Case study k predmetu Evolučné algoritmy

Téma 17.

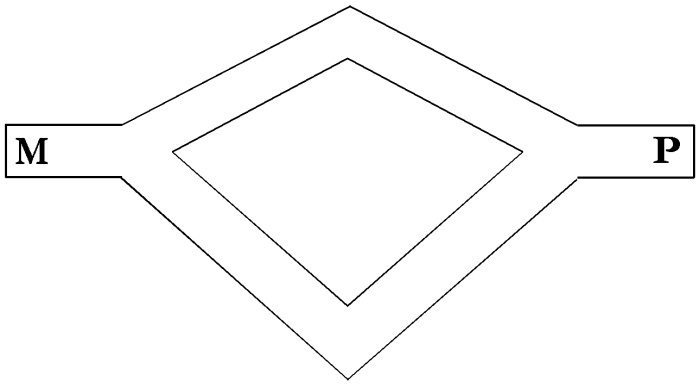
Použitie ANT COLONY optimalizácie na hľadanie najkratšej cesty v bludisku.

Vlado Škopek (skopek@pobox.sk**)**

január 2001

1. Úvod

Algoritmy založené na simulácii správania mravcov prvý krát použil Marcelo Dorigo a kol.[2] ako multiagentový prístup k ťažkým kombinatoricky optimalizačným problémom. Z nich asi najznámejší je *problém obchodného cestujúceho*. V krátkej histórií tejto metódy stále pribúdajú prístupy a aplikácie mraveniskových algoritmov. Posledné aplikácie sa zaoberajú smerovaním vozidiel, ofarbovaním grafov, smerovaním v komunikačných sieťach atď.
Objavenie tohto algoritmu je zrejmý, vznikol pozorovaním skutočných mravcov a spôsobu ich komunikácie v prostredí. Sociálne správanie mravcov v mravenisku je orientované hlavne na prospech kolónie a nie jednotlivca. Zaujímavé na mravcoch ich schopnosť nájsť potravu a potom nájdenie najkratšej cesty medzi potravou a mraveniskom.
Tajomstvo ich šikovnosti sa ukrýva v tom, že vypúšťajú do prostredia látku, ktorá sa nazýva feromón, ktorý za nimi vytvára feromónovú stopu. Mravce tieto stopy aj cítia a pri svojej ceste ich sledujú a ak sa krížia, tak na základe ich intenzity sa rozhodujú ako budú pokračovať. Táto stopa im tiež umožňuje vrátiť sa do mraveniska ak našli potravu (a naopak). Takto vznikne niekoľko ciest medzi mraveniskom a potravou. Mravce sú potom schopné postupne odhaliť najkratšiu cestu k potrave, ktorú začne väčšina používať.
Zaujímavý pokus so skutočnými mravcami spravil Doneubourg a kol.[3]. Urobil pre mravcov jednoduché prostredie kde mali svoje mravenisko a zdroj potravy im umiestnil tak, že sa k nej dalo dostať len cez dva mosty. Cesty medzi potravou a mraveniskom boli cez tieto mosty úplne rovnako dlhé. Výsledkom tohto experimentu bolo to, že mravce po počiatočnom rovnomernom rozdelení (v ktorom mohli byť malé oscilácie) začali konvergovať iba k jednej ceste.
Je to spôsobené tým, že tieto oscilácie narástli tak, že na jednej ceste sa objavovalo viac feromónov ako na druhej, čo spôsobilo odklonenie aj ostatných mravcov.



Obr. 1 – M - mravenisko; P - potrava

Tento experiment môžeme pozmeniť tak, že jeden most predĺžime. Výsledok tohto experimentu je ešte zrejmejší. Je jasné, že mravce si vyberú kratšiu cestu, ale poďme sa pozrieť na to ako sa k takémuto výsledku dostanú.
Na začiatku predpokladajme, že na mostoch nie je žiaden feromónový pach a v takomto prípade sa mravce rozhodujú úplne náhodne. Teda na začiatku sa rozdelia približne rovnomerne. Lenže polovica mravcov, má kratšiu cestu a pri opätovnom príchode na križovatku sa už začnú viac rozhodovať pre silnejšiu stopu z kratšej cesty. Prečo je ale silnejšia? Tu vstupuje do hry ešte jeden dôležitý faktor a to je *vyparovanie* feromónov. Bez toho by sa totiž po príchode pomalšej skupiny opäť vyrovnali množstvá feromónov na križovatke a teda mravce by mali zasa možnosť vyberať z dvoch rovnako silných vetiev. Vďaka vyparovaniu v čase, sa ale viac odparí z dlhšej cesty. Z toho teda vyplýva, že kratšia cesta získava čoraz silnejšiu stopu a teda sa aj viac mravcov rozhoduje pre túto vetvu, až to skonverguje do stavu kedy skoro všetky mravce chodia kratšou cestou.
[1]Z tohto popisu je vidieť, že na tomto princípe môže fungovať distribuovaný optimalizačný mechanizmus, do ktorého každý mravec prispieva len veľmi malým vplyvom. Je zaujímavé, že jediný mravec môže byť schopný nájsť cestu k potrave, len skupina mravcov dokáže nájsť optimálne riešenie teda najkratšiu cestu k potrave. Teda hľadanie najkratšej cesty k potrave je emergentným správaním mraveniska. Mravce toto riešenie nachádzajú pomocou nepriamej komunikácie lokálnou zmenou svojho prostredia[1].
Cieľom tohto článku je ukázať možnosť využitia týchto techník a vlastností v multi-agentových systémoch použitých na riešenie zložitých optimalizačných problémov. V skutočnosti sa to dá hľadanie najkratšej cesty ľahko previesť na riešenie určitých typov problémov agentamy splnením následovných podmienok: (i) priradením

