Elektrotehniški vestnik XX(Y): 1–6, YEAR
Electrotechnical Review, Ljubljana, Slovenija

Optimizacija s pomočjo kolonije mravelj

Ivan Pešl, Viljem Žumer, Janez Brest

Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Smetanova ul. 17, 2000 Maribor, Slovenija

E-pošta: ivan.pesl@hermes.si

Povzetek. V naravi so mravlje sposobne najti najkrajšo pot od izvora hrane do gnezda brez uporabe vizualnih informacij. Zraven tega so se zmožne prilagoditi spremembam v okolju, recimo najti novo najkrajšo pot, ko trenutno pot preseka ovira. Pri tem se pojavi ideja, da bi lahko bilo posnemanje takšnega obnašanja mravelj učinkovito tudi v diskretnem svetu. V članku bomo prikazali reševanje problema trgovskega potnika s pomočjo optimizacije s kolonijo mravelj.

Ključne besede: kolonija mravelj, umetna inteligenca, inteligenca roja, problem trgovskega potnika

ACO - Ant Colony Optimization

Extended abstract. Ant colony optimization is a relatively new approach for solving NP-Hard problems. It has roots in the behavior of real ants. Real ants always find the closest path between nest and a food source. Such behavior can be transferred into the discrete world, were real ants are replaced by simple agents. Such simple agents are placed into the environment where can solve different combinatorial problems.

In this paper we describe an artificial ant colony capable of solving the travelling salesman problem (TSP). Artificial ants are able to generate successively shorter feasible tours by using information accumulated in the form of a phermone trail deposited on the edges of the TSP graph [1]. We can also improve basic ants behavior with added heuristic information e.g. local search.

We describe several different algorithms that are used to solve the TSP (and similar) problems. We start from the first algorithm that was first used in ant optimization named Ant System. After this algorithm many new approaches where added, which result in the better performance of ant colony optimization. The main job is to test the behavior of ants on different graphs, which are taken from *TSPLIB95* library. At the end we also show the comparison of ant algorithms on several instances of TSP.

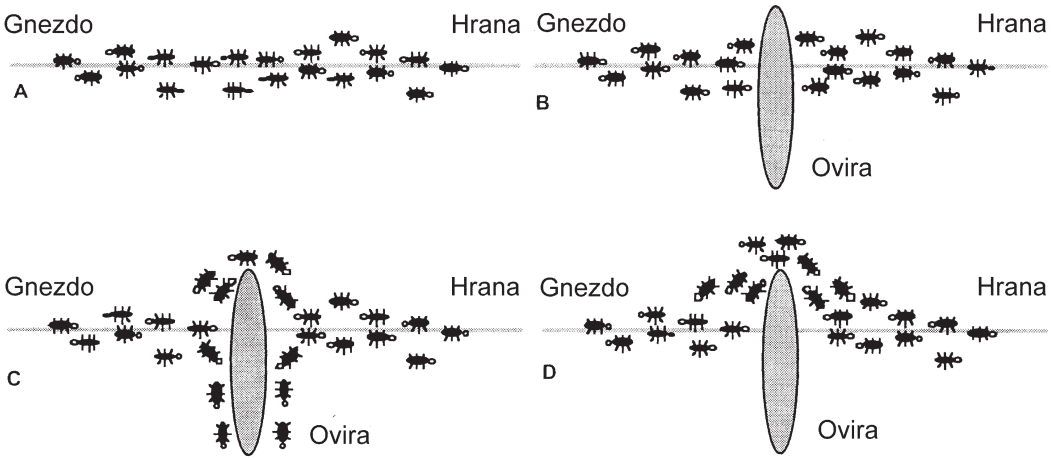
Key words: Ant Colony Optimization, artificial intelligence, swarm intelligence, travelling salesman problem

