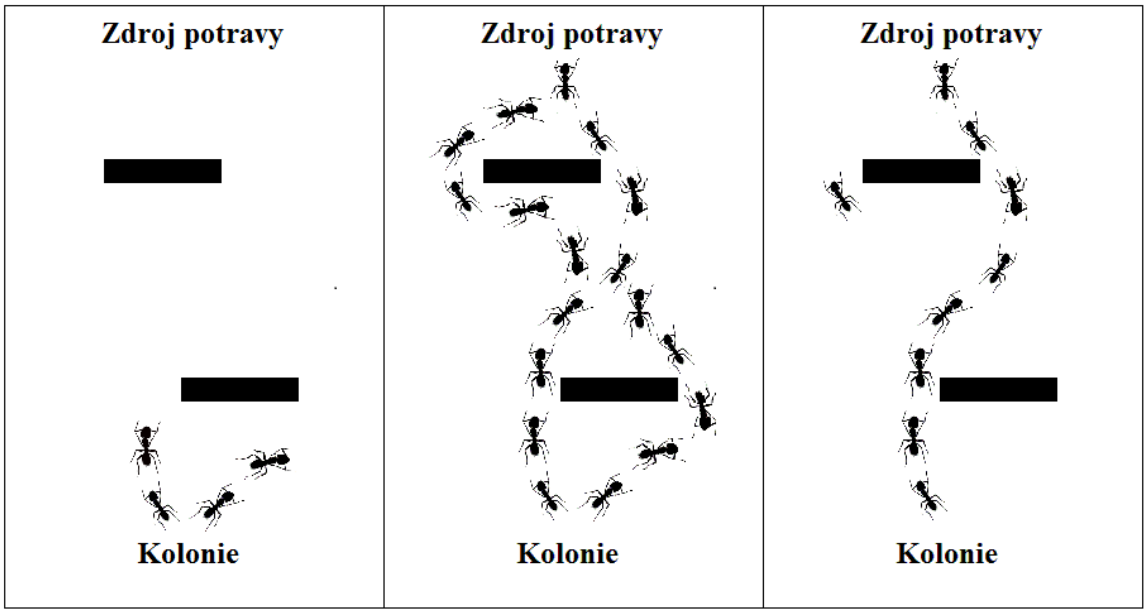
Mravenčí kolonie nejsou *nijak centrálně řízené*, a i přesto mravenčí kolonie vykazují *koordinaci* a *komplexnost*. Jednotliví mravenci se v přírodě dělí do několika skupin, tzv. kast, a to podle svého určení (např. dělník, královna, voják). Komunikace mezi jednotlivými mravenci neprobíhá přímo. Mravenci komunikují chemicky pomocí modifikace prostředí feromonem [07]. Feromon je chemická substance specifická pro každý druh. Vylučování feromonu a reakce na jeho přítomnost je způsob, jak nepřímo komunikovat mezi jedinci stejného druhu, příp. stejné kasty.



Obr. 2 Ukázka pokusu s dvojím mostem při studiu mravenců při hledání zdroje potravy.

3.2 Pohyb reálných mravenců při hledání zdroje potravy

Při pohybu prostorem se mravenec rozhoduje o další cestě majoritně na základě množství feromonu, které na ni uložili jiní mravenci. Jestliže se na budoucí cestě nenachází žádný feromon, mravenec volí cestu náhodně. Tento způsob pohybu je základním kamenem samoorganizace kolonie mravenců [01],[07],[08].



Obr. 3 Příklad průběhu nalezení nejkratší cesty mezi mravenčí kolonií a zdrojem potravy.

Na obr. 3 je zobrazen pohyb reálných mravenců hledajících cestu mezi zdrojem potravy a hnízdem.

Celý proces hledání potravy začíná tím, že se mravenci vydávají z kolonie pátrat po zdroji potravy. Na začátku neexistuje žádná feromonová stopa, mravenci jsou nuceni volit svoji cestu náhodně, a při tom prohledávají prostor v okolí kolonie. Jakmile mravenec nalezne zdroj potravy, vezme její část a vydává se zpět do své kolonie, a zároveň po celou dobu značí feromonem svoji cestu. Ostatní mravenci, kteří se vydávají hledat zdroj potravy, pak reagují na tyto feromonové stopy zanechané mravenci. Feromonová stopa vedoucí ke zdroji potravy se postupem času vytrácí díky odpařovaní (evaporaci), pokud není zvyšována nebo alespoň udržována mravenci přepravující potravu do kolonie. Pokud se zdroj potravy vyčerpá, mravenci si neodnáší část potravy a nezanechávají tak za sebou feromonovou stopu, která se pak na této cestě začne odpařovat, až se zcela vytratí.

V případě, že k jednomu zdroji existuje několik různých cest, mravenci začnou preferovat nejkratší cestu. Nejkratší cesta zabere kratší časový úsek, a proto se na ní nahromadí více feromonu a tím ji zatraktivňuje pro více mravenců. Feromon na delších cestách se začne vypařovat, až úplně vymizí a tím mravenci zapomenou tyto cesty, jako je tomu na obr. 3.

V případě, že mravenec během svého putování směrem ke zdroji potravy nebo na zpáteční cestě narazí na rozdělení feromonové stopy na více cest, je jeho volba následné *i*-té cesty dána pravděpodobností $P_{(i)}(t)$.

$$P_{(i)}(t) = \frac{\left(S_{(i)}(t)+k\right)^{h}}{\sum_{j=1}^{n}\left(S_{(j)}(t)+k\right)^{h}} \quad (1)$$

kde:

$S_{(i)}(t)$ - je intenzita feromonové stopy na větvi *i*,

n - je počet možných pokračování cesty,

h, k – jsou konstanty dle experimentů s reálnými mravenčími koloniemi, jsou hodnoty voleny $k = 20$ a $h = 2$.

Tyto zákonitosti volby cest byly experimentálně zjištěny pozorováním reálných mravenčích kolonií a provedením experimentů (např. experiment s dvojím mostem).

3.3 Umělé mravenčí kolonie

Umělé mravenčí kolonie byly vytvořeny podle vlastností reálných kolonií. Inspirací byly schopnosti mravenců hledat nejkratší cesty ke zdrojům potravy. Optimalizační schopnosti zahrnují soustavu pravidel bez přímé komunikace jedinců (agentů v umělých koloniích) a vlastnosti při takto zadaných pravidlech dosahovat co nejlepšího řešení velikosti délky výsledné cesty.

Jak uvádí práce [07], odlišnosti mezi reálnými a umělými koloniemi lze shrnout v těchto bodech:

- uložení feromonové stopy probíhá až po dokončení hledání cesty mravencem,
- mravenci mají vlastní paměť nalezené cesty,
- rozdíl v odpařování - u reálných mravenců probíhá nepřetržitě. U umělých probíhá až na konec iterace,
- odpařováním feromonové stopy dochází u umělých mravenců k "zapomínání" horších cest. Tento mechanismus rovněž předchází uváznutí algoritmu v lokálním extrému.

Mravenčí algoritmy se snaží přiblížit své optimalizační schopnosti reálným mravenčím koloniím, a tyto schopnosti pak co nejlépe využít k řešení daných problémů, jak uvádí [07], [08].

3.4 Optimalizace pomocí mravenčích kolonií

Schopnosti a vlastnosti reálné mravenčí kolonie v řešení problémů s optimalizací cesty ke zdroji potravy byly předlohou pro tuto metodu.

Optimalizační problém, který má být touto metodou řešen, musí být prezentovatelný jako spojitý nebo diskrétní graf [07].

Dle [10], [11], [12] lze popsat kostru většiny ACO algoritmů v pseudokódu takto:

```
1.      Procedure_ACO()
2.         Inicializace()
3.            do
4.               GenerujReseni()
5.               AktualizujFeromon()
6.            while (podmínka)
7.      end – procedure
```

Obr. 4 Obecná kostra ACO algoritmů

Spouštění procedury ACO probíhá takto:

1) Jako první se spouští *Inicializační metoda*.

Tato metoda má na starost nastavení všech potřebných parametrů pro hledání řešení problému a inicializaci grafové struktury. Rovněž nastavuje počáteční hodnoty feromonové stopy na hranách grafu.

2) Po nastavení všech parametrů, jsou cyklicky volány 3 metody:

a) První metoda *GenerujReseni.*

Procedura spouští vyhledání řešení. Kdy se *m* mravenců snaží vyhledat řešení nezávisle na sobě. Velikost feromonové stopy ovlivňuje pohyb mravenců. Pohyb jednotlivých mravenců je přitom nezávislý na jiných mravencích. Řešení, která tito mravenci naleznou, pak ovlivní množství feromonu, které se v následující etapě ukládá na hrany grafu.

b) Druhá metoda *AktualizujFeromon*.

V této proceduře dochází k aktualizaci feromonových stop na hranách grafové struktury.

Aktualizace probíhá ve dvou etapách:

- Odpařování feromonových stop na hranách grafu.

Dochází ke snižování hodnoty feromonu na hranách. Toto snižování poskytuje formu zapomínání, a zároveň možnost, jak přimět algoritmus k prohledávání neprozkoumaných oblastí grafu.

- Ukládání feromonových stop.

Dochází zde ke zvyšování hodnoty feromonové stopy na hranách řešení úspěšných mravenců. Zvýšení hodnot feromonu vede k větší pravděpodobnosti, že v další iteraci algoritmu si mravenci vyberou cestu s nejvyšší hodnotou feromonové stopy.

3) Ukončení algoritmu proběhne po splnění určité ukončovací podmínky, nebo jejich kombinací. K typickým podmínkám ukončení běhu algoritmu patří: dosažení časového limitu, dosažení maximálního počtu iterací, dosažení definované doby stagnace algoritmu.

3.5 Základní druhy ACO algoritmů

V referenčních zdrojích [07], [08], [09] lze najít mnoho různých variant ACO algoritmů. Mezi základní varianty patří:

- ant system (AS) ,
- elitist ant system (EAS),
- rank-based ant system (RBAS),
- max-min ant system (MMAS) [10],
- ant colony system (ACO).

Rozšíření ACO algoritmů lze provést mnoha způsoby:

- upravením schémat ukládání feromonových stop,
- předefinováním heuristické informace,
- upravením pravidel pro chování agentů,
- spoluprací ACO algoritmu s nějakým vhodně zvoleným optimalizačním algoritmem.

3.5.1 Ant system

Ant systém (AS) je původní algoritmus, na který navazuje většina ACO algoritmů. Mezi první řešené úlohy tímto algoritmem byly úlohy hledání optimální cesty grafem [07].

Průchod mravenců grafovou strukturou je obdobně, jako u reálných mravenců, pravděpodobnostní. Pravděpodobnost výběru hrany je ovlivňována *hodnotou feromonu* a případně *heuristickou informací.*

Hodnota pravděpodobnosti, že mravenec k využije hranu spojující vrcholy (i, j) je dána rovnicí [07]:

$$p_{(\mathrm{i,j})}^{k} = \frac{|\tau_{(i,j)}|^{\alpha} \cdot |\eta_{(i,j)}|^{\beta}}{\Sigma_{z \notin \mathrm{Tabu}^{\mathrm{k}}} |\tau_{(i,j)}|^{\alpha} \cdot |\eta_{(i,j)}|^{\beta}} \qquad (2)$$

kde:

- $\tau_{(\mathrm{i,j})}$ – hodnota intenzity feromonu na hraně (i, j),
- $\eta_{(\mathrm{i,j})}$ – heuristická informace spojená s hranou (i, j),
- z – označuje bezprostřední sousedy vrcholu i,
- α,β – koeficienty, které určují důležitost feromonové stopy a heuristické informace,

- *Tabu* – je množina zakázaných stavů, které jsou složeny z hran a vrcholů použitých během konstrukce řešení. Tato oblast zamezuje opětovnému přechodu do již navštívených stavů a také zacyklení celého algoritmu.

Heuristická informace η spojená s hranou (i,j) je pro většinu problémů uvažována jako vzdálenost vrcholu i, j. Je dána vztahem:

$$\eta_{(i,j)} = \frac{1}{d_{(i,j)}} \tag{3}$$

kde:

- $d_{(i,j)}$ – je vzdálenost mezi vrcholy i, j tj. délka hrany grafu.

Jakmile mravenci naleznou cestu k cíli nebo ukončili hledání, je na grafové struktuře aktualizována hodnota feromonové stopy. Aktualizace feromonových hodnot probíhá ve dvou krocích:

- Snížení hodnoty feromonových stop pomocí tzv. odpařování (evaporace):

$$\tau_{(i,j)} = \rho \cdot \tau_{(i,j)} \tag{4}$$

kde:

- $\tau_{(i,j)}$ – hodnota intenzity feromonu na hraně (i,j),
- ρ – koeficient odpařování, ρ se nachází v *intervalu* (0,1).

- Ukládání feromonových stop na hranách.

Na výslednou cestu úspěšných mravenců je aplikována zvýšená feromonová stopa. Výsledná intenzita je dána vztahem:

$$\tau_{(i,j)} = \tau_{(i,j)} + \sum_{k=1}^{m} \Delta\tau_{(i,j)}^{k} \tag{5}$$

kde:

- $\Delta\tau_{(i,j)}^{k}$ – přírůstek intenzity feromonu na hraně (i, j) mravence k
- $\tau_{(i,j)}$ – hodnota intenzity feromonu na hraně (i, j).

Přírůstek intenzity na hraně:

$$\Delta\tau_{(i,j)}^{k} = \frac{1}{C_k} \qquad \forall(i,j) \in T_k$$

$$0 \qquad \forall(i,j) \notin T_k \tag{6}$$

kde:

- $\Delta\tau_{(i,j)}^{k}$ – hodnota intenzity feromonu *k* mravence na hraně (*i*, *j*),
- k – k–tý mravenec,
- C_k – je délka řešení zkonstruovaného mravencem *k*,
- T_k – řešení zkonstruované mravencem *k*.