[1] V anglickej literatúre sa toto označuje pojmom „stigmergy“

stavovej premennej každému stavu riešenia a (ii) umožniť agentom len lokálny prístup k čítaniu a modifikovaniu týchto premenných.
Toto výsledky tohto správania sú zapríčinené spojením *autokatalýzy* (pozitívnej spätnej väzby) a imlicitného vyhodnocovania riešenia. Pod implicitným vyhodnocovaním riešenia rozumieme fakt, že kratšie cesty budú uskutočnené rýchlejšie ako dlhšie a teda budú posilnené feromónmi častejšie.
Autokatalýza je veľmi silná vlastnosť. Využíva sa aj v evolučných algoritmoch v selekčných a reprodukčných mechanizmoch. Ide o uprednostňovanie lepších jedincov(riešení), ktorý potom určujú smer hľadania. Tu si ale treba dávať pozor na predčasnú konvergenciu (stagnáciu), kedy proces riešenia sa môže zastaviť v nejakom riešení (lokálnom minime), ktoré nie je optimálne. Dôležité parametre, ktoré ovplyvňujú správnosť riešenia, sú vyparovanie a pravdepodobnostné parametre pri rozhodovaní.
V tomto článku zhrniem čo sa dosiahlo a čo sa robí v stratégiách a algoritmoch pracujúcich s mraveniskovými algoritmami, stručne načrtnem ich rôzne aplikácie. Ďalej popíšem algoritmus svojho riešenia, odlišnosti v prístupe k riešeniu a jeho alternatívy. Tiež predstavím výsledky dosiahnuté týmto algoritmom. V závere zhrniem svoju prácu a porovnám ju s inými mraveniskovými algoritmami.

1. Mraveniskové algoritmy

Umelé mravce majú určitú podobnosť s reálnymi, ale existujú aj rozdiely, ktoré nemajú náprotivky u skutočných mravcov. Chceme použiť mraveniskové algoritmy na riešenie aj iných problémov ako hľadanie najkratšej cesty k potrave, a tak si môžeme dovoliť dať mravcom aj iné, efektívnejšie, vlastnosti, ktoré im umožnia rýchlejšie dospievať k správnemu riešeniu. Teraz charakterizujeme "umelého mravca", jeho podobnosti a odlišnosti od skutočných.
Kolónia kooperujúcich jednotlivcov. Tak ako skutočné mravenisko aj virtuálne je zložené z kooperujúcich jednotlivcov, so spoločným cieľom. Mravce kooperujú v tom zmysle, že v stavoch riešenia zapisujú a čítajú informácie potrebné všetkých.
Feromónové stopy a komunikácia cez prostredie. Umelé mravce modifikujú nejakým spôsobom svoje prostredie, tak ako to robia aj skutočné mravce. Takisto dochádza k vyparovaniu informácie u oboch. Rozdiel je ale v tom, že umelé mravce "žijú" v diskrétnom prostredí a teda sa z lokálneho preskúmania prostredia nedá vždy zistiť napríklad smerovanie stopy, čo v spojitom skutočnom svete nie je taký problém. Toto by sa dalo riešiť v diskrétnom priestore rozšírením informácie o ďalšie zložky, ktoré by určovali smer príchodu a odchodu stopy. V mieste kríženia takýchto stôp by muselo dôjsť ale k navrstveniu týchto ciest. V doterajších riešeniach sa takýto problém, ale zatiaľ nenastolil a tak asi nemá veľký vplyv na riešenie. Bolo by to zaujímavé skúmať, ak by sa ukázalo, že skutočné mravce majú jedinečný pach a dokážu jednoznačne identifikovať svoju cestu, alebo cestu iného.
Vyhľadávanie najkratšej cesty a lokálne pohyby. Umelé a skutočné mravce majú spoločný cieľ a to nájsť najkratšiu (najlacnejšiu) cestu spájajúcu mravenisko a zdroj potravy. Pohyby v prostredí sú striktne len po susedných stavoch, čo v spojitom prostredí znamená, že mravce neskáču.
Stochastické a okamihové zmeny stavu. U oboch typov mravcov riešenie vzniká náhodným rozhodovaním v stavoch na základe množstva feromónov v blízkom okolí (susedných stavoch). Rozhodovanie je teda úplne lokálne, v čase aj v priestore.
Ako bolo horeuvedené, existujú aj vlastnosti umelých mravcov, ktoré skutočné mravce nemajú.

- Ako bolo spomínané umelé mravce žijú v diskrétnom prostredí a pohybujú sa skokovo z jedného stavu do druhého.
- Umelé mravce majú vnútorný stav. V tomto stave, záleží od aplikácie, majú uloženú históriu krokov, prípadne doposiaľ najlepšiu cestu a iné údaje využívané na zlepšenie a zrýchlenie riešenia problému.
- Časovanie problému je riešené osobitne v každej aplikácii. V niektorých aplikáciách sa feromónová stopa zanecháva až po nájdení nejakého riešenia (vtedy sa dá kvalita riešenia vyjadriť silou stopy).
- Na zlepšenie celkovej efektivity algoritmu, môžeme obohatiť schopnosti mravca o schopnosti ako napríklad predvídanie, lokálne optimalizácie, backtracking a iné ktoré skutočné mravce nemajú.

V nasledujúcej časti rozoberieme pôvodný algoritmus, ktorý používa Dorigo[1] na vysvetlenie fungovania mraveniskových algoritmov aplikovaných na diskrétne optimalizovatelné problémy.

Algoritmus
Riešenie je predstavované ako minimálna cena (cesta) cez stavy problému. Komplexnosť jedného agenta by mala byť taká, aby bol schopný nájsť nejaké riešenie sám aj keď bude obyčajne dosť slabé. Kvalitné riešenia sa objavujú iba ako emergentný jav globálnej kooperácie všetkých agentov, ktorí konkurentne budujú rôzne riešenia.