1 Uvod

V naravi so mravlje sposobne poiskati najkrajšo pot od izvora hrane do gnezda brez uporabe vizualnih informacij. Zraven tega so se zmožne prilagoditi spremembam v okolju, recimo poiskati novo najkrajšo pot, ko se sedanja prekine zaradi ovire. Na sliki 1A so prikazane mravlje, ki se gibljejo po poti, ki povezuje izvor hrane z njihovim gnezdom. Za medsebojno komunikacijo mravlje uporabljajo feromon. Poti, ki so bolj obiskane, imajo večjo količino feromona. Mravlje odlagajo feromon med hojo in vsaka mravlja daje prednost sledenju smerem, ki so bogatejše s feromonom. To elementarno obnašanje mravelj uporabimo za razlago, kako lahko poiščejo najkrajšo pot, ki ponovno poveže prekinjeno povezavo po tem, ko se je na začetni poti nenadoma pojavila ovira (slika 1B). Ko se pojavi ovira, mravlje pred oviro ne morejo nadaljevati s sledenjem poti. V tej situaciji lahko pričakujemo, da se bo polovica mravelj obrnila desno, druga polovica pa levo. Podobna situacija se pojavi tudi na drugi strani ovire (slika 1C). Zanimivo je, da bodo mravlje, ki po naključju izberejo krajšo pot okoli ovire, hitreje rekonstruirale prekinjeno pot v primerjavi z mravljami, ki so izbrale daljšo pot. Zaradi tega bo krajša pot dobila večjo količino feromona na enoto časa in čez čas bo večje število mravelj izbralo krajšo pot (slika 1D). Mravlje so prednostno nagnjene k potem z večjo vsebnostjo feromona, kar naredi nabiranje feromona še hitrejše na krajših poteh. Pokazali bomo, kako lahko naredimo podoben proces v simuliranem svetu naseljenem z umetnimi mravljami, ki skušajo rešiti problem trgovskega potnika.

Optimizacija s pomočjo kolonije mravelj ACO (Ant Colony Optimization) [1] je del večjega območja, ki se ukvarja z raziskovanjem temelječim na algoritmu mravelj oz. inteligenci roja in se ukvarja z algoritmičnimi pristopi, ki so povzeti po obnašanju kolonij mravelj in ostalih družabnih insektov. Posebnega pomena so kolektivne aktivnosti, kot so nabava hrane, skrb za zarod, izgradnja gnezd itd., ki predstavljajo mehanizme samoorganizacije, komunikacije in razdelitev opravil. Optimizacija ACO je povzeta po obnašanju mravelj pri nabavi hrane. Mravlje

Prejet 10. Februar 2004
Odobren 10. Februar 2004



Slika 1. Potovanje mravelj med izvorom hrane ter gnezdom

Figure 1. Ants travelling between the source of food and nest

uporabljajo posebne sledi feromona za označevanje poti do izvora hrane.

Optimizacija s pomočjo kolonije mravelj se uporablja za reševanje težkih kombinacijskih optimizacijskih procesov, kot so: problem trgovskega potnika, kvadratni dodelitveni problemi, problemi razvrščanja, dinamični usmerjevalni problemi v telekomunikacijskih omrežjih. Na žalost je težko teoretično analizirati algoritme ACO, glavni razlog za to je, da so osnovani na zaporedjih naključnega odločanja, ki običajno ni neodvisno in pri katerem se verjetnosti porazdelitve spreminjajo od iteracije do iteracije.

V nadaljevanju bomo natančneje predstavili algoritme ACO. Pokazali bomo uspešnost reševanje na raznih primerih trgovskega potnika, tudi v kombinaciji z algoritmi za izboljšanje rešitev, kot je 2-opt.

2 Vrste algoritmov ACO

2.1 Sistem mravlje - AS

Sistem mravlje (Ant System) [3] je prednik vseh raziskav o algoritmih mravelj in je bil najprej uporabljen na problemu trgovskega potnika.

Sistem mravlje uporablja predstavitev z grafom, ki je enak grafu trgovskega potnika, dopolnjen s stroškom $\delta(r,s)$ ter z zaželenostjo $\tau(r,s)$ imenovano feromon. Feromon v času izvajanja posodabljajo mravlje. Če je AS uporabljen na simetričnih vrstah TSP, je $\tau(r,s) = \tau(s,r)$, na asimetričnih vrstah pa je možno, da $\tau(r,s) \neq \tau(s,r)$.