3.5.2 Elitism ant system

Vylepšení, které zavádí elitism ant system, spočívá v posílení feromonové stopy na hranách grafu, které provádí pouze mravenec s nejlepší nalezenou cestou v dané iteraci [10],[11]. Vlastnosti a rozhodování mravenců zůstávají stejné jako u *ant systemu*.

Posílení hodnoty feromonové stopy na hranách použitých mravencem s nejlepší délkou cesty:

$$\tau_{(i,j)=}\tau_{(i,j)} + \Delta\tau_{(i,j)}^{bs} \qquad \forall(i,j) \in T_{bs} \tag{7}$$

kde:

- $\tau_{(i,j)}$ – hodnota intenzity feromonu mravence na hraně (*i*, *j*),
- $\Delta\tau_{(i,j)}^{bs}$ – přírůstek intenzity feromonu na hraně (*i*, *j*) mravence s nejlepším nalezeným řešením.

$$\Delta\tau_{(i,j)}^{bs} = \frac{e}{L_{bs}} \tag{8}$$

kde:

- e – koeficient určující váhu řešení,
- L_{bs} – délka nejlepšího řešení.

3.5.3 Rank-Based ant system

Na rozdíl od ant systemu, kde ukládají feromonovou stopu všichni úspěšní mravenci, u rank-based ant systemu ukládá feromonovou stopu pouze určený *rozsah mravenců* s nejlepším řešením, jak uvádí literatura [07], [09]. Po dokončení dané iterace algoritmu jsou všichni mravenci, kteří nalezli cestu k cíli, seřazeni podle délky jejich cesty. Následně jsou pak vybráni *w*-1 mravenci s nejlepšími řešeními, kteří pak ukládají svoji feromonovou stopu podle:

$$\tau_{(i,j)} = \tau_{(i,j)} + \sum_{r=1}^{w-1}(w-r)\cdot\Delta\tau_{(i,j)}^{r} + w\cdot\Delta\tau_{(i,j)}^{bs} \quad (9)$$

kde:

- $\tau_{(i,j)}$ – hodnota intenzity feromonu na hraně (*i, j*),
- w – počet mravenců, kteří ukládají stopu, w <= poctu mravenců N,
- $\Delta\tau_{(i,j)}^{bs}$ – hodnota intenzity feromonu *k*-tého mravence na hraně (*i, j)*,
- $\Delta\tau_{(i,j)}^{r}$ – hodnota intenzity feromonu *k*-tého mravence na hraně (*i, j)*.

$$\Delta\tau_{(i,j)}^{bs} = \frac{1}{L_{bs}} \quad (10)$$

kde:

- L_{bs} – délka nejlepšího řešení.

$$\Delta\tau_{(i,j)}^{r} = \frac{1}{L_{r}} \quad (11)$$

kde:

- L_r – je délka řešení *r*-tého mravence.

3.5.4 Max-min ant system

Max-min ant system vylepšuje původní systém o omezení velikosti feromonové stopy daným intervalem. Tento interval velikosti feromonové stopy má za úkol přimět mravence k většímu prohledávání grafové struktury a tím zlepšit výslednou délku nalezené cesty.

Mezi základní vlastnosti max-min ant systemu lze podle literatury [07], [09], [10] zařadit:

- Vysoká počáteční intenzita feromonu

Intenzita je nastavena v počátku běhu algoritmu na hranách na hodnotu τ_{max} . Tato vlastnost společně s vysokou hodnotou koeficientu odpařování ρ, dovoluje algoritmu více prohledávat stavový prostor.

- Podmínka pro ukládání feromonové stopy

Ukládání feromonové stopy může vést k jejímu hromadění na hranách jednoho řešení, které není optimální a může způsobit uvíznutí výsledné cesty v neoptimální délce. V takovém případě dojde k resetování feromonové stopy na hranách grafu na počáteční hodnotu τ_{max}.

Podmínka pro aktualizaci hodnot feromonové stopy:

$$\tau_{(i,j)=}\left[\tau_{(i,j)} + \Delta\tau_{(i,j)}^{bs}\right]_{(\tau_{\min})}^{(\tau_{\max})} \tag{12}$$

kde:

- $\tau_{(i,j)}$ – hodnota intenzity feromonu na hraně (i, j),
- $\Delta\tau_{(i,j)}^{bs}$ – hodnota intenzity feromonu k, mravence na hraně (i, j) dle rovnice(10),
- $\tau_{\max}$ – hranice maximálního množství feromonu,
- $\tau_{\min}$ – hranice minimálního množství feromonu.

3.5.5 Ant colony system

Ant colony system rozšiřuje max-min ant system o dvojí aktualizaci feromonové stopy. Feromonová stopa je od počátku nastavena na velkou hodnotu, stejně jako je tomu u max-min ant systemu. Lokální aktualizace má za úkol odpudit ostatní mravence hledající cestu do cíle od již nalezené cesty grafovou strukturou v dané iteraci algoritmu. Lokální aktualizaci feromonové stopy mravenci provádí ihned po nalezení cesty. Mravenec, který lokálně aktualizuje feromonovou stopu, snižuje hodnotu feromonu na jím použité cestě a tím odpuzuje ostatní mravence od použití jím využité cesty. Po dokončení vyhledávání cest všemi mravenci v kolonii je provedena globální aktualizace feromonové stopy, kdy je aktualizována cesta grafem nejlepšího mravence v iteraci.

Dalším rozšířením oproti ant systemu je možnost použití dvou pravidel pro pohyb mravenců. Výběr mezi starým pravidlem, který využívá ant system a pravidlem zavedeným v ant colony systemu, je využívána hodnota proměnné q. Ta se následně porovnává s hodnotou parametru q_0 a podle velikosti se vybere pravidlo pro pohyb mravence [07], [09].

- Pravidla pro pohyb mravenců

Mravenci používají kombinaci starého a agresivnějšího pravidla pro výběr cesty a konstrukci řešení.

Pseudonáhodné proporční pravidlo:

$$j = \begin{cases} \text{argmax}_{(l \in N_i^k)}\{\tau_{il} \cdot [\eta_{il}]^{\beta}\} & \text{pokud } q \leq q_0 \\ J & \text{pokud } q > q_0 \end{cases} \qquad (13)$$

kde:

- $\tau_{i,l}$ – hodnota intenzity feromonu na hraně (i, l),
- $\eta_{(i,j)}$ – heuristická informace spojená s hranou (i, l),
- α,β – koeficienty, které určují důležitost feromonové stopy a heuristické informace,
- J – cílový vrchol hrany vybraný na základě rovnice (2),
- q – náhodná proměnná s intervalu (0,1),
- q_0 – je proměnná v rozmezí (0,1).

- Ukládání feromonové stopy

Feromonová stopa je následně aktualizována dle rovnice:

$$\tau_{(i,j)=}\rho \cdot \tau_{(i,j)} + (1 - \rho) \cdot \Delta\tau_{(i,j)}^{bs} \qquad \forall (i,j) \in T_{bs} \qquad (14)$$

Kde:

- ρ – koeficient odpařování, ρ se nachází v intervalu (0,1),
- $\tau_{(i,j)}$ – hodnota intenzity feromonu na hraně (i, j),
- $\Delta\tau_{(i,j)}^{bs}$ – hodnota intenzity feromonu k mravence na hraně (i, j) dle rovnice (10),
- T_{bs} – nejlepší nalezené řešení.

- Aktualizace lokální intenzity feromonových stop

Aktualizace probíhá lokálně na straně mravenců. Vždy když najde mravenec řešení, provede lokální aktualizaci feromonové stopy dle rovnice (15). Velkým přínosem je, že na hranách řešení, které je lokálně aktualizováno, poklesne feromonová stopa a tím snižuje pravděpodobnost, že si ji jiný mravenec v dané iteraci zvolí pro konstrukci řešení.

Aktualizace lokální feromonové stopy proběhne dle rovnice:

$$\tau_{(i,j)=}(1 - \xi) \cdot \tau_{(i,j)} + \xi \cdot \tau_0 \qquad (15)$$

kde:

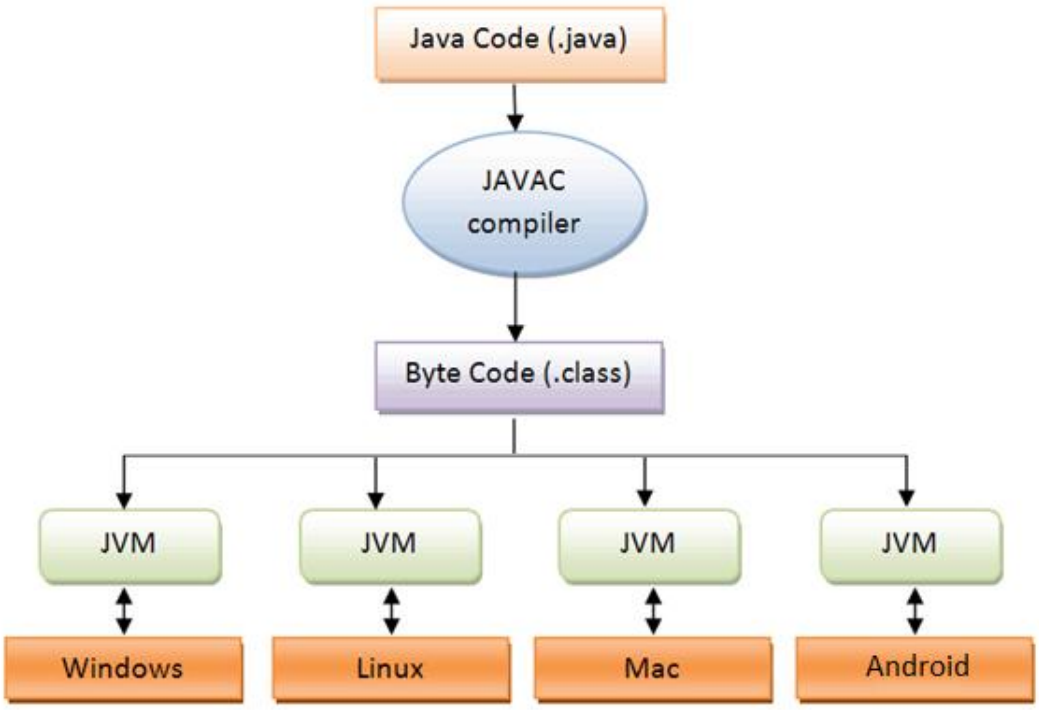
- $\tau_{(i,j)}$ – hodnota intenzity feromonu na hraně (i, j),
- ξ – koeficient v intervalu $<0,1>$,
- τ_0 – hodnota intenzity feromonu použitá při inicializaci.

4. VÝVOJOVÉ PROSTŘEDÍ JAVA

4.1 Popis jazyka

Java je open-source programovací jazyk, který je svou syntaxí podobný jazykům C#, C++. Byl představen v roce 1995 firmou Sun Microsystem a rychle se rozšířil díky svým vlastnostem na operační systémy a hardware. V současné době jsou programy psané v tomto jazyce využívány na osobních *počítačích* (s různým OS) a v mobilních telefonech.

Jazyk Java je tzv. *interpretovaný*. Ze zdrojového *.java* vytváří kompiler tzv. *mezikód* (Byte code), který je následně interpretován na *Java Virtual Machine* (JVM). Celý tento proces demonstruje obr. 6. Tento systém zajišťuje univerzálnost programu, který tak může být spuštěn na různých OS.



Obr. 5 Schéma kompilace programu.

Mezi výhody tohoto systému lze zařadit univerzálnost, nezávislost na architektuře, *generační správu paměti* (která je realizována pomocí *Garbage Collectoru*), *výkonost* (díky „Just-in-time‘‘, kdy je do strojového kódu překládán pouze kód, který je v danou chvíli zapotřebí).

Naopak mezi nevýhody patří *pomalejší start programů*, větší *paměťová a hardwarová náročnost* běhu programů.

4.2 Varianty jazyka Java

Jednotlivé varianty jazyka Java mají vlastní *aplikační rozhraní* (API), které je uzpůsobené pro dané zařízení [11].

Různé platformy:

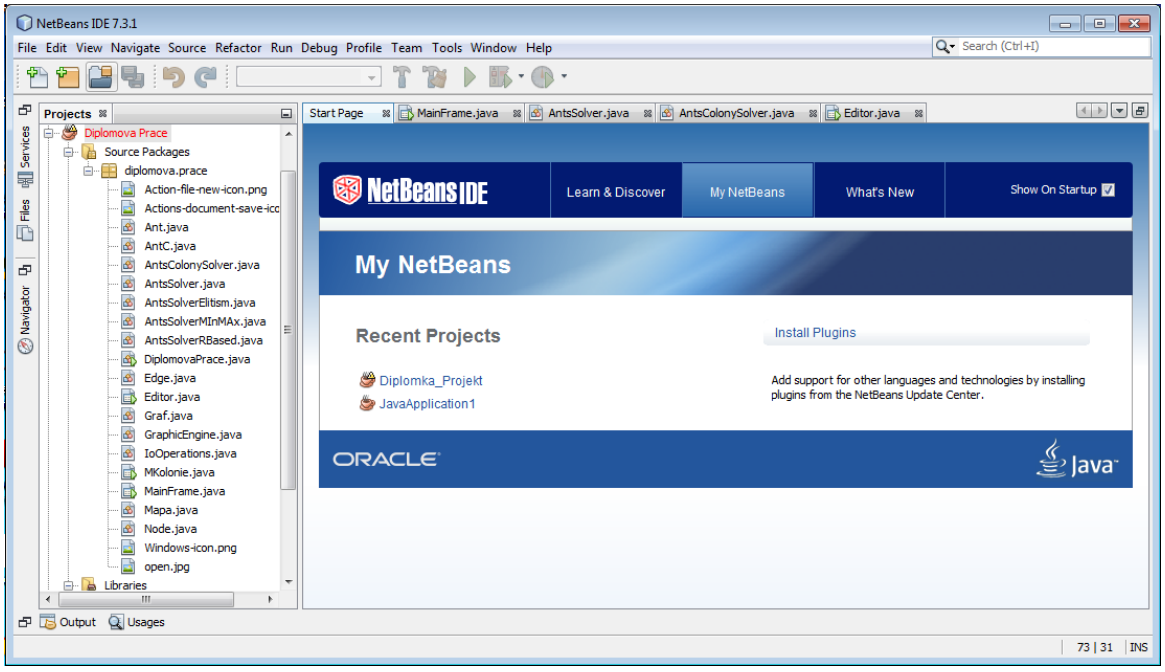
- Java Standart Edition (Java SE) - Java pro aplikace stolních počítačů,
- Java Enterprise Edition (Java EE) - Java pro podnikové a rozsáhlé informační systémy,
- Java Me - Java pro aplikace provozované na mobilních zařízeních (mobilní telefony, PDA, aj.).