Riešenie vzniká pohybom cez stavy riešenia. Agent sa v jednotlivých stavoch rozhoduje na základe privátnych informácii (vnútorný stav, pamäť) a na základe verejnej informácie (feromónová stopa, problémovo-špecifické lokálne informácie).
Vnútorný stav nesie informácie, ktoré majú agentovi pomôcť pri hľadaní správneho riešenia. V určitých typoch riešenia problému to môže agentovi zabrániť vstupovať do stavov nevedúcich k riešeniu alebo v určitých situáciách môže striktne rozhodnúť o smere (v prípade objavenia cyklu v riešení a pod.).
Ukladanie feromónovej stopy sa môže udiať niekoľkými spôsobmi. Jeden spôsob je ukladanie feromónu po každom kroku v bludisku, tiež môžu pozdržať vyhodnotenie cesty až do času keď nájdu riešenie alebo sa používa procedúra **daemon_actions**, ktorá pozoruje globálne situáciu, a na základe globálnej znalosti môže vkladať feromón na určité miesta. Takto sa dá potom množstvo zanechaného feromónu určiť lepšie a cesta sa ohodnotí podľa jej kvality. Tieto prístupy sa dajú medzi sebou kombinovať a vhodnosť ich použitia je závislá od konkrétnej aplikácie.
Keď agent dokončí svoju úlohu, teda nájde riešenie a zanechá feromónovú stopu, tak skončí a je odstránený. Žiadna jeho znalosť sa neprenáša do ďalšej generácie.
Okrem lokálnych akcií, uskutočňovaných algoritmom, je tu jedna už spomínaná procedúra **daemon_actions**, ktorá umožňuje globálny pohľad na situáciu. Tento démon môže pozorovať, mravcov jednotlivo a na základe ich správania zbierať rôzne informácie alebo môže pozerať na všetky riešenia naraz a na ne aplikovať problemovo-špecifické optimalizačné metódy. Vylepšenia dosiahnuté touto procedúrou sa spätne vkladajú do priestoru riešenia, tak že na určité miesta alebo cesty sa pridá nejaké množstvo feromónov.
Algoritmus 1 je napísaný v pseudo-kóde, sú tu popísané len najvyššie procedúry tak, aby bola zrejmá myšlienka. Všetky časti algoritmu sa využívajú len zriedka. Napríklad on-line vkladanie a oneskorené vkladanie feromónov, sú skoro vždy vzájomne sa vylučujúce a taktiež málokedy sú oba neprítomné (feromón je vkladaný len démonom). Algoritmus som trocha skrátil oproti pôvodnému, ktorý môžete nájsť v Dorigo[1] a väčšinu názvov funkcií som preložil do slovenčiny, aby sa slovník použitý v texte dal ľahšie vysledovať v algoritme.

```
procedure ACO_Meta_heuristic()
   while (kritérium_ukončenia_nesplnené)
      vyber_aktivitu:
         generuj_mravca();
         odparovanie_feromónov();
         daemon_action();
      end vyber_aktivitu
   end while
end procedure

procedure generuj_mravca()
   inicializuj_mravca();
   M = aktualizuj_pamäť();
   while (súčasný_stav ≠ cieľový_stav)
      A = lokálna_routovacia_tabuľka();
      P = vypočítaj_pravdepodobnosti_pohybu(A,M,obmedzenia);
      ďalší_krok = vyber_krok(P,obmedzenia);
      presuň_sa_do_ďalšieho_stavu(ďalší_krok);
      if (online_feromónové_vkladanie)
         ulož_feromón();
         oprav_routovaciu_tabuľku();
      end if
      M = uprav_vnútorný_stav();
   end while
   if (online_feromónové_vkladanie)
      vyhodnoť_riešenie();
      ulož_feromón();
      oprav_routovaciu_tabuľku();
   end if
   die();
end procedure
```

algoritmus 1 – mraveniskový algoritmus, tento popis je použiteľný pre distribuované programy

Možnosť využitia tohto algoritmu je v prostrediach, ktoré sa menia počas výpočtu, sú distribuovane stochastické. Takéto problémy sa vyskytujú hlavne v telekomunikačných a dopravných sieťach. Nie sú statické a nie je jednoduché mať pre takúto variabilnosť dostatočne presný model.

Keďže komunikácia medzi mravcami prebieha len lokálne v stavoch, tak tento algoritmus nie je vhodný pre problémy, príliš veľa vrcholmi, pretože pravdepodobnosť, že viaceré mravce navštívia nejaký stav je malá a tak sa stráca význam feromónových stôp.

2. Aplikácie mraveniskových algoritmov

Táto problematika je veľmi zaujímavá a venuje sa jej veľa ľudí, ktorí vytvorili množstvo aplikácii založených na tejto myšlienke. Ich riešenia sa zaoberajú rôznymi problémami. Tieto aplikácie môžeme klasifikovať do dvoch tried rozdelených na statické kombinatoriálne problémy a na dynamické problémy.
Statické sú také kde je problém zadefinovaný raz na začiatku a už sa viac nemení počas riešenia. Klasickým príkladom tohto problému je Problém obchodného cestujúceho v ktorom zadefinovanie vzdialeností miest je dané na začiatku a za behu sa nemenia. Problémy z druhej skupiny sú také, kde problém je zadefinovaný ako funkcia nejakých množstiev, ktorých hodnoty sú dosádzané nejakým systémom, nad ktorým riešenie prebieha. Typickým problémom tejto kategórie je smerovanie v sieťach, kde sa neustále menia hodnoty zaťaženia všetkých trás.
Popíšem teraz niekoľko aplikácií z oboch tried problémov. Najskôr algoritmy riešiace statické kombinatoriálne problémy.
Ant system – bol prvým programom využívajúcim mraveniskové algoritmy. V roku 1991 tento algoritmus publikoval M. Dorigo [1]. Jeho význam je hlavne v tom, že slúžil ako prototyp pre mnoho ďalších aplikácii. Tento algoritmus má tri variácie, ktoré sa líšia v spôsobe rozdeľovania feromónov. Tieto variácie boli medzi sebou porovnané a ďaleko najlepší z nich vyšiel algoritmus, ktorý prideľoval feromónovú stopu až po dokončení trasy. V tomto riešení Dorigo zaviedol aj pojem *elitný mravec*, kde mravec s najlepším riešením dostáva od démona extra prídavok k uvoľňovanému množstvu feromónov.
Ant system bol porovnávaný s inými heuristikami na relatívne malom probléme obchodného cestujúceho(30 až 75 miest). Výsledok bol slušný ale neuspokojivý. Ant system našiel pre malé problémy najlepšie riešenie porovnateľne rýchlo ako všeobecné heuristiky s ktorými bol porovnávaný. Bohužiaľ pre problémy väčších rozmerov nedosiahol najlepšie riešenie nikdy, ale vždy skonvergoval do dobrého riešenia.
Ďalším podobným algoritmom bol **Max-Min AS** od Stützla a Hoosa (1997)[6], ktorý bol ako pôvodný, s niekoľkými odlišnosťami: feromóny prideľoval len démon, vo všetkých stavoch sa hladina feromónu mohla pohybovať len v intervale $|T_{min}, T_{max}|$ a všetky cesty sú inicializované na T_{max}. Výsledky tohto algoritmu boli signifikantne lepšie ako Ant System, pretože sa tak ľahko nedostával do stagnácie.
Ant Colony System od autorov Doriga a Gambardellovej vznikol ako zlepšenie Ant Colony algoritmu. Má niekoľko vylepšení:

- feromónové stopy mení démon off-line, po zbehnutí celej jednej iterácie
- používa iné rozhodovacie pravidlo, ktoré nazývajú pseudonáhodné proporčné pravidlo
- mravce používajú on-line ukladanie feromónov
- zaviedli ďalšiu dátovú štruktúru nazývanú *zoznam kandidátov*, ktorý poskytuje dodatočné informácie pre heuristiku a obsahuje v každom mieste i zoznam jeho d susedov preferovaných pri výbere

Tento algoritmus bol testovaný na veľa parametrov a vo všetkých porovnávaných algoritmov vyšiel ako najlepší.
Ďalším NP-ťažkým riešeným problémom bol problém kvadratického priradenia, kde ide o priradenie n prostriedkov n miestam, tak aby bola cena minimálna. Tento problém sa dá previesť na obchodného cestujúceho. Maniezzo, Colorni a Dorigo vyrobili teda algoritmus **AS-QAP** odvodený z klasického Ant System algoritmu a dosiahli vinikajúce výsledky. Ich algoritmus prekonal dovtedy známe najlepšie heuristiky na riešenie tohto problému. Bližšie pozri [4].
Ďalšie statické problémy v ktorých si vedú mraveniskové algoritmy veľmi dobre sú:

- plánovanie práce a strojov
- problém smerovania dopravy
- distribúcia tovaru nákladnou dopravou
- najmenší spoločný nadreťazec
- farbenie grafov
- sekvenčné objednávanie

Presné zadefinovanie týchto problémov, ako aj stručný popis riešenia a dosiahnutých výsledkov môžete nájsť v článku Doriga, Di Cara a Gambardellovej.

Výskum aplikácii mraveniskových algoritmov pre dynamické problémy sa zameriava hlavne na komunikačné siete, pretože siete sú distribuované, náhodne dynamické a mení sa stav siete a ako také sa hodia na riešenie mraveniskovými heuristikami, ktoré sa zameriavajú na smerovacie problémy.

Implementácie mraveniskových algoritmov v komunikačných sieťach sa delia do dvoch tried:

- orientované siete
- neorientované siete

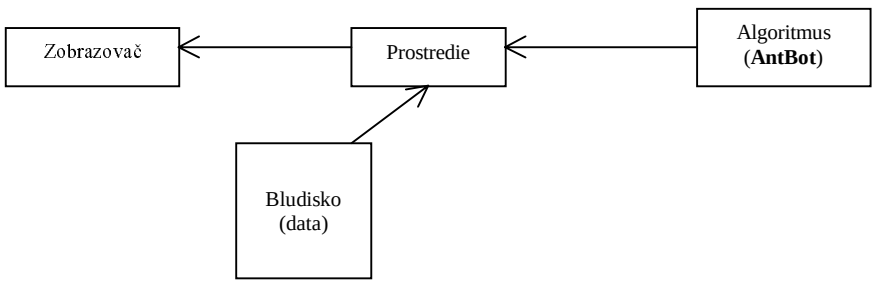
Na smerovanie v orientovaných sieťach sa zamerali Schoonderwoerd, Bruten a Rothkrantz (1996) [5] so svojim algoritmom **ant-based control** (ABC), ktorý použili na modelovanie telefónnej siete British Telecom.
V algoritme ABC sa nepoužíva žiaden démon. Každý nový hovor je prepojený alebo zamietnutý v momente prvého paketu, ktorý hľadá cestu s dostatočnou kapacitou, tak aby sa jej dĺžka približovala najkratšej ceste zapísanej v smerovacích tabuľkách.
ABC bol testovaný na modeli telekomunikačnej siete s 30 uzlami. Bol porovnávaný v percente prepojených hovorov s agentovým systémom vyvinutým v British Telecom. Výsledky boli zaujímavé, ABC dosahoval výrazne lepšiu úspešnosť ako agentový systém a to v rôznych prevádzkových situáciách. Tu chýba ešte porovnanie na zložitejších podmienkach bez obmedzujúcich podmienok a s bežne používanými algoritmami.
V neorientovaných sieťach vzniklo niekoľko algoritmov, ako napríklad od Di Cara a Doriga, ktorí vyvinuli **AntNet** algoritmus pre distribuované adaptívne smerovanie, založený na algoritmoch Ant System a ABC.
Zmena tohto algoritmu je hlavne v tom ako ukladá feromónové stopy. Pri ceste k cieľu používa oneskorený spôsob ukladania feromónu, teda až po dosiahnutí cieľa, tu na rozdiel od minulých algoritmov, mravec nekončí, ale sa otáča a vracia sa späť po svojej stope. Na svojej spiatočnej ceste už pokladá feromónovú stopu spôsobom on-line. Algoritmus odstraňuje cykly z riešenia a tak vylepšuje cestu vracajúceho sa mravca.
Testovanie tohto algoritmu na modely siete s rôznym počtom uzlov (od 8 do 150) a s rôznym zaťažením. Bol porovnávaný s najmodernejšími statickými a adaptívnymi smerovacími algoritmami. Výsledky boli vynikajúce: AntNet predviedol veľký výkon v zmysle priepustnosti siete a oneskorovania paketov. Naviac ukázal sa ako veľmi robustný a jeho vlastné zaťaženie siete bolo zanedbateľné.
V ďalšom popíšem svoj algoritmus **AntBot**, ktorý simuluje pohyb robotov bludiskom za účelom objavenia najkratšej cesty.