Sistem mravlje deluje tako, da vsaka mravlja zgenerira kompletno pot z izbiranjem mest glede na pravilo prehoda stanj (state transition rule) – mravlje se raje premikajo k mestom, ki so povezana s krajšimi povezavami (z nižjim stroškom) in z veliko količino feromona. Ko vse mravlje opravijo svojo pot, se uporabi pravilo globalne posodobitve feromona (global phermone updating rule) – na vseh povezavah del feromona izhlapi. Nato vsaka mravlja odloži količino feromona na povezave, ki pripadajo njeni poti v razmerju dolžine njene poti (povezave, ki pripadajo veliko kratkim potem dobijo večjo količino feromona). Proces se potem ponavlja.

Pravilo prehoda stanj se imenuje naključno proporcijsko pravilo (random-proportional rule) in je podano v (1). Pravilo prehoda stanj podaja verjetnost, s katero mravlja k v mestu r izbere mesto s, v katero se premakne:

$$p_k(r,s) = \begin{cases} \frac{[\tau(r,s)]\cdot[\eta(r,s)]^\beta}{\sum_{u\in J_k(r)}[\tau(r,u)]\cdot[\eta(r,u)]^\beta}, \ \check{c}e\ s \in J_k(r) \\ 0, \ sicer \end{cases} \tag{1}$$

kjer je τ feromon, $\eta = 1/\delta$ je inverzna vrednost stroška, $J_k(r)$ je množica mest, ki ostanejo, da jih obišče mravlja k in so povezane z r (da je rešitev možna), β je parameter, ki določi relativno pomembnost feromona proti strošku ($\beta > 0$).

Če v (1) pomnožimo količino feromona na povezavi (r,s) s pripadajočo hevristično vrednostjo $\eta(r,s)$, damo prednost izbiri povezav, ki so krajša in imajo večjo količino feromona.

Pravilo globalne posodobitve je implementirano kot: ko vse mravlje zgradijo svoje poti, je feromon posodobljen na vseh povezavah glede na:

$$\tau(r,s) \leftarrow (1-\alpha)\cdot\tau(r,s) + \sum_{k=1}^{m}\Delta\tau_k(r,s) \tag{2}$$

kjer je:

$$\Delta\tau_k(r,s) = \begin{cases} \frac{1}{L_k}, \ \check{c}e(r,s) \in \ poti\ od\ mravlje\ k \\ 0, \ sicer \end{cases}$$

$0 < \alpha < 1$ je parameter razpada feromona, L_k je dolžina poti narejena od mravlje k in m je število mravelj.

Posodabljanje feromona je namenjeno dodeljevanju večje količine feromona krajšim potem. Formula posodabljanja feromona je namenjena posnemanju spreminjanja količine feromona zaradi dodajanja novega feromona na povezavah, ki so jih obiskale mravlje ter izhlapevanja feromona.

Feromon postavljen na povezave igra vlogo porazdeljenega dolgo-trajajočega spomina; ta spomin ni lokalno shranjen znotraj posamezne mravlje, ampak je porazdeljen po povezavah grafa, kar dovoljuje indirekten način komunikacije.

Čeprav je sistem mravlje uporaben za iskanje dobrih rešitev za majhne TSP (do 30 mest), je potreben čas za reševanje večjih problemov neustrezen. Zaradi tega so bile predlagane tri glavne spremembe za izboljšanje algoritma, ki so pripeljale do definicije sistema kolonije mravelj (Ant Colony System – ACS).

2.2 Sistem kolonije mravelj - ACS

ACS [4] se od AS razlikuje v treh glavnih pogledih:

- pravilo prehajanja stanj zagotovi direktni način za uravnovešanje med raziskovanjem novih povezav in izkoriščanjem danih ter akumuliranih znanj o problemu,
- pravilo globalne posodobitve je dodano samo povezavam, ki pripadajo samo najboljši mravljini poti, in
- medtem ko mravlje gradijo rešitev, je uporabljeno pravilo lokalne posodobitve feromona.

Inicializacija
do {
  Vsaka mravlja je postavljena na začetno točko
  do {
    Vsaka mravlja doda *pravilo prehajanja stanj*, da naraščajoče zgradi rešitev, in *pravilo lokalne posodobitve feromona*
  } while (Vse mravlje ne zgradijo kompletno rešitev)
  Dodano je *pravilo globalne posodobitve feromona*
} while (Zaključni pogoj)

Slika 2. Algoritem za ACS
Figure 2. