Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Zuzana Grešlíková

Učenie konečno-stavového automatu kolóniou mravcov

(Diplomová práca)

Študijný program:	Informačné systémy
Študijný odbor:	9.2.6 Informačné systémy
Miesto vypracovania:	Ústav informatiky a softvérového inžinierstva, FIIT STU v Bratislave
Vedúci diplomovej práce:	Prof. RNDr. Jiří Pospíchal, DrSc.

STU FIIT
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Zadanie diplomovej práce

Meno študenta: **Bc. Zuzana Grešlíková**

Študijný program: Informačné systémy

Študijný odbor: Informačné systémy

Názov práce: **Učenie konečnostavového automatu kolóniou mravcov**

Samostatnou výskumnou a vývojovou činnosťou v rámci predmetov Diplomový projekt I, II, III vypracujte diplomovú prácu na tému, vyjadrenú vyššie uvedeným názvom tak, aby ste dosiahli tieto ciele:

Všeobecný cieľ:

Vypracovaním diplomovej práce preukážte, ako ste si osvojili metódy a postupy riešenia relatívne rozsiahlych projektov, schopnosť samostatne a tvorivo riešiť zložité úlohy aj výskumného charakteru v súlade so súčasnými metódami a postupmi študovaného odboru využívanými v príslušnej oblasti a schopnosť samostatne, tvorivo a kriticky pristupovať k analýze možných riešení a k tvorbe modelov.

Špecifický cieľ:

Vytvorte riešenie zodpovedajúce návrhu textu zadania, ktorý je prílohou tohto zadania. Návrh bližšie opisuje tému vyjadrenú názvom. Tento opis je záväzný, má však rámcový charakter, aby vznikol dostatočný priestor pre Vašu tvorivosť.

Riaďte sa pokynmi Vášho vedúceho.

Pokiaľ v priebehu riešenia, opierajúc sa o hlbšie poznanie súčasného stavu v príslušnej oblasti, alebo o priebežné výsledky Vášho riešenia, alebo o iné závažné skutočnosti, dospejete spoločne s Vaším vedúcim k presvedčeniu, že niečo v texte zadania a/alebo v názve by sa malo zmeniť, navrhnite zmenu. Zmena je spravidla možná len pri dosiahnutí kontrolného bodu.

Miesto vypracovania: Ústav aplikovanej informatiky FIIT STU v Bratislave

Vedúci práce: **prof. RNDr. Jiří Pospíchal, DrSc.**

Termíny odovzdania:

podľa harmonogramu štúdia platného pre semester, v ktorom máte príslušný predmet (Diplomový projekt I, II, III) absolvovať podľa Vášho študijného plánu

Predmety odovzdania:

V každom predmete dokument podľa pokynov na www.fiit.stuba.sk v časti: home > Informácie o > štúdiu > organizácia štúdia > diplomový projekt.

V Bratislave dňa 17. 2. 2014

prof. Ing. Pavol Návrat, PhD.
riaditeľ Ústavu informatiky a softvérového inžinierstva

STU FIIT
SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce [1]

Študent:

Meno, priezvisko, tituly: Zuzana Grešlíková, Bc.
Študijný program: Informačné systémy
Kontakt: greslikova.zuzana@gmail.com

Výskumník:

Meno, priezvisko, tituly: Jiří Pospíchal, prof. RNDr. DrSc.

Projekt:

Názov: Učenie konečnostavového automatu kolóniou mravcov
Názov v angličtine: Learning Finite-State Machines by ant colony
Miesto vypracovania: Ústav aplikovanej informatiky, FIIT STU, Bratislava
Oblasť problematiky: Optimalizačné algoritmy, umelá inteligencia

Text návrhu zadania[2]

Metódy pre inteligentné rozhodovanie sú súčasťou štúdia inteligentných informačných systémov. Jednou z možností na automatizáciu týchto procesov je aj prehľadávanie priestoru rozhodovacích algoritmov stochastickými optimalizačnými heuristikami, ktoré môžu byť založené na evolučných algoritmoch a im príbuzných technikách.

Konečno-stavové automaty sa môžu aplikovať na riešenie širokej škály rôznych problémov. Ukázali sa ako vhodná reprezentácia agentových stratégií. Učenie konečno-stavových automatov pomocou optimalizačného algoritmu kolónie mravcov je inovatívny prístup. Problém vytvorenia konečno-stavového automatu pre danú účelovú funkciu vyvolal záujem mnohých odborníkov.

Analyzujte vplyv zmeny parametrov algoritmu vytvoreného v článku "Learning Finite-State Machines with Ant Colony Optimalization" na výslednú hodnotu účelovej funkcie.

Navrhnite a implementujte novú metódu na učenie konečno-stavových automatov, založenú na optimalizačnom algoritme kolónie mravcov. Výsledky navrhnutého algoritmu porovnajte s testami zvoleného konkurenčného publikovaného algoritmu.

Navrhnuté riešenie overte na testovacom príklade "John Muir food trail problem", v ktorom agent na toroidálnej mriežke vidí iba jedno políčko vopred a jeho cieľom je nájsť takú stratégiu, aby pri svojej ceste po mriežke odhalil čo najviac „kúskov potravy". Tento problém je štandardom pri porovnávaní rôznych evolučných algoritmov a heuristík.

---

[1] Vytlačiť obojstranne na jeden list papiera

[2] 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra[3]

- Chivilikhin, D., Ulyantsev, V. Learning Finite-State Machines with Ant Colony Optimization. 8th International Conference, ANTS 2012, Brussels, Belgium : Springer Berlin Heidelberg, 2012
- Chivilikhin, D., Ulyantsev, V. MuACOsm – A New Mutation-Based Ant Colony Optimization Algorithm for Learning Finite-State Machine: GECCO '13 Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference. 2013

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Zuzana Grešlíková, konzultoval(a) a osvojil(a) si ho prof. RNDr. Jiří Pospíchal, DrSc. a súhlasí, že bude takýto projekt viesť v prípade, že bude pridelený tomuto študentovi.

V Bratislave dňa 10.2.2014

Podpis študenta

Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / ~~nie~~[4]

Dňa: 14. 2. 2014

Podpis garanta predmetov

[3] 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uveďte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

[4] Nehodiace sa prečiarknite

ANOTÁCIA

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program : Informačné systémy

Autor: Zuzana Grešlíková
Diplomová práca: Učenie konečno-stavového automatu kolóniou mravcov
Vedúci diplomovej práce: Prof. RNDr. Jiří Pospíchal, DrSc.
2015, Máj

Hlavnou náplňou tejto diplomovej práce je vylepšiť algoritmus na učenie konečno-stavových automatov, ktorý využíva metódu optimalizácie kolóniou mravcov. V práci budeme testovať vplyv zmeny jednotlivých parametrov algoritmu na získané výsledky. Pokúsime sa získať výsledok, ktorý by potreboval na riešenie testovacieho problému čo najmenší počet krokov konečno-stavového automatu a bol získaný za čo najmenší počet iterácií algoritmu (fitnes ohodnotení).

Výsledný algoritmus bude testovaný na testovacom príklade „Santa Fe trail problem“ a „John Muir food trail problem“ a bude vyvodený záver podľa dosiahnutých výsledkov.

ANOTATION

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: Information Systems

Author: Zuzana Grešlíková
Master´s Thesis: Learning Finite-State Machines with Ant Colony
Supervisor: Prof. RNDr. Jiří Pospíchal, DrSc.
2015, May

The main goal of this diploma thesis is to improve learning algorithm to synthesize a finite state machine, which uses ant colony optimization. We will test the impact of different parameters of the algorithm on the results obtained. We will try to get a result that would need to make the minimum number of steps of the finite state machine and has been obtained for the minimum number of iterations of the algorithm (fitness evaluation).

Final algorithm will be tested on „Santa Fe trail problem“ and „John Muir food trail problem“ and conclusion, based on results, will be deduced.

Poďakovanie

Na tomto mieste by som sa chcela poďakovať vedúcemu mojej diplomovej práce, Prof. RNDr. Jiřímu Pospíchalovi, DrSc, za jeho odborné vedenie počas písania diplomovej práce, jeho cenné rady, pripomienky, odbornú pomoc a materiály, ktoré mi pri vypracovaní diplomovej práce veľmi pomohli.

Čestné prehlásenie

Čestne prehlasujem, že som prácu vypracovala samostatne s použitím uvedenej literatúry.

V Bratislave, 11. Mája 2015

.....
(podpis autora práce)

Zoznam skratiek

FSM – Finite-state machine (konečno-stavový automat)

ACO- Ant colony optimization (optimalizácia kolóniou mravcov)

SACO – Simple Ant colony optimization algoritmus

AS – Ant System algoritmus

ACS – Ant Colony System algoritmus

MMAS - Max-Min Ant System algoritmus

FANT – Fast Ant System algoritmus

ANTS - Aproximated non-deterministic tree search algoritmus

ACO-MH - Ant Colony Optimization Meta-heuristika

AS-MH - Ant System Meta-heuristika

AP - Ant Programing meta-heuristika

HC-ACO - Hypercube ACO framework

TSP – Traveling Salesman problem (Problém obchodného cestujúceho)

MRTS - Memorized-Random-Tree-Search, algoritmus navrhnutý v článku [34]

Obsah

1. Analýza

1.1 Teória automatov

Teória automatov [7] je súčasťou počítačovej vedy. Svoje korene má v dvadsiatom storočí a vznikla potom, čo matematici začali rozvíjať stroje, ktoré by mali napodobňovať určité vlastnosti človeka, vyhodnocovať výpočty spoľahlivejšie a rýchlejšie. Teória automatov sa zaoberá logikou výpočtu. Počítačoví vedci sú pomocou automatov schopní porozumieť tomu, ako stroje počítajú funkcie a riešia problémy, a čo je dôležitejšie dokážu vyjadriť, čo znamená pre funkciu byť definovaná ako vyčísliteľná a pre otázku byť opísaná ako rozhodnuteľná.

Automaty sú abstraktné modely strojov, ktoré vykonávajú výpočty na základe vstupu pohybom medzi sériou stavov alebo konfigurácií. Obsahujú vstupy, výstupy a stavy. V každom stave výpočtu, prechodová funkcia určí nasledujúci stav (konfiguráciu). Keď sa výpočet dostane do akceptačného stavu (konfigurácie), akceptuje tento vstup.

Hlavným cieľom teórie automatov je vyvinúť metódy, ktorými dokážu počítačoví vedci opísať a analyzovať dynamické správanie diskrétnych systémov, v ktorých sú signály periodické. Správanie týchto diskrétnych systémov je určené spôsobom, akým je systém konštruovaný.

Existujú štyri základné rodiny automatov:
- Konečno-stavové automaty
- Zásobníkové automaty
- Lineárne ohraničené automaty
- Turingove stroje

Práca sa ďalej zameriava na konečno-stavové automaty.

1.1.1 Konečno-stavové automaty (FSM)

Konečno-stavový automat (Finite-state machine, FSM) je matematický výpočtový model používaný pri návrhu počítačových programov a sekvenčných logických obvodov. Stavový priestor konečno-stavových automatov je konečný. FSM pozostáva zo stavov, vstupov a výstupov, pričom počet stavov je fixný. FSM je abstraktný stroj, ktorý sa môže nachádzať práve v jednom zo stavov. Stav, v ktorom sa v danom čase nachádzame sa nazýva aktuálny stav. Po vykonaní určitej akcie alebo podmienky (po prijatí vstupu) sa môže daný stav zmeniť a ak je definovaný, vyprodukuje sa výstup. To sa nazýva prechod. Jednotlivé FSM sú definované pomocou množiny stavov a spúšťacích podmienok pre jednotlivé prechody.

Základný FSM sa dá prirovnať k Turingovmu stroju, ktorého čítacia hlava sa pohybuje len v jednom smere.

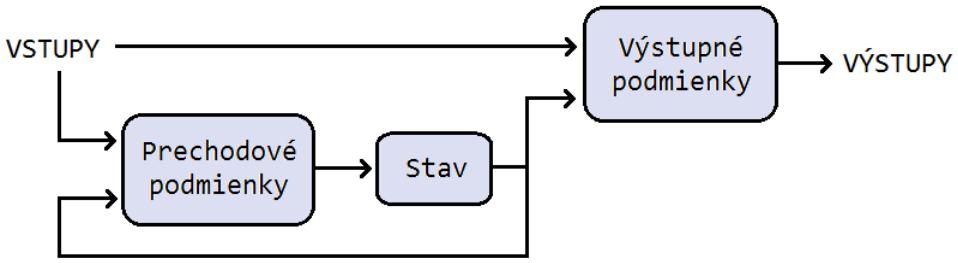
FSM nemajú takú výpočtovú silu, ako niektoré iné výpočtové modely, napr. Turingov stroj. Je to spôsobené tým, že FSM majú limitovanú veľkosť pamäte, ktorá je obmedzená počtom stavov.

1.1.1.1 História konečno-stavových automatov [4]

História toho, ako sa automaty stali časťou počítačovej vedy, naznačuje široké spektrum ich využitia. Ako prví sa zaoberali myšlienkou konečno-stavových automatov tímy biológov, psychológov, matematikov, inžinierov a prví počítačoví vedci. Všetci zdieľali spoločný záujem namodelovať ľudský myšlienkový proces, či už v mozgu, alebo v počítači. Opis konečného automatu prezentovali ako prví neurológ Warren McCulloch a Walter Pitts v roku 1943. Ich práca "A Logical Calculus Immanent in Nervous Activity" bola významným príspevkom v oblasti neurónových sietí, teórie automatov, teórie výpočtov a v kybernetike.

1.1.1.2 Logika konečno-stavových automatov

Nasledujúci stav a výstup FSM sú funkciou vstupu a aktuálneho stavu. Logika FSM je na obrázku 1.1. Konečno-stavový automat pozostáva zo vstupnej pásky a čítacej hlavy. Čítacia hlava môže byť umiestnená na jednom z konečného počtu stavov. Číta slovo tým, že sa postupne pohybuje po vstupnej páske a podľa prečítaných znakov mení svoj stav. Stav po dočítaní pásky rozhodne, či je slovo dobré, alebo nie.

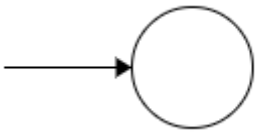


Obrázok 1.1: Logika konečno-stavového automatu

1.1.1.3 Reprezentácia

FSM môže byť reprezentovaný ako graf s konečnou množinou uzlov. Uzly predstavujú stavy. *Stav* je popis pozície systému, ktorý čaká na to, aby sa vykonal *prechod*. Prechod je množina akcií, ktoré sa majú vykonať, keď je splnená podmienka, alebo keď nastane určitá udalosť.

Každý stav je reprezentovaný kruhom a každý prechod je reprezentovaný šípkou. Prechody sú označené vstupom, ktorý prechod spôsobí, no môžu byť aj výstupom, ktorý prechod vyvolá. Konečno-stavové automaty môžu, no nemusia, mať výstup. Štartový stav je označený šípkou smerujúcou do korešpondujúceho uzla (Obrázok 1.2).

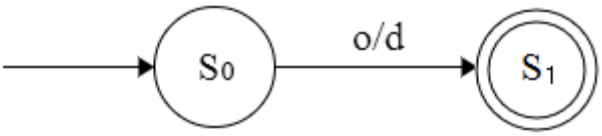


Obrázok 1.2: Zobrazenie štartového uzla

Akceptačný stav je vyjadrený dvojitou kružnicou (Obrázok 1.3). Neakceptačný pomocou jednoduchej čiary. Nie každý automat musí mať konečný stav, a také automaty môžu bežať navždy.

Obrázok 1.3: Zobrazenie koncového uzla

Automat môže byť reprezentovaný ako na obrázku 1.4, kde S_0 a S_1 predstavujú stavy. S_0 je štartový stav, S_1 je akceptačný stav. Prechod medzi stavmi je zobrazený šípkou z jedného stavu do druhého. Prechody sú popísané pomocou o/d, kde o je vstupný symbol a d je výstupný symbol.

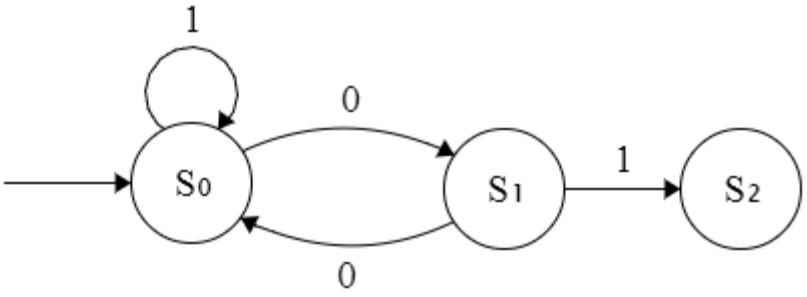


Obrázok 1.4: Zobrazenie reprezentácie automatu

Automat môžeme reprezentovať prechodovou tabuľkou, ktorá pre každý stav a vstup zobrazuje nasledujúci stav a výstup. Výstupy sú väčšinou umiestnené naľavo a oddelené od vstupov, ktoré sú napravo. Na obrázku 1.5 je jednoduchý príklad, ktorý je zobrazený v prechodovej tabuľke č.1.1.

Vstup	Aktuálny stav	Nasledujúci stav	Výstup
0	S_0	S_1	null
1	S_0	S_0	null
0	S_1	S_0	null
1	S_1	S_2	null

Tabuľka 1.1: Prechodová tabuľka pre automat na obrázku 1.5



Obrázok 1.5: Príklad jednoduchého automatu

1.1.1.4 Optimalizácia konečno-stavových automatov

Optimalizácia FSM znamená nájdenie minimálneho počtu stavov, ktoré vykonajú rovnakú funkciu. Najrýchlejší známy algoritmus, ktorý optimalizuje FSM sa nazýva Hopcroft minimization algorithm.

1.1.1.5 Rozdelenie konečno-stavových automatov

Stavové automaty delíme na prevodové, akceptačné, klasifikačné a sekvenčné.

- **Akceptačné** produkujú binárny výstup s tým, že povedia, či bol vstup akceptovaný alebo nie. Každý stav môže byt akceptačný alebo neakceptačný. Ak je aktuálny stav akceptačný, daný vstup je akceptovaný, inak je zamietnutý.
- **Klasifikačné** podobné akceptačným produkujú jeden výstup, no majú viac ako dva koncové stavy.
- **Prechodové** vygenerujú výstup založený na vstupe a stave pomocou akcií. Používajú sa na riadenie akcií a v počítačovej lingvistike. V riadení aplikácií rozlišujeme dva typy, a to Moorove automaty a Mealyho automaty.
- **Sekvenčné** sú podtriedou vyššie uvedených typov, ktorých vstupná abeceda obsahuje jedno písmeno.

Konečno-stavové automaty môžeme tiež rozdeliť na deterministické a nedeterministické.

Deterministické

Pri deterministických má každý stav presne jeden prechod pre každý možný vstup. V nedeterministickom nemusí byť nevyhnutne v určitom stave len jeden prechod pre daný vstup, no prechod nemusí byť žiaden. V praxi sa nedeterministický automat dá previesť na deterministický.

- **Matematicky model**

Deterministický stavový automat (alebo akceptačný FSM) predstavuje pätica $(\Sigma, S, s_0, \delta, F)$, kde

- Σ je vstupná abeceda (konečná neprázdna množina symbolov)

- S je konečná množina stavov (konečná neprázdna množina stavov)
- s_0 je počiatočný stav (prvok množiny S)
- δ - prechodová funkcia $\delta : S \times \Sigma \rightarrow S$ (na základe stavu a symbolu zo vstupnej abecedy vráti nový stav S) V nedeterministickom konečno-stavovom automate to bude dané ako $\delta : S \times \Sigma \rightarrow P(S)$ a δ vráti množinu stavov.
- F – množina akceptačných (koncových) stavov, je to ľubovoľná podmnožina S (aj prázdna)

- **Konfigurácia**

Konfigurácia deterministického konečného automatu je dvojica $(s, w) \in S \times \Sigma^*$, kde w je neprečítaná časť vstupného slova a s je aktuálny stav. Udáva jednoznačný opis akcie, v ktorej sa automat v určitom čase nachádza. Na základe konfigurácie sa dá povedať ako bude pokračovať výpočet.

- **Krok výpočtu**

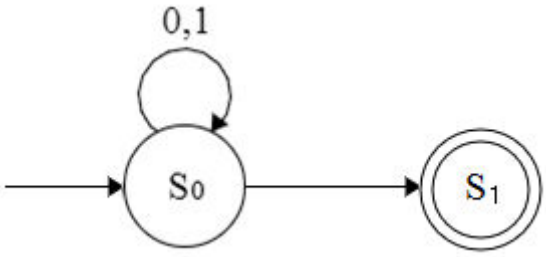
Krok výpočtu hovorí v akom stave sa bude automat A nachádzať v ďalšom kroku výpočtu, ak vieme, kde sa nachádza aktuálne. Krok výpočtu deterministického konečného automatu je relácia $\vdash_A$ definovaná ako $(s, av) \vdash_A (p, v) \Leftrightarrow p = \delta(s, a)$. Pod výpočtom sa rozumie akákoľvek postupnosť na seba nadväzujúcich krokov výpočtu.

- **Akceptovaný jazyk**

Jazyk akceptovaný deterministickým konečno-stavovým automatom A je množina všetkých slov, na ktoré existuje v automate A výpočet končiaci v akceptačnom stave. Tento jazyk je definovaný ako $L(A) = \{w \mid \exists p \in F: (s_0, w) \vdash_A^* (p, \varepsilon)\}$.

Nedeterministické

Zatiaľ čo pri deterministickom konečnom automate pre stav a korektný vstup existuje iba jeden prechod, pri nedeterministickom pre daný vstup môže existovať viacero prechodov. Príklad na obrázku 1.6., kde pre stav S_0 existuje niekoľko možných prechodov.



Obrázok 1.6: Príklad nedeterministického konečného automatu

1.1.1.6 Využitie

FSM sú významné v mnohých oblastiach. Často sa používajú pri návrhu počítačových programov, ale aj v inžinierstve, biológii, počítačovej vede, filozofii, lingvistike, matematike, logike a iných vedách vďaka ich schopnosti rozpoznať sekvencie.

Správanie stavových automatov sa dá pozorovať v mnohých prístrojoch, ktoré vykonávajú preddefinovaný sled činností, ktoré závisia na sekvencií udalostí. Napr. výťahy, semafory, kombinačné zámky, ktoré vyžadujú aby bola zadaná presná kombinácia čísel.

Pomocou FSM môžeme modelovať veľké množstvo problémov, ako napríklad návrh komunikačných protokolov, návrh digitálnych systémov, popis pamäťových obvodov, v kompilátoroch, modelovanie správania aplikácií, rozdeľovanie jazyka, výskum umelej inteligencie, používajú sa na opis neurónových systémov a v lingvistike na opis gramatík a jazykov, na vyhľadávanie v texte.

1.2 Kolónia mravcov

Mravce žijú v kolóniách, v ktorých sa, v niektorých prípadoch, nachádzajú až milióny jedincov. Patria medzi tzv. sociálny hmyz. Ľudí vždy fascinovalo správanie rojov hmyzu, a preto existuje a vzniká množstvo štúdií, ktorých cieľom je lepšie porozumieť a pochopiť toto správanie. Veda, ktorá sa zaoberá správaním živočíchov sa nazýva etológia a za otca tejto vedy sa považuje Konrad Lorenz. Priekopníkom v tejto oblasti je však aj Eugéne Marais, ktorý vydal knihu „The Soul of the Ant (1937)“ [8], kde detailne popísal jeho experimentálne pokusy a výskumy sociálneho spoločenstva termitov. Postupne sa dospelo k záveru, že medzi mravcami (resp. termitmi) existuje určitá forma nepriamej komunikácie. Komunikácia existuje medzi jedincami navzájom, ale aj medzi jedincom a prostredím. Jean- Louis Deneubourg [9] sa zaoberal štúdiou jedného z druhov komunikácie, a to konkrétne feromónovou komunikáciou. Na základe tejto štúdie bol vyvinutý a implementovaný prvý algoritmický model hľadania potravy [10]. Táto forma komunikácie bola skúmaná najmä v kolóniách mravcov. Jednotliví jedinci sociálneho hmyzu sú v podstate agenti, ktorí reagujú odpoveďou (jednoduchá základná akcia) na stimuly, ktoré prijímajú z prostredia.

Systém hľadania potravy u mravcov inšpiroval mnoho algoritmov založených na správaní mravcov, ktoré sa využívajú najmä pri riešení optimalizačných problémov. Prvým správaním, ktorým sa etológovia zaoberali bola schopnosť mravcov nájsť najkratšiu cestu medzi mraveniskom a zdrojom potravy. Z týchto štúdií vznikol prvý algoritmický model vyvinutý Marcom Dorigom. Odvtedy vzniklo množstvo iných algoritmov inšpirovaných správaním mravcov pri hľadaní potravy.

Ako sa mravcom podarí nájsť najkratšiu cestu od mraveniska k jedlu bez akéhokoľvek viditeľného, centrálneho koordinačného mechanizmu? Na začiatku je pohyb mravcov náhodný a chaotický, keď objavia jedlo, ich pohyb sa stáva čoraz viac organizovaný a čím ďalej tým viac mravcov nasleduje rovnakú najkratšiu trasu. Komunikácia prebieha pomocou feromónov. Keď mravec lokalizuje potravu, cestou späť do mraveniska zanechá na trase feromón. Ostatné mravce hľadajúce potravu sa rozhodujú, ktorou cestou sa vybrať na základe feromónovej hodnoty. Cesty s vyššou feromónovou hodnotou tak majú vyššiu šancu na to, aby boli mravcom zvolené. Čím viac mravcov danú trasu použije, tým viac je posilnená jej vhodnosť, keďže sa na nej nachádza viac feromónov, čo priťahuje stále viac mravcov.

Nepriama komunikácia, kde mravce modifikujú svoje prostredie, aby ovplyvnili správanie ostatných jedincov, sa nazýva „stigmergy“.

1.2.1 Binary bridge experiment

Denebourg [9] študoval správanie mravcov pri hľadaní potravy za účelom nájdenia formálneho modelu, ktorý by opísal ich správanie. V jeho experimente je mravenisko oddelené od jedla dvoma rovnako dlhými mostmi. Na začiatku tieto trasy neobsahujú žiaden feromón. V konečnom čase si väčšina mravcov vyberie jednu trasu, aj napriek tomu, že ich

dĺžky sú rovnaké. Selekcia jednej z ciest je spôsobená fluktuáciou (výkyvom) pri selekcii cesty, čo spôsobuje vyššiu koncentráciu na jednej z dráh. Z tohto experimentu vznikol jednoduchý formálny model, ktorý charakterizuje výber cesty. Pre tieto účely sa predpokladá, že mravce zanechávajú rovnaké množstvo feromónu, a že tento feromón sa nevyparuje. Nech $n_A(t)$ a $n_B(t)$ predstavujú počet mravcov na trase A a trase B v čase t. Pasteels [9] pokusne dokázal, že pravdepodobnosť, že si mravec v ďalšom kroku v čase $t+1$ vyberie trasu A, je daná ako

$$P_A(t+1) = \frac{(c + n_A(t))^{\propto}}{(c + n_A(t))^{\propto} + (c + n_B)^{\propto}} = 1 - P_B(t+1), \qquad (1.1)$$

kde c vyčísľuje stupeň atraktivity nepreskúmanej hrany a $\propto$ je tendencia použitia feromónového depozitu v rozhodovacom procese. Čím je $\propto$ väčšie, tým je vyššia pravdepodobnosť, že nasledujúci mravec bude nasledovať cestu s vyššou feromónovou koncentráciou (aj keď je rozdiel len malý).

Experiment rozšíril Goss [11]. Zmena bola v tom, že jedna z hrán bola dlhšia ako druhá. Na začiatku sa hrany vyberajú náhodne a počet mravcov na oboch cestách je približne rovnaký. Postupom času stále viac mravcov nasleduje kratšiu cestu. Selekcia sa prikláňa ku kratšej ceste, keďže sa mravce, ktoré kratšiu cestu použijú vrátia do mraveniska skôr. Kratšia cesta je tým pádom „zosilnená“ skôr.

Goss [11] dokázal, že pravdepodobnosť selekcie kratšej cesty sa zvyšuje s dĺžkovým rozdielom medzi dvoma cestami. To sa nazýva tiež „differential path lenght effect“ popísaný Dorigom.

Hoci celá skupina mravcov vykazuje komplexné adaptívne správanie, jeden mravec má jednoduché správanie. Mravca môžeme vidieť ako agenta, ktorý na stimuly reaguje odpoveďou. Mravec skúma feromónovú koncentráciu a vykoná akciu založenú na feromónovom stimule.

1.2.2 **Druhy optimalizácií kolóniou mravcov**

1.2.2.1 **SACO (simple ACO)**

SACO [12,13] je algoritmická implementácia double bridge experimentu spomenutého v kapitole 1.2.1. Základný problém nájdenia najkratšej cesty je reprezentovaný grafom, $G = (V, E)$, kde V je množina vrcholov a E je matica, ktorá reprezentuje spoje medzi vrcholmi. Dĺžka, ktorú mravec prejde sa určí ako počet hrán, ktoré mravec prejde na ceste do cieľového uzla. Všetky mravce sú na začiatku umiestnené v zdrojovom uzle. V každej iterácii, každý mravec inkrementálne vytvára cestu do cieľového uzla. Každý mravec teda v každej iterácii musí urobiť rozhodnutie, v ktorom určí kam sa presunie. Ak sa mravec nachádza vo vrchole i, nasledujúci uzol $j \epsilon N_i^k$ sa určí na základe pravdepodobnosti

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(t)}{\sum_{j \in N_i^k} \tau_{ij}^{\alpha}(t)} & ak\ j \in N_i^k \\ 0 & ak\ j \notin N_i^k \end{cases}, \tag{1.2}$$

kde N_i^k je množina vrcholov, ktoré sú s vrcholom i spojené hranou. $\propto$ je konštanta, ktorá sa využíva na zosilnenie vplyvu feromónovej koncentrácie.