3. Bludiskový problém

Mojou úlohou bolo nájsť algoritmus, ktorý by našiel najkratšiu cestu v bludisku za podmienok, aby sa dal aplikovať na autonómne mobilné roboty, ktoré by našli najkratšiu cestu v bludisku bez priamej vzájomnej komunikácie, teda len komunikáciu cez prostredie.

Popis aplikácie

Aplikácia je rozdelená do niekoľkých samostatných programov, ktoré majú presne vymedzené úlohy. Progam sa delí na prostredie, zobrazovač a algoritmus. (Viď obr.2)
Prostredie – komunikuje so zobrazovačom aj s algoritmom. Zo súboru si natiahne bludisko, ktoré (ak je vo vizuálnom režime) odošle zobrazovaču. Po spustení čaká na pripojenie algoritmu, ktorý s ním komunikuje na základe niekoľkých funkcií. Tým je pre algoritmus zabezpečené, že má prístup len k lokálnej informácii. Ďalej poskytuje informácie o tom či daný robot sa nachádza pri potrave, v domčeku a či jeho riešenie je najlepšie.
Zobrazovač – Je to grafický server, ktorý čaká na príkazy. Zobrazuje bludisko, ale vlastne iba zobrazuje čiary, ktoré mu posiela prostredie. Teda o tvary a farby sa stará prostredie. Na požiadanie vie oznámiť rozmery svojej grafickej plochy, aby sa dali vypočítať rozmery zobrazovaného bludiska.
Algoritmus – Môže byť, ľubovolný algoritmus ktorý riadi pohyby robotov pomocou funkcií z knižnice prostredia. V tomto prípade sa bude používať algoritmus **AntBot.**

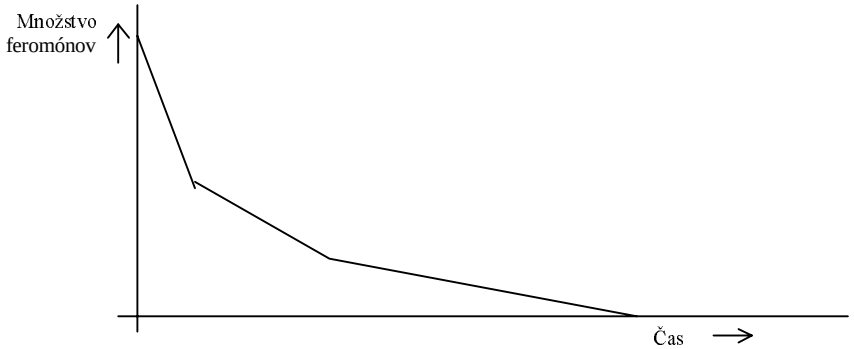


Obr. 2 – Popis architektúry prostredia na simuláciu bludiska

AntBot algoritmus

Tento algoritmus synchrónne pohybuje mravcami a po každom kole, keď sa pohne každý raz, sa spustí funkcia `MakeTimeStep()`, ktorá zrealizuje vyparovanie feromónov v celom bludisku.

Myšlienka tohto algoritmu vychádzala z toho, že malo by byť možné zrealizovať úplne distribuované riešenie mraveniskového algoritmu za takých podmienok aké majú skutočné mravce. Teda hlavne, žiadna komunikácia na veľké vzdialenosti, pokladanie feromónovej stopy on-line, bez znalosti o úspešnosti svojej cesty.
Prvá verzia programu mala na mravca len veľmi malé nároky. Na svojej ceste pokladal feromóny a v momentoch, keď prišiel do hniezda alebo k potrave, tak sa otočil. Rozhodovanie na križovatkách sa robilo ruletovým pravidlom a tak cesta s najsilnejším feromónom mala najväčšiu pravdepodobnosť. Výsledok bol úplný chaos v bludisku a ani náznak, že by niektoré jeho časti boli menej navštevované ako iné.
Každý mravec dostal pamäť svojich krokov a keď prišiel k potrave, tak sa podľa tejto pamäte vrátil. Algoritmus ale začal pracovať až vtedy keď som zaviedol formu *elitného mravca* bol to mravec, čo dosiahol najlepší výsledok, ktorý mu "oznámili" v mravenisku. On teda prechádza svoju (najlepšiu) cestu a posilňuje ju výrazne väčším množstvom feromónov. Zmenou množstva pokladaného feromónu elitným mravcom môžeme regulovať rýchlosť konvergencie riešenia.
Najväčšia zmena oproti pôvodnej metóde je v spôsobe vyparovania feromónov. Tu som vychádzal z predpokladu, že množstvo chemikálií sa nevyparuje lineárne, ale logaritmicky. Nechcel som použiť priamo logaritmickú funkciu, ale spravil som jednoduchú aproximáciu lineárnymi funkciami (obr. 3).



Obr. 3 – Funkcia charakterizujúca vyparovanie feromónov

Procedúra `AntLife()` má za úlohu urobiť jeden krok s mravcom a v špeciálnych prípadoch (v hniezde, pri potrave) robí ešte nejaké dodatočné úlohy. Najskôr si pozrime čo sa deje v týchto špeciálnych prípadoch:

- pri potrave sa len mravec obráti a zmení sa jeho stav na: vracajúci sa (Return)
- v hniezde ak je mravec vracajúci sa, tak si zistí či bol najlepší, ak áno tak si zmení stav na najlepší (Best) a znova sa vydá po svojej trase, v ostatných prípadoch si zmaže svoju históriu rozhodnutí a začína od znova

Tieto dve procedúry sú v celku jednoduché a ich pseudo-kódový prepis je uvedený ako algoritmus 2.