4.3 Vývojová prostředí jazyka Java

Existuje několik vývojových prostředí. Odlišují se navzájem svojí *licencí* (Open-source software, komerční software), *typem vykreslování prostředí* (Swing, SWT), *nástroji* a *pluginy* (moduly pro jiné programovací jazyky, moduly pro testování aj.).

Některá nejpoužívanější vývojová prostředí:

- NetBeans,
- Eclipse,
- BlueJ,
- IntelliJ IDEA.

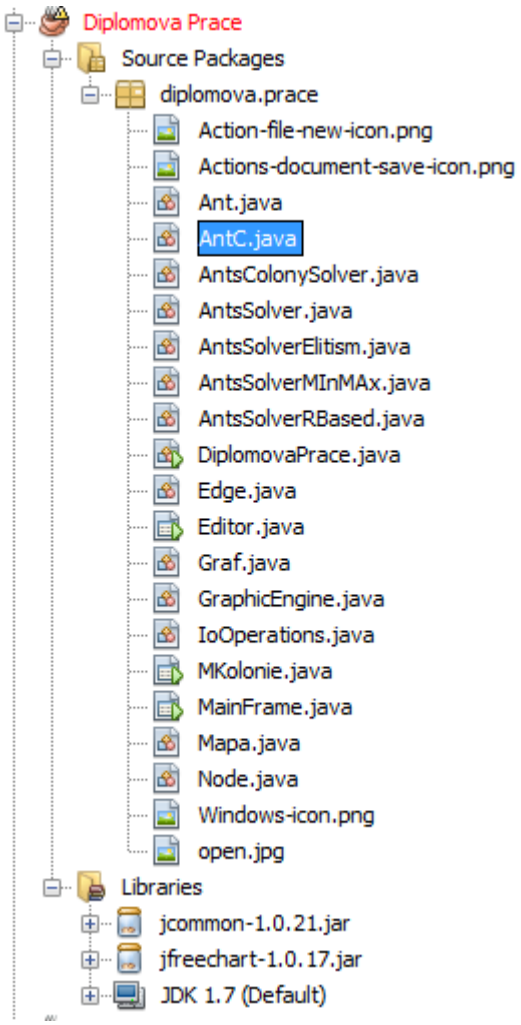


Obr. 6 Ilustrace vývojového prostředí Netbeans

5. IMPLEMENTACE V JAZYCE JAVA

Pro naprogramování aplikace, bylo použito vývojového prostředí NetBeans. Mezi hlavní třídy náleží *Mapa.java*, která datově reprezentuje zadané překážky, souřadnice startu, cíle a metody pro práci s mapou. *GraphicEngine.java* třída obsahuje metody pro grafický výstup/vstup mapy a výsledných cest. Dále třídy *Ant.java* a *AntC.java* tyto třídy zajišťují pohyb, vyhodnoceni cesty apod. Třídy *AntsSolver.java*, *AntsColonySolver.java*, *AntsSolverElitism.java*, *AntsSolverMaxMin.java*, *AntsSolverRBased.java* zajišťují řešení daného problému. Třídy *Main.java*, *MainFrame.java*, *EditorFrame.java* a *OptionsFrame.java* zajišťují komunikaci mezi řešiči, zobrazováním a uživatelem.

5.1 Rozložení tříd



Obr. 7 Strom tříd programu.

5.2 Implementace algoritmů

5.2.1 Třída Ant

Třída *Ant* implementuje mravence, jeho chování a hledání cesty. Základními metodou je `RunAnt`. Tato metoda vyhledává cestu, ze startovní pozice do cílové pozice. Výslednou nalezenou cestu pak ohodnotí pomocí `CalcuateFinalRoute` pomocí metody `CheckRoutes` a `ChooseRoute` se starají o kontrolu a volbu cest. Podrobnější popis metod je uveden v tab. 1.

Tab. 1 Vybrané metody třídy Ant.java a jejich popis.

Metoda	`Ant(int _alfa, int _beta, int _reset)`
Parametry	`_alfa, _beta, _reset` – Hodnoty klíčových parametrů AS.
Popis	Konstruktor třídy – vytváří instance třídy s danými hodnotami parametrů.
Metoda	`RunAnt()`
Parametry	-
Popis	Metoda, která vyhledává cestu mravence. Nejprve nastaví startovní pozici mravence. A následně postupně volá metody pro nalezení možných cest, kontrolu a posunu do další pozice. Použitou cestu ukládá do paměti mravence. V případě, že mravenec uvázne je proveden reset mravence. Počet resetů mravence je omezen parametrem `_reset`.
Metoda	`CheckRoutes(List<> _routes)`
Parametry	`List<> _routes` – list obsahující hrany k vyberu.
Popis	Metoda kontroluje a odstraňuje cesty, které byly již použity při konstrukci celkové cesty mravence, vrací cesty vhodné k výběru.
Metoda	`ChooseRoute(List<> _posibleRoutes)`
Parametry	`List<> _posibleRoutes` – List možných cest k výběru.
Popis	Metoda provádí výpočet pravděpodobností výběru cest. Následně pak vybere cestu podle velikosti vypočtené pravděpodobnosti.
Metoda	`CalcuateFinalRoute()`
Parametry	-
Popis	Vypočte celkovou cestu mravence.

Metoda	`ResetAnt()`
Parametry	-
Popis	Resetuje mravence, jeho paměť a pozici.

5.2.2 Třída AntsSolver

Třída *AntsSolver* třída implementuje ant system. Inicializuje jednotlivé mravence dle parametrů. Spouští vyhledávání cest mravenci, jejich reset. Taktéž aplikuje feromonovou stopu na mapu. Podrobnější popis metod v tab. 2.

Tab. 2 Vybrané metody třídy AntsSolver.java a jejich popis.

Metoda	`AntsSolver(int _alfa, int _beta, _numberOfAnts, int _reset,double _evaporation, double _foromone)`
Parametry	`_alfa,_beta, _numberOfAnts, _reset, _evaporation, _feromone` – Hodnoty klíčových parametrů AS.
Popis	Konstruktor třídy – vytváří instance třídy s danými hodnotami parametrů.
Metoda	`CreateColony()`
Parametry	–
Popis	Vytváří mravenčí kolonii s parametry, nastavuje parametry mravenců a ukládá je do listu mravenců.
Metoda	`SolveProblem()`
Parametry	-
Popis	Metoda řešící daný problém. V každé iteraci volá metody `RunAnts`, `UpdateFeromone`, `ResetAnts`.
Metoda	`RunAnts()`
Parametry	-
Popis	Spouští hledání cesty na každém mravenci.
Metoda	`ResetAnts()`
Parametry	-
Popis	Resetuje každého mravence.

Metoda	`ResultRoute()`
Parametry	–
Popis	Metoda zajišťuje vypočtení konečné cesty.
Metoda	`UpdateFeromone()`
Parametry	–
Popis	Aktualizace feromonové stopy na mapě má dvě fáze. První fází je odpařování, kdy se hodnoty feromonové stopy sníží. Následně je aplikován přírůstek feromonové stopy na cestách mravenců.

5.2.3 Třída AntSolverElitism

Třída implementuje ant elitism systém. Tato třída je rozšířením třídy *AntsSolver*. Pozměňuje metodu `UpdateFeromone`, jelikož je rozdílné nanášení feromonové stopy. Ostatní metody jako je například *`CreateColony`*, *`SolveProblem`*, *`RunAnts`*, *`ResetAnts`*, *`ResultRoute`* jsou stejné, jako u třídy *AntsSolver*. Podrobnější popis metod se nachází v tab. 3.

Tab. 3 Vybrané metody třídy AntsSolverElitism.java a jejich popis.

Metoda	`AntsSolverElitism(int _alfa, int beta, double _weight, int _reset, int _numberOfAnts, double _evaporation, double _foromone)`
Parametry	`_alfa, _beta, _numberOfAnts, _reset, _weight, _evaporation, _feromone` – Hodnoty klíčových parametrů EAS.
Popis	Konstruktor třídy – vytváří instance třídy s danými hodnotami parametrů.
Metoda	`UpdateFeromone()`
Parametry	–
Popis	Metoda aktualizuje feromonovou stopu. Prvně sníží hodnotu feromonové stopy na mapě a následně vybere pomocí metody *`GetEliteAnt`* nejlepšího mravence na jehož cestě aplikuje přírůstek feromonové stopy.

Metoda	`GetEliteAnt(List <> _listOfAnts)`
Parametry	`_listOfAnts` – list mravenců.
Popis	Metoda porovná cesty všech mravenců z `_listOfAnts`, vybere mravence s nejlepší cestou a porovná cesty s dosud nejlepší nalezenou cestou. Jestliže má mravenec z `_listOfAnts` lepší cestu vrací tohoto mravence. V opačném případě vrací uloženého mravence.

5.2.4 Třída AntsSolverRankBased

Třída *AntsSolverRankBased* implementuje rank-based ant system a liší od třídy *AntsSolverElitism* pozměněnou metodou `SolveProblem`, `UpdateFeromone` a metodou `GetEliteAnts`. Podrobnější popis viz. Tab. 4.

Tab. 4 Vybrané metody třídy AntsSolverRankBased.java a jejich popis.

Metoda	`AntsSolverRankBased(int _alfa, int beta, int _reset, int _numberOfAnts, double _evaporation, double _foromone, double _numberOfAnts)`
Parametry	`_alfa, _beta, _feromone, _reset, _evaporation, _feromone, _numberOfAnts` – Hodnoty klíčových parametrů RBAS.
Popis	Konstruktor třídy – vytváří instance třídy s danými hodnotami parametrů.
Metoda	`UpdateFeromone()`
Parametry	-
Popis	Metoda aktualizuje feromonovou stopu. Prvně sníží hodnotu feromonové stopy na mapě a následně vybere pomocí metody *`GetEliteAnts`* nejlepší mravence na jehož cestě aplikuje přírůstek feromonové stopy.
Metoda	`GetEliteAnts(List <> _listOfAnts)`
Parametry	`_listOfAnts` – list mravenců.
Popis	Metoda porovná cesty všech mravenců z *`_listOfAnts`*, vybere mravence s nejlepší cestou. Vrací počet nejlepších mravenců dle parametru *`_numberOfAnts`*.

5.2.5 Třída AntsSolverMaxMin

Třída AntsSolverMaxMin implementuje max-min ant systém a liší od předchozích metod díky jinému způsobu práce s feromonovou stopou. Metoda `SolveProblem` na začátku svého běhu volá metodu `setMaxFeromone` pro počáteční nastavení feromonové stopy. V tab. 5 se nachází podrobnější popis implementovaných metod.

Tab. 5 Vybrané metody třídy AntsSolverMaxmin.java a jejich popis.

Metoda	`AntsSolverMaxMin(int _alfa, int _beta, double _foromone, int _reset, _numberOfAnts, double _evaporation, double _tauMax, double _tauMin)`
Parametry	`_alfa, _beta, _feromone, _numberOfAnts, _reset, _evaporation, _tauMax , _tauMin` – Hodnoty klíčových parametrů MMAS.
Popis	Konstruktor třídy – vytváří instance třídy s danými hodnotami parametrů.
Metoda	`SolveProblem()`
Parametry	-
Popis	Metoda řešící daný problém. V každé iteraci volá metody *`RunAnts`*, *`UpdateFeromone`*, *`ResetAnts`*.
Metoda	`setMaxFeromone()`
Parametry	-
Popis	Nastavuje maximální hodnotu feromonové stopy podle parametru `_tauMax`.
Metoda	`UpdateFeromone()`
Parametry	-
Popis	Provádí aktualizaci feromonové stopy. Provádí odpařování a nanášení feromonové stopy, ale ta nesmí překročit hodnotu `_tauMax`.

5.2.6 Třída AntsColonySolver

Tato třída se liší způsobem *aktualizace feromonové stopy*. Tuto aktualizaci neprovádí pouze třída *AntsColonySolver*, ale průběžně i jednotlivý mravenci. Popis jednotlivých implementovaných metod viz tab. 6.

Tab. 6 Vybrané metody třídy AntsColonySolver.java a jejich popis.

Metoda	`AntsColonySolver(int _alfa, int beta, double _evaporation, _numberOfAnts, int _reset, double _foromone, double _q, double _ksi)`
Parametry	`_alfa, _beta, _numberOfAnts _reset, _evaporation, _feromone, _q, _ksi` – Hodnoty klíčových parametrů ACO.
Popis	Konstruktor třídy – vytváří instance třídy s danými hodnotami parametrů.
Metoda	`CreateColony()`
Parametry	–
Popis	Vytváří mravenčí kolonii s parametry, nastavuje parametry mravenců a ukládá je do listu mravenců. Tito mravenci se vytváří ze třídy `AntC`.
Metoda	`SolveProblem()`
Parametry	-
Popis	Metoda řešící daný problém. V každé iteraci volá metody `RunAnts, UpdateFeromone, ResetAnts.`
Metoda	`RunAnts()`
Parametry	-
Popis	Spouští hledání cesty na každém mravenci třídy *AntC*.
Metoda	`ResetAnts()`
Parametry	-
Popis	Resetuje každého mravence.
Metoda	`ResultRoute()`
Parametry	–
Popis	Metoda zajišťuje vypočtení konečné cesty.

Metoda	`UpdateFeromone()`
Parametry	-
Popis	Aktualizace feromonové stopy na mapě má dvě fáze. První fází je odpařování, kdy se hodnoty feromonové stopy sníží. Následně je aplikován přírůstek feromonové stopy na cestách mravenců.

5.2.7 Třída AntC

Popis jednotlivých implementovaných metod viz tab. 7.

Tab. 7 Vybrané metody třídy AntsC.java a jejich popis.

Metoda	`AntC(int _alfa, int beta, double _q, double _ksi)`
Parametry	`_alfa, _beta, double _q, _ksi` – hodnoty parametrů
Popis	Konstruktor třídy – vytváří instance třídy s danými hodnotami parametrů.
Metoda	`RunAnt()`
Parametry	-
Popis	Metoda, která vyhledává cestu mravence. Nejprve nastaví startovní pozici mravence. A následně postupně volá metody pro nalezení možných cest, kontrolu a posunu do další pozice. V případě, že mravenec nalezne cestou rovnou aktualizuje feromonovou stopu ostatním mravencům. Použitou cestu ukládá do paměti mravence. V případě, že mravenec uvázne je proveden reset mravence. Počet resetů mravence je omezen parametrem *_reset.*
Metoda	`CheckRoutes(List<> _routes)`
Parametry	`List<> _routes` – list obsahující možné cesty k výběru.
Popis	Metoda kontroluje a odstraňuje cesty, které byly již použity při konstrukci celkové cesty mravence, vrací cesty vhodné k výběru.
Metoda	`ChooseRoute(List<> _posibleRoutes)`
Parametry	`List<> _posibleRoutes` – List možných cest k výběru.
Popis	Metoda provádí výpočet pravděpodobností výběru cest. Následně pak vybere cestu podle velikosti vypočtené pravděpodobnosti.

Metoda	`CalcuateFinalRoute()`
Parametry	-
Popis	Vypočte celkovou cestu mravence
Metoda	`RunAnt()`
Parametry	–
Popis	Metoda, která vyhledává cestu mravence. Nejprve nastaví startovní pozici mravence. A následně postupně volá metody pro nalezení možných cest, kontrolu a posunu do další pozice. Použitou cestu ukládá do paměti mravence. V případě, že mravenec uvázne je proveden reset mravence. Počet resetů mravence je omezen parametrem *_reset.*
Metoda	`CheckRoutes(List<> _routes)`
Parametry	`List<> _routes` – list obsahující možné cesty k výběru.
Popis	Metoda kontroluje a odstraňuje cesty, které byly již použity při konstrukci celkové cesty mravence, vrací cesty vhodné k výběru.
Metoda	`ChooseRoute(List<> _posibleRoutes)`
Parametry	`List<> _posibleRoutes` – List možných cest k výběru.
Popis	Metoda provádí výpočet pravděpodobností výběru cest. Následně pak vybere cestu podle velikosti vypočtené pravděpodobnosti.
Metoda	`CalcuateFinalRoute()`
Parametry	-
Popis	Vypočte celkovou cestu mravence.
Metoda	`UpdateFeromoneToAllAnts()`
Parametry	–
Popis	Aktualizuje feromonovou stopu ostatním mravencům

5.3 Implementace prostoru

5.3.1 Třída Mapa

Obsahuje položky pro ukládání *překážek*, *startu* a *cíle*, *textových názvů překážek*, *pseudo3D mapy*. Hlavními metodami jsou `addObstacle`, `removeObstacle` které se starají o přidávání a odebíraní překážek. Dále `expandObstacles`, která se stará o výpočet bodů překážek. `ParseMap`. Popis metod se nachází v tab. 8.

Tab. 8 Výběr metod třídy Mapa.java a jejich popis.

Metoda	`Mapa()`
Parametry	-
Popis	Konstruktor třídy – vytváří instance třídy.
Metoda	`setStart (int _x, int _y)`
Parametry	`_x, _y` – hodnoty parametrů
Popis	Nastavuje startovní pozici na hodnoty parametrů `_x, _y`.
Metoda	`setGoal (int _x, int _y)`
Parametry	`_x, _y` – hodnoty parametrů
Popis	Nastavuje cílovou pozici na hodnoty parametrů `_x, _y`.
Metoda	`addObstacle(List<Bod> _obstacle)`
Parametry	`_obstacle` – list s body překážky
Popis	Přidává překážku do seznamu překážek.
Metoda	`removeObstacle(int _obstacle)`
Parametry	`_obstacle` – index překážky
Popis	Odebírá překážku do seznamu překážek.
Metoda	`ExpandObstacles()`
Parametry	-
Popis	Metoda vypočítává body všech překážek.

Metoda	`setPseudo3D(Bod _bod, int _vyska)`
Parametry	`Bod _bod, int _indexVysky` – parametry.
Popis	Nastavuje výšku v daném kvadrantu.
Metoda	`ParseMap()`
Parametry	–
Popis	Zpracovává mapu k jejímu využití jinými třídami.

5.4 Implementace zobrazování

5.4.1 Třída GraphicEngine

Obsahuje metody pro vykreslení mapy. Dále slouží jako instance poskytující grafický kontext jednotlivých panelům hlavní aplikace a editorům. Mezi hlavní metody patří `Repaint` starající se o překreslování scény aktuální mapou. Dále metody `DrawStart` a `DrawCil` vykreslují start a cíl. Metoda `PaintPseudo3D` vykresluje pseudo 3D mapu. A metoda `DrawAntRoute` vykresluje cestu mravence. Popis metod viz Tab. 9.

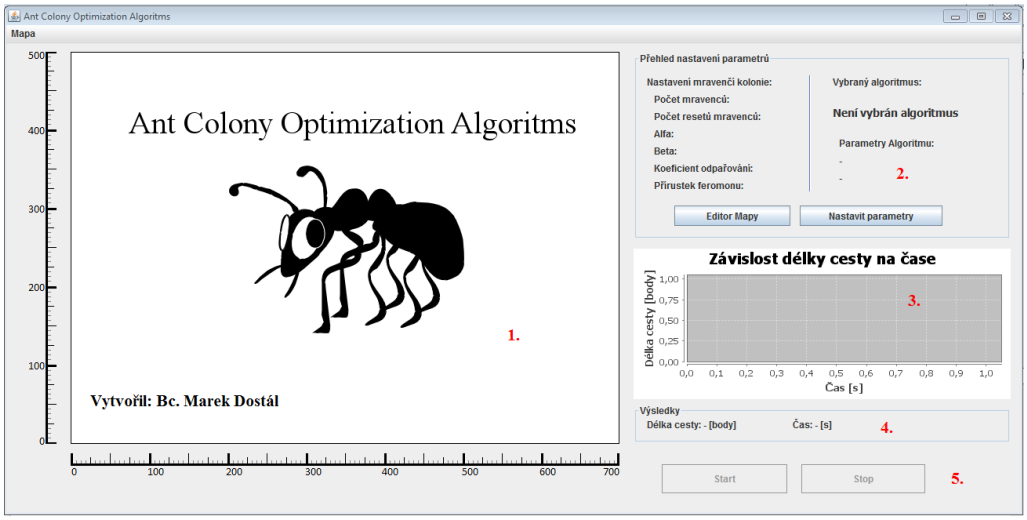
Tab. 9 Vybrané metody třídy GraphicEngine.java a jejich popis.

Metoda	`GraphicEngine(Mapa _mapa,Graphics _graphics)`
Parametry	`Mapa _mapa, Graphics _graphics`
Popis	Konstruktor třídy – vytváří instance třídy s danými hodnotami parametrů.
Metoda	`Repaint()`
Parametry	–
Popis	Metoda překresluje plochu aktuální mapou.
Metoda	`paintPseudo3D(boolean _color)`
Parametry	`boolean _color` – hodnota určující způsob vykreslení pseudo 3D mapy.
Popis	Metoda vykreslující pseudo 3D mapu.
Metoda	`ClipToImage()`
Parametry	-
Popis	Metoda vytváří ze scény obrázek k rychlejšímu vykreslování scén.

Metoda	`ClipToImage3D()`
Parametry	-
Popis	Metoda vytváří ze scény s pseudo 3D mapou obrázek k rychlejšímu vykreslování scény.
Metoda	`DrawStart()`
Parametry	–
Popis	Vykresluje na scénu start.
Metoda	`DrawGoal()`
Parametry	–
Popis	Vykresluje na scénu cíl.
Metoda	`DrawAntRoute()`
Parametry	–
Popis	Vykresluje na scénu cestu mravence.

5.5 Uživatelské rozhraní

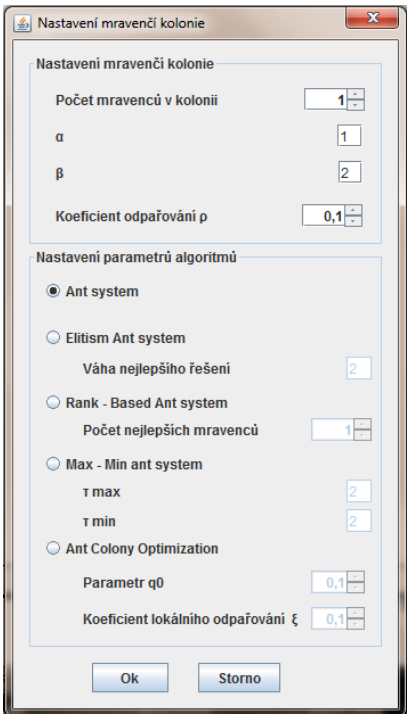
Celá aplikace se skládá ze tří oken. Hlavní okno slouží pro sledování běhu algoritmu na mapě a zobrazení výsledků v podobě grafu závislosti délky nalezené cesty na čase výpočtu. Z hlavního okna lze vyvolat okno editoru pseudo 3D prostředí a editoru překážek, toto okno v podstatě slouží k návrhu "světa" ve kterém bude probíhat optimalizace. Třetím oknem, rovněž aktivovaným z hlavního okna, je okno umožňující nastavení parametrů optimalizace, tj. volbu a vlastní parametrizaci zvolených mravenčích strategii.



Obr. 8 Hlavní okno aplikace.

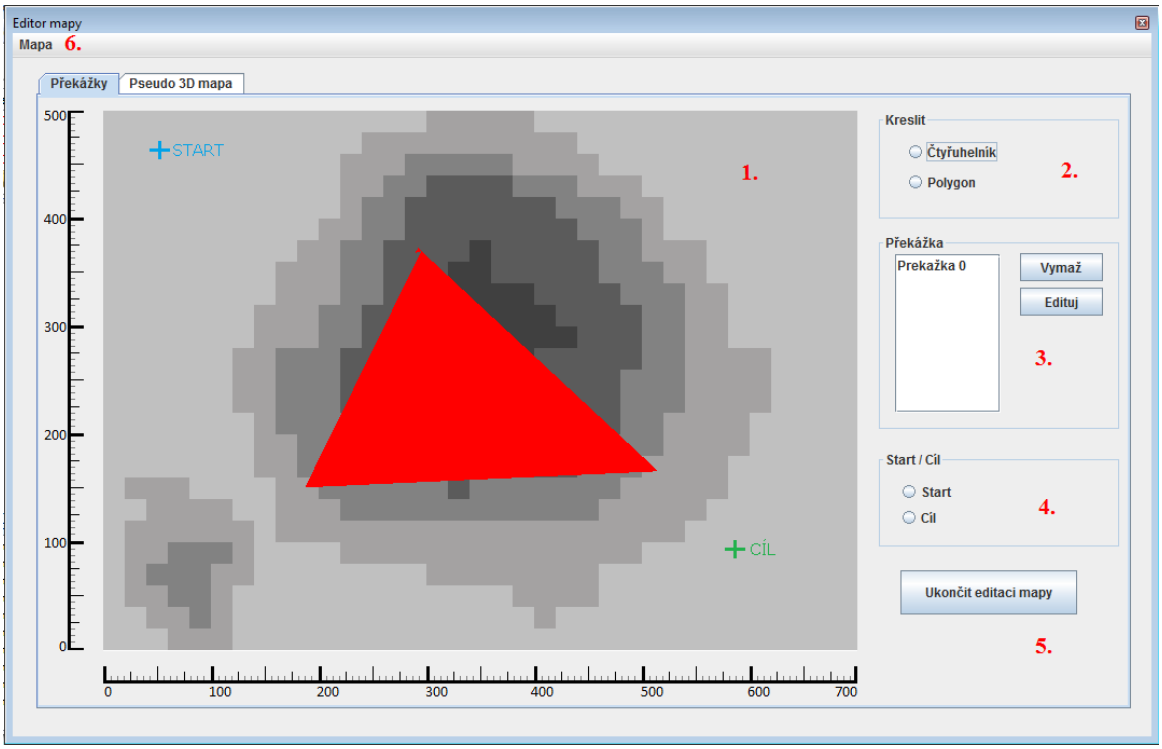
Popis hlavního okna aplikace:

1. Panel se zobrazením mapy a cest mravenců.
2. Informační panel s přehledem vybraných parametrů mravenčí kolonie a algoritmu.
3. Graf závislosti délky cesty v čase.
4. Panel s výsledky.
5. Tlačítka pro start a stop běhu algoritmů.



Obr. 9 Okno nastavení parametrů mravenčí kolonie.