Keď všetky mravce skonštruujú kompletnú cestu do koncového vrcholu, každý z mravcov deterministicky zrekonštruuje svoju cestu do zdrojového vrcholu a zanechá na nej feromónovú hodnotu,

$$\Delta\tau_{ij}^{k}(t) = \frac{1}{L^k(t)}, \tag{1.3}$$

kde $L^k(t)$ je dĺžka cesty skonštruovanej daným mravcom k v čase t. Nová feromónová hodnota na hrane (ij) je

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{n_k} \Delta\tau_{ij}^{k}(t), \tag{1.4}$$

kde n_k je počet mravcov.

Prvé experimenty na Binary bridge probléme ukázali, že mravce rapídne konvergujú k jednému riešeniu a venujú len málo času preskúmaniu alternatívnych ciest. Na to, aby sa tomu zamedzilo a riešenie neskončilo len v lokálnom optime, bolo umožnené, aby sa intenzita feromónu na trase mohla „vyparovať“, ešte kým bude hodnota posilnená novo skonštruovanými cestami mravcov. Pre každú hranu

$$\tau_{ij}(t) \leftarrow (1 - \rho)\tau_{ij}(t), \tag{1.5}$$

kde ρ patrí do intervalu [0,1] a špecifikuje intenzitu akou sa feromóny vyparujú. Pre $\rho = 1$ je hľadanie náhodné.

1.2.2.2 Early Ant algoritmy

Sú vylepšením algoritmu SACO implementáciou jednoduchých zmien. Je pridaná heuristická informácia na určenie pravdepodobnosti selektovania cesty. Obsahuje pamäť, aby sa zamedzilo cyklom a iné pravidlá feromónovej aktualizácie s použitím lokálnych alebo globálnych informácií z prostredia.

Ant system (AS)

Bol vytvorený Dorigom [10] a pomenovaný ako Ant system [14]. Zlepšuje SACO zmenou určenia pravdepodobnosti p_{ij}^{k}, tak aby obsahovala heuristickú informáciu, a taktiež bol

pridaný tabu list. Pravdepodobnosť toho, že sa mravec presunie z vrcholu i do vrchola j je daná ako

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}(t)\,\eta_{ij}^{\beta}(t)}{\sum_{j \in N_i^k(t)} \tau_{ij}^{\alpha}(t)\;\eta_{ij}^{\beta}(t)} & ak\ j \in N_i^k \\ 0 & ak\ j \notin N_i^k \end{cases}, \tag{1.6}$$

kde η_{ij} reprezentuje *a posteriori* účinnosť pohybu z vrcholu i do vrcholu j vypočítanú na základe určitých heuristík. τ_{ij} predstavuje koncentráciu feromónu a indikuje aké výhodné bolo v minulosti spraviť pohyb z vrcholu i do j.

Vyparovanie sa robí rovnako ako v SACO. Feromónová hodnota sa aktualizuje na základe

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \Delta\tau_{ij}(t), \tag{1.7}$$

kde

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{n_k} \Delta\tau_{ij}^{k}(t), \tag{1.8}$$

kde $\Delta\tau_{ij}^{k}(t)$ je množstvo feromónu zanechaného mravcom k na trase (i,j) v čase t. Celkovo boli vyvinuté tri verzie AS, ktoré sa líšia v spôsobe vypočítania $\Delta\tau_{ij}^{k}$.

1. Ant-cycle AS:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{f(x^k(t))} & ak\ sa\ hrana\ (i,j) \in x^k(t) \\ 0 & inak \end{cases} \tag{1.9}$$

Výsledok je nepriamo úmerný kvalite $f\left(x^k(t)\right)$ kompletnej cesty skonštruovanej mravcom. Q je pozitívna konštanta.

2. Ant-density AS:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} Q & ak\ sa\ hrana\ (i,j) \in\ x^k(t) \\ 0 & inak \end{cases} \tag{1.10}$$

3. Ant-quantity AS:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{d_{ij}} & ak\ sa\ hrana\ (i,j)\ \in x^k(t) \\ 0 & inak \end{cases} \tag{1.11}$$

d_{ij} môže reprezentovať dĺžku hrany (i,j).

Ant Colony System (ACS)

Bol vyvinutý Gambardelom a Dorigom na zlepšenie výkonu AS [14]. Od AS sa líši v štyroch aspektoch.

- Iné prechodové pravidlo
- Iné pravidlo na aktualizáciu feromónovej hodnoty
- Lokálna aktualizácia feromónovej hodnoty
- List kandidátov použitý na podporu špecifických uzlov

Prechodové pravidlo bolo navrhnuté tak aby vyvažovalo pomer skúmania grafu a rozvíjania získaných najlepších riešení. Mravec *k*, ktorý sa nachádza v uzle *i* si vyberie nasledujúcu hranu *j* použitím pravidla

$$j = \begin{cases} \arg max_{u \in N_i^k(t)} \left\{ \tau_{iu}(t) \eta_{iu}^{\beta}(t) \right\} & if\ r \leq r_0 \\ J & if\ r > r_0 \end{cases}, \tag{1.12}$$

kde $r \sim U(0,1)$ a $r_0 \in [0,1]$ je parameter špecifikovaný používateľom. $J \in N_i^k(t)$ je uzol náhodne vybraný na základe pravdepodobnosti

$$p_{ij}^k = \frac{\tau_{iJ}(t)\ \eta_{iJ}^{\beta}(t)}{\sum_{u \in N_i^k} \tau_{iu}(t)\ \eta_{iu}^{\beta}(t)} \tag{1.13}$$

Globálna aktualizácia feromónovej hodnoty: Na rozdiel od AS iba globálne najlepší jedinec (ten, ktorý zostrojil najkratšiu cestu $x^+(t)$) má právo ovplyvniť feromónovú koncentráciu na korešpondujúcej najlepšej ceste. Feromónová hodnota sa aktualizuje na základe globálneho aktualizačného pravidla

$$\tau_{ij}(t+1) = (1-\rho_1)\tau_{ij}(t) + \rho_1 \Delta\tau_{ij}(t), \tag{1.14}$$

kde

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{f(x^+(t))} & ak\ (i,j) \in x^+(t) \\ 0 & inak \end{cases} \tag{1.15}$$

Lokálna aktualizácia feromónovej hodnoty:

$$\tau_{ij}(t) = (1-\rho_2)\tau_{ij}(t) + \rho_2 \tau_0, \tag{1.16}$$

kde $\rho_2 \in (0,1)$ a τ_0 je malá pozitívna konštanta.

Max-Min Ant System (MMAS)

Ukázalo sa, že AS predčasne stagnuje pri komplexných problémoch. Stagnácia znamená, že všetky mravce prechádzajú po rovnakej ceste. Stane sa, že mravce preskúmajú iba malú časť svojho okolia a rapídne začnú využívať len cestu s najvyššou feromónovou koncentráciou. MMAS [15,16] predstavili Stützle a Hoss. Hlavnou zmenou oproti AS je, že feromónová intenzita je obmedzená intervalom. Ďalší rozdiel je, že iba ten najlepší mravec môže feromónovú hodnotu ovplyvniť. Na začiatku sú feromóny nastavené na najvyššiu možnú hranicu a potom sa použije mechanizmus, ktorý feromóny uhladzuje.

Globálna aktualizácia feromónovej hodnoty je rovnaká ako pri ACS.

Ant-Q

Jedná sa o variant ACS vyvinutý Dorigom a Cambardellom [17], v ktorom je lokálne aktualizačné pravidlo inšpirované Q-learning. Myšlienka feromónov je nahradená Ant-Q hodnotou. Cieľom Ant-Q je naučiť AQ-hodnoty tým, že sa preferuje preskúmanie dobrých riešení.

Fast Ant System (FANT)

Vyvinutý Taillardom a Gambardelom [18] na riešenie špecifických kvadratických priraďovacích problémov. Rozdielom oproti predošlým zmieneným algoritmom je, že FANT používa iba jedného mravca. To znižuje výpočtovú zložitosť. Pri výpočte pravdepodobnosti pohybu z i do j sa používa rovnaký vzorec ako pri AS, s tým, že $\beta = 0$. Aktualizácia feromónu je daná ako

$$\tau_{ij}(t+1) = \tau_{ij}(t) + w_1 \Delta\tilde{\tau}_{ij}(t) + w_2 \Delta\hat{\tau}^{+}_{ij}(t), \tag{1.17}$$

kde w_1 a w_2 sú parametre vyjadrujúce relatívny vplyv aktuálneho riešenia v čase t a celkovo najlepšieho riešenia.

$$\Delta\tilde{\tau}_{ij}(t) = \begin{cases} 1 & ak\ (i,j) \in \tilde{x}(t) \\ 0 & inak \end{cases} \tag{1.18}$$

a

$$\Delta\hat{\tau}_{ij}(t) = \begin{cases} 1 & ak\ (i,j) \in \hat{x}(t) \\ 0 & inak \end{cases}, \tag{1.19}$$

kde $\tilde{x}(t)$ a $\hat{x}(t)$ sú najlepšia cesta nájdená v čase t a globálne najlepšia cesta nájdená od začiatku vyhľadávania.

Antabu

Roux [19] a Kaji [20] pozmenili AS pridaním lokálneho vyhľadávania s použitím tabu search na to, aby dosiahli vylepšenie riešení nájdených po každej iterácii.

Každý mravec aktualizuje feromónovú hodnotu pomocou použitím

$$\tau_{ij}(t+1) = (1-\rho)\,\tau_{ij}(t) + \left(\frac{\rho}{f\left(x^k(t)\right)}\right)\left(\frac{f\left(x^-(t)\right) - f\left(x^k(t)\right)}{f\left(\hat{x}(t)\right)}\right), \quad (1.20)$$

kde $f\left(x^-(t)\right)$ je cena najhoršieho doteraz známeho riešenia, $f\left(\hat{x}(t)\right)$ je cena najlepšieho doteraz známeho riešenia a $f\left(x^k(t)\right)$ je cena riešenia mravca k. Výpočet sa použije pre každého mravca k pre každú hranu $(i,j) \in x^k(t)$.

AS – rank

Modifikácia AS predstavená Bullnheimerom [21], ktorá umožňuje len najlepšiemu mravcovi aktualizáciu feromónovej koncentrácie na hranách najlepšej globálnej trasy. Využíva elitistických mravcov a umožňuje aktualizácie feromónovej hodnoty na základe zoradenia mravcov podľa ich úspešnosti. Globálne optimalizačné pravidlo je dané ako

$$\tau_{ij}(t+1) = (1-\rho)\,\tau_{ij}(t) + n_e \Delta\tilde{\tau}_{ij}(t) + \Delta\tau_{ij}^r(t), \quad (1.21)$$

kde n_e je počet elitistických mravcov a

$$\Delta\tilde{\tau}_{ij}(t) = \frac{Q}{\bar{f}(\hat{x}(t))}, \quad (1.22)$$

kde $\hat{x}(t)$ je dosiaľ najlepšia skonštruovaná cesta.

Použijeme n^e najlepších mravcov a n^k mravcov je zoradených podľa získanej fitnes hodnoty ako $f\left(x^1(t)\right) \leq f\left(x^2(t)\right) \leq \cdots \leq f(x^{n^k}(t))$, potom

$$\Delta\tau_{ij}^r(t) = \sum_{k=1}^{n_e} \Delta\tau_{ij}^k(t), \quad (1.23)$$

kde

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{(n_e - k) * Q}{f(x^k(t))} & ak\ (i,j) \in x^k(t). \\ 0 & inak \end{cases} \quad (1.24)$$

Aproximated non-deterministic tree search (ANTS)

Bol vyvinutý Maniezzom a Carbonarom [22] ako rozšírenie AS. Od AS sa odlišuje v určovaní pravdepodobnosti presunu, v globálnom aktualizačnom pravidle a prístupe na vyhnutie sa stagnácii. Feromónové intenzity sa zmenia, až keď svoju cestu vytvoria všetky mravce. Pravdepodobnosť prechodu je daná vzťahom

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\alpha\tau_{ij}(t) + (1-\alpha)\eta_{ij}}{\sum_{u \in N_i^k(t)} \left(\alpha\tau_{iu}(t) + (1-\alpha)\eta_{iu}(t)\right)} & ak\ j \in N_i^k(t) \\ 0 & inak \end{cases}, \qquad (1.25)$$

kde množina N_i^k obsahuje všetky možné prechody z aktuálneho uzla i. Intenzity feromónov sa aktualizujú, keď všetky mravce svoje cesty ukončia. Pri výpočte feromónovej intenzity sa používajú vzorce (1.5) a (1.8), ale

$$\Delta\tau_{ij}^{k}(t) = \tau_0 \left(1 - \frac{f(x^k(t)) - \epsilon)}{\bar{f}(t) - \epsilon}\right), \qquad (1.26)$$

kde $f\left(x^k(t)\right)$ je cena cesty $x^k(t)$ mravca k v iterácií t. $\bar{f}(t)$ je pohyblivý priemer ceny posledných nájdených $\hat{n}_t$ globálne najlepších riešení. ϵ je spodná hranica ceny optimálneho riešenia.

1.2.3 Charakteristiky ACO algoritmov

ACO algoritmy sú všeobecne stochastické prehľadávacie algoritmy založené na populácii. Používajú sa na riešenie špecifických kombinačných problémov, ktoré sú charakterizované nasledovne:

- prehľadávaný priestor je diskrétny (aj keď už boli vyvinuté aj ACO algoritmy pre spojité prehľadávacie priestory)
- existuje množina ukončovacích ohraničení
- riešenie je zvyčajne usporiadaná postupnosť komponentov
- ohodnocovacia funkcia každému riešeniu dokáže priradiť hodnotu
- je konečná množina komponentov
- je konečná množina možných prechodov medzi komponentmi
- je konečná množina usporiadaní komponentov

Cieľom optimalizačného algoritmu je vytvoriť prípustnú postupnosť komponentov, tak, aby hodnota ohodnocovacej funkcie bola čo najlepšia.

ACO vyžaduje, aby problém, ktorý rieši, bol reprezentovaný grafom, ktorý pozostáva z konečnej množiny uzlov a prepojení medzi nimi. Každý uzol predstavuje jeden komponent a ich prepojenie znamená prechod z jedného uzla do ďalšieho. Každý prechod (prepojenie)

má priradenú hodnotu. Cieľom ACO je nájsť najmenej nákladnú cestu v grafe. Skonštruovaná cesta predstavuje sériu komponentov. Cesta sa vytvára inkrementálne.

Kolónia mravcov sa využíva na paralelné hľadanie rôznych riešení. Mravce prechádzajú graf a každý z nich inkrementálne buduje riešenie, pričom používajú lokálne informácie vo forme feromónového depozitu a heuristickú informáciu, pomocou ktorej sa rozhodujú, ktorú hranu navštívia. Mravce tiež modifikujú koncentráciu feromónov na každej hrane ich cesty, čo umožňuje kooperáciu medzi mravcami. Na to, aby bola vytvorená optimálna cesta, mravce majú tieto vlastnosti a charakteristiky:

- Mravec má pamäť, aby si uložil informácie o jeho vytvorenej ceste. Používa sa najmä na to, aby sa dodržali obmedzenia a mravec nenavštívil rovnaký uzol viac krát. Pamäť sa tiež využíva pri ovplyvňovaní feromónovej hodnoty hrán tejto cesty.
- Každý mravec si pre nový stav určí možných susedov. Obsahuje tie uzly, pre ktoré je definovaná hrana z aktuálneho uzla.
- Každému mravcovi je daný inicializačný stav, ktorý korešponduje štartovaciemu uzlu. Mravce sa môžu do grafu umiestniť náhodne alebo deterministicky.
- Každému mravcovi je priradená jedna alebo viac ukončovacích podmienok. Môže obsahovať napríklad maximálny počet uzlov, cieľový uzol, akceptovateľné riešenie alebo ukončenie pri stagnácii.
- Každý mravec používa pravdepodobnostné prechodové pravidlo pri výbere nového uzla z množiny susedných uzlov.
- Každý mravec má možnosť upraviť koncentráciu feromónov na hranách, ktoré navštívil.

ACO algoritmy majú všeobecne viacero komponentov. Tými sú kolónia mravcov, mechanizmus na generovanie a aktivovanie mravcov, mechanizmus vyparovania feromónov, ukončovacie podmienky, „daemon“ akcie (akcie, ktoré nemôžu byť vykonané jedným mravcom).

Na to aby mohli mravce zostaviť cestu sú potrebné tieto komponenty:

- Prechodové pravidlo (rozhodnutie o nasledujúcom stave)
- Priama aktualizácia feromónov, poznáme 2 typy. Priama postupná aktualizácia feromónov (lokálna aktualizácia) a priama oneskorená aktualizácia (globálna aktualizácia).

Na základe tejto charakteristiky vzniklo niekoľko rôznych ACO algoritmov, ktoré sa líšia v prechodových pravidlách, vo feromónovej aktualizácii, či v implementácii „daemon“ akcií.

1.2.4 Všeobecný Framework

1.2.4.1 Ant Colony Optimization Meta-heuristika (ACO-MH)

Bola vytvorená Dorigom a Di Carom [12,23] a predstavovala jeden z prvých frameworkov pre ACO algoritmy. ACO algoritmy sú rôznymi inštanciami ACO-MH.

Obsahuje tri hlavné komponenty ACO algoritmu a to generáciu mravcov a aktiváciu, vyparovanie feromónov, "daemon" akcie (sú nepovinné), bez toho, aby hovorila o detailoch ako ich vykonať.

1.2.4.2 Ant System Meta-heuristika (AS-MH)

Bola vytvorená Taillardom [24] a je podobná ACO-MH. Hlavným rozdielom je, že AS-MH obsahuje explicitnú referenciu na proces kráľovnej. Proces kráľovnej sa používa v analógii s reálnymi kolóniami mravcov, kde má kráľovná rolu koordinátora pri produkcii optimálneho množstva mravcov na vykonávanie určitých úloh (napr. ochrana, hľadanie potravy). Bez kráľovnej by kolónia zlyhala.

Je založená na predpoklade, že správanie mravca môže byť popísané množinou procesov, ktoré spolupracujú pomocou spoločnej pamäte, vo forme feromónových depozitov. Obsahuje 2 procesy:

- Proces mravcov − skonštruovanie riešenia. Proces každého mravca skonštruuje nejaké riešenie s použitím akcie pseudonáhodného proporcionálneho výberu na určenie nasledujúceho uzla. Po skonštruovaní riešenia sa vykoná zodpovedajúci lokálny prehľadávací algoritmus, ktorý vylepší riešenie. Toto riešenie sa následne vráti do procesu kráľovnej.

- Proces kráľovnej − koordinácia konštrukcie riešení. Všetky feromónové hodnoty sa inicializujú na maximálnu možnú hodnotu τ_{max}, po čom sa aktivujú procesy mravcov. Každý proces mravca vráti riešenie a aktualizuje sa pamäť. To nasleduje po prvej aplikácii vyparovania feromónov, po ktorom sa globálne najlepšie riešenie použije na ovplyvnenie feromónov. Napokon sa proces kráľovnej uistí, že všetky feromónové hodnoty ostali v intervale $[\tau_{min}, \tau_{max}]$.

AS-MH tak ako ju uviedol Taillard priraďuje všetku zodpovednosť za aktualizáciu kolektívnej pamäte výhradne procesom kráľovnej.

1.2.4.3 Ant Programing (AP)

Metóda bola nedávno špecifikovaná Birattarim [25]. Charakterizuje črty ACO pomocou formálnej matematiky pre jednoduchšiu teoretickú analýzu. Je definovaná pre špecifickú triedu optimalizačných problémov, menovite pre diskrétne problémy (problémy najkratšej

cesty). AP vyžaduje aby bol optimalizačný problém definovaný ako viacstupňový rozhodovací proces, podobne ako v dynamickom programovaní. Inštancie AP predstavujú iteratívny algoritmus.

1.2.4.4 Hypercube ACO framework (HC-ACO)

HC-ACO [26] bol navrhnutý pre kombinačné optimalizačné problémy, ktorých riešenia sú kódované binárne. Feromóny sú v tomto prípade asociované s uzlami, nie s hranami.

1.2.5 Aplikácie ACO

Prvý ACO algoritmus AS bol vyvinutý na riešenie klasického problému obchodného cestujúceho. Odvtedy bol ACO algoritmus použitý na riešenie širokej škály rôznych optimalizačných problémov, najmä diskrétnych optimalizačných problémov. Optimalizačné problémy zahŕňajú klasické problémy, ako kvadratické priradenie, plánovanie práce, problém podmnožín, ale aj reálne problémy zo života ako smerovanie v sieti, riadenie dopravy, dolovanie dát, bioinformatika a mnohé iné. Zatiaľ čo ACO algoritmy boli vyvíjané najmä na riešenie diskrétnych optimalizačných problémov, varianty algoritmu boli vyvinuté aj na riešenie spojitých optimalizačných problémov ako aproximácia funkcií, učenie neurónových sietí a lokalizovanie optima spojitých funkcií.

1.2.5.1 Diskrétne optimalizačné problémy:

Problémy usporiadania

ACO sa môže použiť na riešenie problémov, kde riešenie je definované špecifickým poradím komponentov. Patria sem problémy optimalizácie cesty (napr. TSP), problém riadenia dopravy, plánovanie práce, problém najkratšej spoločnej sekvencie, alokácia zdrojov v komunikačných sieťach, zostrojenie evolučného stromu, zostavenie DNA fragmentu a pod.

Problémy priradenia

Zahŕňajú tie optimalizačné problémy, kde cieľom je priradiť hodnoty ku konečnej množine premenných tak, že daná vyhodnocovacia funkcia je optimalizovaná. Patria sem problémy ako segmentácia obrázkov, tréning neurónových sietí, návrh obvodov, priradenie frekvencie a pod.

Problémy podmnožín

Problémy podmnožín sa podstatne líšia od problémov usporiadania a priradenia. Ich cieľom je vybrať najlepšiu podmnožinu n_s z n položiek, kde $n > n_s$. Dané reálne funkcie sú optimalizované a obmedzenia nie sú porušené. Nie je tu koncepcia cesty. Čiastkové riešenie neobsahuje žiadnu postupnosť komponentov. Komponent, ktorý

bude zvolený ako ďalší nie je ovplyvnený tým, ktorý bol zvolený ako posledný do čiastkového riešenia. Riešenia nemusia byť rovnakej veľkosti. Príkladom problémov podmnožín sú problémy batohu a problémy „Hitting-set“.

Problémy zoskupovania

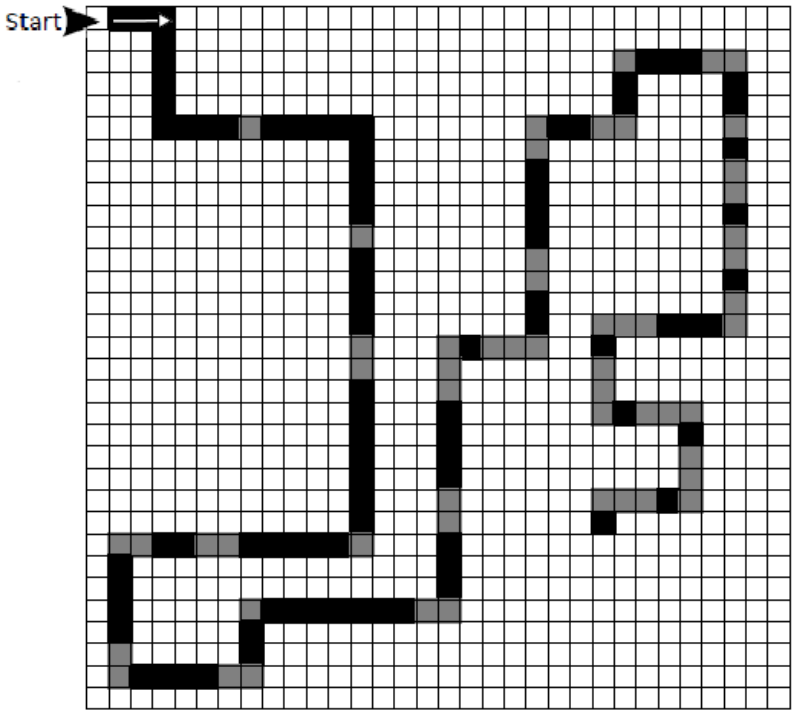
Cieľom problémov zoskupovania je rozdeliť množinu položiek do niekoľkých skupín tak, aby daná ohodnocovacia funkcia bola optimalizovaná a obmedzenia neboli porušené. Používa sa napríklad na problém ofarbovania grafu, klastrovanie dát, alebo problém „Bin packing“.

1.3 Problém hľadania jedla (Food trail problem)

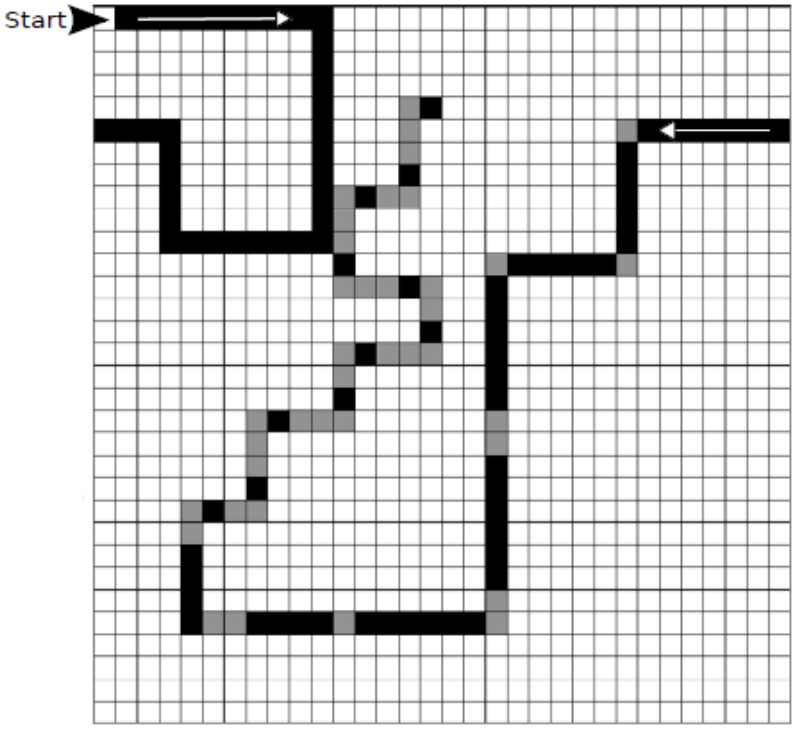
K známym optimalizačným problémom patrí úloha navigácie agenta, s cieľom nájsť čo najviac kusov potravy ležiacej pozdĺž nepravidelnej cesty. Problém hľadania potravy je považovaný za základný problém na testovanie výkonu evolučných algoritmov. Hrá sa na toroidálnej mriežke veľkosti 32 x 32, na ktorej sa nachádza 89 kúskov potravy (jabĺk), rozmiestnených pozdĺž určitej cesty, v jednotlivých bunkách mriežky. Cieľom je nájsť takého agenta v hre, ktorý kúskov potravy nájde čo najviac.

Na začiatku hry je agent umiestnený v ľavom hornom rohu a pozerá na východ, jeho koordináty sú (0,0). Agent dokáže určiť, či nasledujúca bunka obsahuje jedlo alebo nie. V každom kroku môže ísť agent vpravo, vľavo, alebo sa pohnúť vpred.

Mriežka, cesta, pozícia agenta na začiatku hry pre Santa Fe problém je zobrazená na obrázku 1.7, kde čierne štvorce indikujú jedlo, biele sú prázdne a sivé zobrazujú cestu agenta. Pri testovaní bude vyskúšaný aj test pre problém John Muir Food Trail, ktorý je zobrazený na obrázku 1.8.



Obrázok 1.7: Príklad Santa Fe – (Chivilikhin, 2013, (s. 531) [29])



Obrázok 1.8: Príklad John Muir Food trail problem (Chivilikhin, 2012, (s. 274) [1])

2. Súvisiace práce

Problematika vytvárania konečno-stavových automatov pre danú účelovú funkciu vyvolala záujem viacerých odborníkov. V mojej práci sa budem zameriavať najmä na algoritmus vytvorený v článku [1], ktorého autormi sú Daniil Chivilikhin a Vladimír Ulyantsev. V časti 2.1 je rozobraté toto riešenie podrobnejšie. V časti 2.2 sú popísané iné prístupy k riešeniu učenia konečno-stavových automatov. V časti 2.3 sú popísané iné prístupy na riešenie problému „John Muir Food Trail Problem“ a v časti 2.4 sú spomenuté diskrétne problémy, ktoré boli riešené pomocou kolónie mravcov.

2.1 Pôvodné riešenie

V článku [1] bola uvedená úplne nová metóda na učenie konečno-stavových automatov s použitím špecifickej hodnoty danej fitnes funkcie, ktorá je založená na optimalizácii kolóniou mravcov a na grafovej reprezentácii prehľadávaného priestoru. Cieľom algoritmu je maximalizovať danú fitnes funkciu, ktorá je definovaná na množine všetkých konečno-stavových automatov s danými parametrami.

Optimalizačný problém je sformulovaný nasledovne: Nech je daný počet stavov N, množina udalostí $\sum$, množina akcií Δ, ktoré tvoria konečno-stavový automat, s danou cieľovou hodnotou fitnes funkcie f.

Bola vytvorená heuristická metóda lokálneho prehľadávania so zameraním na učenie konečno-stavového automatu pre danú fitnes funkciu, založená na optimalizácii kolóniou mravcov (Ant Colony Optimization - ACO). Efektívnosť riešenia bola porovnávaná s výsledkami genetického algoritmu a test bol zrealizovaný na „John Muir food trial problem“, popísanom v časti 1.3. Novo vytvorená metóda prekonala výsledky získané pomocou genetického algoritmu alebo fungovala rovnako dobre.

V častiach 2.1.1 až 2.1.7 sa práca venuje jednotlivým častiam algoritmu [1] a rozoberá ich podrobnejšie.

2.1.1 Reprezentácia prehľadávaného priestoru

Množina konečno-stavových automatov s danými špecifickými parametrami reprezentuje prehľadávaný priestor algoritmu [1]. Prehľadávaný priestor je reprezentovaný formou orientovaného grafu G a platí, že uzly grafu G predstavujú konečno-stavové automaty a hrany, ktorými sú uzly (konečno-stavové automaty) prepojené predstavujú drobnú zmenu (mutáciu) v štruktúre automatu. Pre každú dvojicu konečno-stavových automatov A_1, A_2 a korešpondujúcu dvojicu vrcholov *u* a *v,* existuje cesta v grafe G z u do v a zároveň aj z *v* do *u,* pokiaľ je možné odvodenie konečno-stavového automatu A_1 z automatu A_2 pomocou jednej mutácie a tiež naopak. Mutácie sú popísané v časti 2.1.3.

2.1.2 **Konštrukcia cesty**

Agent (mravec) si vyberá cesty v grafe a do hrán grafu vkladá feromónové hodnoty, kým sa nedosiahne stagnácia. Riešenie stagnácie je popísané v časti 2.1.6.

V prvom kroku sa vyberá uzol, z ktorého každý agent začína vytvárať svoju cestu. Ak je graf prázdny, náhodne sa vygeneruje počiatočné riešenie a všetky mravce sú umiestnené do uzla spojeného s týmto riešením. Náhodné prvotné riešenie je získané náhodným definovaním prechodovej funkcie konečno-stavového automatu s určitým počtom stavov. V prípade, že počet hrán grafu je vyšší ako nula, štartovací uzol pre mravce je vybraný náhodne z množiny všetkých uzlov v najlepšej ceste (dráha prejdená určitým mravcom, ktorá vedie k doteraz najlepšiemu získanému riešeniu, a teda dráha s najvyššou hodnotou účelovej fitnes funkcie). Následne mravce vytvárajú svoju cestu týmito spôsobmi:

- Expanzia prehľadávaného priestoru: s pravdepodobnosťou danou p_{new} mravec vytvára N_{mut} nových hrán, mutáciami hrany, v ktorej sa aktuálne nachádza. Postup vytvárania mutácií je bližšie popísaný v časti 2.1.3. Po vytvorení všetkých mutácií si mravec vyberie ten najlepší (vzhľadom na fitnes hodnotu) novo skonštruovaný uzol v a presunie sa do tohto uzla.

- Stochastický výber cesty: S pravdepodobnosťou $1 - p_{new}$ mravec stochasticky vyberá niektorého z nasledovníkov N_x hrany x, v ktorej sa aktuálne nachádza, a to na základe ich feromónových hodnôt. Výber nasledujúcej hrany je popísaný v časti 2.1.5.

Keď sa mravec nachádza v takej hrane, ktorá nemá žiadnych nasledovníkov, dochádza ku vytvoreniu nových hrán (expanzii prehľadávaného priestoru) s pravdepodobnosťou rovnou 1.

2.1.3 **Mutácia FSM**

Pokiaľ sa mravec nachádza v uzle, ktorý nemá nasledovníkov, vytvára N_{mut} mutácií uzla, v ktorom sa nachádza. S pravdepodobnosťou p_{new} mravec, ktorý sa nachádza vo vrchole, ktorý má nasledovníkov vytvára nových N_{mut} mutácií aktuálneho stavu. Aktuálne sú v riešení použité tieto druhy mutácií:

- Mutácia prechodu koncového stavu: Zvolený je náhodne jeden z prechodov prechodovej tabuľky. Následne je tento prechod zamenený za iný prechod z množiny stavov.

- Mutácia akcie: Zvolená je náhodne jedna z akcií v prechodovej tabuľke a je nahradená inou akciou z množiny $\{M,R,L\}$.

Mutácia konečno-stavového automatu sa vykoná nasledovne:

- Vytvorenie mutácie konečno-stavového automatu A_{mutated}.
- Nájdenie uzla t v grafe G, ktorý je prepojený s A_{mutated}. Ak graf G takýto uzol neobsahuje bude takýto uzol vytvorený a prepojený s A_{mutated}.
- Pridanie hrany (u, t) do G.

2.1.4 **Výpočet Fitnes FSM**

Je dané, že mravec má nájsť riešenie v menej ako s_{max} (v algoritme [1] je $s_{max} = 200$) krokoch. Vzorec na výpočet fitnes je daný ako:

$$Fitness(FSM_A) = n_{apples} + \frac{s_{max} - s_{last} - 1}{s_{max}}, \tag{2.1}$$

kde FSM_A je konečno-stavový automat, ktorému určujeme jeho fitnes hodnotu. n_{apples} je počet jabĺk, ktoré boli daným konečno-stavovým automatom nazbierané a s_{last} je počet krokov, za ktoré sme dospeli k danému riešeniu (respektíve bol zjedený posledný kúsok jedla) a s_{max} je maximálny počet krokov, ktoré dávame algoritmu na to, aby zjedol toto jedlo (v pôvodnom riešení to bolo 200 krokov).

2.1.5 **Výber nasledujúcej hrany**

Pokiaľ pri konštrukcii cesty dôjde k stochastickému výberu hrany, mravec stochasticky vyberá jedného zo svojich nasledovníkov. Samotné hrany majú pravdepodobnosť výberu danú formulou:

$$p_{uv} = \frac{\tau_{uv}^{\alpha} * n_{uv}^{\beta}}{\sum_{w \in N_u} \tau_{uw}^{\alpha} * n_{uw}^{\beta}}, \tag{2.2}$$

kde $v \in N_u$ a $\alpha, \beta \in [0,1]$. V tomto algoritme sú všetky hrany rovnako dlhé a preto n_{uv} neovplyvní výber cesty, β neovplyvní výber cesty a α je vždy rovná 1.

2.1.6 **Aktualizácia feromónovej hodnoty**

Hrany (*i,j*) grafu *G* disponujú feromónovou hodnotou τ_{ij}^{best}, ktorá predstavuje najvyššiu feromónovú hodnotu, ktorá bola kedy zanechaná na hrane (*i,j*). Pre jej určenie sa pre každú cestu, ktorú mravce vytvorili, vezme „pod-cesta“ od počiatočného uzla až po uzol, ktorý získal najlepšiu hodnotu fitnes na tejto ceste. Hodnoty τ_{ij}^{best} sa aktualizujú pre každú hranu, ktorá sa nachádza na tejto „pod-ceste“. Následne sa počíta feromónová hodnota pre každú hranu (*i,j*) grafu *G* podľa formuly:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau_{ij}^{best} \tag{2.3}$$

kde $\rho \in [0,1]$ predstavuje stupeň odparovania.

2.1.7 **Riešenie stagnácie algoritmu**

Aby pri vykonávaní algoritmu nedochádzalo k stagnácii, sú dané dve ukončovacie podmienky, pričom prvá podmienka je špecifická pre každého mravca zvlášť a druhá je daná pre celú generáciu mravcov v jednej iterácii algoritmu.

- Každý mravec má k dispozícii n_{stags} krokov, ktoré môže prejsť bez navýšenia najlepšej získanej fitnes hodnoty. Pokiaľ dôjde k prekročeniu tejto hodnoty, mravec je zastavený a ukončuje vytváranie cesty.

- Taktiež platí, že celá kolónia mravcov má daný počet iterácií, ktoré sa môžu vykonať bez navýšenia najlepšej fitnes hodnoty získanej v celej populácii mravcov N_{stags}. Pokiaľ je tento počet prekročený, vytváranie cesty je v danej iterácii ukončené pre všetkých mravcov a algoritmus sa reštartuje.

2.2 Iné prístupy na učenie konečno-stavových automatov

V článku [2] bol prezentovaný nový optimalizačný algoritmus MuACOsm, založený na mutácii využívajúci optimalizačný algoritmus „Ant Colony“ na učenie konečno-stavových automatov. Cieľom bolo maximalizovať danú účelovú funkciu, ktorá je definovaná na množine všetkých konečno-stavových automatov s danými parametrami.

Vyvinutá bola nová metóda učenia konečno-stavových automatov, v ktorej sa využíva optimalizácia kolóniou mravcov. Tento problém bol redukovaný na úlohu nájdenia optimálneho uzla v grafe, v ktorom sú vrcholy grafu asociované s konečno-stavovými automatmi a hrany grafu sú asociované s mutáciami konečno-stavových automatov. Efektívnosť algoritmu [2] (vzhľadom k počtu ohodnotení fitnes funkcie, ktoré boli potrebné na nájdenie optimálneho riešenia), bola lepšia v porovnaní s tradičnými algoritmami Artifical Ant Colony. Porovnanie ukázalo, že vyvinutý algoritmus bol niekoľko krát rýchlejší, ako iné riešenia.

Oproti článku [1] sa algoritmus v článku [2] líši v použití (1+1) evolučnej stratégie na vylepšenie náhodne vytvoreného koreňa, v použití heuristickej informácie a minimálnej hranice feromónových hodnôt na vyhnutie sa veľkému poklesu feromónových hodnôt na hranách, ktoré mravcami neboli navštívené, po tom čo boli skonštruované.

V článku [29] autori dosiahli nájdenie 7-stavového konečno-stavového automatu, ktorý nájde riešenie za menej ako 400 krokov, po približne 10500 ohodnoteniach fitnes. V

približne 12% prípadov však algoritmus nebol úspešný (nenašiel riešenie ani za 30000 ohodnotení fitnes)

V článku [3] bol vyvinutý algoritmus, v ktorom boli na učenie konečno-stavových automatov využité evolučné stratégie pri riešení problému „The Competition for Resources Problem“. Bolo dokázané, že konečno-stavové automaty môžu byť efektívnou reprezentáciou agentových stratégií.

2.3 Iné prístupy na riešenie „John Muir Food Trail“ a „Santa Fe Trail“

V článku [34] bol predstavený algoritmus MRTS (Memorized-Random-Tree-Search), ktorý systematicky pracuje s malými stromami v prehľadávanom priestore pomocou genetického programovania. Tieto malé stromy boli použité na generovanie malých pod-problémov genetického programovania. Tento algoritmus bol testovaný na probléme „Santa Fe Trail problem“, ktorý je popísaný v časti 1.3. Najlepší získaný výsledok bol pre tento problém 20696 fitnes ohodnotení.

Problém „John Muir Food Trail“ je popísaný v časti 1.3. Jedným z prístupov k jeho riešeniu je aj náhodný prístup [33], avšak pri náhodnom vygenerovaní bilióna riešení, najväčší počet jabĺk, ktoré dokázalo najlepšie riešenie zjesť, je 81, pričom možné maximum je 89.

Iným prístupom môže byť využitie evolučného algoritmu [33], kde sa vygenerovala náhodná populácia mravcov veľkosti 65536. Každý mravec je reprezentovaný ako konečno-stavový automat a na základe stavov prechádza svoju trasu po mriežke. Fitnes hodnotou mravca je počet jabĺk, ktoré na mriežke našiel. Keď všetky mravce dokončia prechádzanie grafu, vyberie sa 1- 10% najlepších jedincov, ktoré sú prenesené do ďalšej generácie a prebehne na nich proces kríženia a mutácie. Z jedincov, ktoré sú vybrané na kríženie a mutáciu sa vyberajú jedince náhodne, nezáleží pritom na tom, ako boli pri hľadaní potravy úspešné.

V nultej generácii bol týmto algoritmom dosiahnutý výsledok 58 nájdených jabĺk, po 200 generáciách to už bolo 89 jabĺk. Vyvinutý algoritmus teda dokázal nájsť riešenie po 200 generáciách. Ohodnotení fitnes však vzhľadom k veľkosti populácie bolo viac než 13 miliónov.

2.4 Diskrétne problémy a ich riešenie kolóniou mravcov

Článok Doriga a Di Cara [27] poskytol prehľad prác slúžiacich na optimalizáciu diskrétnych problémov, ktoré boli inšpirované hľadaním potravy v kolónii mravcov a uviedol meta-heuristiku optimalizácie kolóniou mravcov (ACO meta-heuristic). Bol opísaný biologický postup mravcov k hľadaniu potravy a boli popísané rôzne aplikácie ACO.

V článku [28] je popísané riešenie problému obchodného cestujúceho (TSP) pomocou optimalizácie kolóniou mravcov. V ACO sú mravce jednoduchí agenti, ktorí v prípade TSP vytvárajú cesty prechodom z jedného mesta do druhého. Mravce z kolónie sú postupne schopné úspešne generovať kratšie cesty pomocou informácií uložených vo forme feromónových hodnôt na jednotlivých hranách. Simulácie ukázali, že tento algoritmus je schopný generovať dobré riešenia na symetrické aj asymetrické inštancie TSP. V práci bola použitá metóda AS, metóda ACS, metóda MMAS a Rank-AS, ktoré sú všetky popísané v časti 1.2.2.2.

3. Návrh riešenia

Definovaný je problém nájdenia potravy na toroidálnej mriežke v čo najmenšom počte krokov, ktorý je popísaný v časti 1.3. Na riešenie zadaného problému je použitý modifikovaný ACO algoritmus popísaný v časti 2.1.

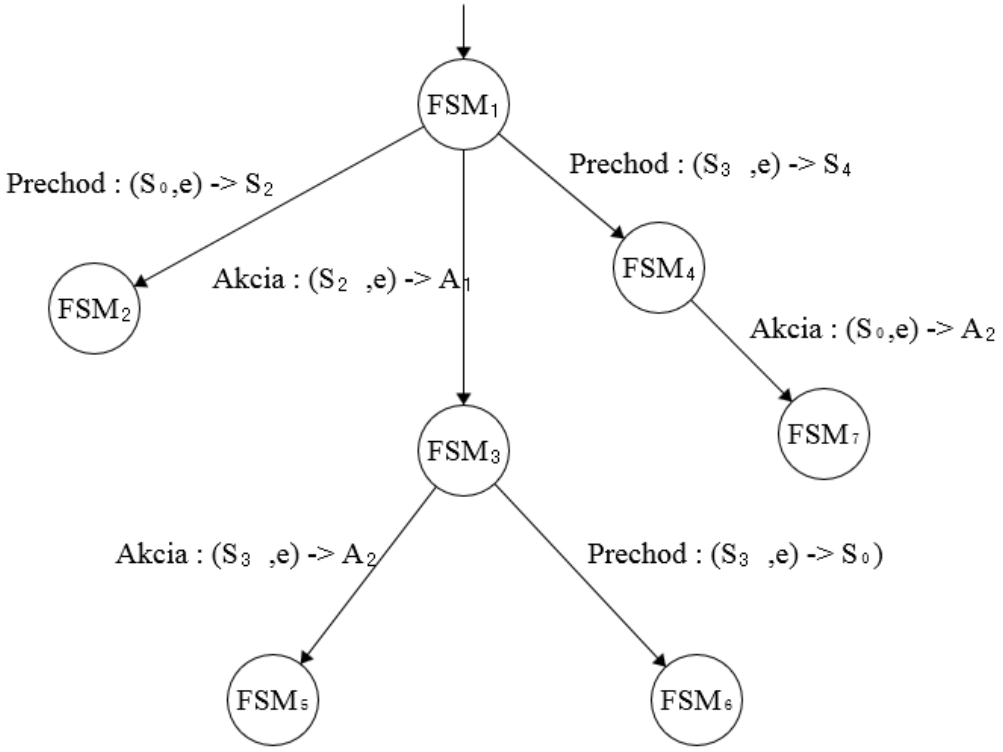
Rozdiel oproti klasickému ACO algoritmu je najmä v konštrukcii grafu. V klasickom ACO algoritme hrany a uzly predstavujú komponenty riešenia. Riešenie mravce vytvárajú postupne v procese hľadania potravy („foraging“). V použitom riešení sa pristupuje ku grafu iným spôsobom. Uzly v grafe predstavujú samotné kompletné riešenia a mravce prechádzajú pomedzi ne. Nové uzly, riešenia, sú vytvárané mutáciami. Takto skonštruované grafy môžu byť veľmi veľké. Keď si predstavíme, že hľadáme FSM, ktorý by mal 7 stavov, graf, ktorý by obsahoval všetky možné riešenia by mohol dosiahnuť veľkosť viac než $3x10^{18}$ uzlov, pričom každý uzol predstavuje jeden FSM, čo je veľmi veľké množstvo na uloženie v pamäti bežného počítača. Aby sme sa tomuto problému vyhli, na začiatku algoritmu je vytvorené jediné riešenie. Postupne mutáciami uzlov mravce rozširujú prehľadávací priestor, až kým nie sú splnené ukončovacie podmienky.

Prehľadávací priestor riešení je množina konečno-stavových automatov (ďalej len FSM z finite-state machine) so zadanými parametrami, ktoré sú reprezentované formou orientovaného grafu, v ktorom je bod asociovaný s FSM. Každá hrana má priradenú feromónovú hodnotu. Platí, že dva FSM (body), FSM_A a FSM_B, sú vzájomne prepojené hranou, pokiaľ ich vzájomne delí jediná mutácia (bližšie v časti 3.1.2). Existuje vtedy hrana z FSM_A -> FSM_B a tiež hrana FSM_B -> FSM_A.

Príklad prehľadávacieho priestoru je zobrazený na obrázku 3.1. V zobrazenom grafe FSM_1 predstavuje inicializačný FSM. Postup jeho vytvorenia je popísaný v časti 3.1.1. Uzly v grafe predstavujú jednotlivé FSM, ktoré sú vytvárané mravcami v procese konštrukcie cesty, popísanej v časti 3.3. Hrany sú prechodmi medzi jednotlivými FSM a predstavujú jedinú mutáciu. Každá hrana grafu je popísaná mutáciou, ktorá vedie k vytvoreniu nového FSM. Hrany sú popísane podľa tejto notácie:

- Prechod : $(S_x, e) \rightarrow S_y$: Znamená, že ak nastane udalosť e, stav S_x sa zmení na stav S_y. e predstavuje udalosť, ktorá je v našom prípade daná tým, či mravec vidí na políčku pred sebou jedlo alebo nie.

- Akcia : $(S_x, e) \rightarrow A_y$: Znamená, že ak nastane udalosť e, zo stavu S_x nasleduje akcia A_y.

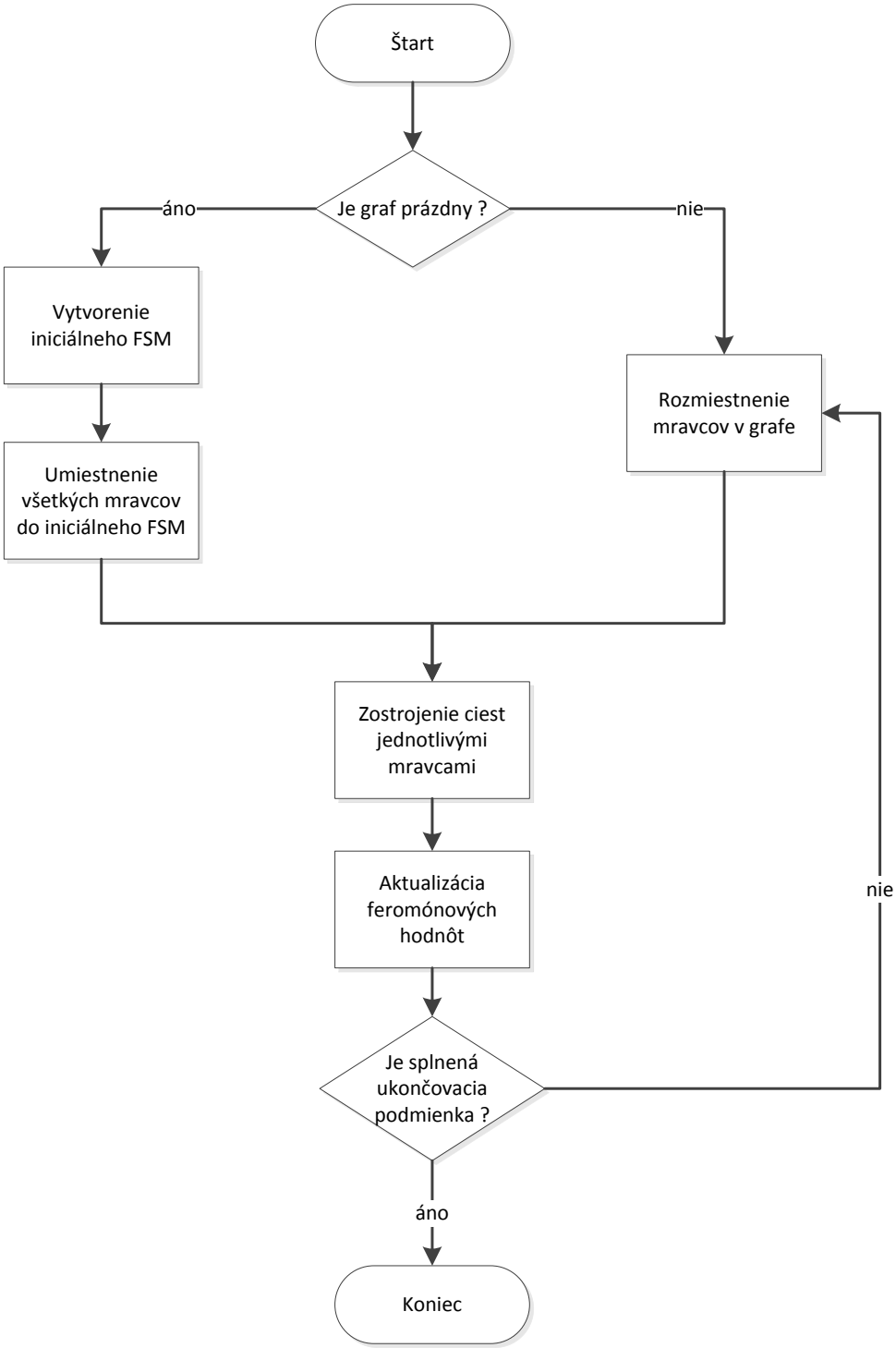


Obrázok 3.1: Príklad prehľadávacieho priestoru

Postup riešenia je zobrazený vo forme diagramu aktivít na obrázku 3.2. Na začiatku programu sa vytvorí iniciálne riešenie (FSM). Postup vytvorenia iniciálneho FSM je popísaný v časti 3.1.1. Všetky mravce sú následne umiestnené do tohto iniciálneho FSM. Pokiaľ graf nie je prázdny, mravce sú rozmiestnené v grafe. Postup rozmiestnenia mravcov v grafe je popísaný v časti 3.2. Po rozmiestnení mravcov, každý z mravov skonštruuje svoju cestu grafom. Postup konštrukcie cesty je popísaný v časti 3.3. Každý mravec aktualizuje feromónovú hodnotu jednotlivých hrán, ktoré navštívi. U feromónových hodnôt dochádza tiež k ich odparovaniu. Postup určenia a aktualizácie feromónových hodnôt v jednotlivých iteráciách je popísaný v časti 3.4.

Každá iterácia končí, keď dochádza k stagnácii algoritmu. Ukončovacia podmienka je v tomto prípade daná počtom krokov N_{stags}, je to počet krokov, ktoré môžu prebehnúť bez zvýšenia najlepšej fitnes hodnoty. Pokiaľ prebehne N_{stags} krokov bez navýšenia najlepšej fitnes, iterácia sa vtedy skončí.

Algoritmus končí, keď je nájdené optimálne riešenie, alebo ak dôjde k vyčerpaniu výpočtových prostriedkov.



Obrázok 3.2: Diagram aktivít pre celkový priebeh algoritmu

3.1 FSM (konečno-stavový automat)

V tejto časti bude podrobnejšie popísaná počiatočná konštrukcia FSM, spôsob mutácie FSM a výpočet fitnes pre daný FSM.

3.1.1 **Konštrukcia FSM**

V pôvodnom riešení [1] sa FSM vytvára na začiatku algoritmu náhodne (keď nemáme daný žiadny uzol grafu) a následne tento vytvorený uzol tvorí koreň grafu. Náhodne sa zvolia prechody a taktiež sa náhodne zvolia akcie, ktoré mravec vykoná. Pri vytváraní FSM, ktorý by mal *N* stavov by FSM bol reprezentovaný ako 4 rozmerné pole s *N* prvkami, pričom prvé dve dimenzie definujú prechod a akciu mravca, ktorý reaguje na jedlo, ktoré vidí pred sebou a ďalšie 2 dimenzie definujú prechod a akciu mravca, ktorý pred sebou jedlo nevidí. Akcia je definovaná pohybom na mriežke. Mravec môže isť rovno (Move forward – M) respektíve pokračovať v smere predošlého pohybu. Môže odbočiť vpravo (Turn right - R), alebo vľavo (Turn left - L).

Príklad takejto prechodovej tabuľky je zobrazený v tabuľke 3.1. Tabuľku možno interpretovať nasledovne. Každý mravec sa na začiatku nachádza v stave S_0, ktorý je počiatočným stavom. Aktuálny stav je teda stav S_0 (riadok vyznačený svetlo sivou farbou). Ak mravec na políčku pred sebou vidí jedlo, prechádza do stavu S_0 (respektíve aktuálny stav sa nemení a ostávame v stave) a na mriežke sa posunie vľavo ‚L'. Ak mravec po presunutí sa na inú pozíciu pred sebou jedlo nevidí znamená to preňho prechod do stavu S_4 a akciou je presun na mriežke vpravo ‚R'. V stave S_4 (riadok vyznačený tmavo sivou farbou) sa teraz pri videní jedla môže presunúť do stavu S_2 a vykonať presun na mriežke vľavo, alebo keď pred sebou jedlo nevidí ostať v stave S_4 a presunúť sa na mriežke vpravo. Ďalej je postup rovnaký.

	Videnie jedla		**Nevidenie jedla**	
Aktuálny stav	Prechod do stavu	Akcia	Prechod do stavu	Akcia
S_0	S_0	L	S_4	R
S_1	S_3	M	S_1	M
S_2	S_1	R	S_2	M
S_3	S_3	L	S_3	L
S_4	S_2	L	S_4	R

Tabuľka 3.1: Príklad FSM

Vygenerovanie náhodného riešenia je jednoduché a rýchle, no nedá sa považovať za najefektívnejšie. Pokiaľ sa vygeneruje riešenie s nízkou hodnotou fitnes, algoritmus potrebuje niekoľko iterácií navyše, aby sa z oblasti nízkej hodnoty fitnes dostal. Keďže sa inicializácia algoritmu vykonáva len raz, jej náročnosť nezohráva veľkú rolu pri náročnosti celkového algoritmu. Pri riešení algoritmu som preto plánovala využiť aj iný prístup k vytvoreniu iniciálneho riešenia respektíve koreňa grafu.

- Jedným z prístupov je vytvorenie $N_{initial}$ náhodných riešení, pričom za štartovný uzol je zvolený najlepší z nich (vzhľadom na hodnotu fitnes).

- Iným prístupom je tiež vytvorenie $N_{initial}$ náhodných riešení. Na vybratie štartovného uzla je však použitý výber pomocou rulety. Pravdepodobnosť na výber má tak každé riešenie. Pravdepodobnosť výberu lepšieho riešenia je však väčšia. Tento prístup dáva

šancu aj riešeniam, ktoré aj keď potenciálne nemajú najlepšiu fitnes hodnotu môžu byť bližšie k správnemu riešeniu ako riešenie s vyšším fitnes.

- Ďalším z prístupov je použitie série mutácií na vygenerovaný koreň.

3.1.2 **Mutácia FSM**

Pokiaľ sa mravec nachádza v stave, ktorý nemá nasledovníkov, alebo robí expanziu stavového priestoru, vytvára N_{mut} mutácií stavu, v ktorom sa aktuálne nachádza. V pôvodnom riešení sa využívajú druhy mutácií popísané v časti 2.1.3.

Pri mojom riešení som zvažovala rozšírenie týchto mutácií o nové typy. Problém by však mohol nastať v tom prípade, že graf, ktorý mravce konštruujú, by mal spĺňať podmienku, že FSM, ktoré vzájomne susedia, sa odlišujú v jednom prvku celej prechodovej tabuľky. Znamenalo by to, že mnou vytvorená mutácia by mala zachovať toto pravidlo a zmutovaný FSM by sa nemal líšiť vo viac ako v jednom prvku. Tým pádom je najvhodnejšie použiť mutácie, ktoré boli použité aj v pôvodnej práci.

3.1.3 **Výpočet fitnes FSM**

Výpočet fitnes hodnoty na základe pôvodného algoritmu [1] je popísaný v časti 2.1.4. Výpočet prebiehal podľa vzorca (2.1). V programe však často dochádzalo k tomu, že algoritmus ostal uväznený v lokálnom optime. Aj keď boli získané všetky kusy jedla, bol prekročený maximálny počet krokov (s_{max}) a algoritmus z tohto miesta nedokázal nájsť východisko, keďže fitnes takéhoto riešenia bola vzhľadom na daný problém pomerne vysoká. Napríklad ak algoritmus nájde všetky kusy potravy za 800 krokov, jeho fitnes je 88, pričom nami vyžadovaná fitnes je minimálne 89, čo predstavuje nájdenie všetkých kusov potravy za menej ako s_{max}. Aj keď sa zdá, že s riešením s hodnotou fitnes 88 sme veľmi blízko k nájdeniu hľadaného riešenia, nie je to tak, len veľmi ťažko sa z takéhoto riešenia dostaneme k lepším hodnotám, a často algoritmus končí v lokálnom optime. Z tohto dôvodu som navrhla nový prístup.