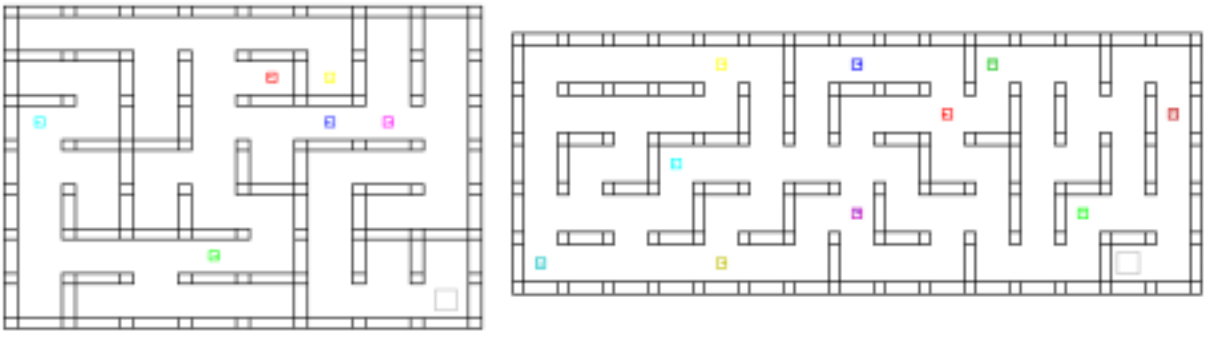
```
program AntBot()
   call_server();
   init_ants();
   while (TRUE)
      i = 0;
      while ( i < Num_Ants)
         AntLife(i);
         i = i + 1;
      end while
      MakeTimeStep();
   end while
end program

procedure AntLife(Ant)
   sensors = getSens(Ant);
   if ( križovatka )
      move = behave(sensors);
      add_to_history(move);
   else
      move = free_way(sensor);
   end if
   make_move(move);
   if (isFood(Ant))
      turn_back(Ant);
      Ant_status(Ant, Return);
   end if
   if (AtStart(Ant))
      if (Astat(Ant) = Return)
         if (AmIBest(Ant))
            Ant_status(Ant,
            Best);
         else
            Clear_history(Ant);
         end if
      else
         Clear_history(Ant);
      end if
      turn_back(Ant);
      Ant_status(Ant, Forward);
   end if
end procedure
```

algoritmus 2 – stručný prepis programu AntBot do pseudo kódu

Funkcia, ktorej ešte budeme venovať trocha pozornosti je `behave()`. V tejto funkcií sa rozlišuje stav mravca. Ak je v stave hľadania, tak má zabudované správanie ktoré nazývam **spokojnosť.** Pri použití tejto vlastnosti som vychádzal z nasledujúcej úvahy:
Mravec primárne sleduje feromónové stopy (pokiaľ tam nejaké sú) a vlastne takto sa dostáva k potrave. Ale v prípade, keď takéto stopy sleduje príliš dlho (napríklad spravil niekoľko cyklov), tak rastie jeho "nespokojnosť" s tým, že stále nič nenašiel a začne experimentovať. To znamená, že na križovatkách nepôjde pravdepodobne za najsilnejšou stopou, ale za najslabšou.
V tejto procedúre je to naimplementované takto: Pri každom rozhodovaní (jedná sa o smer k potrave) sa zväčšuje premenná **content** (spokojnosť) a rozhodovanie prebieha ruletovým rozhodovaním zo všetkých možných ciest. Keď dosiahne kritickú hranicu, tak sa s určitou pravdepodobnosťou rozhoduje na odvážny krok smerom k slabšej ceste. Ak sa rozhodne pre slabšiu, tak sa opäť ruletou vyberá, ale teraz najslabšia cesta. A zároveň sa o nejakú hodnotu zníži premenná content (trocha sa uspokojí).

4. Výsledky algoritmu

Podobný problém som už riešil aj pomocou evolučných algoritmov, ktoré vyvíjali 4-stavové automaty na prechádzanie bludiska. Podmienky v ktorých boli oba algoritmy použité, neboli úplne rovnaké, a tak ich nemôžem jednoznačne porovnávať. Najdôležitejšia odlišnosť je v tom, že 4-stavové algoritmy sa museli natrénovať aj na to či je pred nimi stena alebo nie (túto časť úlohy ale zvládli veľmi rýchlo). Na porovnanie teda môžem uviesť, že na rovnakom bludisku uvedenom na obrázku 3a), evolučný algoritmus hľadal najkratšiu cestu s populáciou 100 agentov priemerne 30 generácii a mraveniskovému algoritmu to trvalo pri populácii 6 mravcov asi 70 ciest bludiskom. Keď to prerátame na počet sekvenčných krokov tak agentom to trvalo 45 000 krokov a mravcom to trvalo iba 6 300 krokov. Je to významný rozdiel. Navyše so zväčšovaním bludiska agenti prestávajú nachádzať najlepšie riešenie.
Toto riešenie by bolo teda relatívne najvhodnejšie aj na použitie v skutočných robotoch hľadajúcich najkratšiu cestu, len je tu technický problém, s vylučovaním feromónov.
Pri ďalšom porovnávaní som sa zameral na sledovanie času potrebného na nájdenie najlepšej cesty v závislosti od počtu mravcov podieľajúcich sa na riešení. Tieto testy bežali na zložitejšom bludisku s rozmermi 5 x 15, ktoré je na obrázku 3b). Toto riešenie prebieha paralelne, ale na sledovaný čas sa môžeme pozerať aj sekvenčne – z hľadiska prostredia, ktoré posúva v každej časovej jednotke iba jedným mravcom. Uvádzam to preto, lebo výsledky týchto prístupov sú rôzne. Keďže v týchto výpočtoch hrá veľkú úlohu aj náhodnosť, tak som sa snažil výsledky mierne objektivizovať. Pre každý sledovaný počet mravcov som zbehol algoritmus tri-krát a bral som do úvahy stredné riešenie.

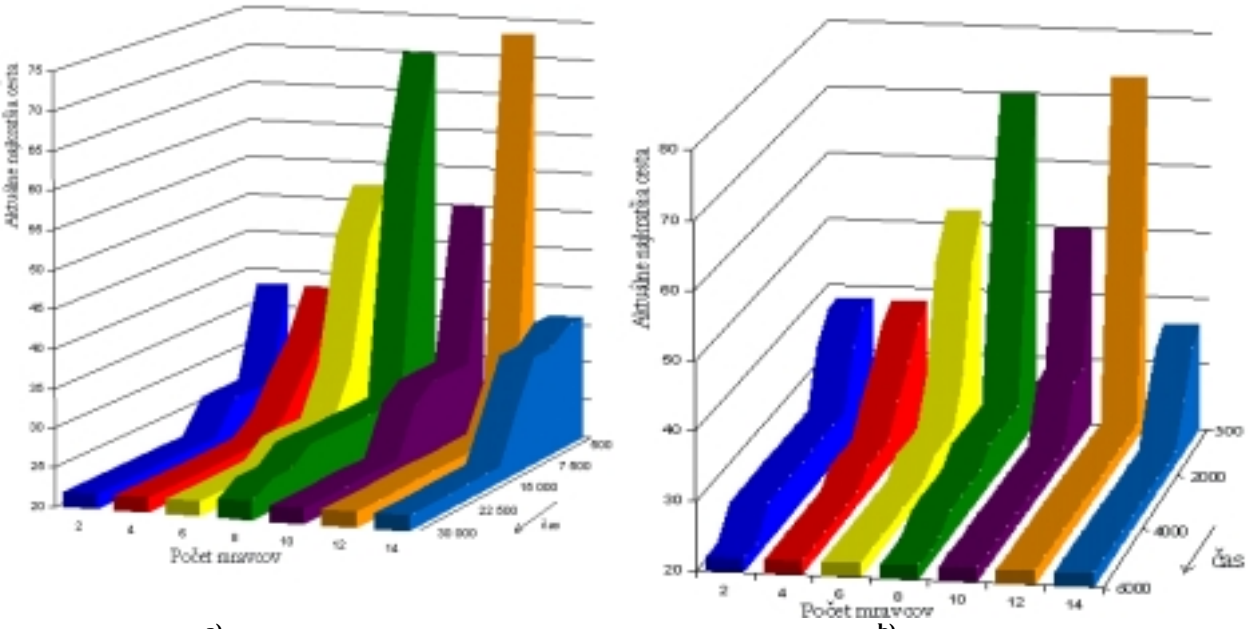


a) **b)**

Obr. 3 – Ukážka výstupu z vizuálnej časti programu

Výsledky sú uvedené v tabuľkách 1 a 2 a príslušne je ich vidieť v grafoch 1 a) a b). V tabuľke 1 (grafe 1a) je čas chápaný z pohľadu prostredia, teda koľko spravili jednotliví agenti krokov spolu, kým našli najlepšie riešenie. Ako vidieť pre nízke počty mravcov sú tieto výsledky veľmi dobré. Ale príliš často tieto riešenia skonvergovali do suboptimálneho riešenia, z ktorého sa už nedostali. Taktiež je prekvapivé, že pri väčšom počte agentov sú výsledky také dobré. Pri pohľade na simuláciu totiž bludisko pôsobí "preplneným" dojmom a zdá sa, že sa príliš zahlcuje feromónom. Opak je pravdou a je vidieť, že aj väčšia skupina dokáže účinne spolupracovať pri hľadaní riešenia.

Skupiny s väčším počtom mravcov majú aj ďalšiu výhodu a to je robustnosť. Tieto riešenia aj pri konvergencií do suboptima sa dokázali po čase z neho vymaniť a nájsť najlepšie riešenie.
Tabuľka 2 (graf 1b) zachytáva hodnoty z pohľadu reálneho času. Teda tu uvažujeme, že počas jednej časovej jednotky spravia všetky mravce jeden krok. Tu už sú výsledky jednoznačne v prospech väčšieho počtu mravcov.
Ďalšou sledovanou skutočnosťou bol rozptyl času potrebného na dosiahnutie optima. Na sledovanie som si vybral kolóniu s 12 mravcami, ktorá sa javila ako najúčinnejšia.



a) b)

Graf 1 – Časová závislosť sledovaná z hľadiska: a) počtu všetkých vykonaných krokov, b) reálneho času

2 mravce	38	26	26	26	22	22	22	22	22	22	22
4 mravce	38	32	28	24	22	22	22	22	22	22	22
6 mravcov	52	46	30	24	24	24	22	22	22	22	22
8 mravcov	70	56	26	26	26	26	26	26	24	24	22,5
10 mravcov	50	30	30	30	28	22	22	22	22	22	22
12 mravcov	73	28	22	22	22	22	22	22	22	22	22
14 mravcov	36	36	34	34	28	22	22	22	22	22	22
čas	0	3000	6000	9000	12000	15000	18000	21000	24000	2700	30000

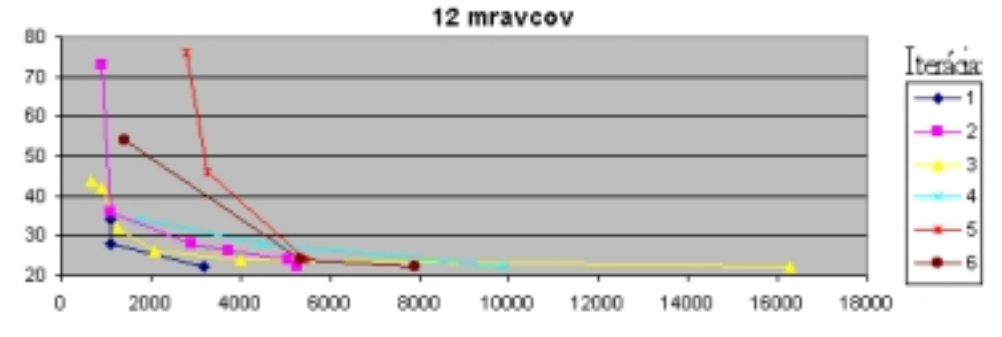
Tabuľka 1 – Sledovanie času z hľadiska všetkých vykonaných krokov

2 mravce	38	38	34	26	26	26	26	26	26	26	26	22
4 mravce	38	36	32	28	24	24	24	22	22	22	22	22
6 mravcov	52	46	30	24	24	24	22	22	22	22	22	22
8 mravcov	70	52	26	26	26	26	24	24	24	24	22	22
10 mravcov	50	30	30	22	22	22	22	22	22	22	22	22
12 mravcov	73	22	22	22	22	22	22	22	22	22	22	22
14 mravcov	36	34	28	22	22	22	22	22	22	22	22	22
čas	0	500	1000	1500	2000	2500	3000	3500	4000	4500	5000	5500