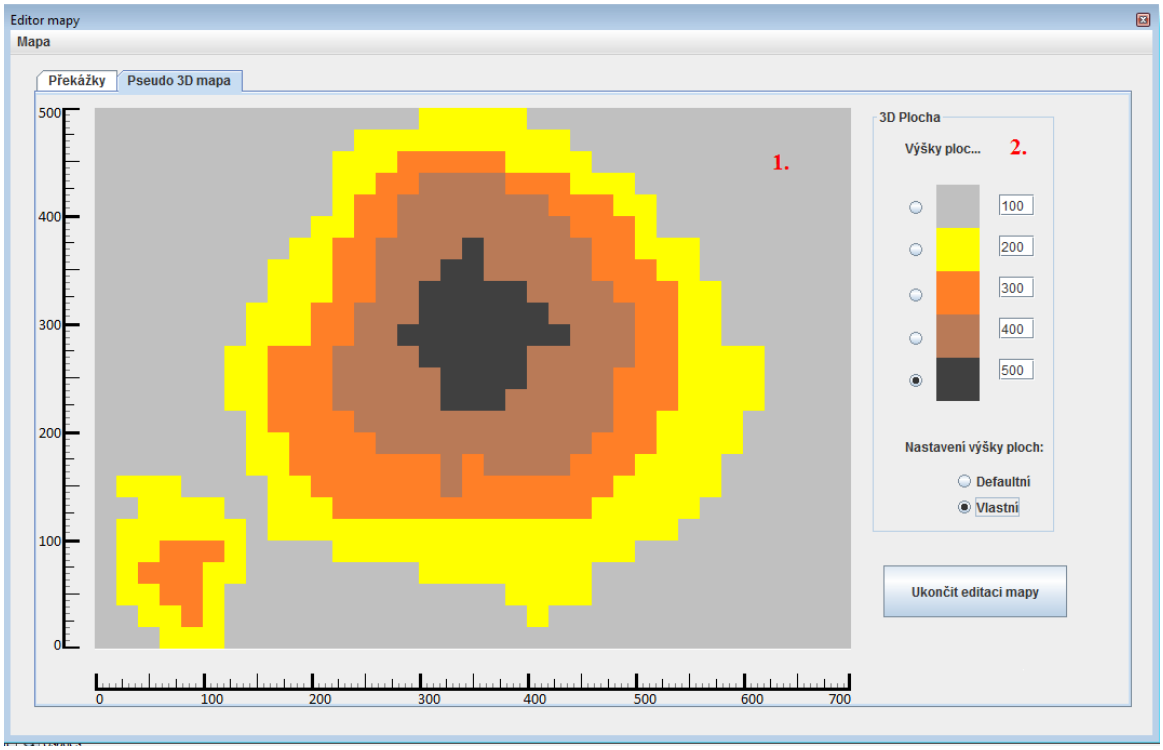
5.5.1 Popis editoru překážek a pseudo 3D prostoru



Obr. 10 Editor pro vytváření překážek.

Popis editoru překážek:

1. Panel se zobrazením překážek, startu a cíle.
2. Panel pro kreslení překážek typu čtyřúhelníku a polygonu.
3. Panel pro editaci překážek.
4. Panel pro nastavení startu / cíle.
5. Tlačítko pro pokračování na hlavní okno.



Obr. 11 Panel s editací Pseudo 3D mapy.

3d prostor byl nahrazen pseudo 3d, který byl implementován v 5ti úrovních, které v rámci výpočtu cesty znamenají jiný výpočet dle úrovně terénu. Tyto úrovně si uživatel může zvolit. A pak jsou zde nepřekonatelné překážky, jako druhý krok tvorby mravenčího světa.

Popis editoru Pseudo 3D mapy:

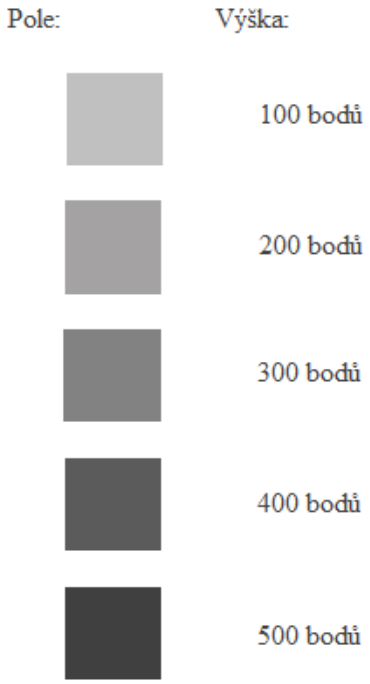
1. Panel se zobrazením pseudo 3D prostoru.
2. Panel pro kreslení a nastavení výšek pseudo 3D prostoru.

6. VÝSLEDKY EXPERIMENTŮ

K porovnání jednotlivých mravenčích algoritmů, je vhodné znát jejich nejlepší možné nastavení. Tento úkol je pochopitelně vázaný na danou mapu prostředí. Z tohoto důvodu byly vytvořeny 4 rozdílné mapy prostředí, které vhodným způsobem ilustrují rozličné obtížnosti a charakteristiky prohledávaného prostoru. Z další části práce je patrná povaha těchto 4 testovacích úloh. V rámci optimalizace parametrů mravenčích algoritmů je vhodné uvést, že mnohé parametry jsou ve významu shodné pro všechny prezentované mravenčí strategie. Pochopitelně mohou tyto parametry obecně dle dané strategie vykazovat rozdílná optima. Dále byly shodně voleny parametry α, β, ρ a počet mravenců N a testován výkon jednotlivých strategii. Každá z testovaných variant běžela 300 iterací daného algoritmu. Testována byla škála N={10,50,100} mravenců v rámci každé strategie. Testovaná parametrizace α a β byla na množině hodnot {1,2,3,4,5} a následná hodnota ρ byla parametrizována na množině hodnot {0.1,0.2,...0.9}. Pro statistickou objektivitu výsledku byl každý iterační běh zopakován 100x. Vítězné parametrizace byly poté rozšířeny o dalších 100 běhů a vzájemně porovnány. Získané výsledky jsou tedy založeny na dostatečném, statisticky významném počtu měření.

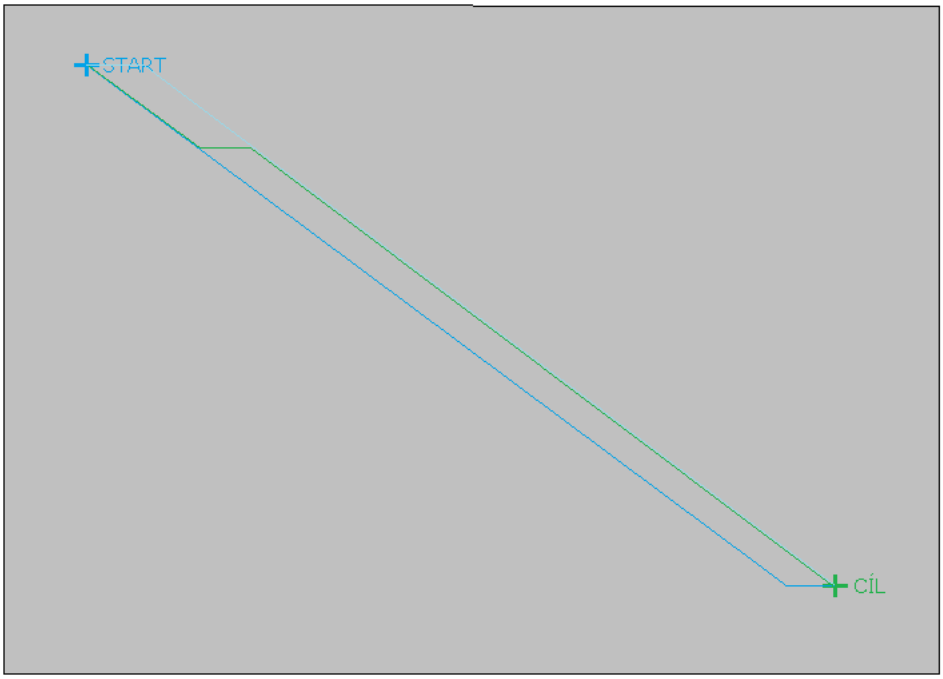
6.1 Přehled testovaných map

Každá mapa je tvořena základní pseudo 3D plochou, tj. terénem, překážkami a definovaným startem a cílem. Plocha mapy byla volena v matici 700x500 bodů, ale případná modifikace je pochopitelně programově možná. Pseudo 3D predstavuje na dané oblasti diskretizaci na mřížku 35x25 políček, přičemž u každého políčka je možné definovat výšku v příslušném rozsahu. Základní nastavení výšky terénu je následující:

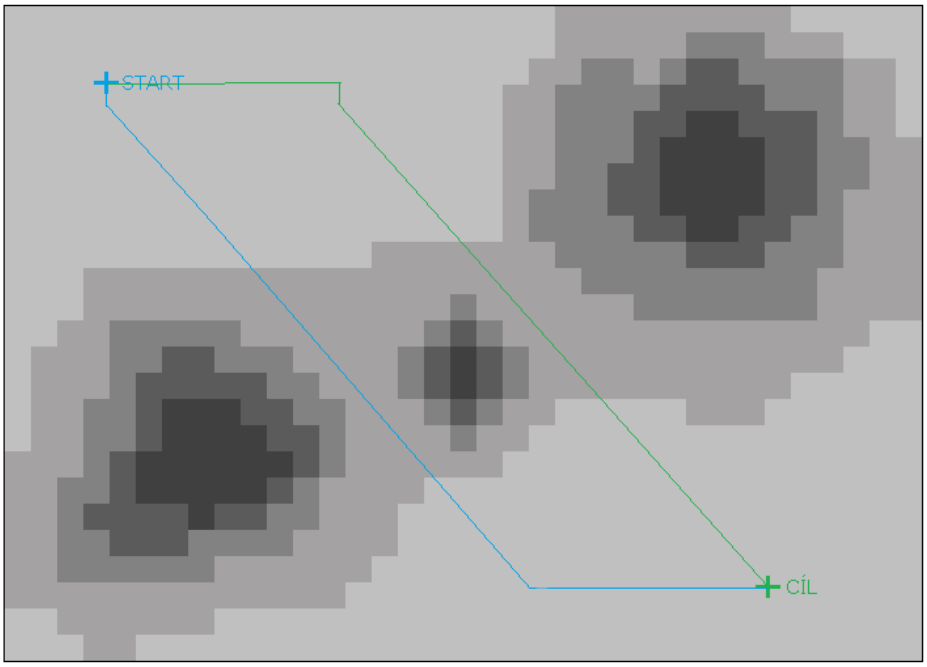


Obr. 12 Nastavení výšek jednotlivých polí na pseudo 3D ploše.

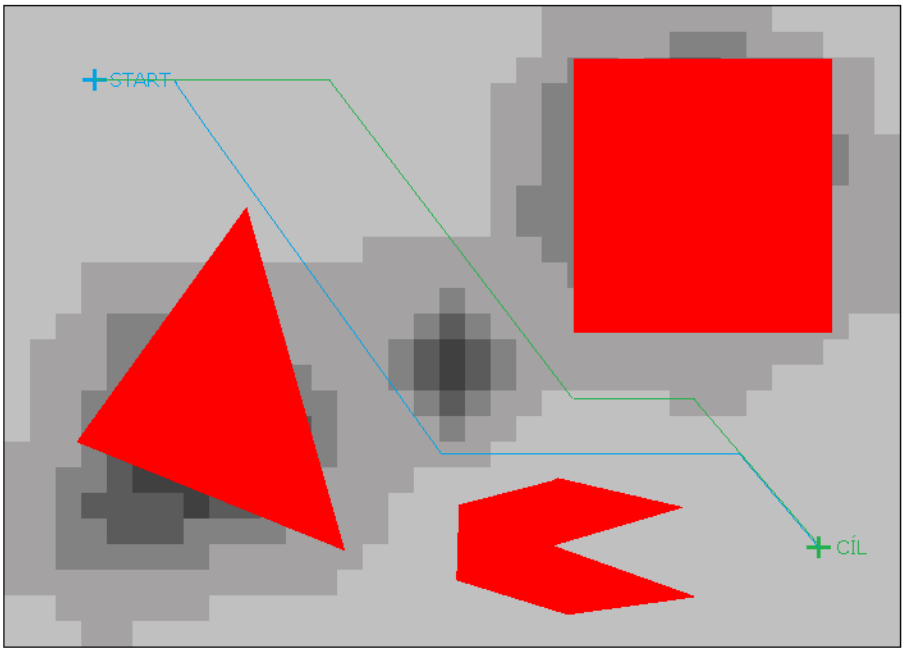
Po-té co je mapa doeditována, program mapu převede do diskrétních kroků a následně převedena na grafovou strukturu. Kde délka hrany je vypočítána, jako přímá vzdálenost mezi vrcholy vytvořeného grafu, které mají svojí z souřadnici.



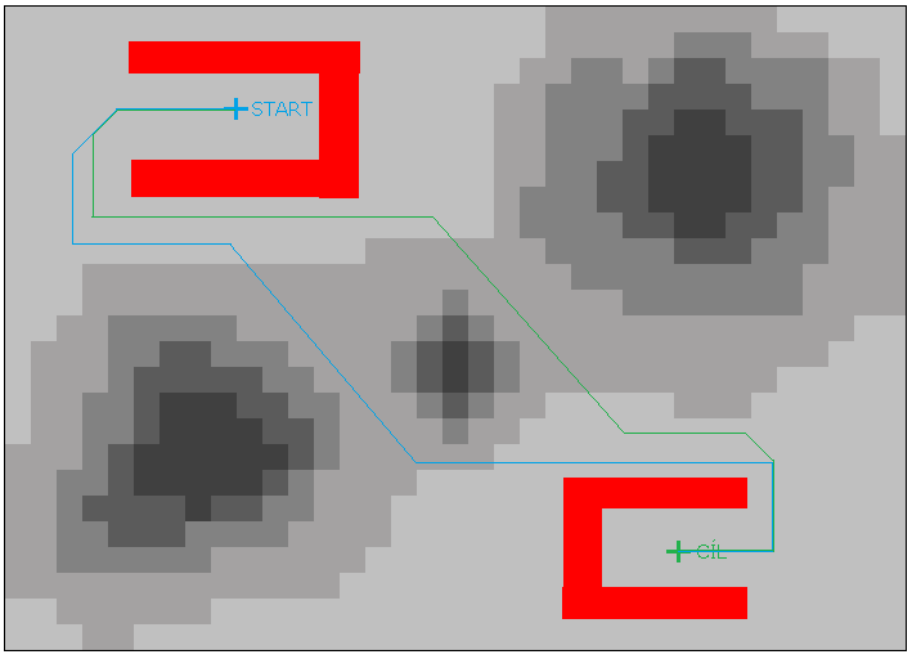
Obr. 13 Ukázka mapy FLAT s nalezenými cestami.