Vo vzorci (2.1) hrá dĺžka cesty mravca minimálnu rolu, keďže sa n_{apples} pohybuje v intervale <0,89> a druhá časť vzorca, ktorá je asociovaná s dĺžkou cesty, pre veľmi veľký počet krokov rastie len veľmi pomaly. K tomuto vzorcu som preto zvolila nový parameter γ, ktorý dáva váhu dĺžke nájdeného riešenia. Kratšie trasy zvýhodňuje a naopak dlhšie sú výraznejšie znevýhodnené. Vzorec by s týmto parametrom vyzeral takto:

$$Fitness(FSM_A) = n_{apples} + \gamma \frac{s_{max} - s_{last} - 1}{s_{max}}. \qquad (3.2.1)$$

Pracujeme s dvoma stratégiami. Prvá pridáva konštantne váhu krokom podľa formuly (3.2.1). V druhej pracujeme s penalizáciou. Tento prístup by mal ešte viac znevýhodniť také automaty, ktoré prekročia maximálny počet krokov. Postupuje sa podľa vzorca:

$$Fitness(FSM_A) = \begin{cases} n_{apples} + \gamma_1 \dfrac{s_{max} - s_{last} - 1}{s_{max}}, & s_{max} > s_{last} \\ n_{apples} + (\gamma_1 + \gamma_2) \dfrac{s_{max} - s_{last} - 1}{s_{max}}, & s_{max} \leq s_{last} \end{cases} \quad (3.2.2)$$

3.2 Rozmiestnenie mravcov v grafe

V práci, z ktorej vychádzam [1] sú na začiatku každej iterácie všetky mravce rozmiestnené po hranách cesty, po ktorej išiel mravec, ktorý dosiahol globálne najlepšie výsledky. Pokiaľ taká cesta neexistuje, a teda sa jedná o prvú iteráciu, náhodne sa vytvorí prvý bod grafu a to náhodným skonštruovaním nového FSM (popísané v časti 3.1.1) a do tohto novo vytvoreného bodu sú umiestnené všetky mravce. Výber počtu mravcov je popísaný v časti 3.2.2.

V mojom riešení je vyskúšaný aj iný prístup a určitá časť populácie mravcov je rozmiestnená v grafe aj na iné pozície ako pozdĺž najlepšej cesty, z dôvodu, že pri pôvodnom riešení často dochádzalo k uväzneniu v lokálnom optime.

Jedným z prístupov je rozmiestnenie mravcov pozdĺž najlepšej cesty dosiahnutej v predošlej iterácii. Vyskúšané sú aj prístupy, keď bude určité percento mravcov umiestnené na globálne najlepšiu cestu a opačné percento na iteratívne najlepšiu cestu. Vyskúšali sme tiež rozmiestnenie mravcov pomocou turnaja (bližší popis v časti 3.2.1), ktorý sa využíva najmä pri evolučných algoritmoch. Pri rozmiestňovaní majú väčšiu šancu na úspech tie hrany grafu, ktoré dosiahli vyššiu hodnotu fitnes. Táto metóda je výpočtovo nenáročná, keďže sa bude pracovať s veľkým grafom a teda veľkým počtom FSM, nebolo by vhodné použiť metódu Ruleta, ktorá je výpočtovo náročnejšia, keďže vyžaduje pred výberom inicializáciu a teda určenie veľkosti výseku v rulete pre každý FSM.

Pracujeme tiež s počtom iterácií, v priebehu ktorých algoritmus stagnuje. Po prekročení určitého množstva globálnych stagnácií, prebehne zmena v rozmiestnení mravcov.

3.2.1 Metóda výberu – Turnaj

Pri *n*-árnom turnaji sa náhodne vyberie *n* jedincov, pričom každý z nich je vybraný s rovnakou pravdepodobnosťou. Títo jedinci následne medzi sebou súperia. Vybraný je najlepší z nich. Jedinec, ktorý turnaj vyhral môže byť vybraný znova a teda nie je odstránený pri ďalšom výbere. Platí tu, že najhorší jedinec turnaj môže vyhrať iba v prípade, že by bol zvolený do turnaja *n*-krát.

3.2.2 **Počet mravcov**

Vplyv počtu mravcov bol študovaný vo viacerých článkoch [32,30]. Viditeľný vplyv má počet mravcov na komplexnosť celého algoritmu, čím viac máme mravcov, tým viac musíme skonštruovať ciest a vypočítať množstvá feromónu, ktoré zanechajú. Napríklad výpočtová zložitosť AS (1.2.2.2) je daná $O(n_c n_G^2 n_k)$, kde $n_c = n_t . n_k$, je celkový počet cyklov, n_t je celkový počet iterácií, a n_G je počet uzlov v riešení. Čím je však počet mravcov menší, tým je aj nižšia schopnosť preskúmať prehľadávací priestor. Mravce navzájom menej komunikujú, a tým máme o priestore menej informácií. Malé n_k tak môže viesť k nájdeniu sub-optimálneho riešenia, alebo k rýchlej stagnácii. Veľmi veľký počet mravcov však tiež nie je nevyhnutne výhodou, celková zložitosť algoritmu sa zvyšuje a dochádza k tomu, že nárast feromónových hodnôt na najlepších trasách nie je dostatočne výrazný. V práci [30] pri riešení problému TSP pomocou ACS (1.2.2.2) dokázali, že $n_k \approx n_G$ pracuje dobre. Dorigo a Gambardella [32] odvodili, že optimálny počet mravcov sa dá určiť na základe:

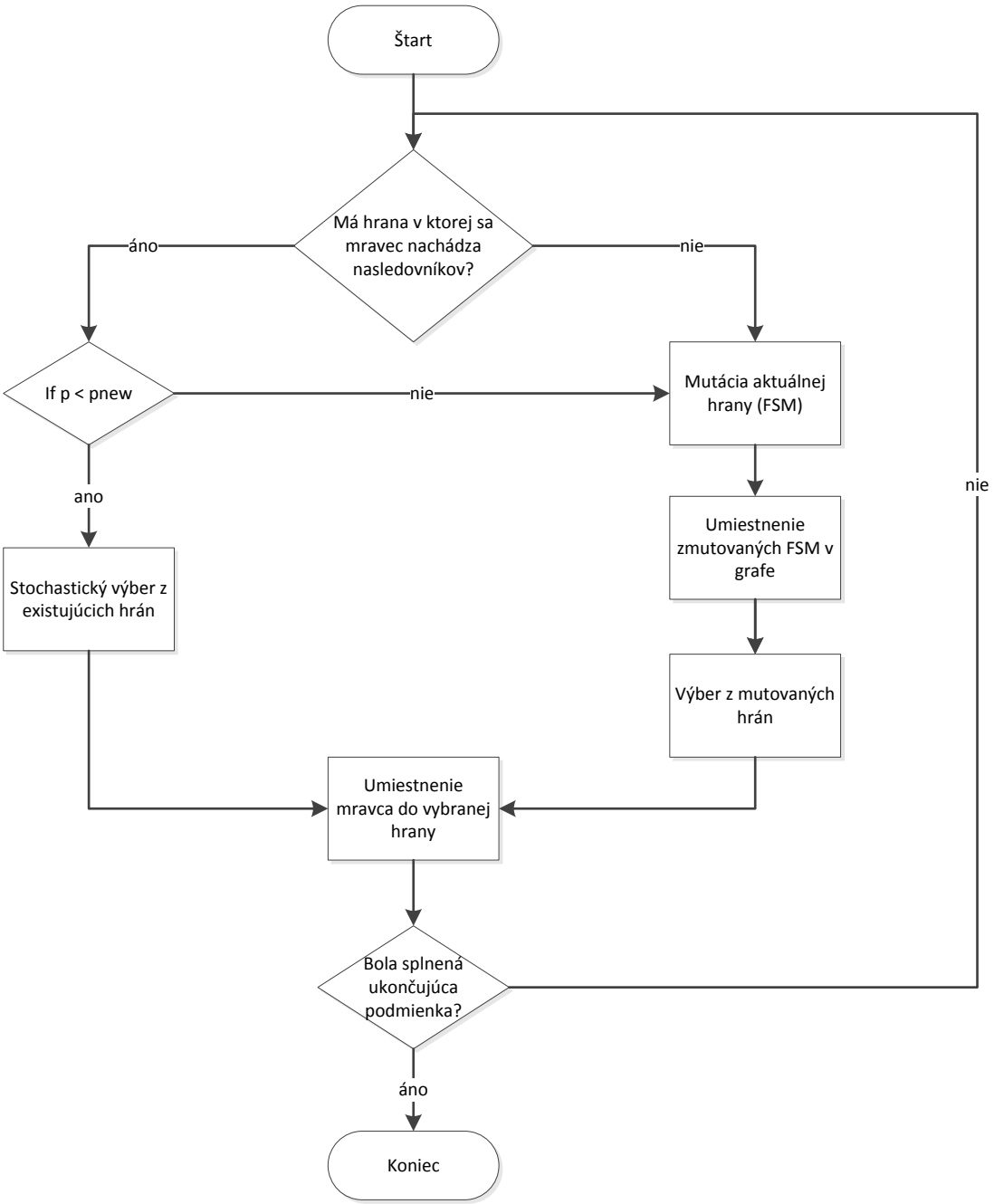
$$n_k = \frac{log(\sigma_1 - 1) - log(\sigma_2 - 1)}{r_0 log(1 - \rho_2)}, \tag{3.3}$$

kde $\sigma_1 \tau_0$ je priemerná koncentrácia feromónu na hranách poslednej najlepšej cesty pred globálnou aktualizáciou a $\sigma_2 \tau_0$ po nej. Empirické analýzy ukázali, že ACS pracuje najlepšie keď $(\sigma_1 - 1)/(\sigma_2 - 1) \approx 0{,}4$, čo dáva $n_k = 10$. Tu však samozrejme záleží na probléme, ktorý riešime a nedá sa povedať, aký počet mravcov n_k bude vhodný. V pôvodnom riešení boli použité hodnoty 5 a 10. V riešení sme sa zamerali aj na iné hodnoty.

3.3 Konštrukcia cesty

Kolónia mravcov prechádza grafom, aby našla riešenie zadaného problému. Každý mravec (agent) si v jednotlivých iteráciách konštruuje svoju vlastnú cestu. Mravec postupne pridáva jednotlivé komponenty do svojho riešenia prechádzaním grafu. Nasledujúcu hranu si mravec vyberá na základe pravdepodobnostného pravidla. Mravce prechádzajú v iterácii graf, až kým všetci nevytvoria kompletné riešenia. Na to, aby sme sa vyhli stagnácii, je použitý postup popísaný v časti 3.3.3. Postup konštrukcie cesty je vyjadrený diagramom aktivít, ktorý je zobrazený na obrázku 3.3.

Vychádzame z toho, že mravce sú už rozmiestnené v grafe (popísané v časti 2.2). Mravec sa nachádza v hrane *x*. Ak predpokladáme, že daná hrana má nasledovníkov, mravec si nasledujúcu hranu môže vybrať dvoma spôsobmi, ktoré sú popísané v časti 3.3.1. V mojom riešení sme sa v tomto smere zamerali na modifikácie pravdepodobnosti p_{new} a jej vplyv na výsledok algoritmu.



Obrázok 3.3: Diagram aktivít pre konštrukciu cesty

3.3.1 **Výber nasledujúcej hrany**

Výber nasledujúcej hrany sa posudzuje rôzne, a pri rôznych formách ACO algoritmov existuje viacero prístupov. V tejto práci by sa venujeme aj iným formám výberu ako boli použité pri pôvodnom algoritme.

Výber nasledujúcej hrany v pôvodnej práci je popísaný v časti 2.1.4. Keďže však bola v pôvodnej práci použitá heuristická vzdialenosť $\eta = 1$, ktorú mala určenú každá hrana, je možné túto heuristickú vzdialenosť vo výpočte úplne zanedbať. V tomto prípade by vzorec

zodpovedal výpočtu nasledujúcej hrany, ktorý bol použitý pri vôbec prvom ACO algoritme SACO (Simple Ant Colony Optimization časť 1.2.2.1) a teda vzorcu (1.2).

Parameter α dáva vyšší význam feromónovej hodnote s tým, že pre veľké α majú výsledky tendenciu konvergovať k sub- optimálnemu riešeniu.

V algoritme AS popísanom v časti 1.2.2.2 už je zahrnutá heuristická informácia (vzorec (1.6)) a je vyjadrená formou:

$$\eta_{ij} = \frac{1}{d_{ij}}, \tag{3.4}$$

kde d_{ij} je dĺžka hrany *ij*, v tomto prípade by samotná dĺžka hrany mohla byť posudzovaná ako rozdiel medzi fitnes hodnotou uzla *i* a uzla *j*. Bližšie vyjadrenie k popisu získavania heuristickej vzdialenosti je uvedené v časti 3.5.

Pri ACS sa využíva určitá forma elitizmu (vzorec (1.12)) a mravec si s určitou pravdepodobnosťou priamo vyberá najlepšiu hranu. S opačnou pravdepodobnosťou si však hranu vyberá na základe pravdepodobnostného pravidla vyjadreného vzorcom (1.13). Na rozdiel od AS je pri tomto výbere $\propto$ zanedbaná, respektíve rovná 1.

V riešení som tiež použila dve modifikácie ACS. V prvej modifikácii by si mravec s určitou pravdepodobnosťou vybral nasledujúcu hranu úplne náhodne, a v druhej modifikácii by si ju zvolil turnajom.

Pri ANT-Q prístup ku výberu hrany je podobný ako pri ACS, s tým rozdielom, že sa dá využiť aj prístup, kde sa s určitou pravdepodobnosťou vyberie najlepšie riešenie a s opačnou pravdepodobnosťou sa vyberie ďalšia hrana úplne náhodne. V tomto smere som vyskúšala prístup, kde by boli stanovené dve hranice r_0 a r_1, kde r0<r1 a bola by náhodne určená hranica r. Ak platí, že $r \leq r_0$ vybrali by sme najlepšiu hranu. Ak by $r_0 < r \leq r_1$ vybrali by sme nasledujúcu hranu na základe klasického pravdepodobnostného pravidla definovaného formulou (1.13). V prípade, že by $r > r_1$ vybrali by sme nasledujúcu hranu úplne náhodne.

V riešení som tiež použila modifikáciu ANT-Q, a ak by $r > r_1$ nasledujúcu hranu by som vybrala turnajom.

3.3.2 **Stratégie kolónie.**

V pôvodnom článku stratégie mravcov neboli popísané, v mojom riešení som sa ich rozhodla využiť a ich výsledky porovnať. Stratégie kolónie boli popísané v článku [29]. Stratégia výberu cesty predstavuje stratégiu jedného individuálneho mravca. Sú však stratégie, ktoré popisujú správanie celej kolónie. Stratégia popisuje kedy a ako sa jednotlivým mravcom umožní pohyb v grafe. Popisujú sa dve používané stratégie:

- Postupná stratégia mravcov: Všetky mravce budujú svoju cestu postupne, nasledujúc po sebe navzájom. Každý mravec prebehne ním zvolenú cestu, až kým sa nezastaví, a až po ukončení vytvárania cesty sa vyberie nasledujúci mravec, ktorý začne svoju cestu vytvárať.

- Paralelná stratégia mravcov: Každý mravec spraví v jednotlivých iteráciách jeden krok, až pokým sa všetky mravce nerozhodnú zastaviť. Mravce teda vytvárajú svoje riešenie navzájom paralelne, nejedná sa však o paralelizáciu algoritmu.

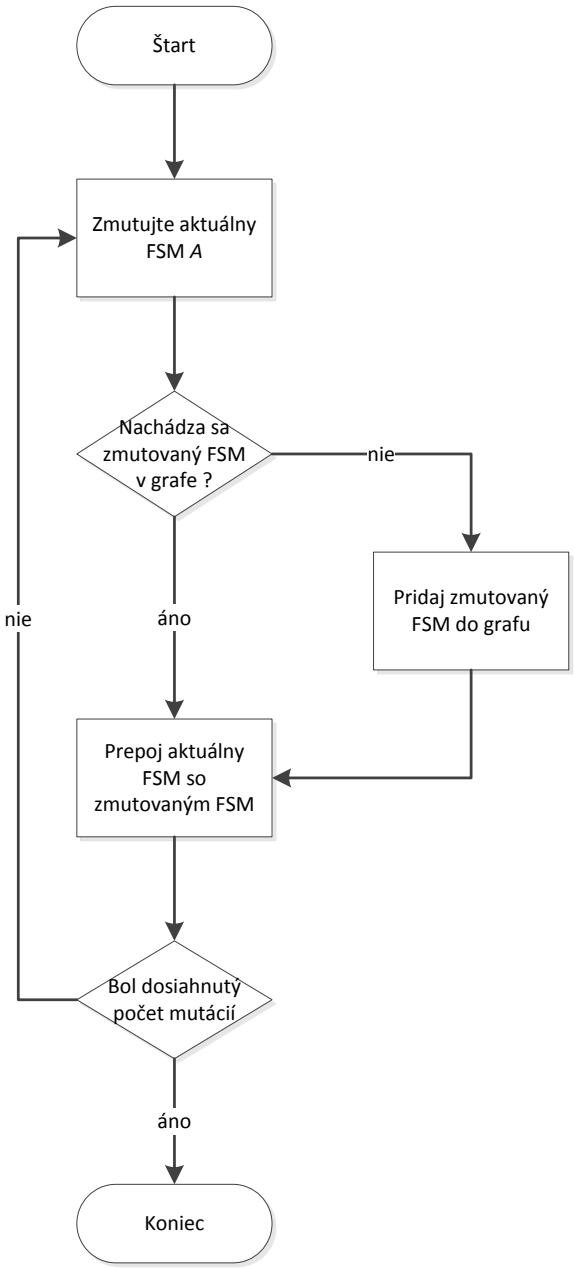
Pokiaľ mravce modifikujú feromónové hodnoty, až keď všetky mravce preskúmajú graf, nezáleží na tom, ktorá stratégia sa využije. Výber z týchto stratégií má vplyv na riešenie problému v prípade, že mravce modifikujú feromónové hodnoty už počas vytvárania svojho riešenia.

3.3.3 **Vyhnutie sa stagnácii**

Aby sme sa pri vykonávaní algoritmu vyhli stagnácii, sú dané dve ukončovacie podmienky, pričom prvá podmienka je špecifická pre každého mravca a druhá je daná pre celú generáciu mravcov v jednej iterácii algoritmu. Tieto podmienky sú popísané v časti 2.1.7, tak ako boli definované v pôvodnom riešení, v ktorom pre premenné n_{stags} a N_{stags} boli použité hodnoty 5, 10 a 20. V mojom riešení sme vytvorili histogram, ktorý predstavuje, koľkokrát sa pre daný bod stagnácie podarí zlepšiť fitnes hodnotu, na základe toho sa pokúšame určiť vhodné N_{stags}.

3.3.4 **Tvorba mutovaných FSM**

Na obrázku 3.4 je zobrazený postup pri vytváraní mutácií aktuálneho stavu. Mutácia FSM je popísaná v časti 3.1.2, je to malá zmena v štruktúre FSM. Po vytvorení mutovaného FSM sa prehľadaním stavového priestoru zisťuje, či sa daný FSM už v grafe nachádza. Pokiaľ sa v grafe už nachádza, vytvoríme hranu medzi aktuálnym FSM a nájdeným FSM. Pokiaľ sa v grafe doposiaľ takýto uzol (FSM) nenachádza, vytvoríme novú hranu grafu a prepojíme ju s aktuálnym FSM.



Obrázok 3.4: Diagram aktivít pre tvorbu mutovaných FSM

3.4 Feromónová hodnota hrany

V ACO algoritmoch, každé prepojenie, alebo každá hrana (*i, j*) konštruovaného grafu má priradenú feromónovú hodnotu τ_{ij}. Feromónová hodnota predstavuje dlhodobú pamäť kolónie. Priradené feromónové hodnoty sa behom algoritmu aktualizujú. Môžu svoju hodnotu zvyšovať, keď nimi prejde mravec (agent), ktorý napríklad konštruuje dobré riešenie, alebo sa môže znížiť vplyvom vyparovania.

3.4.1 **Určenie počiatočnej feromónovej hodnoty:**

Získanie počiatočnej feromónovej hodnoty v článku [1] nie je popísané, na základe toho som sa rozhodla autora článku kontaktovať pričom som zistila, že počiatočná feromónová hodnota je určená na základe počtu jabĺk, ktoré daným FSM vieme získať. Počiatočná feromónová hodnota sa určuje pri vytváraní nových FSM pri expanzii prehľadávaného priestoru popísanom v časti 2.3. Pokiaľ máme daný FSM_A, ktorý je asociovaný s uzlom *i*, a expanziou stavového priestoru vytvoríme mutáciou FSM_A nový FSM_B, ktorý bude asociovaný s hranou *j*, tak feromónová hodnota, ktorá bude priradená hrane (*i, j*) bude daná ako

$$\tau_{ij} = n_{apples}(FSM_B)$$

a naopak

$$\tau_{ji} = n_{apples}(FSM_A),$$

kde n_{apples} je počet vyzbieraných jabĺk daným FSM.

3.4.2 **Aktualizácia feromónovej hodnoty**

V každej iterácii je feromónová hodnota jednotlivých hrán aktualizovaná. V pôvodnom riešení [1] sa postupovalo tak, ako je popísané v časti 2.1.6. V literatúre je však popísaných množstvo iných prístupov k výpočtu modifikácie feromónovej hodnoty.

Pri algoritme EAS (Elitist Ant System) [30] sú feromónové hodnoty jednotlivých hrán aktualizované podľa formuly:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{N_{ants}} \Delta\tau_{ij}^{k} + w_{elit} * \Delta\tau_{ij}^{best}, \tag{3.5}$$

kde $w_{elit} \in [0,1]$ dáva váhu elitistickému riešeniu a platí, že

$$\Delta\tau_{\mathrm{ij}}^{\mathrm{k}} = \begin{cases} f(s_k), (i,j) \in L_t^k \\ 0, inak \end{cases}, \tag{3.6}$$

a

$$\Delta\tau_{\mathrm{ij}}^{\mathrm{best}} = \begin{cases} f(s^{best}), (i,j) \in L_t^{best} \\ 0, inak \end{cases}, \tag{3.7}$$

kde s_k je *k*-te riešenie v aktuálnej iterácii a L_t^{best} je množina prepojení prejdených na najlepšej získanej ceste. Keď porovnáme tento prístup s pôvodným, tak v pôvodnom sa využíva len najlepšie získané riešenie a $w_{elit} = 1$. Pri oboch prístupoch je spodná hranica zanechaného feromónu udržiavaná nad hranicou $\tau_{\min}$.

Vhodné by mohlo byť aj použitie viacerých riešení, ktoré dosiahli uspokojivé výsledky, a teda použitie viacerých najlepších trás získaných v danej iterácii, a teda v čase *t*.

V algoritme SACO, popísanom v časti 1.2.2.1 sa postupuje podľa vzorca (1.4), a všetky mravce, ktoré hranou prešli majú vplyv na výslednú feromónovú hodnotu v danej iterácii. Pri SACO sa využíva aj odparovanie dané vzorcom (1.5).

V algoritme AS na rozdiel od SACO využívame malú pozitívnu konštantu Q, ktorá sa využíva pri určovaní zmeny feromónovej hodnoty (Ant cycle vzorec (1.9)). „Ant density" (vzorec (1.10)) a „Ant quantity" (vzorec (1.11)) nebudeme uvažovať, keďže prvé zmienené berie ohľad iba na to, či danou hranou mravec prešiel bez ohľadu na to, aké riešenie vytvoril a druhé zmienené berie do úvahy iba dĺžku vytvoreného riešenia.

V algoritme ACS (1.2.2.2) feromónovú hodnotu ovplyvňuje iba mravec, ktorý v danej iterácii dosiahol najlepšie riešenie. Feromón sa aktualizuje na základe formuly (1.14), kde je zmena feromónu vyjadrená formulou (1.15). Najlepšia cesta môže byť posudzovaná globálne, a teda berie sa tá, ktorá bola najlepšia doteraz vo všetkých iteráciách, alebo sa môže vybrať cesta, ktorá je najlepšia v danej iterácii.

V algoritme MMAS (1.2.2.2), sú feromónové hodnoty dané do určitého intervalu aby sme sa vyhli stagnácii. Feromónové hodnoty môže ovplyvniť iba najlepší mravec, tak ako pri ACS. Na začiatku sú všetky hodnoty feromónových trás nastavené na maximálnu možnú hranicu. Tento prístup pre náš problém pravdepodobne nebude ideálny, keďže graf je konštruovaný postupne a nie ako pri klasickom ACO algoritme, kde je graf vytvorený už počas inicializácie. Zaujímavým však na MMAS je prístup k aktualizácii feromónu globálne a iteratívne najlepším riešením. Keby bolo použité iba samotné globálne najlepšie riešenie, limitovali by sme preskúmanie nových trás a prehľadávali by sme iba okolie tohto globálne najlepšieho riešenia Pri MMAS sú spomenuté stratégie, kde sa využívajú obe prístupy. Štandardne sa využíva najlepšie riešenie v iterácii no s určitou frekvenciou (ktorá sa behom algoritmu zvyšuje) sa využíva aj globálne najlepšie riešenie. Tento prístup bude vyskúšaný aj v našom riešení.

Úplne iný prístup využíva algoritmus ANTABU (1.2.2.2), je použitá formula 1.20, kde feromón aktualizuje každý mravec.

AS-rank (1.2.2.2) využíva elitistické mravce. Aktualizácia feromónu je vykonaná na základe formuly (1.21) a príslušných podformúl ((1.22),(1.23),(1.24)). Je použitých n_e elitistických mravcov, ktoré môžu feromónovú hodnotu aktualizovať. Oproti predošlým zmieneným prístupom je rozdiel práve v tom, že sa využíva viacero najlepších mravcov a nie len jeden najlepší.

3.5 Heuristická informácia hrany

V ACO algoritmoch je bežné, že hrany majú okrem feromónovej hodnoty priradenú aj heuristickú informáciu η. Heuristická informácia predstavuje apriórnu (nezávislú od skúseností) znalosť o probléme. Na rozdiel od feromónovej hodnoty (kapitola 3.4), ktorá sa behom algoritmu mení a aktualizuje, heuristická informácia zostáva po priradení nezmenená.

V pôvodnom riešení [1], z ktorého práca vychádza, sa s heuristickou hodnotou hrany nepracuje. Heuristická informácia hrany je stanovená pre každú hranu na 1, a teda neovplyvňuje výber cesty a mohla sa zo vzorcov úplne zanedbať. Dá sa povedať, že je ťažké definovať rozumné stanovenie heuristickej informácie pre problém učenia konečno-stavových automatov.

Nový prístup k riešeniu priniesol článok [2], kde sa hrane asociovala aj heuristická informácia špecifická pre jednotlivé hrany. Heuristická vzdialenosť je statická hodnota hrany. Heuristická vzdialenosť hrany $i \rightarrow j$, kde i a j sú uzly v konštruovanom grafe G, sa posudzuje na základe rozdielu fitnes hodnôt, ktoré sú asociované s týmito hranami. Heuristická informácia η_{ij} sa určí ako

$$\eta_{ij} = \max\left(\eta_{min}, f(i) - f(j)\right), \tag{3.8}$$

kde η_{min} je malá pozitívna konštanta. Zaistíme tým, že heuristická informácia bude vždy pozitívna.

4. Implementácia

4.1 Implementačné prostredie

Program na učenie konečno-stavových automatov je vytváraný v implementačnom prostredí Microsoft Visual Studio 2010 v programovacom jazyku C#. Tento jazyk patrí do tretej úrovne abstrakcie programovacích jazykov. Túto kategóriu tvoria procedurálne a objektovo orientované jazyky. Jazyk C# je v dnešnej dobe relatívne často používaný, poskytuje vhodnú pamäťovú a časovú zložitosť. Jedným z dôležitých kritérií pri výbere implementačného jazyka bola najmä znalosť tohto jazyka.

Program je vytváraný a testovaný na počítači s procesorom Intel Core i7-2630QM 2,9GHz s operačnou pamäťou 8GB na 64-bitovom operačnom systéme Windows 7.

4.2 Opis hlavných častí programu

Program je rozdelený na dve hlavné balíky Models, Controllers a triedy:

- Program.cs – spustiteľná časť programu s metódou Main(). Spúšťanie jednotlivých behov programu, výstup programu.
- Parameters.cs – obsahuje nastavenia všetkých parametrov algoritmu. Podrobnejšie sú jednotlivé parametre opísané v Prílohe B.
- Statistics.cs – obsahuje premenné potrebné pre štatistické vyhodnotenie programu.
- FSMLearning.cs – obsahuje logiku celého algoritmu. Spustenie behu jedného algoritmu.

4.2.1 Balík Models

Balík obsahuje triedy, ktoré predstavujú jednotlivé objekty, využité v programe. Obsahuje triedy:

- Ant.cs – predstavuje objekt mravca. Každý mravec má priradenú cestu (objekt triedy Path.cs), tabulist, najlepšiu fitnes, ktorú na danej ceste získal, index tejto hodnoty na ceste, a tiež počet krokov, ktoré pri vytváraní cesty stagnuje.
- Edge.cs – predstavuje objekt hrany. Hrana má priradený vrchol, do ktorého smeruje, feromónovú hodnotu, najlepšiu feromónovú hodnotu, aká bola v danej hrane kedy získaná, heuristickú hodnotu, a kvôli určitým stratégiám aktualizácie feromónov aj sumu feromónov, ktoré na nej boli zanechané.
- FSM.cs – predstavuje objekt konečno-stavového automatu. Má priradenú prechodovú tabuľku, fitnes hodnotu, počet nazbieraných kusov potravy na mriežke a počet krokov, ktoré boli potrebné na nazbieranie daného počtu kusov potravy.

- Hash.cs – predstavuje objekt hashovacej tabuľky. Obsahuje uzol grafu (Vertex.cs) a jeho identifikátor.
- Path.cs – predstavuje objekt reprezentujúci cestu v grafe. Obsahuje koreň danej cesty (Vertex.cs), hrany, ktorými prechádza (list Edge.cs) a najlepšiu získanú hodnotu na tejto ceste s jej indexom.
- Position.cs – predstavuje pozíciu mravca na mriežke. Obsahuje vertikálnu a horizontálnu polohu mravca na mriežke.
- Vertex.cs – predstavuje uzol grafu. Obsahuje konečno-stavový automat, ktorý daný uzol reprezentuje (FSM.cs), susedov daného uzla (list Edge.cs), predchodcu daného uzla (Edge.cs) a jeho jedinečný identifikátor.

4.2.2 **Balík Controllers**

Balík zahŕňa funkcionalitu programu. Obsahuje nasledujúce balíky:

FSMMethods: balík obsahuje triedy, ktoré pracujú s konečno-stavovými automatmi.
- FSMCreation.cs – trieda, ktorá sa používa na vytvorenie konečno-stavových automatov.
- FitnessCounter.cs – trieda na výpočet fitnes hodnoty konečno-stavového automatu.
- FSMComparer.cs - trieda na porovnávanie konečno-stavových automatov.
- FSMMutation.cs – trieda na mutáciu konečno-stavových automatov.