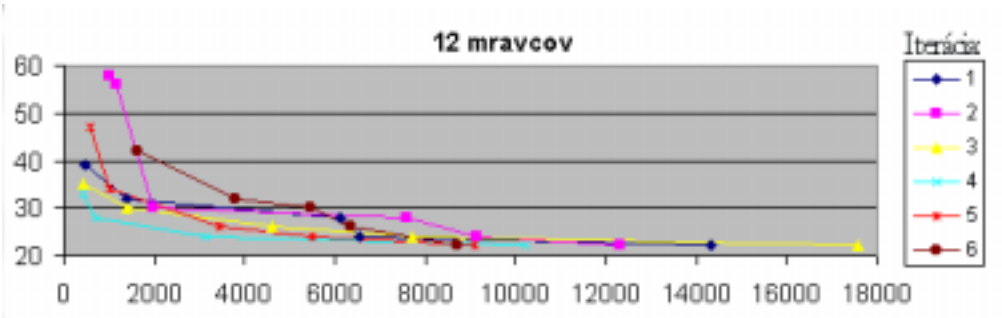
Tabuľka 2 – Sledovanie času z hľadiska reálneho času

Porovnanie rýchlosti hľadania najlepšieho riešenia je v grafe 2. Graf je zostrojený zo šiestich nezávislých behov programu. Tu je vidieť, že rozptyl riešení je dosť veľký. Oproti grafu 1a) sa nám stredná hodnota nijak významne neposunula.
Posledný graf (č. 3) nám ukazuje tiež šesť behov algoritmu AntBot, ale s malou úpravou. Algoritmus som upravil tak, že sa na začiatku inicializovala úroveň feromónu v celom bludisku na hodnotu približne dvojnásobnú oproti bežne uvoľňovanému množstvu feromónu jednotlivým mravcom. Túto črtu použili vo svojom algoritme MMAS Stützl a Hoos [6], ktorý inicializovali prostredie na maximálnu hodnotu. V mojom riešení nie je horné ohraničenie a preto som volil dvojnásobok vylučovanej hodnoty.

Ako je vidieť k zlepšeniu nedošlo, dokonca sa priemerné riešenie zhoršilo o takmer 700 krokov na hlavu. Naopak tento upravený algoritmus je výrazne lepší v dosiahnutí prvého správneho riešenia. Tiež tento algoritmus má menší rozptyl výsledkov ako porovnávaný, čo je dôležitá vlastnosť pravdepodobnostného algoritmu.



Graf 2 – 6 iterácii algoritmu AntBot



Graf 3 – 6 iterácii modifikovaného algoritmu

5. Záverečné zhrnutie

Algoritmus som testoval na bludiskách s cyklami aj bez cyklov. Oba druhy zvládal celkom uspokojivo, ale prekvapivo bludiská bez cyklov mu robili trocha väčšie problémy. Aj keď som očakával, že optimalizovanie zachádzania do slepých vetiev by mala byť vynikajúca, nebolo to vždy tak a práve relatívna vyrovnanosť úrovne feromónov v rozhodnutí sa pre správnu cestu neustále zavádzala nových agentov. V slepých cestách je o niečo viac feromónu, pretože sa tam v krátkom čase prechádza dva razy tou istou cestou.

Našli sa aj bludiská, ktoré tento algoritmus vôbec nezvládal. Paradoxne boli to bludiská ktoré nemali žiadne steny, teda to bola len sieť, mravenisko a potrava boli na jednej úrovni kde najkratšia cesta viedla priamo. Pri hlbšom zamyslení to je ale celkom pochopiteľné, pretože vlastne každý bod je rozhodovacím miestom a pravdepodobnosť, že mravec sa rozhodne viac krát ísť priamo prudko klesá s veľkosťou bludiska. V týchto prípadoch riešenia konvergovali do cesty, ktorá išla takmer po obvode bludiska.

Tiež som skúšal aj bludiská s väčšími rozmermi (najväčšie 10 x 20), ktoré algoritmus tiež zvládal. Veľkým problémom je zastavenie v suboptimálnom riešení. Tento problém sa ale bežne vyskytuje v evolučných algoritmoch a jeho riešenie všeobecne nie je jednoduché, ak je vôbec možné.

Ako sa ukazuje tento druh swarm algoritmov má veľké možnosti a aj priestor pre praktické využitie. Z uvedeného je vidieť, že na tejto téme pracuje veľa ľudí na rôznych univerzitách, čo tiež hovorí o životaschopnosti tejto myšlienky.

Ja som sa pokúšal pridržiavať sa čo najviac prírodného modelu a používať len prostriedky, ktoré majú biologické mravce. Nakoniec som musel niektoré vlastnosti porušiť, aby som dosiahol požadovaný výsledok. Táto cesta nie je tak úplne na zahodenie a keďže nevieme ako sa presne skutočné mravce rozhodujú (určite si nenosia so sebou náhodný generátor), tak musíme túto neznalosť nahradiť heuristikami, ktoré umožňujú zlepšovať výsledky a urýchľovať konvergenciu riešenia.

Zlepšenie pomocou heuristík je možné aj v tomto algoritme. Mohli by sa napríklad cesty očisťovať od cyklov, zlepšiť rozhodovanie a správanie v slepých cestách a veľa ďalších vylepšení, ktorých množstvo sa dá čerpať z uvedenej literatúry.

6. Literatúra

[1] M. Dorigo, G. Di Caro, L. Gambardella: Ant Algorithms for Discrete Optimisation, 1998

[2] M.Dorigo: Optimization,Learning and Natural Algorithms (in Italian).PhD thesis, 1992.

[3] J.-L.Deneubourg,S.Aron,S.Goss,and J.-M.Pasteels: The self-organizing ex-ploratory pattern of the argentine ant, 1990.

[4] V.Maniezzo,A.Colorni,and M.Dorigo:The ant system applied to the quadratic assignment problem, 1994.

[5] R.Schoonderwoerd,O.Holland,J.Bruten,and L.Rothkrantz.Ant-based load balancing in telecommunications networks, 1996.

[6] T.Stutzle and H.Hoos.Improvements on the ant system:Introducing MAX – MIN ant system, 1997.