Obr. 14 Ukázka mapy PSEUDO_3D s nalezenými cestami.



Obr. 15 Ukázka mapy OBSTACLES s nalezenými cestami.



Obr. 16 Ukázka mapy TRAP s nalezenými cestami.

6.2 Určení vhodných parametrů AS

Jednotlivé parametry:

- α, β – exponenty určující důležitost feromonové stopy a heuristické informace při výpočtu pravděpodobnosti cesty podle rovnice (2), byla testována na množině hodnot{1,2,3,4,5},
- ρ – koeficient odpařování, ρ byla parametrizována na množině hodnot {0.1,0.2,...0.9},
- N – počet jednotlivých mravenců, hledajících řešení, N={10,50,100}.

Pro testování parametrů α, β bylo použito nastavení nejvhodnějšího koeficientu odpařování podle literatury [10], kde $\rho = 0{,}5$.

Tab. 10 Průměrná délka cesty pro nastavení α, β pro N = 10 mravenců na mapě FLAT.

α / β	1	2	3	4	5
1	2201	1275	2295	2301	2560
2	1023	2156	1250	2135	2365
3	2295	1303	2293	1420	2001
4	2635	1963	1420	2036	1485
5	3021	2903	1823	1389	2096

Tab. 11 Průměrná délka cesty pro nastavení α, β pro N = 50 mravenců na mapě FLAT.

α / β	1	2	3	4	5
1	1952	1023	1562	2023	2123
2	996	1856	1023	1852	1756
3	1096	1130	2132	1430	1326
4	1345	1452	1320	2013	1200
5	1963	1899	1689	1320	1996

Tab. 12 Průměrná délka cesty pro nastavení α, β pro N = 100 mravenců na mapě FLAT.

α / β	1	2	3	4	5
1	1635	952	1230	1520	1852
2	895	1596	1098	1350	1789
3	1027	1123	1723	1698	1521
4	1398	1536	1635	1568	1103
5	1852	1785	1623	1482	1652

Při porovnání výsledků pro nastavení počtu mravenců N, lze vidět, že délka výsledného řešení závisí na počtu mravenců a to tak, že při vyšší hodnotě N je dosahováno lepšího řešení. Pro další testování bude použito nastavení $N = 100$ mravenců.

Tab. 13 Průměrná délka cesty pro nastavení α, β pro N = 100 mravenců na mapě PSEUDO_3D.

α / β	1	2	3	4	5
1	1457	1120	1362	1456	1568
2	1007	1542	1201	1345	1565
3	1230	1351	1578	1644	1542
4	1345	1455	1358	1498	1485
5	1523	1422	1399	1287	1604

.

Tab. 14 Průměrná délka cesty pro nastavení α, β pro N = 100 mravenců na mapě OBSTACLES.

α / β	1	2	3	4	5
1	1687	1453	1587	1687	1898
2	1257	1752	1599	1570	1736
3	1358	1568	1698	1432	1625
4	1485	1687	1785	1698	1544
5	1677	1798	1835	1766	1785

Tab. 15 Průměrná délka cesty pro nastavení α, β pro N = 100 mravenců na mapě TRAP.

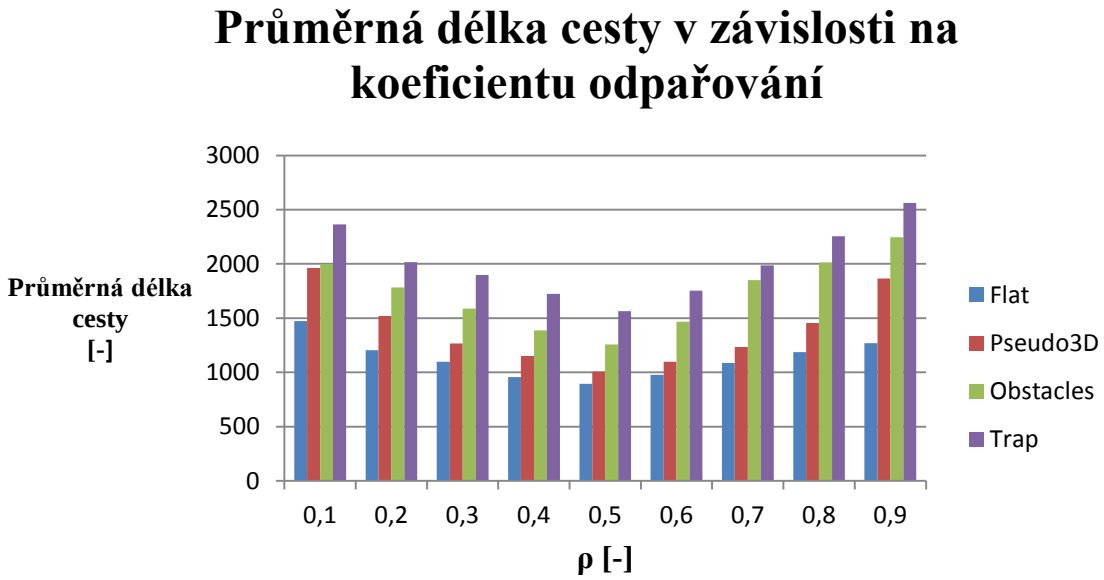
α / β	1	2	3	4	5
1	2235	1687	1899	1988	2240
2	1523	2452	1752	1842	2110
3	1782	1866	2368	1736	2035
4	1820	1987	1866	2458	1927
5	1968	2100	1987	1702	2556

Z tabulek testování exponentů *α, β* je patrné, že nejlepších výsledků bylo dosaženo při nastavení exponentů $\alpha = 1, \beta = 2$ a počtu mravenců $N = 100$.

Zjištěné nastavení exponentů *α, β,* bylo využito k testování koeficientu odpařování ρ. Kde $\rho = (0,1)$.

Tab. 16 Průměrná délka cesty pro nastavení ρ pro N = 100 mravenců.

ρ	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
Flat	1472	1203	1098	957	895	976	1085	1186	1268
Pseudo3D	1963	1520	1267	1150	1008	1098	1235	1456	1866
Obstacles	1998	1783	1587	1387	1257	1468	1852	2013	2245
Trap	2365	2017	1899	1725	1563	1753	1985	2256	2563



Obr. 17 Graf závislosti průměrné délky cesty na velikosti koeficientu odpařování ρ pro N =100 mravenců.

Při nastavení koeficientu odpařování na malou hodnotu vede k rychlejšímu odpařování feromonové stopy a tím dochází prohledávání menšího prostoru s velkou možností uváznutí řešení na neoptimální délce cesty. Naopak vysoká hodnota ρ, způsobí pomalé odpaření feromonové stopy a tím zvýší prohledávaný prostor.

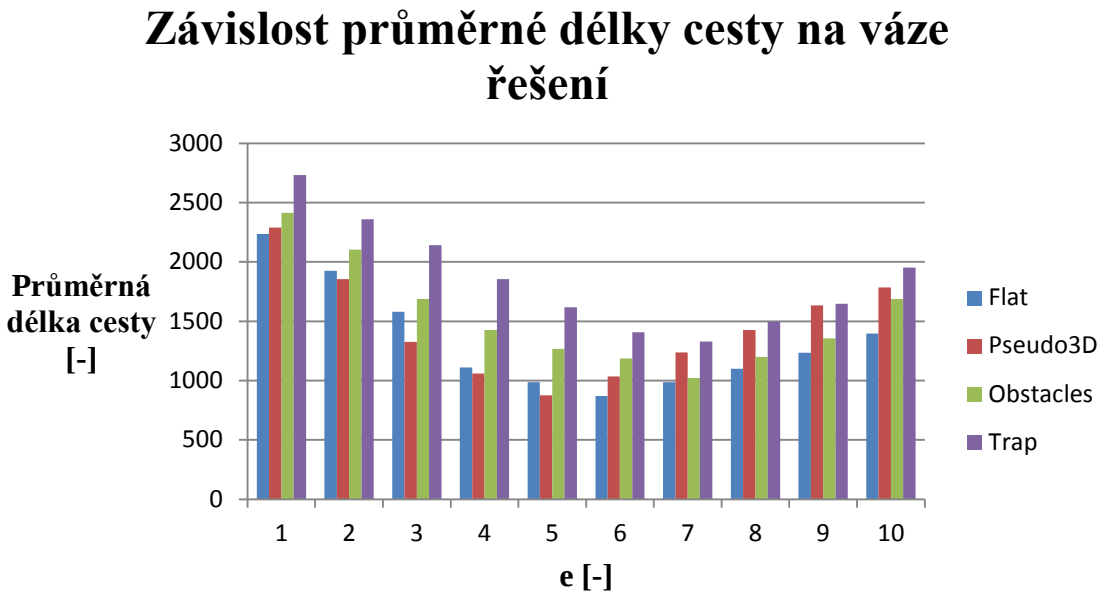
Tyto zjištěné parametry AS, byly použity jako základní nastavení parametrů mravenčí kolonie algoritmů EAS, RBAS, MMAS, ACO.

6.3 Určení vhodných parametrů EAS

Parametr *e*, určuje váhu řešení elitního mravence, který jako jediný nanáší feromonovou stopu. Tato váha pak posiluje feromonovou stopu elitního mravence podle rovnice (8) a (7).

Tab. 17 Průměrná délka cesty pro e při N = 100.

e	1	2	3	4	5	6	7	8	9	10
Flat	2236	1926	1579	1110	985	869	987	1099	1235	1396
Pseudo3D	2289	1854	1326	1058	935	1036	1237	1425	1633	1784
Obstacles	2412	2102	1687	1427	1268	1185	1102	1199	1356	1687
Trap	2732	2358	2141	1854	1618	1526	1428	1496	1647	1952



Obr. 18 Graf závislosti průměrné délky cesty na velikosti e pro N =100 mravenců.

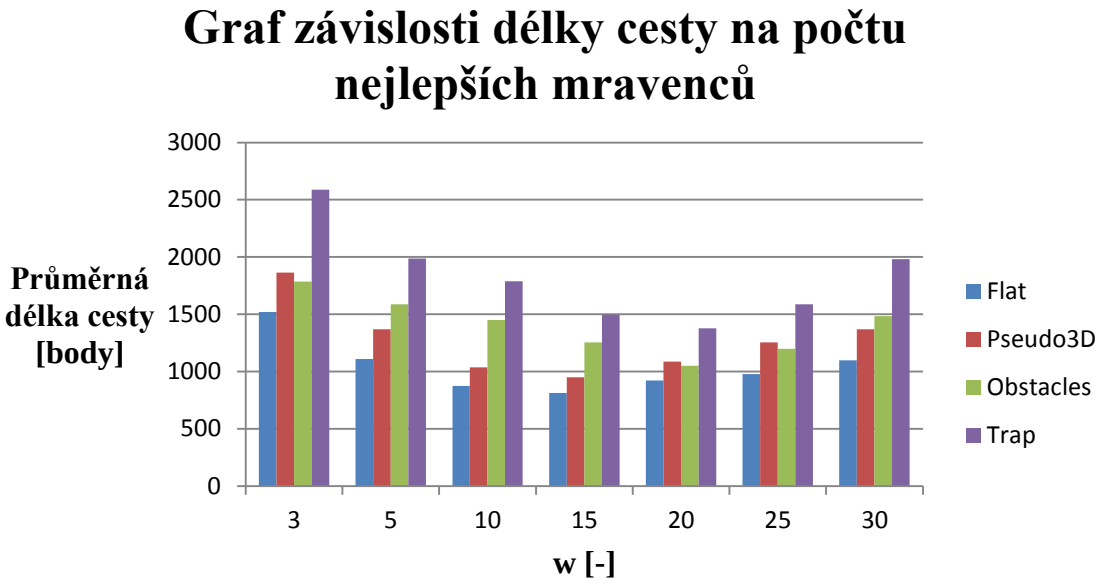
Z výsledků je patrné, že malá hodnota váhy řešení vedla k menšímu zvyšování feromonové stopy elitního mravence a tím způsobila ustálení v neoptimální cestě. Opačná vysoká hodnota vedla k zvýšení feromonové stopy a to zapříčinilo ustálení v této cestě danou feromonovou stopou.

6.4 Určení vhodných parametrů RBAS

Parametr *w* určuje počet nejlepších mravenců, kteří nanášejí feromonovou stopu. Velikost nanášené stopy je dána rovnicemi (9), (10), (11).

Tab. 18 Průměrná délka cesty při N = 100 mravenců.

w	3	5	10	15	20	25	30
Flat	1520	1109	875	815	923	979	1098
Pseudo3D	1865	1369	1038	950	1088	1254	1369
Obstacles	1785	1587	1452	1254	1050	1198	1483
Trap	2587	1987	1789	1498	1379	1587	1982



Obr. 19 Graf závislosti délky cesty na počtu nejlepších mravenců.