GraphController – balík obsahuje triedy na prácu s grafom.
- PheromonUpdate.cs – trieda na aktualizáciu feromónových hodnôt hrán.

PathController: balík obsahuje triedy, ktoré slúžia na vytváranie cesty mravcom.
- PathConstruction.cs – trieda na vytvorenie cesty v grafe.
- PathSelection.cs – trieda na výber nasledujúcej hrany pri vytváraní cesty v grafe.
- AntsLayout.cs – trieda slúži na rozmiestnenie mravcov v grafe.

ToroidalFieldController
- SetToroidalField.cs – trieda na vytvorenie toroidálnej mriežky pre daný problém.

VertexController
- VertexComparer.cs – trieda na porovnávanie uzlov grafu.
- RootCreation.cs – trieda na vytvorenie koreňa grafu.
- RootReader.cs – čítanie koreňov grafu zo súboru.

5. Riešenie

V tejto časti je popísaný postup zmeny parametrov algoritmu [1]. **Riešenie je rozdelené do dvoch častí**. V prvej časti sa skúma vplyv jednotlivých zmien parametrov na algoritmus [1] a pozoruje sa, ktorá zo zmien má najväčší vplyv na výsledok algoritmu. Druhá časť je zameraná na celkový návrh vylepšenia parametrov algoritmu [1] a jednotlivé zmeny sa vykonávajú iteratívne, v nadväznosti na predošlé zmeny, s cieľom čo najviac znížiť počet fitnes ohodnotení algoritmu. Testovaný je vplyv zmeny jednotlivých parametrov na počet fitnes ohodnotení, za ktoré dospejeme k očakávanému výsledku.

Pre vyššiu presnosť je, pre každé nastavenie parametrov, vykonaných 100 nezávislých behov algoritmu. Jednotlivé testy sú porovnávané na základe priemerného počtu ohodnotení, ktorý bol potrebný na získanie hľadaného konečno-stavového automatu. Pre každé meranie je uvedený priemerný počet fitnes ohodnotení po 100 behoch, počet behov, v ktorých sme nedosiahli výsledok ani po 100000 ohodnoteniach a tiež koľkokrát bol v 100 behoch prekročený počet ohodnotení 30000. Pokiaľ algoritmus nedosiahne výsledok ani po 100000 ohodnoteniach, algoritmus končí a tento výsledok nie je zarátaný do priemeru.

V každom pokuse bol hľadaný konečno-stavový automat, ktorý bude obsahovať 7 stavov, a ktorý by našiel cestu za menej ako 400 (s_{max}) krokov. Ak nie je povedané inak používa sa problém hľadania cesty „Santa Fe trail“, popísaný v časti 1.3.

Vo výsledku sa získané výsledky porovnajú z výsledkami dosiahnutými v algoritme [1], v tomto prípade však autori článku hľadali konečno-stavový automat, ktorý by našiel riešenie za menej ako 200 krokov. Ich algoritmus dokázal nájsť vhodný automat so 7 stavmi 2-krát, a to po 143 a 221 miliónoch fitnes ohodnotení. V mojom riešení som sa rozhodla hľadať konečno-stavový automat, ktorý by našiel vhodné riešenie za menej ako 400 krokov a porovnať ho s výsledkami, ktoré dosiahne pre s_{max} 400 pôvodný algoritmus [1].

Porovnávať sa budeme aj s výsledkami získanými v algoritme [2], ktorý už je vylepšením algoritmu [1] a je vytvorený rovnakými autormi pričom autori použili s_{max} 400, 500 aj 600. Dosiahnuté výsledky porovnáme aj s algoritmami [29] a [34].

5.1 Časť 1: Vplyv zmeny parametrov na algoritmus [1]

V časti 5.1.1 až 5.1.10 sa zameriavame na zmenu jednotlivých parametrov navrhnutých v časti Návrh riešenia (kapitola 3). Ich vplyv na algoritmus [1], z ktorého práca vychádza, sa skúma samostatne. Zameriavame sa na stratégie kolónie (5.1.1), výpočet fitnes (5.1.2), generovanie koreňa grafu (5.1.3), rozmiestnenie mravcov v grafe (5.1.4), zmenu počtu krokov stagnácie N_{stags} (5.1.5), typ aktualizácie feromónových hodnôt (5.1.6), pridanie heuristickej informácie (5.1.7), výber nasledujúcej hrany (5.1.8), zmenu pravdepodobnosti P_{new} (5.1.9) a zmenu počtu mravcov N_{ants} (5.1.10).

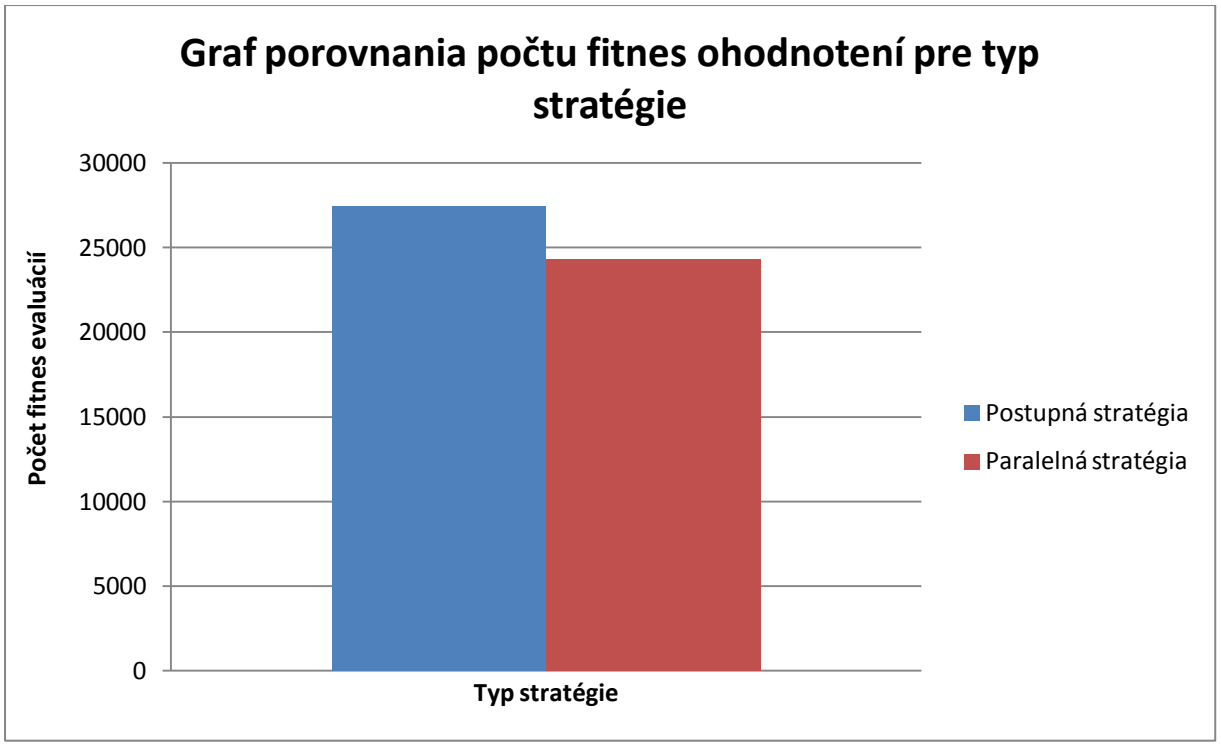
5.1.1 Zmena stratégie. Manažment riadenia mravcov

V prvom kroku sa zameriavam na stratégiu kolónie. Je to najmä z dôvodu, že v pôvodnom článku [1] nie je stratégia zmienená. Vychádzať budeme z tej stratégie, v ktorej získame lepšie výsledky. Podrobnejšie je stratégii mravcov venovaný popis v časti 3.3.2. Táto zmena nemá vždy vplyv na algoritmus kolónie mravcov. V tomto prípade, keď mravce zanechávajú feromónové hodnoty už počas vytvárania cesty a graf je vytváraný postupne, však vplyv má.

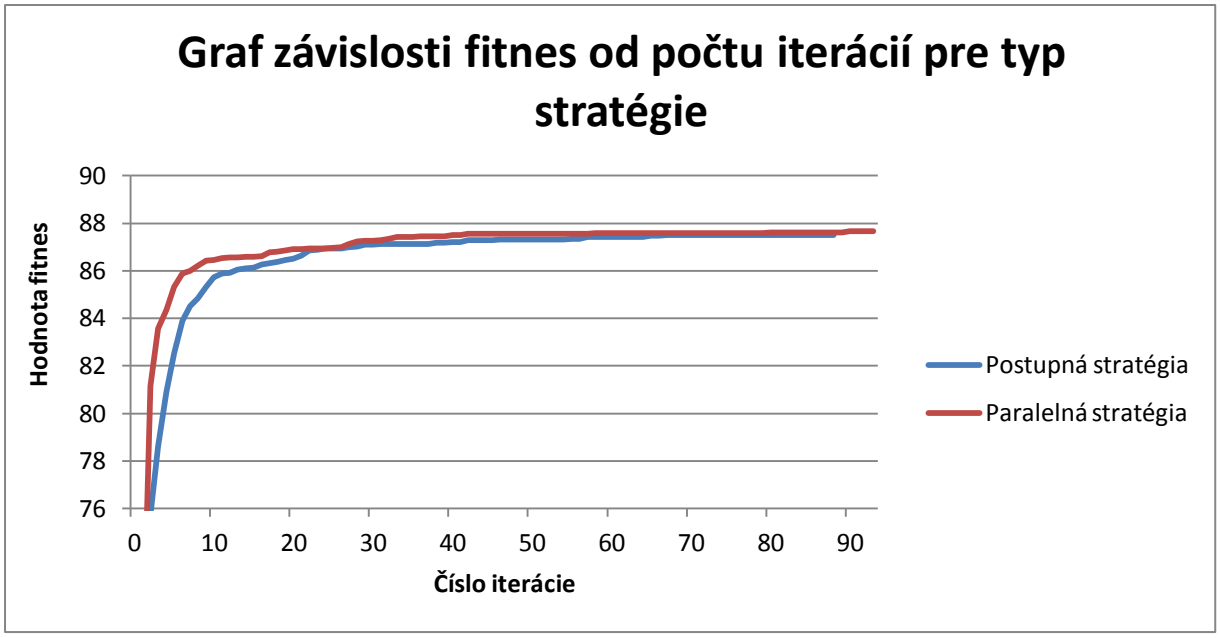
Tabuľka 5.1 reprezentuje výsledky získané postupnou a paralelnou stratégiou (nejedná sa o paralelizáciu, ale o súbežné vytváranie cesty. Mravec nevykoná ďalší krok kým nevykonal krok mravec pred ním). Graf 5.1 reprezentuje graf porovnania počtu fitnes ohodnotení pre daný typ stratégie. Graf 5.2 predstavuje graf závislosti fitnes od počtu iterácií algoritmu.

Číslo stratégie	Typ stratégie	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	Postupná	4	35	27401
2	Paralelná	4	30	24292

Tabuľka 5.1: Porovnanie výsledkov pre typ stratégie



Graf 5.1: Graf porovnania počtu fitnes ohodnotení pre typ stratégie



Graf 5.2: Graf závislosti fitnes od počtu iterácií pre typ stratégie

Lepšie výsledky boli dosiahnuté paralelnou stratégiou. Celkovo bol dosiahnutý priemer 24292 ohodnotení a 4-krát sa nám nepodarilo dospieť k výsledku ani po 100000 ohodnoteniach. Budeme predpokladať, že v pôvodnom algoritme [1] bola použitá paralelná stratégia a budeme sa porovnávať s týmto výsledkom.

5.1.2 **Zmena výpočtu fitnes hodnoty**

Výpočet fitnes hodnoty na základe pôvodného algoritmu [1] je popísaný v časti 2.1.4. Výpočet prebiehal podľa vzorca (2.1). Na výpočet fitnes som navrhla nový prístup, popísaný v časti 3.1.3. Bude sa postupovať podľa vzorca (3.2.1) a (3.2.1). Do vzorca na výpočet fitnes bol pridaný parameter, ktorý pridá vyššiu váhu dĺžke skonštruovaného riešenia.

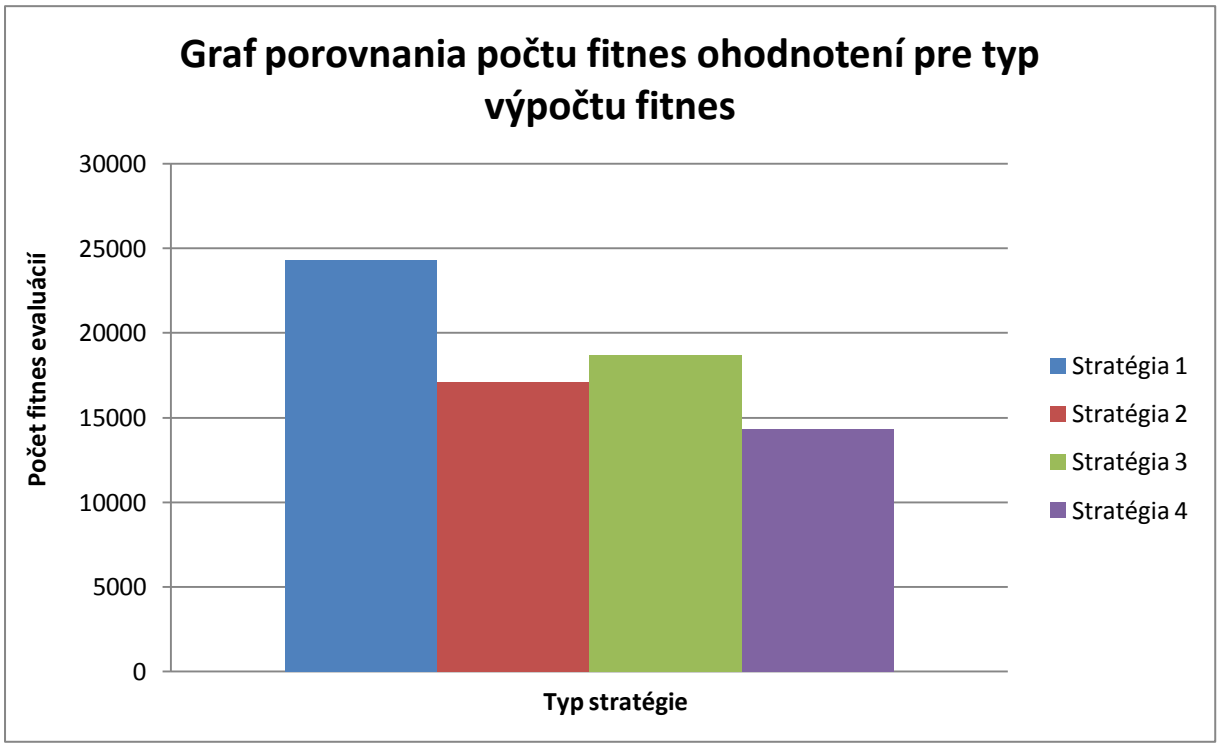
Použité sú tieto stratégie:

1- Klasický výpočet fitnes hodnoty daný pôvodným algoritmom podľa vzorca (2.1).
2- Výpočet so zvýšením váhy počtu krokov podľa vzorca (3.2.1) a $\gamma_1 = 10$.
3- Výpočet fitnes hodnoty daný pôvodným algoritmom s pridaním penalizácie podľa vzorca (3.2.2), $\gamma_2 = 10$.
4- Výpočet so zvýšením váhy počtu krokov s pridaním penalizácie podľa vzorca (3.2.2), $\gamma_1 = 10$ a $\gamma_2 = 10$.

Tabuľka 5.2 prezentuje výsledky získané pre rôzne stratégie výpočtu fitnes. Graf 5.3 predstavuje porovnanie počtu fitnes ohodnotení pre rôzne typy výpočtu fitnes. Graf závislosti fitnes od počtu iterácií algoritmu nie je v tomto prípade smerodajný, keďže sa výpočet fitnes mení.

Číslo stratégie	γ_1	γ_2	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	n/a	n/a	6	30	24633
2	10	n/a	0	14	17099
3	1	10	1	18	18704
4	**10**	**10**	**0**	**10**	**14334**

Tabuľka 5.2: Porovnanie výsledkov pre zmenu výpočtu fitnes hodnoty



Graf 5.3: Graf porovnania počtu fitnes ohodnotení pre typ výpočtu fitnes

Výsledky ukázali, že najlepšie bude použiť stratégiu č. 4 so znevýhodnením počtu krokov nad s_{max} s využitím $\gamma_1 = 10$ a $\gamma_2 = 10$.

Zmenou výpočtu fitnes hodnoty sme dosiahli zníženie počtu fitnes ohodnotení o 10299, čo predstavuje 41.8%. Celkovo sa počet testov, v ktorých výsledok nebol nájdený ani po 100000 fitnes ohodnoteniach znížil o 6, čo znamená, že v každom pokuse sme boli schopní nájsť výsledok pred prekročením 100000 ohodnotení.

5.1.3 Vytvorenie koreňa grafu

V pôvodnom algoritme bolo použité náhodné vytvorenie koreňa. Nový postup vytvárania koreňa je popísaný v časti 3.1.1.

V riešení sú použité tieto stratégie vytvorenia koreňa:

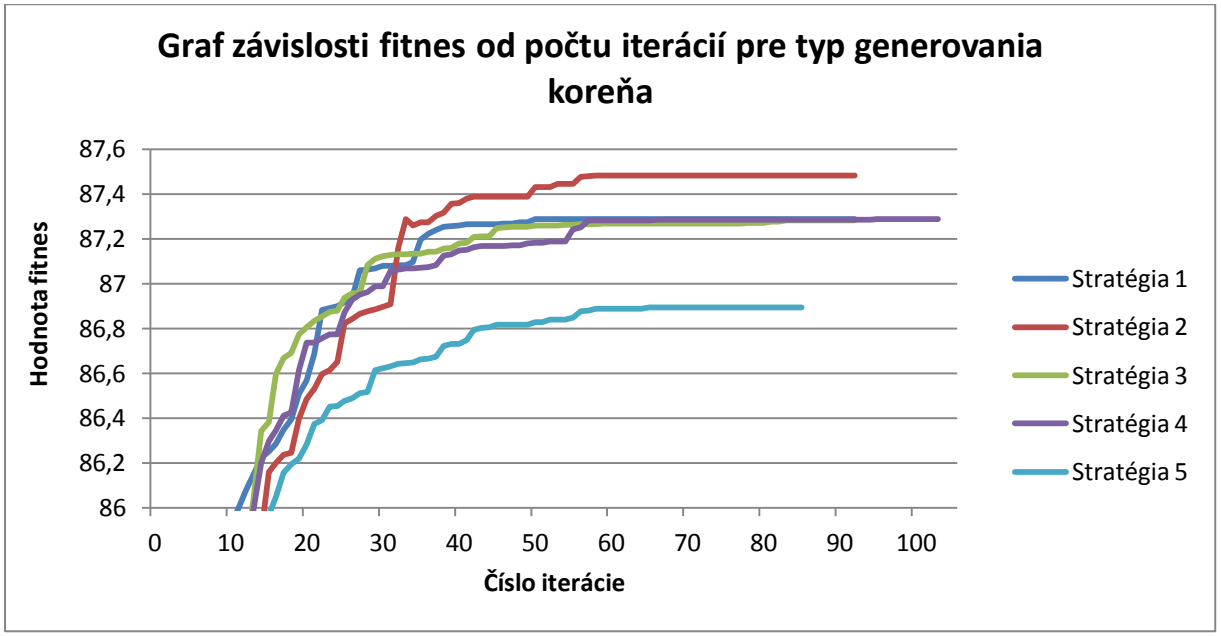
1 – Náhodné vytvorenie koreňa, podľa pôvodného algoritmu [1].

2 – Náhodne sa vytvorí konečno-stavový automat. Ten je následne v piatich iteráciách 10-krát náhodne zmutovaný, pričom do nasledujúcej iterácie sa vyberie najlepší zmutovaný automat.
3 – Náhodne sa vytvorí 5 konečno-stavových automatov. Vybraný je najlepší z nich.
4 - Náhodne sa vytvorí 10 konečno-stavových automatov. Vybraný je najlepší z nich.
5 - Náhodne sa vytvorí konečno-stavový automat, v piatich iteráciách sa náhodne zvolí pozícia v tabuľke konečno-stavového automatu a sú vyskúšané všetky modifikácie hodnôt na tejto pozícii. Do ďalšej iterácie sa vyberie automat, ktorý získa najvyššiu fitnes hodnotu.

V tabuľke 5.3 je zobrazené porovnanie výsledkov pre rôzne typy určenia koreňa grafu. Graf 5.4 predstavuje graf závislosti fitnes od počtu iterácií pre typ generovania koreňa.

Číslo stratégie	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	6	30	24633
2	**2**	**31**	**24256**
3	5	33	26042
4	4	30	24367
5	5	34	27702

Tabuľka 5.3: Porovnanie výsledkov pre typ generovania koreňa



Graf 5.4: Graf závislosti fitnes od počtu iterácií pre typ generovania koreňa

Zmenou generovania koreňa sme nedosiahli výrazné zníženie počtu fitnes ohodnotení, jednalo sa o zníženie o 377, čo predstavuje 1.5%. Celkovo sa však počet testov, v ktorých výsledok nebol nájdený ani po 100000 fitnes ohodnoteniach znížil až o 4.

5.1.4 **Rozmiestnenie mravcov v grafe**

V pôvodnom algoritme bolo použité rozmiestnenie mravcov popísané v časti 2.1.2 a mravce sú rozmiestnené náhodne pozdĺž globálne najlepšej trasy. K rozmiestneniu mravcov som sa rozhodla použiť aj iný prístup, ktorý je popísaný v časti 3.2. Pri sledovaní algoritmu som si totiž všimla, že umiestňovanie mravcov na globálne najlepšiu trasu môže spôsobiť, že algoritmus ostane uväznený v lokálnom optime.

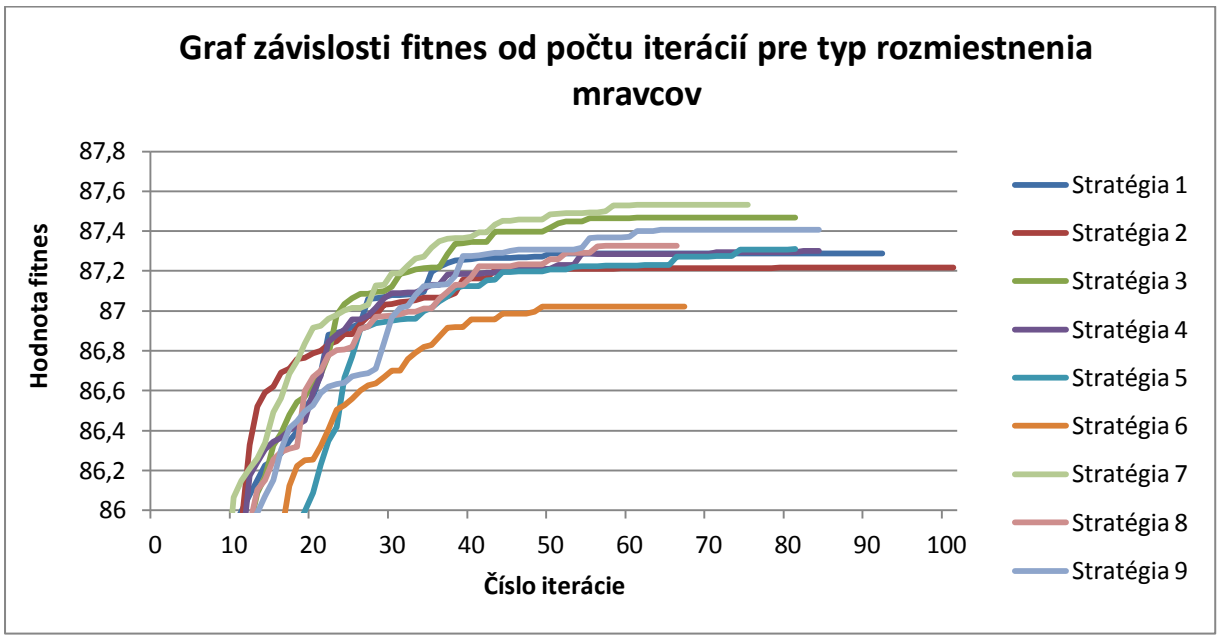
Sú použité nasledujúce stratégie:

1 – Na základe rozmiestnenia mravcov v pôvodnom algoritme [1]. Využité je rozmiestenie mravcov pozdĺž globálne najlepšej získanej trasy,
2 – Rozmiestnenie mravcov pozdĺž najlepšej získanej trasy v predošlej iterácii.
3 – Mravec je s pravdepodobnosťou 10% umiestnený na globálne najlepšiu cestu, s opačnou pravdepodobnosťou na najlepšiu trasu z predošlej iterácie.
4 - Mravec je s pravdepodobnosťou 10% umiestnený na globálne najlepšiu cestu, s pravdepodobnosťou 10% je použitý na výber počiatočného bodu turnaj veľkosti 10, pričom sa vyberá zo všetkých vrcholov.
5 – Využité je rozmiestnenie mravcov pozdĺž najlepšej získanej trasy v predošlej iterácii, vyberú sa dve vrcholy z tejto trasy, a ako štartovací bod sa vyberie lepší z nich.
6 – Podobne ako stratégia č. 5. Využité je rozmiestnenie mravcov pozdĺž najlepšej získanej trasy v predošlej iterácii. Pokiaľ algoritmus stagnuje viac ako 10 iterácií vyberú sa dve vrcholy z tejto trasy, a ako štartovací bod sa vyberie lepší z nich. Pokiaľ stagnuje menej ako 10 iterácií vyberie sa štartovací vrchol náhodne.
7 – Pokiaľ algoritmus stagnuje presne $N_{stags}/2$ iterácií, mravce sa rozmiestnia v tomto kroku na globálne najlepšiu trasu. V opačnom prípade sa mravce umiestnia na najlepšiu trasu z predošlej iterácie.
8 – Pokiaľ algoritmus stagnuje presne $N_{stags}/2$ iterácií mravce sa rozmiestnia v tomto kroku na globálne najlepšiu trasu. Pokiaľ mravec stagnuje menej ako 10 iterácií, vyberú sa dve vrcholy z iteratívne najlepšej trasy, a ako štartovací bod sa vyberie lepší z nich. V opačnom prípade sa mravce umiestnia na najlepšiu trasu z predošlej iterácie.
9 – Podobne ako v stratégii č. 8, s výnimkou toho, že v opačnom prípade sa mravec s pravdepodobnosťou 30% umiestni v grafe turnajom veľkosti 10, zo všetkých vrcholov.

V tabuľke 5.4 je zobrazené porovnanie výsledkov pre rôzne stratégie rozmiestnenia mravcov. Graf 5.5 predstavuje graf závislosti fitnes od počtu iterácií pre typ rozmiestnenia mravcov.

Číslo stratégie	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	6	30	24633
2	4	29	22012
3	5	30	22328
4	9	28	23797
5	8	29	23535
6	5	35	25445
7	**3**	**30**	**21431**
8	4	30	23961
9	5	30	24146

Tabuľka 5.4: Porovnanie výsledkov pre typ rozmiestnenia mravcov



Graf 5.5: Graf závislosti fitnes od počtu iterácií pre typ rozmiestnenia mravcov

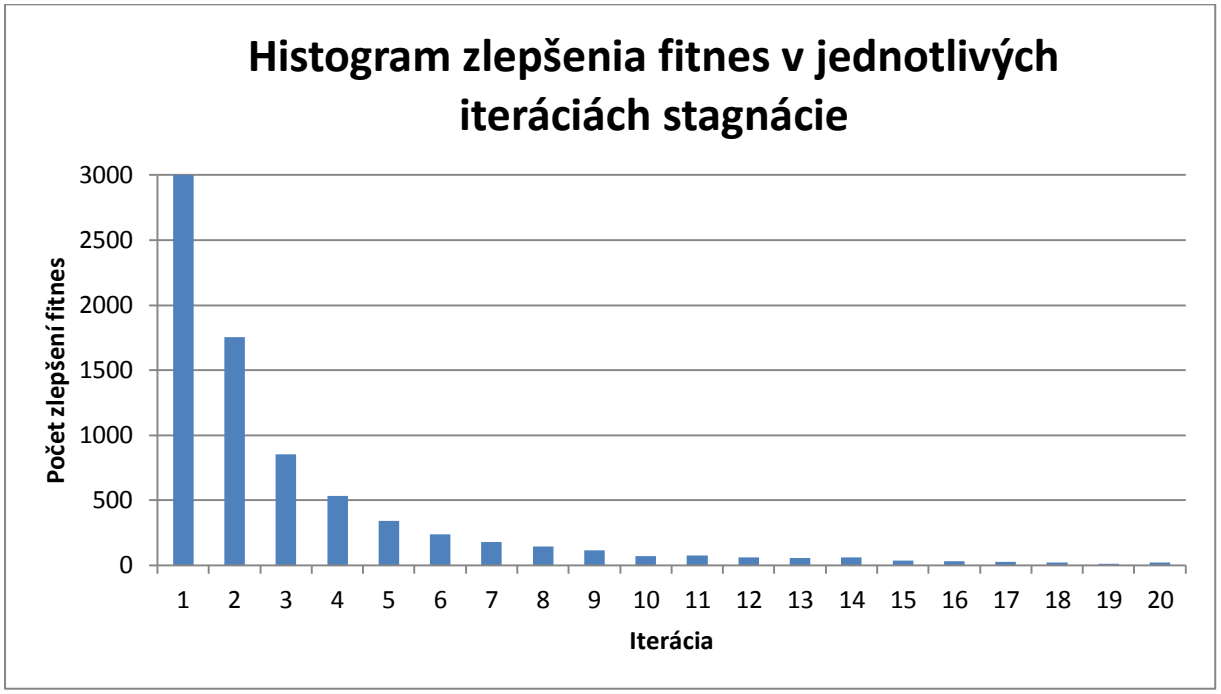
Zmenou počiatočného rozmiestnenia mravcov v grafe sme dosiahli zníženie počtu fitnes ohodnotení o 3202, čo predstavuje 13%. Celkovo sa počet testov, v ktorých výsledok nebol nájdený ani po 100000 fitnes ohodnoteniach znížil o 3.

5.1.5 Zmena veľkosti N_{stags}

Pri sledovaní vývoja algoritmu sa dalo pozorovať, že so zvyšujúcim sa počtom iterácií, bez zlepšenia globálneho maxima, sa znižuje šanca toto globálne maximum zvýšiť. V pôvodnom algoritme mal algoritmus na toto zlepšenie 20 krokov (N_{Stags}). Pokiaľ globálne maximum za 20 krokov nedokázal zvýšiť, reštartoval sa. Bližší popis je v časti 2.1.7. Rozhodla som sa preto zostaviť histogram zlepšenia najlepšej fitnes v jednotlivých iteráciách, keď zlepšenie fitnes stagnuje. Histogram je zobrazený v grafe 5.6. Prvá iterácia pre prehľadnosť nie je zobrazená celá, no došlo v nej k zlepšeniu celkovo 8454 krát. Na základe tohto pozorovania

som sa rozhodla znížiť počet N_{Stags} na 15 a 10, keďže v posledných iteráciách došlo k zlepšeniu len vo veľmi malom množstve prípadov.

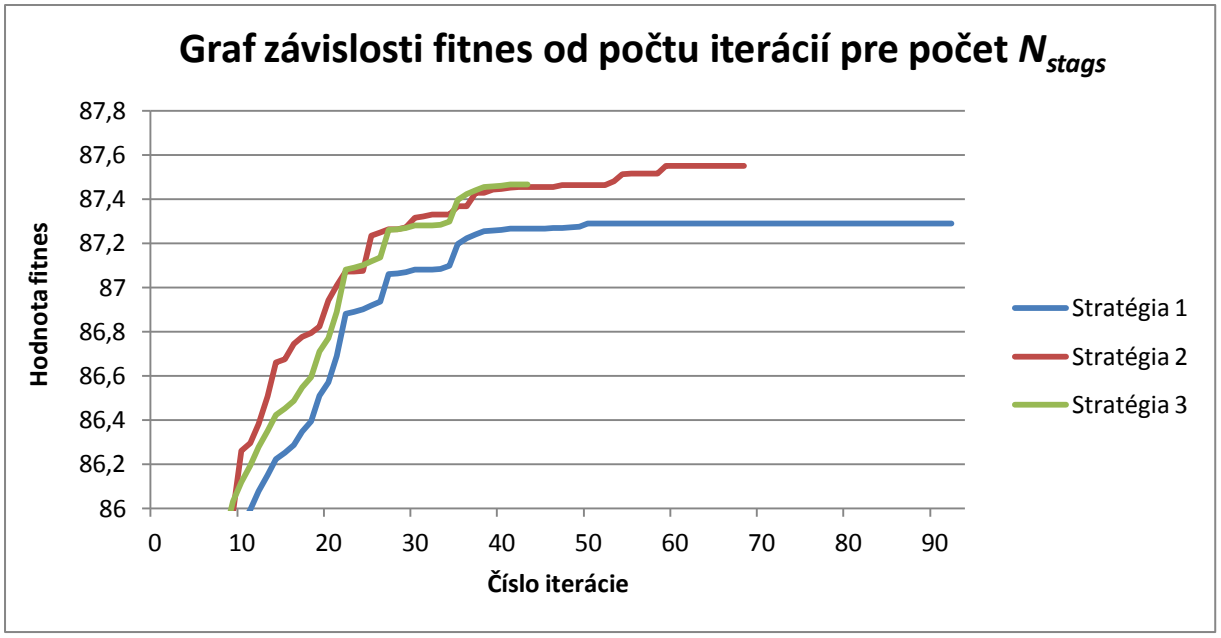
V tabuľke 5.5 je zobrazené porovnanie výsledkov pre rôzne hodnoty N_{Stags}. Graf 5.7 predstavuje graf závislosti fitnes od počtu iterácií pre počet krokov globálnej stagnácie N_{stags}.



Graf 5.6: Histogram zlepšenia fitnes v jednotlivých iteráciách stagnácie

Číslo stratégie	N_{Stags}	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	20	6	30	24633
2	15	3	23	19995
3	10	**2**	**22**	**19096**

Tabuľka 5.5: Porovnanie výsledkov pre počet N_{stags}



Graf 5.7: Graf závislosti fitnes od počtu iterácií pre počet N_{stags}

Znížením N_{Stags} sme dosiahli zníženie počtu ohodnotení o 5537, čo činí 22.5%. Celkovo sa počet testov, v ktorých výsledok nebol nájdený ani po 100000 fitnes ohodnoteniach, znížil o 4.

5.1.6 Typ aktualizácie feromónových hodnôt

Aktualizácia feromónovej hodnoty pôvodného algoritmu je popísaná v časti 2.1.4. V mojom riešení som navrhla zmeny popísané v časti 3.4.2.

Použité sú tieto stratégie:

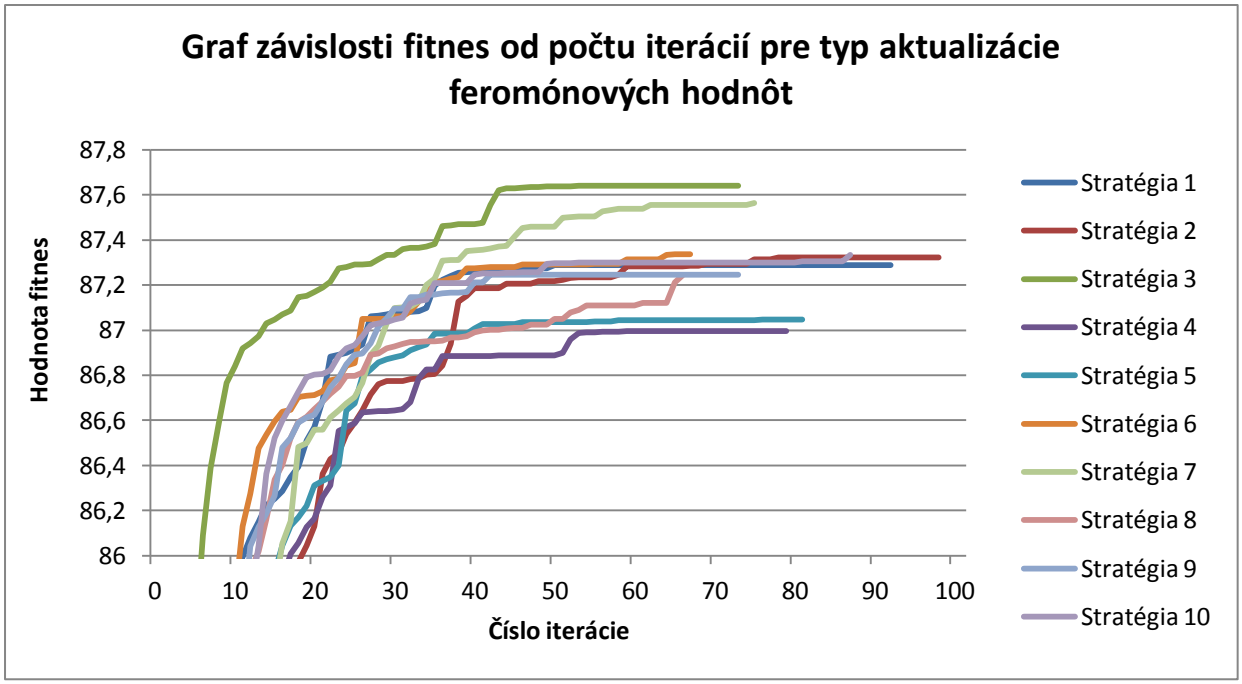
1 – Na základe aktualizácie pôvodného algoritmu [1], popísanej v časti 2.1.4.
2 – Feromónová aktualizácia na základe algoritmu EAS, hodnoty sú aktualizované pomocou formuly (3.5).
3 – Feromónová aktualizácia na základe algoritmu SACO, ktorý je popísaný v časti 1.2.2.1. Postupuje sa na základe formuly (1.4).
4 - Feromónová aktualizácia na základe algoritmu AS Ant cycle, ktorý je popísaný v časti 1.2.2.2. Postupuje sa na základe formuly (1.7) a na určenie zmeny feromónovej hodnoty sa využíva formula (1.9).
5 - Feromónová aktualizácia na základe algoritmu ACS, ktorý je popísaný v časti 1.2.2.2. Postupuje sa na základe formuly (1.14). Zmena feromónovej hodnoty je vyjadrená formulou (1.15) a hodnoty aktualizuje mravec, ktorý dosiahol najlepšie riešenie v predošlej iterácii.
6 – Rovnako ako stratégia č. 5. Feromónová aktualizácia na základe algoritmu ACS, s tým rozdielom, že hodnoty aktualizuje mravec, ktorý dosiahol globálne najlepšie riešenie.

7 - Feromónová aktualizácia na základe algoritmu MMAS. Prístup je podobný ako ACS, ale feromónové hodnoty aktualizuje aj globálne najlepší mravec a aj mravec, ktorý bol najlepší v predošlej iterácii. Globálne najlepšie riešenie sa využíva na začiatku algoritmu s 10% pravdepodobnosťou, táto pravdepodobnosť sa behom programu zvyšuje (každá iterácia znamená nárast o 1%).
8 – Feromónová aktualizácia na základe algoritmu ANTABU, ktorý je popísaný v časti 1.2.2.2, postupuje sa na základe formuly (1.20).
9 – Podobne ako stratégia č. 1 ale s vyparovaním 0.05. Využíva sa parameter $MUACOsmQ = 5$, ktorým je delený prírastok feromónovej hodnoty.
10 – Rovnako ako stratégia č. 9, ale $MUACOsmQ = 10$.

V tabuľke 5.6 je zobrazené porovnanie výsledkov pre rôzne určenie feromónových hodnôt. Graf 5.8 predstavuje graf závislosti fitnes od počtu iterácií pre typ aktualizácie feromónových hodnôt.

Číslo stratégie	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	6	30	24633
2	8	25	21507
3	**5**	**26**	**19760**
4	5	34	27626
5	5	35	26798
6	9	27	23460
7	7	27	21344
8	8	33	23552
9	4	25	20541
10	10	28	22559

Tabuľka 5.6: Porovnanie výsledkov pre rôzne typy aktualizácie feromónových hodnôt



Graf 5.8: Graf závislosti fitnes od počtu iterácií pre typ aktualizácie feromónových hodnôt

Zmenou feromónovej hodnoty sme dosiahli zníženie počtu fitnes ohodnotení o 4873, čo predstavuje 19.8%. Celkovo sa počet testov, v ktorých výsledok nebol nájdený ani po 100000 fitnes ohodnoteniach, znížil o 1.

5.1.7 **Pridanie heuristickej informácie**

V pôvodnom riešení [1] heuristická informácia nie je využitá, je však využitá v článku [2]. Bližší popis je uvedený v časti 3.5. Hodnoty heuristickej informácie môžu byť posilnené parametrom β a môže byť tiež zmenená minimálna hodnota heuristickej informácie η_{min}.

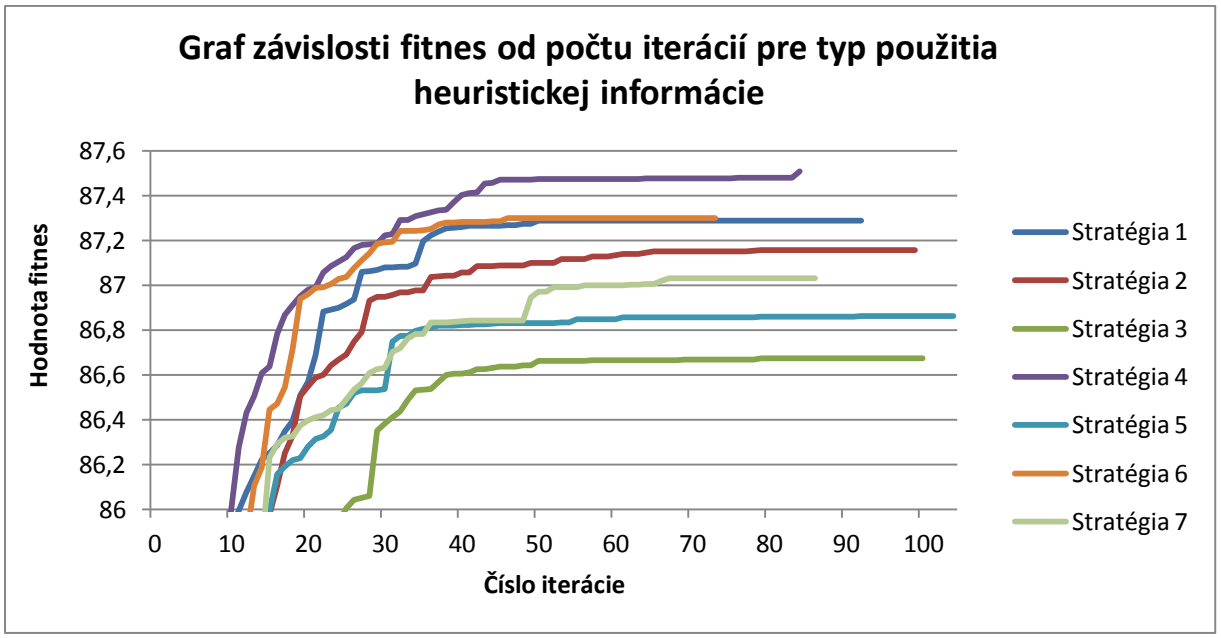
Boli vyskúšané 4 rôzne stratégie použitia heuristickej informácie:

1 – Heuristická informácia nebola využitá.
2 - Heuristická informácia bola využitá, parameter β bol rovný 1.
3 - Heuristická informácia bola využitá, parameter β bol rovný 2.
4 - Heuristická informácia bola využitá, parameter β bol rovný 3.

V tabuľke 5.7 je zobrazené porovnanie výsledkov pre rôzny prístup k heuristickej informácii hrany. Graf 5.9 predstavuje graf závislosti fitnes od počtu iterácií pre typ použitia heuristickej informácie.

Číslo stratégie	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	6	30	24633
2	8	32	24888
3	**5**	**27**	**23310**
4	5	32	23983

Tabuľka 5.7: Porovnanie výsledkov pre rôzne typy použitia heuristickej informácie



Graf 5.9: Graf závislosti fitnes od počtu iterácií pre typ použitia heuristickej informácie

Pridaním heuristickej informácie sme nedosiahli výrazné zníženie počtu fitnes ohodnotení. Počet fitnes ohodnotení sa podarilo znížiť o 1323, čo predstavuje 5.4%. Celkovo sa počet testov, v ktorých výsledok nebol nájdený ani po 100000 fitnes ohodnoteniach, znížil o 1.

5.1.8 Výber nasledujúcej hrany

V pôvodnom riešení [1] sa používa výber nasledujúcej hrany popísaný v časti 2.1.5 a hrana sa vyberá na základe formuly (2.2). Bude však vyskúšaný aj iný prístup navrhnutý v časti 3.3.1.

Sú použité tieto stratégie:

1 – Na základe výberu pôvodného algoritmu [1], popísaného v časti 2.1.4.
2 – Ako stratégia č. 1, ale ak je $r < r_{ACS}$, je automaticky vybraná najlepšia hrana, $r_{ACS} = 20$.
3 – Ako stratégia č. 1, ale ak je $r < r_{ACS}$, je vybraná nasledujúca hrana náhodným výberom, $r_{ACS} = 10$.
4 - Ako stratégia č. 1, ale ak je $r < r_{ACS}$, je nasledujúca hrana vybraná turnajom, $r_{ACS} = 20$.

5 - Ako stratégia č. 1, ale ak je $r < r_{ACS}$ je nasledujúca hrana vybraná turnajom, $r_{ACS} = 30$.

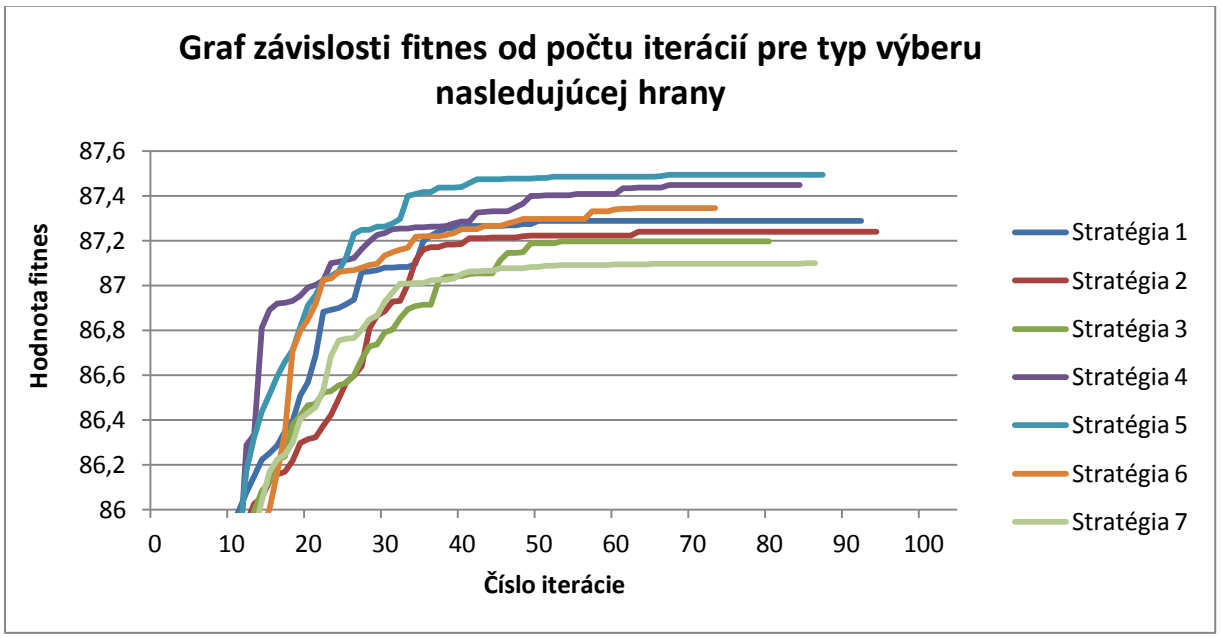
6 - Ako stratégia č. 1, ale ak je $r < r1_{ANTQ}$, je vybraná najlepšia hrana, ak je $r < r2\_ANTQ$ nasledujúca hrana je vybraná turnajom. Pričom platí, že $r1\_ANTQ = 20$, $r2\_ANTQ = 80$.

7 - Ako stratégia č. 6, no $r1\_ANTQ = 20, r2\_ANTQ = 70$.

V tabuľke 5.8 je zobrazené porovnanie výsledkov pre rôzny pre typ výberu nasledujúcej hrany. Graf 5.10 predstavuje graf závislosti fitnes od počtu iterácií pre typ výberu nasledujúcej hrany.

Číslo stratégie	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	6	30	24633
2	5	35	28477
3	4	31	26205
4	5	29	22416
5	**6**	**25**	**21791**
6	8	28	22201
7	6	32	26269

Tabuľka 5.8: Porovnanie výsledkov pre typ výberu nasledujúcej hrany



Graf 5.10: Graf závislosti fitnes od počtu iterácií pre typ výberu nasledujúcej hrany

Zmenou výberu nasledujúcej hrany sme dosiahli zníženie počtu fitnes ohodnotení o 2542, čo predstavuje 10.3%. Celkovo sa však počet testov, v ktorých výsledok nebol nájdený ani po 100000 fitnes ohodnoteniach, neznížil.

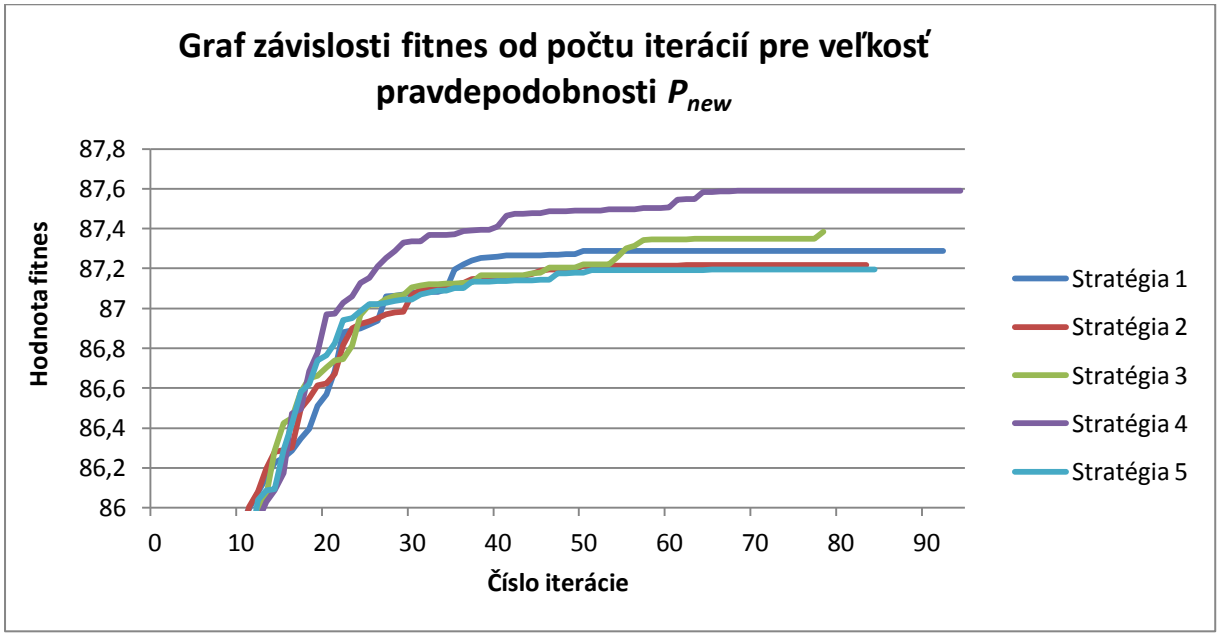
5.1.9 **Zmena pravdepodobnosti P_{new}**

Parameter P_{new} predstavuje pravdepodobnosť vytvorenia nových uzlov pri konštruovaní grafu. Popis konštrukcie cesty v pôvodnom algoritme [1] je popísaný v časti 2.1.2. V riešení sme sa rozhodli vyskúšať modifikovanie pravdepodobnosti P_{new}, a vplyv zmeny na počet fitnes ohodnotení.

V tabuľke 5.9 je zobrazené porovnanie výsledkov pre rôzny rôzne veľkosti pravdepodobnosti P_{new}. Graf 5.11 predstavuje graf závislosti fitnes od počtu iterácií pre veľkosť pravdepodobnosti P_{new}.

Číslo stratégie	P_{new}	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	50	6	30	24633
2	40	2	34	24950
3	60	5	29	22878
4	**70**	**5**	**23**	**19512**
5	80	6	35	25046

Tabuľka 5.9: Porovnanie výsledkov pre rôzne veľkosti P_{new}



Graf 5.11: Graf závislosti fitnes od počtu iterácií pre veľkosť pravdepodobnosti P_{new}

Zmenou parametra P_{new} sme dosiahli zníženie počtu fitnes ohodnotení o 5121, čo predstavuje 20.8%. Celkovo sa počet testov, v ktorých výsledok nebol nájdený ani po 100000 fitnes ohodnoteniach znížil o 1.

5.1.10 Zmena počtu mravcov N_{ants}

Parameter N_{ants} predstavuje počet mravcov, ktoré sa používajú pri konštrukcii ciest. V riešení sme sa rozhodli vyskúšať zmenu počtu mravcov N_{ants} a skúmať jej vplyv na počet fitnes ohodnotení.

V tabuľke 5.10 je zobrazené porovnanie výsledkov pre rôzny počet mravcov N_{ants}. Graf závislosti fitnes od počtu iterácií algoritmu v tomto prípade nie je relevantný, keďže so zvyšujúcim sa počtom mravcov, fitnes hodnota narastá rýchlejšie. Výraznejšie však narastá aj počet fitnes ohodnotení.

Číslo stratégie	N_{ants}	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	5	6	30	24633
2	**3**	**5**	**37**	**24172**
3	4	6	27	23813
4	6	7	27	24538
5	7	5	31	25295
6	8	6	28	24967

Tabuľka 5.10: Porovnanie výsledkov pre rôzny počet mravcov N_{ants}

Zmenou parametra N_{ants} sa nepodarilo dosiahnuť výraznejšie zníženie počtu fitnes ohodnotení. Za najlepšie zlepšenie by sa dala považovať stratégia, kde bol daný počet mravcov 3 (respektíve stratégia č. 2). Nejedná sa síce o najvýraznejšie zníženie počtu fitnes ohodnotení (zlepšenie o 461 fitnes ohodnotení, čo predstavuje 1.9%), najvýraznejšie bolo dosiahnuté pri stratégii č. 3, no celkovo sa počet testov, v ktorých výsledok nebol nájdený ani po 100000 fitnes ohodnoteniach znížil o 1, a to bol v tomto prípade rozhodujúci faktor.

5.1.11 Časť 1: Zhodnotenie

V prvej časti riešenia, ktorá bola zameraná na pozorovanie vplyvu zmien jednotlivých parametrov, sme zmenou parametrov dosiahli výsledky uvedené v tabuľke 5.11. Najväčšie zlepšenie dosiahla úprava fitnes výpočtu (5.1.2), kde sme pri výpočte fitnes pridali parameter, ktorý zvýšil váhu počtu krokov oproti počtu nazbieraných kusov potravy a pridali penalizáciu.

	Zmena parametra (kapitola)	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení	Percentuálne zlepšenie počtu fitnes ohodnotení	Zníženie počtu ohodnotení nad 100000
1	Pôvodný algoritmus [1] (2.1)	6	30	24633	n/a	n/a
2	**Zmena výpočtu fitnes (5.1.2)**	**0**	**10**	**14334**	**41,8%**	**6**
3	Zmena N_{stags} (5.1.3)	2	22	19096	22.5%	4
4	Zmena pravdepodobnosti P_{new} (5.1.9)	5	23	19512	20.8%	1
5	Zmena aktualizácie feromónových hodnôt (5.1.6)	5	26	19760	19.8%	1
6	Zmena rozmiestnenia v grafe (5.1.4)	3	30	21431	13%	3
7	Zmena výberu nasledujúcej hrany (5.1.8)	6	25	22091	10.3%	0
8	Pridanie heuristickej informácie (5.1.7)	5	27	23310	5.4%	1
9	Zmena počtu mravcov (5.1.10)	5	37	24172	1.9%	1
10	Zmena vytvorenia koreňa grafu (5.1.3)	2	31	24256	1.5%	4

Tabuľka 5.11: Porovnanie zlepšení pre jednotlivé zmeny parametrov

5.2 Časť 2: Iteratívna optimalizácia parametrov

V časti 5.2.1 až 5.2.9 sa postupne zameriame na zmenu jednotlivých parametrov navrhnutých v časti Návrh riešenia (kapitola 3), pričom sa budeme snažiť počet fitnes ohodnotení potrebných na nájdenie požadovaného riešenia čo najviac znížiť, a to tým, že parametre budeme zlepšovať iteratívne a zmenu ďalšieho parametra vykonáme s nastavením parametrov z predošlého získaného zlepšenia. Zameriavame sa na výpočet fitnes (5.2.1), generovanie koreňa grafu (5.2.2), rozmiestnenie mravcov v grafe (5.2.3), zmenu počtu krokov stagnácie N_{stags} (5.2.4), typ aktualizácie feromónových hodnôt (5.2.5), pridanie heuristickej informácie (5.2.6), výber nasledujúcej hrany (5.2.7), zmenu pravdepodobnosti P_{new} (5.2.8) a zmenu počtu mravcov N_{ants} (5.2.9).

5.2.1 Zmena výpočtu fitnes hodnoty

Keďže je výpočet fitnes prvým krokom zmeny parametrov algoritmu, a teda nemá žiadnu nadväznosť na iný predošlý krok, budú použité výsledky získané v časti 5.1.2.

5.2.2 Vytvorenie koreňa grafu

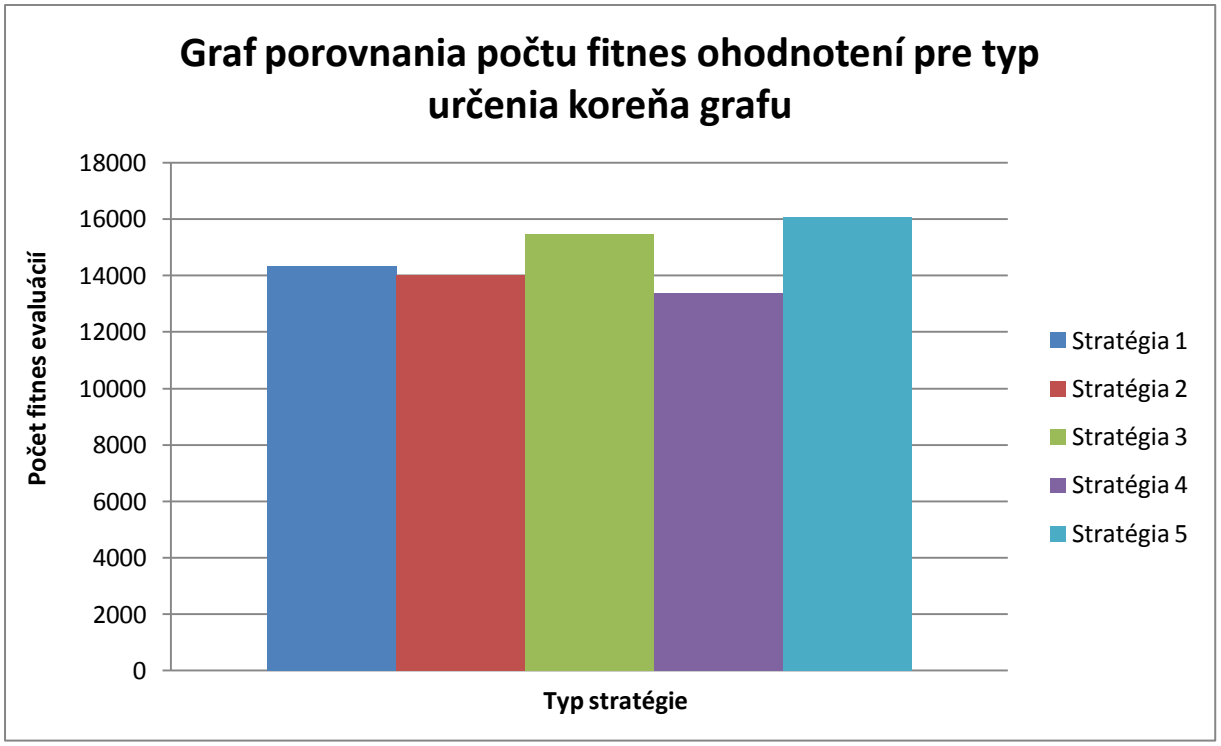
Vychádzame zo stratégie č. 4 použitej na výpočet fitnes hodnoty, ktorá získala najlepšie výsledky v časti 5.1.2.