Při menším počtu nejlepších mravenců dochází k horším výsledkům, protože mimo nastavený rozsah, mohou být mravenci sice s horší cestou, ale i s potenciálem na lepší řešení. Naproti tomu při větším počtu mravenců se do rozsahu dostanou i mravenci se špatným řešením a tím způsobí ustálení výsledné cesty na neoptimální délce.

6.5 Určení vhodných parametrů MMAS

U tohoto algoritmu je feromonová stopa dána hranicí τ_{max} *a* τ_{min}, kdy je na počátku běhu feromonová stopa inicializována na hodnotu τ_{max}. Při průběhu dochází k odpařování feromonové stopy až na hodnotu τ_{min}. Feromonové stopy na nejlepších cestách, mají pak hodnotu τ_{max}.

K určení hodnoty τ_{max} byl využit podle literatury [09] rovnice:

$$\tau_{\max} = \frac{1}{(\rho \cdot L_{mn})} \tag{16}$$

Kde:

- ρ – koeficient odpařování feromonové stopy,
- L_{mn} – nejkratší vzájemná vzdálenost mezi startovními a cílovými body.

Hodnota τ_{min} byla určena jako podíl hodnoty τ_{max}.

Tab. 19 Průměrná délka cesty pro N = 100 mravenců na mapě FLAT.

τ_{max} / τ_{min}	0,1	0,08	0,06	0,05	0,025	0,015
0,007	1924	1832	1532	1458	1854	1965
0,012	1856	1796	1352	1308	1758	1906
0,025	1752	1632	1108	1287		
0,03	1699	1527	959	1325		
0,04	1796	1789	990	1564		
0,05	1885	1876	1086			

Tab. 20 Průměrná délka cesty pro N = 100 mravenců na mapě PSEUDO3D.

τ_{max} / τ_{min}	0,1	0,08	0,06	0,05	0,025	0,015
0,007	1987	1936	1603	1785	1854	1925
0,012	1789	1725	1132	1602	1768	1895
0,025	1598	1627	950	1236		
0,03	1458	1585	1320	1361		
0,04	1569	1632	1563	1456		
0,05	1665	1782	1703			

Tab. 21 Průměrná délka cesty pro N = 100 mravenců na mapě OBSTACLES.

τ_{max} / τ_{min}	0,1	0,08	0,06	0,05	0,025	0,015
0,007	1927	1885	1236	1287	1374	1496
0,012	1887	1768	1120	1138	1265	1368
0,025	1756	1321	987	1076		
0,03	1352	1084	899	963		
0,04	1185	1117	949	1036		
0,05	1274	1303	1163			

Tab. 22 Průměrná délka cesty pro N = 100 mravenců na mapě TRAP.

τ_{max} / τ_{min}	0,1	0,08	0,06	0,05	0,025	0,015
0,007	2958	2613	1987	1856	1952	2035
0,012	2836	2585	1923	1532	1953	2631
0,025	2246	2234	1786	1130		
0,03	1723	1921	1652	1225		
0,04	1635	1858	1838	1530		
0,05	1596	1785	2236			

Interval mezi τ_{max} a τ_{min} určuje kvalitu nalezeného řešení. Velké a malé rozpětí intervalu způsobuje prohledávání většího prostoru a tím zhoršení výsledné délky cesty.

6.6 Určení vhodných parametrů ACO

ACO algoritmus kombinuje staré a agresivnější pravidlo pro výběr cesty. Výběr pravidla záleží na náhodné hodnotě q, která se porovnává s parametrem q_0.

Aktualizace feromonové stopy probíhá ve dvou fázích. V lokální fázi mravenec aktualizuje feromonovou stopu okamžitě po nalezení cesty. Další mravenci pak mohou reagovat na tuto aktualizovanou feromonovou stopu a to podle rovnice (15). A globální kdy se aktualizuje feromonová stopa mravence s nejlepší nalezenou cestou.

Parametry:

- q_0 – proměnná výběru pravidla $q_0 = (0,1)$,
- ξ – koeficient lokálního odpařování feromonové stopy $\xi = (0,1)$.

Tab. 23 Průměrná délka cesty pro N = 100 mravenců na mapě FLAT.

q_0 / ξ	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
0,1	1254	1223	1120	1235	1250	1352	1523	1785	2123
0,2	1351	1298	1253	1125	1023	1247	1452	1638	2036
0,3	1456	1326	1356	1035	890	1128	1320	1574	1912
0,4	1544	1384	1444	987	845	1057	1248	1453	1854
0,5	1598	1458	1530	1023	936	1123	1124	1485	1743
0,6	1687	1568	1678	1128	1098	1234	1269	1564	1896
0,7	1754	1742	1785	1254	1236	1485	1458	1682	1955
0,8	1895	1854	1799	1456	1458	1789	1785	1736	2058
0,9	1904	1985	1852	1752	1748	1965	1963	1845	2189

Tab. 24 Průměrná délka cesty pro N = 100 mravenců na mapě PSEUDO_3D.

q0 / ξ	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
0,1	1784	1587	1484	1399	1362	1236	1352	1488	1784
0,2	1624	1452	1352	1201	1254	1136	1241	1352	1547
0,3	1585	1365	1247	1154	1102	1025	1103	1241	1455
0,4	1466	1299	1209	1047	1003	936	1052	1125	1354
0,5	1573	1365	1299	1087	978	940	987	1098	1244
0,6	1698	1475	1368	1125	935	925	1069	1125	1358
0,7	1787	1527	1478	1205	1085	998	1174	1235	1574
0,8	1863	1628	1524	1287	1154	1058	1293	1457	1715
0,9	1998	1798	1632	1396	1236	1196	1354	1542	1823

Tab. 25 Průměrná délka cesty pro N = 100 mravenců na mapě OBSTACLES.

q0 / ξ	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
0,1	1874	1652	1518	1478	1532	1687	1699	1789	1934
0,2	1785	1521	1475	1356	1445	1576	1523	1702	1874
0,3	1655	1487	1354	1203	1364	1455	1499	1665	1785
0,4	1544	1447	1244	1123	1254	1401	1405	1582	1720
0,5	1698	1573	1399	968	1123	1503	1485	1499	1754
0,6	1782	1694	1482	1020	1298	1589	1568	1535	1820
0,7	1882	1785	1502	1035	1357	1689	1677	1687	1903
0,8	1977	1844	1687	1124	1465	1721	1789	1799	1994
0,9	2014	1992	1744	1236	1498	1785	1892	1904	2064

Tab. 26 Průměrná délka cesty pro N = 100 mravenců na mapě TRAP.

q0 / ξ	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
0,1	2130	1950	1784	1687	1788	1865	1994	2098	2215
0,2	2036	1895	1602	1554	1701	1801	1874	2010	2198
0,3	1987	1702	1547	1345	1654	1723	1823	1988	2103
0,4	1894	1632	1503	1235	1544	1644	1788	1842	2054
0,5	1966	1751	1468	1150	1674	1785	1879	1906	1984
0,6	2045	1842	1387	1203	1703	1800	1922	1999	2099
0,7	2136	1995	1428	1347	1777	1899	2006	2072	2185
0,8	2354	2014	1498	1458	1822	1922	2109	2136	2354
0,9	2471	2201	1599	1588	1854	2011	2213	2256	2523

Výsledky ukazují, že nejvhodnější velikost parametru q_0 se pohybuje v rozmezí 0,4 až 0,6. Kdy na daných mapách algoritmus má nejlepší výsledky. Koeficient lokálního odpařování feromonové stopy ξ ovlivňuje hledání cest ostatních mravenců. Jeho nastavení na malou hodnotu způsobí, že cesty mravenců se více přibližují těm mravencům, kteří již hledání dokončili a lokálně aktualizovali svoji cestu. Opačné nastavení zapříčiní

větší odpuzování mravenců hledající cesty od cest mravenců, kteří již dokončili hledání a tím zapříčiní ustálení v neoptimální délce cesty.

6.7 Výsledky testování parametrů na mapách

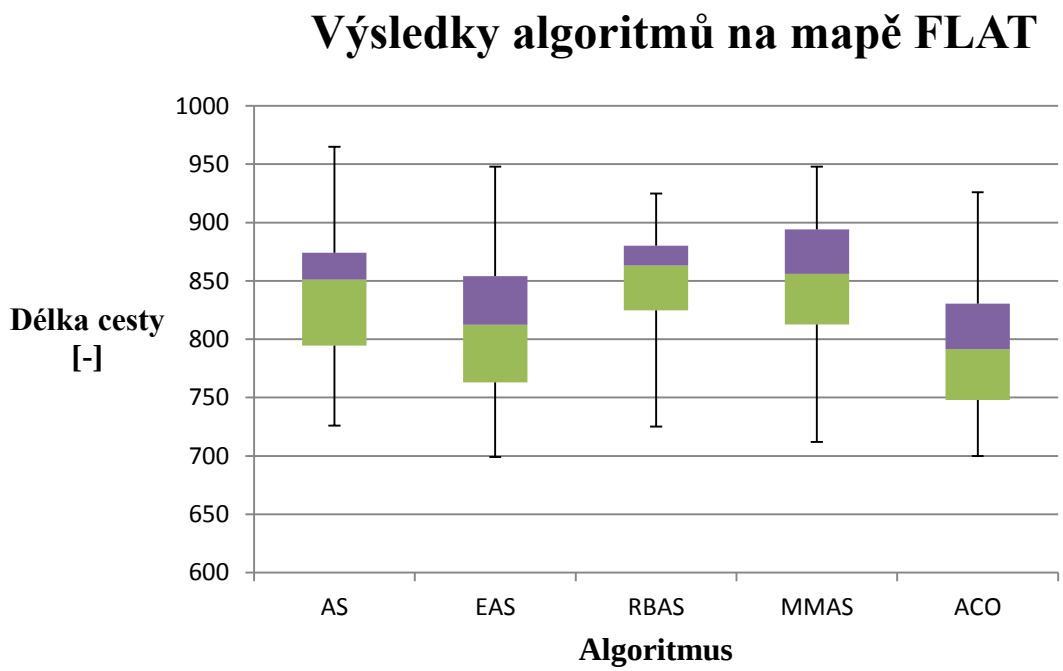
Jednotlivá nastavení se následně testovala, každá po 200 testech na jednotlivých mapách.

6.7.1 Mapa FLAT

Nastavení algoritmů:

Tab. 27 Nastavení algoritmů na mapě FLAT.

Parametry	AS	EAS	RBAS	MMAS	ACO
α	1	1	1	1	1
β	2	2	2	2	2
ρ	0,5	0,5	0,5	0,5	0,5
		$e = 6$	$w = 15$	$\tau_{max} = 0,06$ $\tau_{min} = 0,03$	$q_0 = 0,5$ $\xi = 0,4$



Obr. 20 Graf výsledků algoritmů na mapě FLAT.

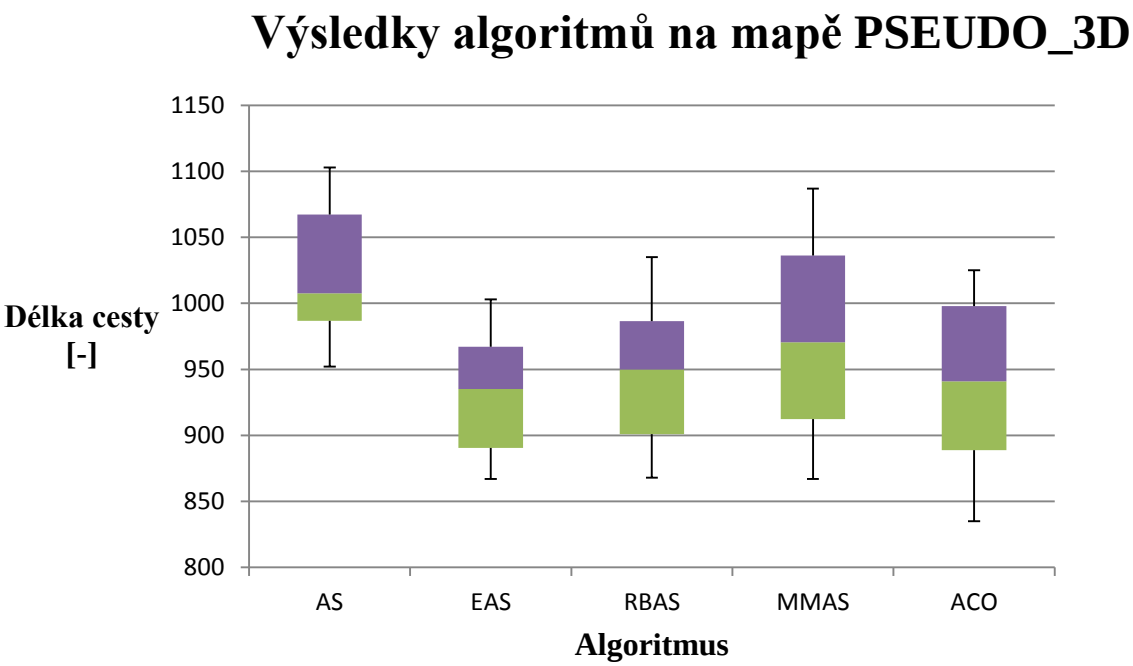
Výsledky algoritmů na mapě FLAT jsou přibližně stejné. Nepatrně lepší výsledky poskytují EAS a ACO. Poměrně stejné výsledky, lze přičíst k topografii mapy, která je dána plochým pseudo 3D prostředím bez překážek, kde prohledávání většího prostoru nevede k lepším výsledkům.

6.7.2 Mapa PSEUDO_3D

Nastavení parametrů:

Tab. 28 Nastavení algoritmů na mapě PSEUDO_3D.

Parametry	AS	EAS	RBAS	MMAS	ACO
α	1	1	1	1	1
β	2	2	2	2	2
ρ	0,5	0,5	0,5	0,5	0,5
		$e = 5$	$w = 15$	$\tau_{max} = 0,06$ $\tau_{min} = 0,025$	$q_0 = 0,6$ $\xi = 0,6$



Obr. 21 Graf výsledků algoritmů na mapě PSEUDO_3D.