V pôvodnom algoritme bol koreň vytvorený na začiatku algoritmu náhodne. V riešení budú použité stratégie na vytvorenie koreňa popísané v časti 5.1.3.

V tabuľke 5.12 je zobrazené porovnanie výsledkov pre rôzne typy určenia koreňa grafu. Graf 5.12 predstavuje porovnanie počtu fitnes ohodnotení pre typ určenia koreňa grafu.

Číslo testu	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	0	10	14334
2	0	9	13988
3	0	10	15453
4	**0**	**10**	**13352**
5	0	13	16087

Tabuľka 5.12: Porovnanie výsledkov pre rôzne typy určenia koreňa grafu



Graf 5.12: Graf porovnania počtu fitnes ohodnotení pre typ určenia koreňa grafu

Ako najlepšia sa ukázala stratégia č. 4. Oproti výsledkom získaným v časti 5.2.1 sme výsledok zlepšili o 982 fitnes ohodnotení, čo predstavuje 6.9%.

Celkovo sme oproti pôvodnému algoritmu dosiahli zlepšenie o 11281 fitnes ohodnotení, čo predstavuje zlepšenie o 45.8%.

5.2.3 **Rozmiestnenie mravcov v grafe**

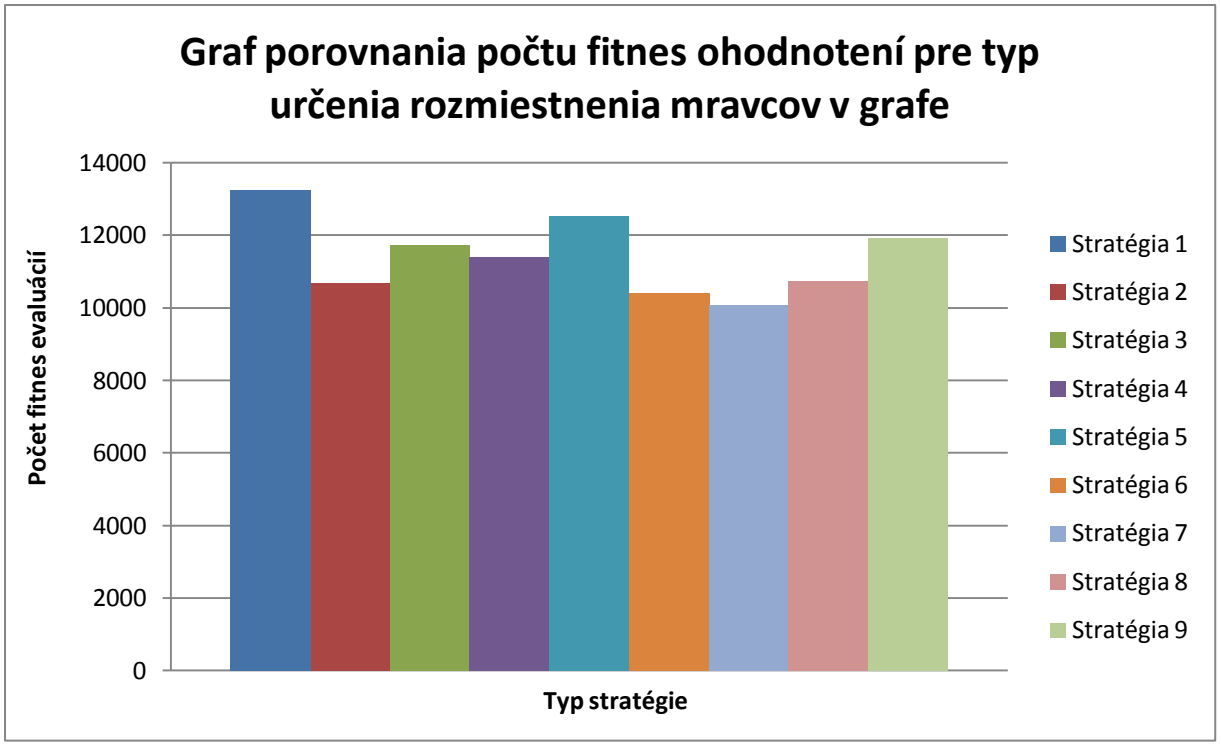
Vychádzame zo stratégie č. 4, ktorá sa ukázala ako najlepšia pri vytváraní koreňa grafu v časti 5.2.2 a tiež zo stratégie č. 4 použitej pri výpočte fitnes hodnoty 5.1.2.

V pôvodnom algoritme bolo použité rozmiestnenie mravcov popísané v časti 2.1.2, a mravce sú rozmiestnené náhodne pozdĺž globálne najlepšej trasy. Na rozmiestnenie mravcov boli použité stratégie popísané v časti 5.1.4.

V tabuľke 5.13 je zobrazené porovnanie výsledkov pre rôzne stratégie rozmiestnenia mravcov. Graf 5.13 predstavuje porovnanie počtu fitnes ohodnotení pre typ určenia rozmiestnenia mravcov v grafe.

Číslo testu	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	0	10	13252
2	0	6	10669
3	0	7	11719
4	0	6	11404
5	0	10	12534
6	0	8	10407
7	**0**	**6**	**10070**
8	0	7	10723
9	0	9	11912

Tabuľka 5.13: Porovnanie výsledkov pre rôzne stratégie rozmiestnenia mravcov



Graf 5.13: Graf porovnania počtu fitnes ohodnotení pre typ určenia rozmiestnenia mravcov v grafe

Ako najlepšia sa ukázala stratégia číslo 7. Oproti výsledkom získaným v časti 5.2.2 sme výsledok zlepšili o 3182 fitnes ohodnotení, čo predstavuje 24%.

Celkovo sme oproti pôvodnému algoritmu dosiahli zlepšenie o 14563 fitnes ohodnotení, čo predstavuje zlepšenie o 59.1%.

5.2.4 Zmena veľkosti N_{stags}

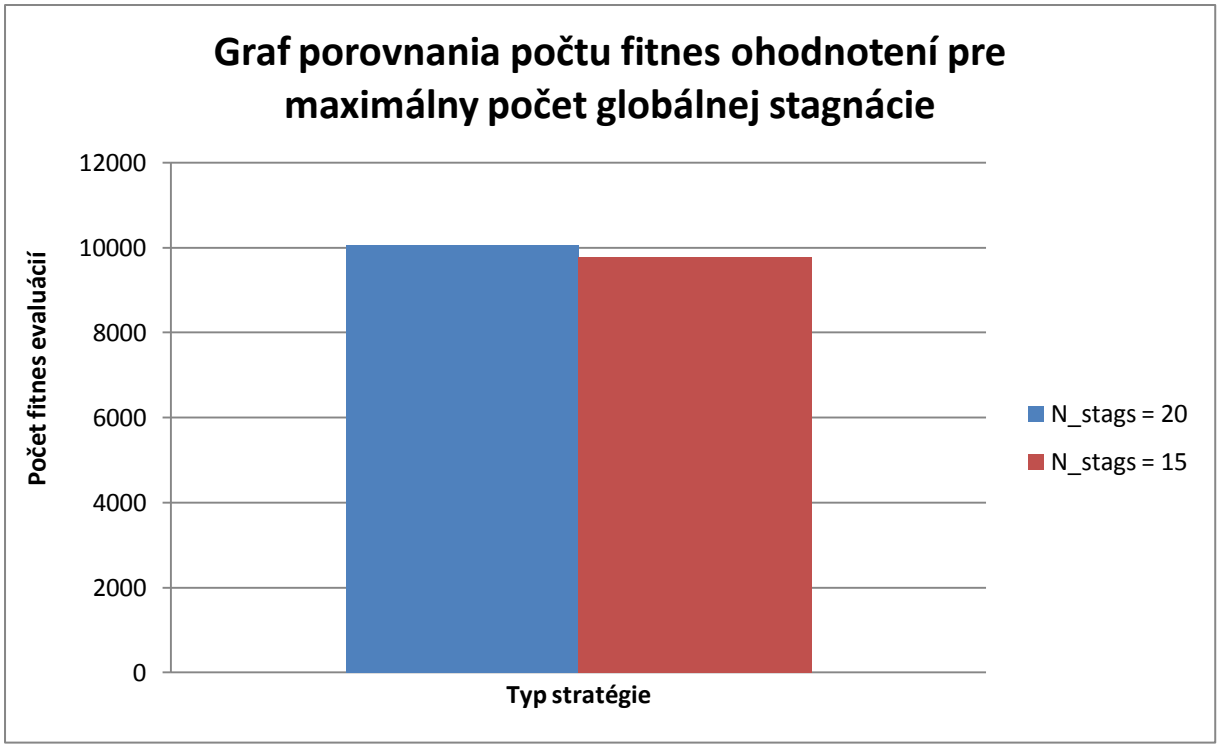
Vychádzame zo stratégie č. 7, ktorá sa ukázala ako najlepšia pri rozmiestnení mravcov v časti 5.2.3, zároveň zo stratégie č. 4, ktorá sa ukázala ako najlepšia pri vytváraní koreňa grafu v časti 5.2.2 a tiež zo stratégie č. 4 použitej pri výpočte fitnes hodnoty 5.1.2.

Dôvod na zmenu N_{Stags} je popísaný v časti 5.1.5, kde sa tiež nachádza histogram zlepšenia najlepšej fitnes v jednotlivých iteráciách, keď zlepšenie fitnes stagnuje (graf 5.6).

V tabuľke 5.14 je zobrazené porovnanie výsledkov pre rôzne hodnoty N_{Stags}. Graf 5.14 predstavuje porovnanie počtu fitnes ohodnotení pre maximálny počet globálnej stagnácie.

Číslo testu	N_{Stags}	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	20	0	6	10070
2	15	0	5	9777

Tabuľka 5.14: Porovnanie výsledkov pre rôzne hodnoty N_{Stags}



Graf 5.14: Graf porovnania počtu fitnes ohodnotení pre maximálny počet globálnej stagnácie

Znížením N_{Stags} sa podarilo dosiahnuť zlepšenie. Oproti výsledkom získaným v časti 5.2.3 sme výsledok zlepšili o 293 fitnes ohodnotení, čo predstavuje 2.9%.

Celkovo sme oproti pôvodnému algoritmu dosiahli zlepšenie o 14856 fitnes ohodnotení, čo predstavuje zlepšenie o 60.3%.

5.2.5 Typ aktualizácie feromónových hodnôt

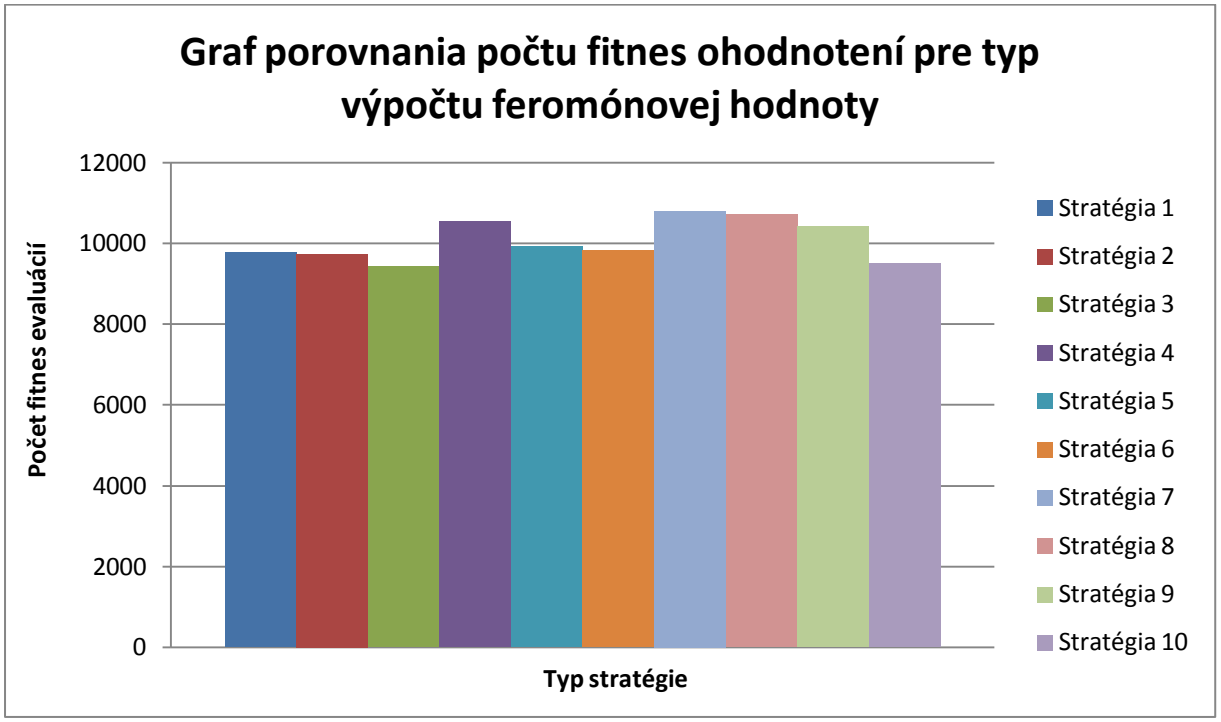
Použitá bude veľkosť $N_{Stags} = 15$, vzhľadom na výsledky dosiahnuté v časti 5.2.4. Vychádzať budeme tiež zo stratégie č. 7, ktorá sa ukázala ako najlepšia pri rozmiestnení mravcov v časti 5.2.3, zároveň zo stratégie č. 4, ktorá sa ukázala ako najlepšia pri vytváraní koreňa grafu v časti 5.2.2 a tiež zo stratégie č. 4 použitej pri výpočte fitnes hodnoty 5.1.2.

Aktualizácia feromónovej hodnoty pôvodného algoritmu je popísaná v časti 2.1.4. V mojom riešení som navrhla zmeny popísané v časti 3.4.2. Použité budú stratégie popísané v časti 5.1.6.

V tabuľke 5.15 je zobrazené porovnanie výsledkov pre rôzne určenie feromónových hodnôt. Graf 5.15 predstavuje porovnanie počtu fitnes ohodnotení pre typ výpočtu feromónovej hodnoty.

Číslo testu	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	0	5	9777
2	0	3	9743
3	**0**	**5**	**9424**
4	0	5	10548
5	0	2	9933
6	0	6	9825
7	0	7	10805
8	0	6	10711
9	0	5	10426
10	0	4	9684

Tabuľka 5.15: Porovnanie výsledkov pre rôzne určenie feromónových hodnôt



Graf 5.15: Graf porovnania počtu fitnes ohodnotení pre typ výpočtu feromónovej hodnoty

Ako najlepšie sa ukázalo použitie stratégie 3. Oproti výsledkom získaným v časti 5.2.4 sme výsledok zlepšili o 353 fitnes ohodnotení, čo predstavuje 3.6%.

Celkovo sme oproti pôvodnému algoritmu dosiahli zlepšenie o 15209 fitnes ohodnotení, čo predstavuje zlepšenie o 61.7%.

5.2.6 Pridanie heuristickej informácie

Využitá bude stratégia č. 3 použitá na aktualizáciu feromónových hodnôt, ktorá získala najlepšie výsledky v časti 5.2.5. Použitá bude veľkosť $N_{Stags} = 15$, vzhľadom na výsledky dosiahnuté v časti 5.2.4. Vychádzať budeme tiež zo stratégie č. 7, ktorá sa ukázala ako najlepšia pri rozmiestnení mravcov v časti 5.2.3, zároveň zo stratégie č. 4, ktorá sa ukázala

ako najlepšia pri vytváraní koreňa grafu v časti 5.2.2 a zo stratégie č. 4 použitej pri výpočte fitnes hodnoty 5.1.2.

V pôvodnom riešení [1] heuristická informácia nie je využitá, je však využitá v článku [2]. Bližší popis je uvedený v časti 3.5. Popis použitých stratégií je v časti 5.1.7.

Výsledky sú zobrazené v Prílohe A v tabuľke A.1, kde je zobrazené porovnanie výsledkov pre rôzny prístup k heuristickej informácii hrany. Graf A.1 v Prílohe A predstavuje porovnanie počtu fitnes ohodnotení pre typ použitia heuristickej informácie.

Najlepší výsledok sme dosiahli s použitím stratégie č. 1, kde heuristická informácia nebola použitá. K zlepšeniu predošlých výsledkov teda nedošlo, a preto sme sa rozhodli v ďalších testoch heuristickú informáciu nepoužiť.

5.2.7 **Výber nasledujúcej hrany**

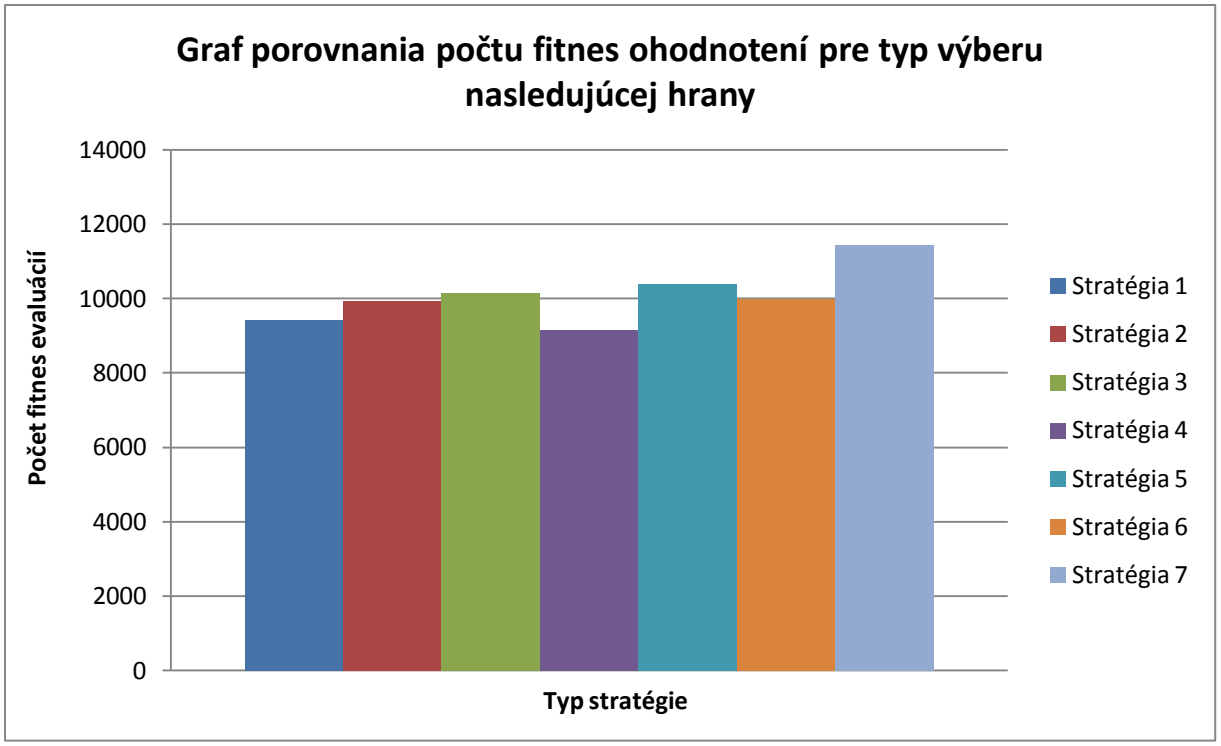
Heuristická informácia nebude vzhľadom na výsledky dosiahnuté v časti 5.2.6 použitá. Využitá bude stratégia č. 3 použitá na aktualizáciu feromónových hodnôt, ktorá získala najlepšie výsledky v časti 5.2.5. Použitá bude veľkosť $N_{Stags} = 15$, vzhľadom na výsledky dosiahnuté v časti 5.2.4. Vychádzať budeme tiež zo stratégie č. 7 rozmiestnenia mravcov, ktorá sa ukázala ako najlepšia v časti 5.2.3, zároveň zo stratégie č. 4, ktorá sa ukázala ako najlepšia pri vytváraní koreňa grafu v časti 5.2.2 a zo stratégie č. 4 použitej pri výpočte fitnes hodnoty 5.1.2.

V algoritme [1] sa používa výber nasledujúcej hrany popísaný v časti 2.1.5 a hrana sa vyberá na základe formuly (2.2). Bude však vyskúšaný aj iný prístup navrhnutý v časti 3.3.1. Popis stratégií výberu nasledujúcej hrany je uvedený v časti 5.1.8.

V tabuľke 5.16 je zobrazené porovnanie výsledkov pre rôzne stratégie výberu nasledujúcej hrany. Graf 5.16 predstavuje porovnanie počtu fitnes ohodnotení pre typ výberu nasledujúcej hrany.

Číslo stratégie	**Dosiahnutie 100000 ohodnotení (%)**	**Nad 30000 ohodnotení (%)**	Priemer fitnes ohodnotení
1	0	4	9424
2	0	6	9930
3	0	7	10154
4	**0**	**3**	**9159**
5	0	8	10382
6	0	6	10002
7	0	10	11443

Tabuľka 5.16: Porovnanie výsledkov pre rôzne stratégie výberu nasledujúcej hrany



Graf 5.16: Graf porovnania počtu fitnes ohodnotení pre typ výberu nasledujúcej hrany

Ako najlepšie sa ukázalo použitie stratégie č. 4. Oproti výsledkom získaným v časti 5.2.5 sme výsledok zlepšili o 265 fitnes ohodnotení, čo predstavuje 2.8%.

Celkovo sme oproti pôvodnému algoritmu dosiahli zlepšenie o 15474 fitnes ohodnotení, čo predstavuje zlepšenie o 62.8%.

5.2.8 Zmena pravdepodobnosti P_{new}

Vychádzame zo stratégie č. 4 výberu nasledujúcej hrany, na základe výsledkov získaných v časti 5.2.7. Heuristická informácia nebude vzhľadom na výsledky dosiahnuté v časti 5.2.6 použitá. Využitá bude stratégia č. 3 použitá na aktualizáciu feromónových hodnôt, ktorá získala najlepšie výsledky v časti 5.2.5. Použitá bude veľkosť $N_{Stags} = 15$, vzhľadom na výsledky dosiahnuté v časti 5.2.4. Vychádzať budeme tiež zo stratégie č. 7 rozmiestnenia mravcov, ktorá sa ukázala ako najlepšia v časti 5.2.3, zároveň zo stratégie č. 4, ktorá sa ukázala ako najlepšia pri vytváraní koreňa grafu v časti 5.2.2 a zo stratégie č. 4 použitej pri výpočte fitnes hodnoty 5.1.2.

V pôvodnom algoritme bola hodnota parametra $P_{new} = 50$. V riešení sme sa rozhodli vyskúšať vplyv zmeny parametra P_{new} na počet fitnes ohodnotení.

Výsledky sú zobrazené v Prílohe A v tabuľke A.2, kde je zobrazené porovnanie výsledkov pre rôzne hodnoty P_{new}. Graf A.2 v Prílohe A predstavuje porovnanie počtu fitnes ohodnotení pre veľkosť parametra P_{new}.

Najlepší výsledok sme dosiahli s použitím stratégie č. 1, kde P_{new} malo pôvodnú hodnotu. K zlepšeniu predošlých výsledkov teda nedošlo, a preto sme sa rozhodli, v ďalších testoch ponechať $P_{new} = 50$.

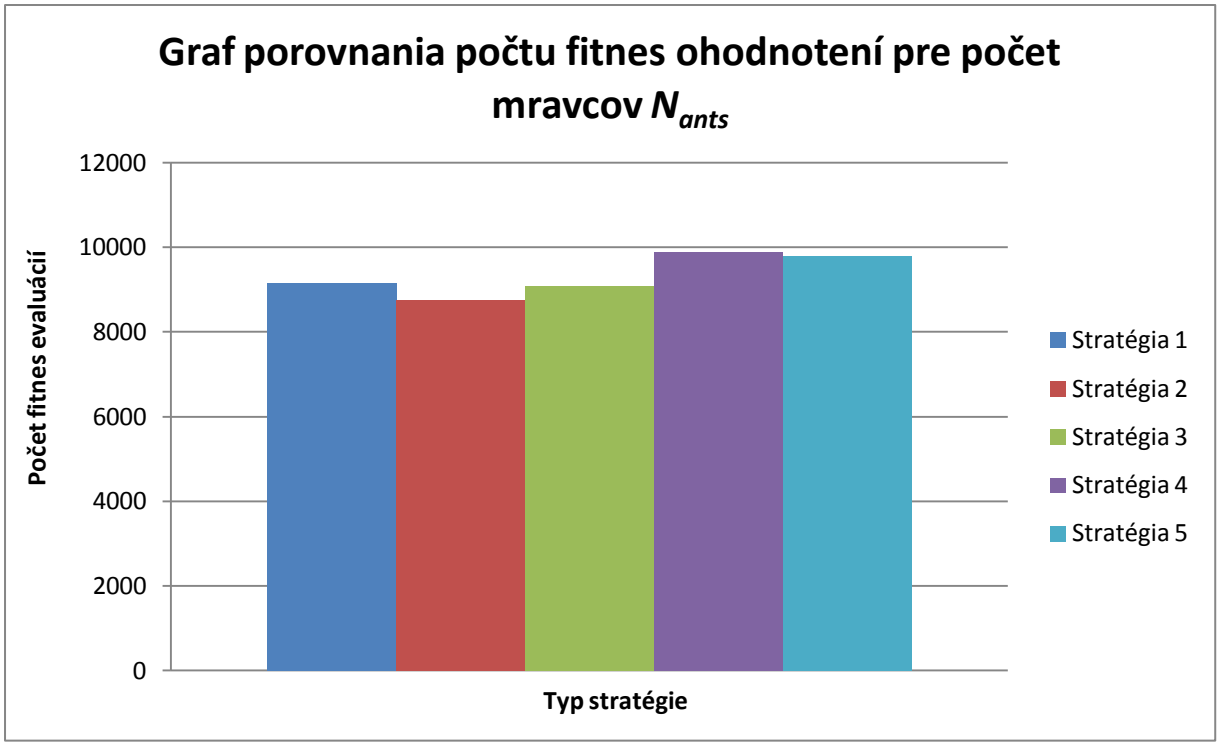
5.2.9 Zmena počtu mravcov N_{ants}

Vychádzame z $P_{new} = 50$, vzhľadom na výsledky dosiahnuté v časti 5.2.8. Použitá je stratégia č. 4 výberu nasledujúcej hrany, na základe výsledkov získaných v časti 5.2.7. Heuristická informácia nebude vzhľadom na výsledky dosiahnuté v časti 5.2.6 použitá. Využitá bude stratégia č. 3 použitá na aktualizáciu feromónových hodnôt, ktorá získala najlepšie výsledky v časti 5.2.5. Použitá bude veľkosť $N_{Stags} = 15$, vzhľadom na výsledky dosiahnuté v časti 5.2.4. Vychádzať budeme tiež zo stratégie č. 7 rozmiestnenia mravcov, ktorá sa ukázala ako najlepšia v časti 5.2.3, zároveň zo stratégie č. 4, ktorá sa ukázala ako najlepšia pri vytváraní koreňa grafu v časti 5.2.2 a zo stratégie č. 4 použitej pri výpočte fitnes hodnoty 5.1.2.

V tabuľke 5.17 je zobrazené porovnanie výsledkov pre rôzny počet mravcov N_{ants}. Graf 5.17 predstavuje graf porovnania počtu fitnes ohodnotení pre počet mravcov N_{ants}.

Číslo stratégie	N_{ants}	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	5	0	4	9159
2	**3**	**0**	**3**	**8747**
3	4	0	4	9068
4	6	0	5	9877
5	7	0	6	9786

5.17: Porovnanie výsledkov pre rôzny počet mravcov N_{ants}



Graf 5.17: Graf porovnania počtu fitnes ohodnotení pre počet mravcov N_{ants}

Ako najlepšie sa ukázalo použitie stratégie č. 2, kde bol daný počet mravcov $N_{ants} = 3$. Oproti výsledkom získaným v časti 5.2.8 sme výsledok zlepšili o 412 fitnes ohodnotení, čo predstavuje 4.5%.

Celkovo sme oproti pôvodnému algoritmu dosiahli zlepšenie o 15886 fitnes ohodnotení, čo predstavuje zlepšenie o 64.5%.

5.2.10 Časť 2: Zhodnotenie

V druhej časti sa nám oproti pôvodnému algoritmu [1] podarilo dosiahnuť zlepšenie o 15886 fitnes ohodnotení, čo predstavuje zlepšenie o 64.5%.

Ako najlepšie nastavenie algoritmu sa ukázalo:

- Paralelná stratégia vytvárania cesty (nejedná sa o paralelizáciu).
- Fitnes ohodnotenie so zvýšením váhy počtu krokov a penalizáciou riešení s počtom krokov nad s_{max}.
- Koreň grafu vytvorený turnajom veľkosti 10, kde sa vyberie najlepší z náhodne vytvorených koreňov.
- Rozmiestnenie mravcov, v ktorom, ak algoritmus stagnuje presne $N_{stags}/2$ iterácií, mravce sa rozmiestnia v tomto kroku na globálne najlepšiu trasu. V opačnom prípade sa mravce umiestnia na najlepšiu trasu z predošlej iterácie.

- Výber nasledujúcej hrany sa robí klasickým stochastickým výberom (podľa formuly (2.2)), ale je dodefinovaná hranica *r*. Ak je $r < r_{ACS}$, je nasledujúca hrana vybraná turnajom, $r_{ACS} = 20$.
- Počet krokov, ktoré algoritmus môže stagnovať $N_{stags} = 15$.
- Heuristická informácia nie je využitá.
- Feromónová aktualizácia na základe algoritmu SACO, ktorý je popísaný v časti 1.2.2.1. Postupuje sa na základe formuly (1.4).
- Počet mravcov 3.
- Pravdepodobnosť vytvorenia nových hrán $P_{new} = 50$.

6. Záver

V diplomovej práci sa mi podarilo dosiahnuť výrazné zlepšenie výsledkov algoritmu [1], z ktorého práca vychádza, ale tiež algoritmov navrhnutých v článkoch [2], [29], [34].

V prvej časti riešenia, zameranej na pozorovanie vplyvu zmien jednotlivých parametrov, som zmenou parametrov dosiahla výsledky uvedené v tabuľke 5.11. Najväčší prínos mala úprava fitnes výpočtu, kde som pri výpočte fitnes pridala parameter, ktorý zvýšil váhu počtu krokov oproti počtu nazbieraných kusov potravy a pridala penalizáciu. Návrh zmeny výpočtu fitnes je popísaný v časti 3.1.3 a výsledky testu v časti 5.1.2. Zmenou výpočtu fitnes hodnoty som dosiahla zníženie počtu fitnes ohodnotení o 41.8%. Celkovo sa počet testov, v ktorých výsledok nebol nájdený ani po 100000 fitnes ohodnoteniach znížil o 6, čo znamená, že v každom pokuse som bola schopná nájsť výsledok. Vzhľadom k tomu, že bol algoritmus pri dosiahnutí hranice 100000 prerušený, neboli tieto hodnoty pri pôvodnom algoritme ani započítané do priemeru. Dosiahnuté zlepšenie je tým ešte výraznejšie.

V druhej časti riešenia som celkovo oproti pôvodnému algoritmu [1] dosiahla zlepšenie o 15886 fitnes ohodnotení, čo predstavuje zlepšenie o 64.5%, pričom žiaden z behov programu nepresiahol hranicu 100000 ohodnotení a algoritmus nebol prerušený. V tabuľke 6.1 je porovnaný dosiahnutý výsledok s inými algoritmami, riešiacimi rovnaký problém.

	Typ algoritmu	**Dosiahnutie 100000 ohodnotení (%)**	**Nad 30000 ohodnotení (%)**	**Priemer fitnes ohodnotení**
1	Pôvodný algoritmus [1]	6	30	24633
2	MuACOsm [2]	n/a	n/a	9200
3	Mutation based ACO [29]	n/a	12	10500[1]
4	MRTS, GP + Subroutines [34]	n/a	n/a	43000
5	MRTS, Random Search + Subroutines [34]	n/a	n/a	20696
6	**Nami dosiahnuté výsledky (kapitola 5.2)**	**0**	**3**	**8747**

Tabuľka 6.1: Porovnanie výsledkov s inými algoritmami

[1] počet fitnes ohodnotení nad 30000 sa pri určovaní výsledkov algoritmu [29] považuje za neúspech a ukončenie algoritmu, tým pádom v priemere nie je započítaný. K neúspechu v tomto prípade došlo v približne 12% prípadov.

Zo všeobecného hľadiska je prínosom, že pri dosiahnutých výsledkoch sa ukázalo, ako veľmi podstatná je váha počtu krokov a penalizácia, ktoré spôsobili zlepšenie o viac ako 40%. Veľmi dobré výsledky (zlepšenie o približne 20%), však prinieslo aj zníženie počtu krokov, ktoré algoritmus môže stagnovať pri zvyšovaní najlepšej fitnes, tiež zvýšenie pravdepodobnosti vytvárania nových hrán a typ aktualizácie feromónových hodnôt. Určité zlepšenie však bolo dosiahnuté pri každej zmene parametra. Dá sa predpokladať, že zmenou týchto parametrov by bolo dosiahnuté zlepšenie aj u iných riešení.

7. Literatúra

1. **Chivilikhin, D., Ulyantsev, V**. (2012): Learning Finite-State Machines with Ant Colony Optimization: 8th International Conference, ANTS 2012. Brussels, Belgium : Springer Berlin Heidelberg, 2012 (s. 268-275).

2. **Chivilikhin, D., Ulyantsev, V.** (2013). MuACOsm – A New Mutation-Based Ant Colony Optimization Algorithm for Learning Finite-State Machine: GECCO '13 Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference. New York, NY, USA: ACM, 2013 (s. 511-518).

3. **Spears, W.M., Gordon, D.E.** (2010). Evolving Finite-State Machine Strategies for Protecting Resources: 12th International Symposium, ISMIS 2000 Charlotte, NC, USA, October 11–14, 2000 Proceedings. Brussels, Belgium: Springer Berlin Heidelberg, 2010 (s. 16- 175).

4. **Engelbrecht, A. E.** (2005). Fundamentals of Computational Swarm Intelligence. Chichester, Endland: Sons Ltd., 2005 (s. 361-478).

5. **Hopcroft, J. E., Ullman J. D**. (1969). Formal Languages and their Relation to Automata. Massachusetts, USA: Addison– Weasley Publishing Company, 1969 (s. 47-57).

6. **Chytil, M.** (1984). Automaty a Gramatiky. Praha: SNTL – Nakladatelství technické literatúry, 1984 (s. 15-73).

7. **Aziz A. D., Cackler J., Yung R**. (2004). Základná teória automatov [cit 2014-05-01] URL: <http://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/automata-theory/basics.html>

8. **Marais E. N.** (1948). Die Siel van die Mier (The Soul of Ant). Pretoria, South Africa: J.L. van Schaik, 1948 piate vydanie (prvý krát publikované 1937).

9. **Deneubourg J. L., Aron S., Goss S., Pasteels J. M.** (1990). The Self-Organizing Exploratory Pattern of the Argentine Ant. Journal of Insect Behaviour, 1990 (s. 159-168).

10. **Dorigo M.** (1992). Optimization, Learning and Natural Algorithms. PhD. Thesis: Politecnico di Milano, 1992.

11. **Goss A. Aron S., Deneubourg J.L., Pasteels J.M.** (1989). Self-Organized Shortcuts in the Argentine Ant: Naturwissenschaften 76. Brussels, Belgium: Springer,1989 (s. 579-581).

12. **Dorigo M., Di Caro G.** (1999). The Ant Colony Optimization Meta-Heuristic: New Ideas in Optimization. Maidenhead, England, UK: McGraw-Hill Ltd. (s. 11-32).

13. **Dorigo M., Stützle T.** (2001). An Experimental Study of the Simple Ant Colony Optimization Algorithm: Proceedings of the 2001 WSES International Conference on Evolutionary Computation. EC'01 (s. 253-258).

14. **Bonabeau E., Dorigo M., Theraulaz G.** (1999). Swarm Intelligence: From Natural to Artificial System. England: Oxford University Press.

15. **Stützle T.** (1997). MAX-MIN Ant System for Quadratic Assignment Problems. Technical report, FG Intellektik, FB Informatik, TH Darmstadt.

16. **Stützle T., Hoos H.** (1997). MAX-MIN Ant System and Local Search fot the Traveling Salesman Problem: Proceedings of the IEEE International Conference on Evolutionary Computation (s. 309-314).

17. **Gambardella L. M., Dorigo M.** (1995). Ant-Q: A Reinforcement Learning Approach to the TSP: Proceedings of ML-95, Twelfth International Conference on Machine Learning (s. 252-260).

18. **Taillard É. D.** (1998). FANT: Fast Ant System. Technical report. Lugano, Switzerland: IDSIA.

19. **Roux O., Fonlupt C., Talbi E. G.** (1998). ANTabu. Technical report LIL-98-04. Calais, France: Laboratorie d'Informatique du Littoral, Université du Littoral.

20. **Kaji T.** (2001). Approach by Ant Tabu Agents for Traveling Salesman Problem: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics vol.5, October 2001 (s. 3429-3434).

21. **Bullnheimer B.**, Kotsis G., Strauss C. (1997). Parallelization Strategies for the Ant System: Tolardo G., Murli P., Pardalos P, editor, Kluwer Series on Applied Optimization (s. 87-100).

22. **Maniezzo V., Carbonaro A.** (2000). An ANTS Heuristic for the Frequency Assignment Problem. Future Generation Computer System (s. 927-935).

23. **Dorigo M., Di Caro G.** (1999). Ant Colony Optimization: A New Meta-Heuristic: Proceedings of the IEEE International Conference on Evolutionary Computation vol. 2, Júl 1999 (s. 1477).

24. **Taillard É. D.** (1999). Ant System. Technical report. Lugano, Switzerland: IDSIA.

25. **Birattari M., Dorigo M., Di Caro G.** (2003). Toward the Formal Foundation of Ant Programming: M. Dorigo, G. Di Caro, M. Samples, editors: Proceedings of the Third International Workshops on Ant Algorithms: Lecture Notes in Computer Science vol. 2463. Springer-Verlag, 2003 (s. 188-201).

26. **Blum C., Dorigo M., Roli A.** (2001). HC-ACO: The Hyper-Cube Framework for Ant Colony Optimization: Proceedings of the Fourth Meta-heuristic International Conference, 2001 (s. 399-403).

27. **Dorigo M., Di Caro G.** (2006). Ant Algorithms for Discrete Optimization: Journal Arifical Life, Vol. 5, No. 2, Marec 2006. MIT Press Cambridge, MA, USA (s. 137-172).

28. **Stützle T.**, **Dorigo M.** (1999) ACO Algorithms for the Traveling Salesman Problem: IRIDIA, Universite Libre de Bruxelles, Belgium.

29. **Chivilikhin, D., Ulyantsev, V.** (2013). Learning Finite-State Machines with Classical and Mutation-Based Ant Colony Optimization : Experimental Evaluation: Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC), 2013 BRICS Congress on. Ipojuca IEEE, 2013 (s. 528-533).

30. **Dorigo M., Maniezzo, V., Colorni, A.** (1996). Ant System: Optimization by a Colony of Cooperating Agents, IEEE Transactions on Systems, Man,and Cybernetics, Part B: Cybernetics, vol. 26, no. 1, 1966 (s. 29-41).

31. **BROWNLEE, J**.: Stochastic Hill Climbing. www.cleveralgorithms.com, 2014 [cit. 9.12.2014]. Dostupné na webovskej stránke (world wide web): http://www.cleveralgorithms.com/nature-inspired/stochastic/hill_climbing_search.html

32. **Dorigo M., Gambardella L. M** (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman problem : IEEE Transactions on Evolutionary Computation, 1(1), 1966 (s. 53-66).

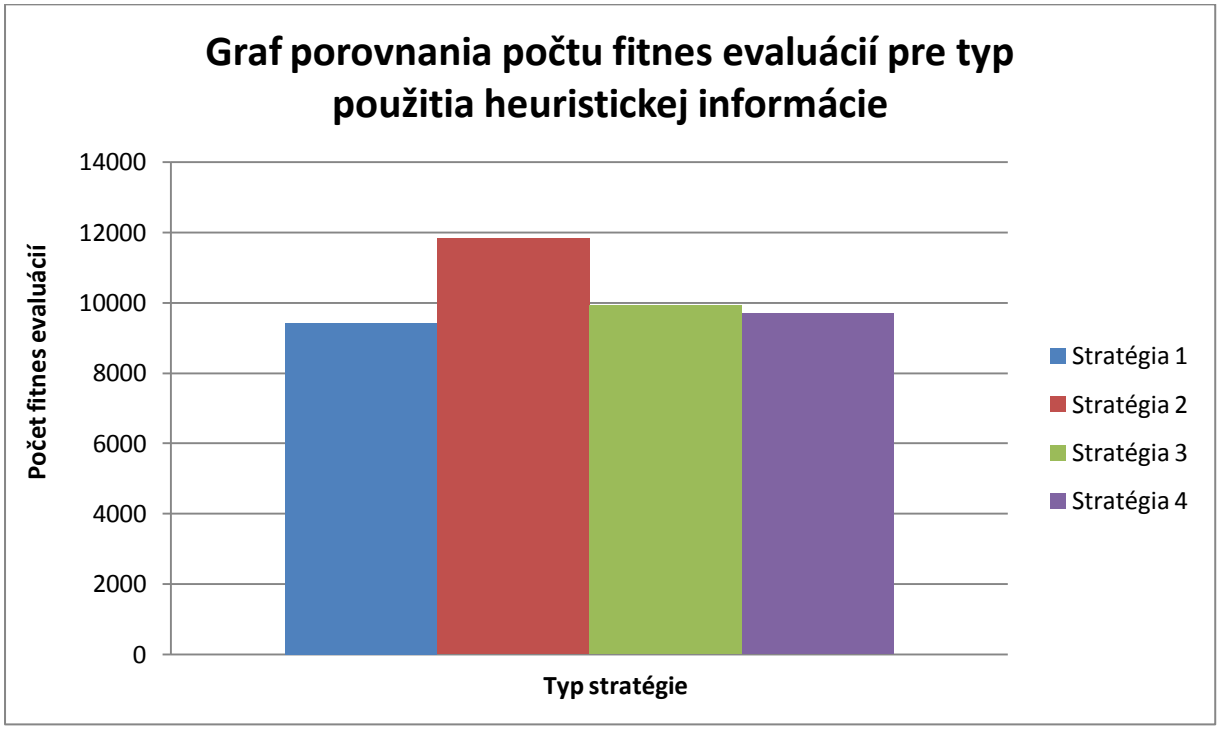
33. **Jefferson, D., Collins, R., Cooper, C., Dryer, M., Flowers, M., Korf, R., Taylor, Ch., Wang, A** (1990): The Genesys System: Evolution as a Theme in Artificial Life: UCLA, Los Angeles, California 90024.

34. **Christensen, S., Oppacher, F.** (2007). Solving the artificial ant on the santa fe trail problem in 20,696 fitness evaluations: GECCO'07, New York, USA (s. 1574-1579).

Príloha A. Výsledky

Číslo stratégie	β	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	**n/a**	**0**	**5**	**9424**
2	1	0	7	11829
3	2	0	6	9924
4	3	0	5	9701

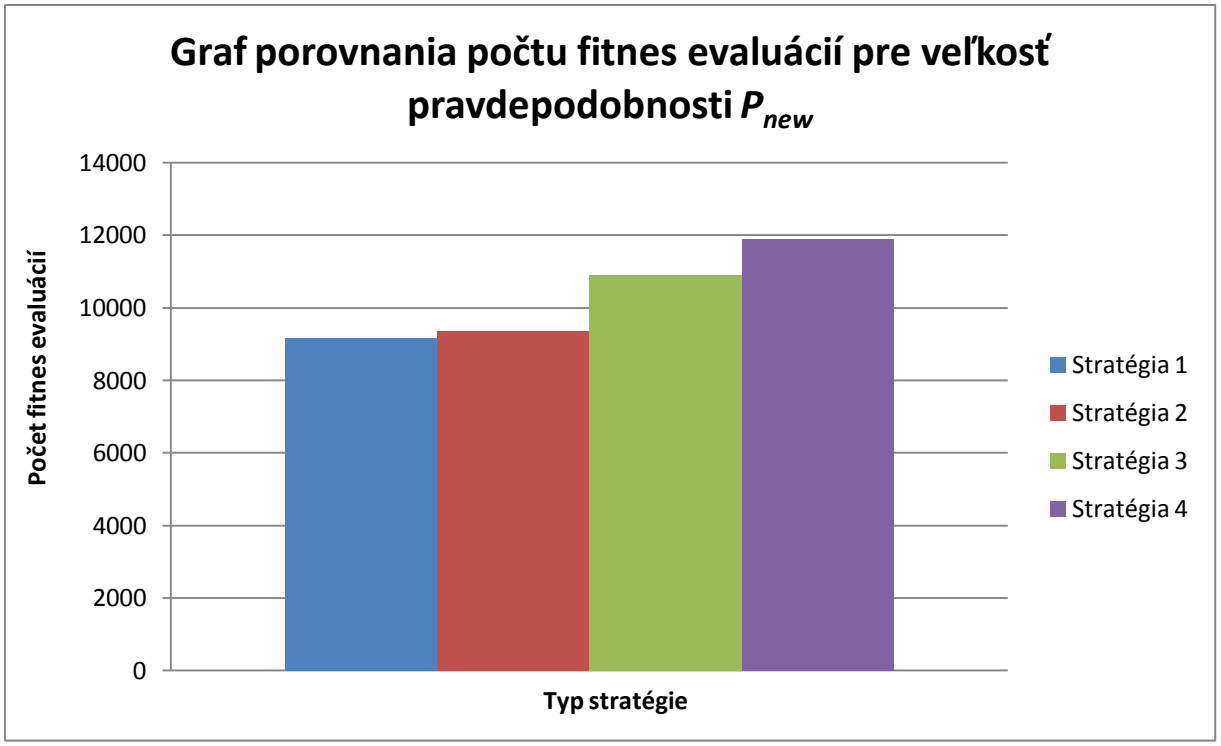
Tabuľka A.1: Porovnanie výsledkov pre rôzny prístup k heuristickej informácii hrany



Graf A.1: Graf porovnania počtu fitnes ohodnotení pre typ použitia heuristickej informácie

Číslo stratégie	P_{new}	Dosiahnutie 100000 ohodnotení (%)	Nad 30000 ohodnotení (%)	Priemer fitnes ohodnotení
1	**50**	**0**	**4**	**9159**
2	40	0	5	9364
3	60	0	8	10900
4	70	0	9	11883

Tabuľka A.2: Porovnanie výsledkov pre rôznu veľkosť pravdepodobnosti P_{new}



Graf A.2: Graf porovnania počtu fitnes ohodnotení pre veľkosť pravdepodobnosti P_{new}

Príloha B. Zoznam parametrov algoritmu

V tabuľke B.1 je zobrazený zoznam parametrov algoritmu, ktoré umožňuje program zmeniť. Parametre je možné zmeniť v triede Parameters.cs.

Typ	Názov premennej	Popis	Podmienky pre zadané hodnoty
int	$numberOfStates$	počet stavov výsledného konečno-stavového automatu	$numberOfStates > 0$
int	$problemType$	Typ riešeného problému, momentálne implementované pre John Muir food trail a Santa Fe	0- John Muir food trail 1- Santa Fe trail
int	s_max	Počet krokov, ktoré môže agent na mriežke prejsť aby našiel všetky kúsky potravy na mriežke	Vhodné je $s\_max > 200$
int	n_ants	Počet mravcov	$n\_ants > 0$
int	$numberOfMutation$	Počet vytvorených mutácií pri vytváraní cesty	$numberOfMutation > 0$
int	p_new	Pravdepodobnosť mutovania vrcholu pri vytváraní cesty mravcom	$< 0{,}100 >$
int	n_stags	počet krokov, ktoré môže vykonať mravec bez navýšenia ním získanej fitnes hodnoty	$n\_stags > 0$
int	N_stags	Počet iterácií hľadaní cesty mravcami, ktoré sa môžu vykonať bez navýšenia fitnes. Po prekročení *N_stags* sa algoritmus reštartuje.	$N\_stags > 0$
		FITNESS:	
int	$stepsWeight$	váha pridaná krokom pri výpočte fitnes	$steps\_weight \geq 0$
int	$fitnessType$	typ výpočtu fitnes	0- Bez znevýhodnenie počtu krokov 1- So znevýhodnením počtu krokov

ROZMIESTNENIE MRAVCOV:

int	$startPositionOfAnt$	Typ rozmiestnenia mravcov v grafe	0 – Rozmiestnenie pozdĺž globálne najlepšej trasy 1 - Rozmiestnenie pozdĺž iteratívne najlepšej trasy 2 – Určité percento mravcov je rozmiestnené na globálne najlepšej trase (podľa parametra $probabilityOfAntPlacedGlobaly$) a zvyšok mravcov je rozmiestnený na najlepšiu cestu z predošlej iterácie. 3 - Určité percento mravcov je rozmiestnené na globálne najlepšej trase (podľa parametra $probabilityOfAntPlacedGlobaly$), určité percento je umiestnené turnajom (podľa parametra $probabilityOfAntPlacedTournament$) a zvyšok mravcov je rozmiestnený na najlepšiu cestu z predošlej iterácie. 4 – Využité je rozmiestnenie mravcov pozdĺž najlepšej získanej trasy v predošlej iterácii, vyberú sa dve vrcholy z tejto trasy, a ako štartovací bod sa vyberie lepší z nich. 5 – Podobne ako 4. Využité je rozmiestnenie mravcov pozdĺž najlepšej získanej trasy v predošlej iterácii. Pokiaľ algoritmus stagnuje viac ako $N_{stags}/2$ iterácií vyberú sa dve vrcholy z tejto trasy, a ako štartovací bod sa vyberie lepší z nich. Pokiaľ stagnuje menej ako $N_{stags}/2$ iterácií vyberie sa štartovací vrchol z tejto cesty náhodne. 6 – Pokiaľ algoritmus stagnuje presne $N_{stags}/2$ iterácií mravce sa rozmiestnia v tomto kroku na globálne najlepšiu trasu. V opačnom prípade sa mravce umiestnia na najlepšiu trasu z predošlej iterácie. 7 – Pokiaľ algoritmus stagnuje presne $N_{stags}/2$ iterácií mravce sa rozmiestnia v tomto kroku na globálne najlepšiu trasu. Pokiaľ mravec stagnuje menej ako 10 iterácií, vyberú sa dve vrcholy z iteratívne najlepšej trasy, a ako štartovací bod sa vyberie lepší z nich. V opačnom prípade sa mravce umiestnia na najlepšiu trasu z predošlej iterácie. 8 – To isté ako v 7, s výnimkou toho, že v opačnom prípade sa mravec s pravdepodobnosťou $probabilityOfAntPlacedTournament$ umiestni v grafe turnajom veľkosti 10 zo všetkých vrcholov.
int	$probabilityOfAntPlaced Globaly$	Pravdepodobnosť umiestnenia mravca globálne	$< 0{,}100 >$
int	$probabilityOfAntPlacea Tournament$	Pravdepodobnosť umiestnenia mravca turnajom	$< 0{,}100 >$
double	$evaporation$	Hodnota odparovania	$< 0{,}1 >$
double	$minHeuristicValue$	Hodnota minimálnej heuristickej hodnoty hrany	Malé nezáporné číslo

GENEROVANIE KOREŇA:

int	$rootgeneratorType$	Typ generovania koreňa grafu	0 - Koreň sa generuje náhodne 1 -Vytvorí sa „$CountOfMutation$“ mutácií koreňa v „$CountOfGenerationMutation$“ iteráciách 2- Vytvorenie koreňa turnajom veľkosti „$tournamentSize$“ 3- Vytvorenie koreňa hill climbingom v „$HillClimbingCount$“ krokoch
int	$tournamentSize$	Veľkosť turnaja	$tournamentSize > 0$
int	$CountOfGenerationMut$	Počet iterácií mutácií	$CountOfGenerationMutation > 0$
int	$CountOfMutation$	Počet mutácií v jednej iterácií	$CountOfMutation > 0$
int	$HillClimbingCount$	Počet krokov hill climbing	$HillClimbingCount > 0$

VÝBER NASLEDUJÚCEJ HRANY:

	$pathSelectionType$	Typ výberu nasledujúcej hrany	0 – Na základe výberu pôvodného algoritmu [1], popísanej v časti 2.1.4. 1 –Ak $r < r_{ACS}$ je automaticky vybraná najlepšia hrana. 2 – Ak $r < r_{ACS}$ je vybraná nasledujúca hrana náhodným výberom. 3 - Ak $r < r_{ACS}$ je nasledujúca hrana vybraná turnajom. 4 - Ak $r < r1\_ANTQ$ je vybraná najlepšia hrana, ak $r < r2\_ANTQ$ nasledujúca hrana je vybraná turnajom
int	r_ACS	Hranica použitá pri výbere nasledujúcej hrany	$< 0{,}100 >$
int	$r1\_ANTQ$	Spodná hranica použitá pri výbere nasledujúcej hrany	$< 0{,}100 >$
int	$r2\_ANTQ$	Horná hranica použitá pri výbere nasledujúcej hrany	$< 0{,}100 >$
int	$alfa$	parameter použitý pri výbere dáva váhu feromónovej hodnote	$alfa \geq 0$
int	$beta$	parameter pri výbere dáva váhu heuristickej informácii	$beta \geq 0$

AKTUALIZÁCIA FEROMÓNOVÝCH HODNÔT:			
Int	$pheromonUpdateType$	typ aktualizácie feromónových hodnôt	0 – Na základe algoritmu MUACOsm 1 – Na základe algoritmu EAS s parametrom „evaporationEAS“ 2 – Na základe algoritmu SACO 3 – Na základe algoritmu AS Ant-cycle 4 – Na základe algoritmu ACS s využitím najlepšieho riešenia v predošlej iterácií 5 – Na základe algoritmu ACS s využitím globálne najlepšieho riešenia 6 – Na základe algoritmu MMAS 7 – Na základe algoritmu ANTABU 8 – ako typ 0 ale pridaná feromónová hodnota je delená.
double	$evaporationEAS$	veľkosť odparovania pre EAS	$< 0{,}1 >$
double	$evaporationACS$	veľkosť odparovania pre ACS	$< 0{,}1 >$
double	EAS_w_elit	váha elitistického riešenia v algoritme EAS ($pheromonUpdateType = 1$)	$< 0{,}1 >$
double	$MinPheromonValue$	minimálna hranica feromónu, ktorá je na hrane zanechaná	Malá pozitívna konštanta
int	AS_Q	veľkosť konštanty Q použitej pri algoritme AS Ant-cycle ($pheromonUpdateType = 3$)	$AS\_Q > 0$
int	$MMAS\_probabilityOf UseGlobal$	počiatočná pravdepodobnosť využitia globálneho riešenia	$< 0{,}100 >$
HEURISTICKÁ INFORMÁCIA			
boolean	$useOfHeuristic - Information$	Vyjadruje či heuristická informácie bude alebo nebude použitá	False – heuristická informácia nie je použitá True- heuristická informácie je použitá
STRATÉGIE KOLONIE:			
int	$colonyStrategy$	Typ stratégie kolónie	0 – Postupná stratégia 2 – Paralelná stratégia

Tabuľka B.1: Tabuľka parametrov algoritmu triedy Parameters.cs

Príloha C. Manuál na spustenie programu

Program je spustiteľná aplikácia. Vzhľadom k tomu, že je program vytvorený v jazyku C#, na jeho spustenie je potrebný .NET Framework 4.

Keďže je práca vedeckého charakteru, program je vytvorený ako konzolová aplikácia. Program sa spustí po otvorení:
\DP_LearningFSM\DP2_LearningFSM\bin\Release\DP2_LearningFSM.exe

Používateľ je pri spustení programu vyzvaný, aby zadal počet behov algoritmu a názov výstupného súboru (akceptuje sa iba alfanumerický názov súboru). Pre zmenu parametrov algoritmu je potrebné použiť triedu Parameters.cs. Bližší popis jednotlivých parametrov je v Prílohe B.

Výsledky sa ukladajú vo formáte .txt do priečinka:
DP_LearningFSM\DP2_LearningFSM\bin\Release\Results

Formát výstupného súboru :

---EVOLUTION OF FITNES IN ALGORITHM ITERATIONS---
$fitnes_1$
.
.
.
$fitnes_n$
---NUMBER OF EVALUATIONS SUMMARY---
AVERAGE - $evaluations_{average}$
MAXIMUM - $evaluations_{maximum}$
MINIMUM - $evaluations_{minimum}$
More then 100000 evaluations: $evaluations_{over100000}$ from $test\_count$
More then 30000 evaluations: $evaluations_{over30000}$
---EVALUATIONS COUNT INDIVIDUALLY---
$evaluations\_count_1$
.
.
.
$evaluations\_count_{test\_count}$
---PARAMETER SETTINGS---
$Parameters\_settings$

$fitnes_x$ $-\ x \in (0, n)$, kde n je počet iterácií v prvom štarte algoritmu. Predstavuje hodnotu fitnes po iterácii x.
$evaluations_{average}$ - predstavuje priemerný počet fitnes ohodnotení vo všetkých testoch
$evaluations_{maximum}$ - predstavuje maximálny počet fitnes ohodnotení, ktorý boli dosiahnutý
$evaluations_{minimum}$ - predstavuje minimálny počet fitnes ohodnotení, ktorý bol dosiahnutý

$evaluations_{over100000}$ - počet testov, v ktorých sme nedosiahli výsledok ani po 100000 ohodnoteniach fitnes
$test\_count$ - celkový počet testov
$evaluations_{over30000}$ - počet testov, v ktorých sme presiahli 30000 fitnes ohodnotení
$evaluations\_count_y - \ y \in < 1, test\_count >,$ Počet fitnes ohodnotení v teste y.
$Parameters\_settings$ - nastavenie jednotlivých parametrov.

Výstup programu v konzole je podobný ako formát výstupu do súboru, neobsahuje však priemernú fitnes hodnotu ani nastavenie parametrov. Na rozdiel od výstupu do súboru však program do konzoly vypisuje počet fintes ohodnotení po každej iterácii algoritmu. Okrem toho, po skončení jedného behu, program vypíše koľko fitnes ohodnotení bolo potrebných na nájdenie riešenia a koľko krokov bolo potrebných na nájdenie všetkých kusov potravy na mriežke.

Príloha D. Obsah priloženého média

1. Samotný dokument diplomovej práce vo formáte pdf a docx
2. Zdrojový kód DP_LearningFSM
3. Manuál na spustenie aplikácie Readme.txt