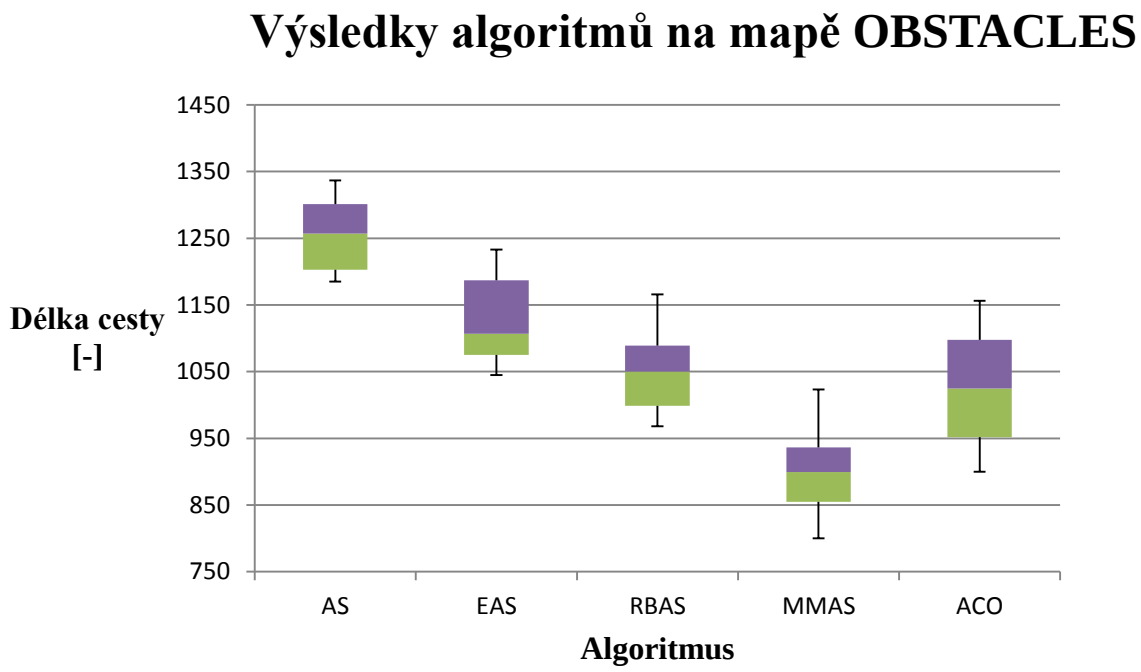
Mapa PSEUDO_3D je podobně jako mapa FLAT bez překážek, avšak již není plochá, ale obsahuje tři kopce. Tyto kopce vytváří „údolí“ s nižší výškou, které poskytují kratší cestu ze startovního do cílového bodu.

6.7.3 Mapa OBSTACLES

Nastavení parametrů:

Tab. 29 Nastavení na mapě OBSTACLES.

Parametry	AS	EAS	RBAS	MMAS	ACO
α	1	1	1	1	1
β	2	2	2	2	2
ρ	0,5	0,5	0,5	0,5	0,5
		$e = 7$	$w = 20$	$\tau_{max} = 0,06$ $\tau_{min} = 0,03$	$q_0 = 0,4$ $\xi = 0,5$



Obr. 22 Graf výsledků algoritmů na mapě OBSTACLES.

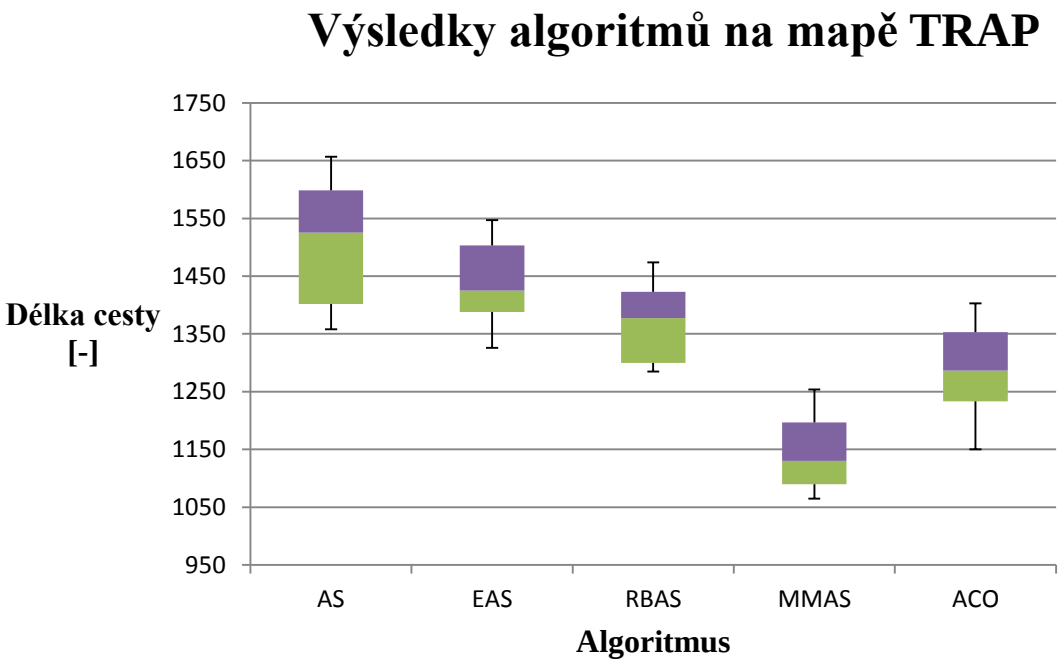
Mapa OBSTACLES je dána kopcovitým pseudo 3D prostředím s několika překážkami. Nejlepší výsledky na této mapě dosáhl algoritmus MMAS.

6.7.4 Mapa TRAP

Nastavení parametrů:

Tab. 30 Nastavení na mapě TRAP.

Parametry	AS	EAS	RBAS	MMAS	ACO
α	1	1	1	1	1
β	2	2	2	2	2
ρ	0,5	0,5	0,5	0,5	0,5
		$e = 7$	$w = 20$	$\tau_{max} = 0,05$ $\tau_{min} = 0,025$	$q_0 = 0,4$ $\xi = 0,5$



Obr. 23 Graf výsledků algoritmů na mapě TRAP.

Mapa TRAP je tvořena kopcovitým pseudo 3D prostředím s překážkami ve tvaru pasti. Nejlepší výsledky zde dosahují obdobně jako u mapy OBSTACLES algoritmy MMAS a ACO. Protože více prohledávají prostor a tím nacházejí lepší cestu z výchozího do cílového bodu.

7. ZÁVĚR

Základním cílem diplomové práce bylo abstrahovat a následně popsat principy základních i vybraných pokročilých mravenčích algoritmů (Ant System, Elitism Ant System, Rank–Based Ant System, Max–Min Ant System, Ant Colony Optimization) a následně navrhnout vlastní programovou aplikaci, která diskutované algoritmy implementuje. Prostředí bylo zvoleno jako tzv. pseudo 3D prostředí, čímž je míněna vizuální náhrada třetího rozměru pomocí barvy tak, jak je zvykem v mapách. V rámci grafové interpretace je poté příslušná výška terénu nahrazena vyšším ohodnocením příslušné hrany, tj. v interpretaci délky cest dojde k jejímu prodloužení. Tímto způsobem byl elegantně vyřešen problém reprezentace 3D plochy pomocí 2D grafu. Vlastní definované překážky jsou pak tzv. nepřekonatelné a v rámci grafové struktury představují odebrané hrany, či hrany s tzv. nekonečnou délkou. Díky takto vytvořené aplikaci byly realizovány rozličné testy, jejichž cílem bylo nalézt optimální nastavení daných mravenčích strategií a prezentovat nejúspěšnější algoritmus při hledání nejkratší cesty.

Z principu implementace 3D prostředí a vlastní grafové interpretace je zřejmé, že na dané grafové struktuře mohou být testovány i jiné, exaktní, či heuristicky založené algoritmy. Typickým příkladem možného algoritmu je například algoritmus Dijkstra nebo A*. Při statickém prostředí, a technickým prostředkům přiměřené velikosti grafu, by tyto algoritmy patrně neměly konkurenci jak v kvalitě řešení, tak v rychlosti jeho nalezení. U rozsáhlých prostor nebo v dynamických či pravděpodobnostně založených prostředích, by však tyto algoritmy byly obtížně implementovatelné. Zde by se ukázala jasná výhoda implementace mravenčích strategií, které jsou schopny efektivněji vymezit svoji výpočetní složitost, resp. přímou dobu výpočtu vůči danému problému. Přesto že vlastní implementace např. Dijkstrova algoritmu nebyla cílem této práce, její případné zavedení by vzhledem k objektové struktuře programu nebylo obtížné.

Mravenčí algoritmy popsané a implementované v této práci vycházejí ze základního Ant Systému definovaného Dorigem v roce 1992. Rozšíření a modifikace původního ant systemu spočívají zejména ve změnách ukládání a odpařování feromonové stopy. Dalším rozšířením je zavedení dvojí aktualizace feromonové stopy a volba pravidla pro výběr cesty grafovou strukturou mravenci. Jednotlivé modifikace feromonové stopy spočívají ve výběru mravenců, kteří nanášejí feromonovou stopu na grafovou strukturu svého řešení. Provádí ho pouze úspěšní mravenci s nalezenou cestou nebo mravenec s nejlepší nalezenou cestou. Feromonová stopa může být taktéž omezena intervalem, který zajišťuje větší prohledávání prostoru. Všechny tyto modifikace byly testovány a je si je možné efektivně prověřit v navržené a na platformě nezávislé programové aplikaci napsané v jazyce Java. Experimentální ověření dokázalo, že implementované algoritmy jsou schopny řešit problémy s hledáním nejkratší cesty na zadaných mapách s pseudo 3D prostorem a překážkami. Jednotlivé algoritmy dosahovaly podobných výsledků v délce cesty na mapách FLAT a PSEUDO_3D, avšak s narůstající složitostí prostoru danou překážkami a pseudo 3D prostorem testy ukázaly, že nejlepším algoritmem pro hledání nejkratší cesty na mapách OBSTACLES a TRAP je tzv. MAX-MIN Ant System.

Závěrem je možné konstatovat, že všechny body zadání diplomové práce byly naplněny. V práci bylo použito nejen nové techniky metaheuristické optimalizace, ale i programování ve velmi efektivním vývojovém prostředí Java.

SEZNAM POUŽITÉ LITERATURY

[01] BONABEAU E., DORIGO M., THERAULAZ G. *Swarm Intelligence: FNatural to Artificial System.* Oxford University Press, 1999. 320 s. ISBN 0-19-513159-2.

[02] ČELEDA,T. *Optimalizace pářením včelí královny*. Praha 2011, 66s. Diplomová práce na Fakultě elektrotechnické, Českého Vysokého učení technického v Praze. Dostupné z www: <https://cyber.felk.cvut.cz/research/theses/papers/132.pdf>

[03] KARABOGA, D., BASTURK, B. A. *Powerful Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm*. Springer Science+Business Media B.V. 2007. 13 s.

[04] MARINAKIS, Y.,MARINAKIS, M. A. *Hybrid Honey Bees Mating Optimization Algorithm for the Probabilistic Traveling Salesman Problem*. IEEE Congress on Evolutionary Computation, 2009, 8 s.

[05] SCHIMITZEK, A. *Plánování cesty robotu pomocí rojové inteligence.* Brno, 2013. 75 s. Diplomová práce na Fakultě strojního inženýrství, vysokého učení technického v Brně. Vedoucí diplomové práce RNDr. Jiří Dvořák, CSc.

[06] RICHTR, R. *Vizualizace optimalizačních algoritmů.* Praha 2010, 101 s. Diplomová práce na Fakultě elektrotechnické, Českého Vysokého učení technického v Praze. Dostupné z www: < https://dip.felk.cvut.cz/browse/pdfcache/richtrad_2011dipl.pdf>

[07] DORIGO M., STÜTZLE T.. *Ant Colony Optimization*. MIT Press, 2004. 305 s. ISBN 0262042193.

[08] DORIGO M., BIRATTARI M., STÜTZLE T.. Ant colony optimization : Artificial Ants as a Computational Intelligence Technique. *Computational Intelligence Magazine, IEEE*, 2006 vol. 1, Issue: 4, s. 28-39. ISSN 1556-603X.

[09] KOVÁŘÍK, O. *Ant Colony Optimization for Continuous Problems*. Praha, 2006. 69 s. Diplomová práce na Fakultě elektrotechnické, Českého Vysokého učení technického v Praze. Vedoucí diplomové práce Ing. Pavel Kordík. Dostupné z www: <https://dip.felk.cvut.cz/browse/pdfcache/kovaro1_2007dipl.pdf >

[10] STÜTZLE T., HOOS H. H.. MAX-MIN Ant System. *Future Generation Computer Systems*, 2000 vol. 16, issue 8, s. 889-914. ISSN 0020-0255.

[11] ORACLE. *Přehled.* [online]. [cit. 2014-3-11]. Dostupné z: <http://www.oracle.com/cz/technologies/java/overview/index.html>

[12] MIŠKAŘÍK, K. *Včelí algoritmus.* Brno, 2010, 37s. Diplomová práce na Fakultě strojního inženýrství, vysokého učení technického v Brně. Vedoucí diplomové práce doc. Ing. Radomil Matoušek, Ph.D.

[13] GIZMAG. *Ant-algorithm-transport-robots.* [online].[cit. 2014-4-11].Dostupné z: < http://www.gizmag.com/ant-algorithm-transport-robots/21933/>

PŘÍLOHY:

CD s elektronickou verzí práce, Java aplikací.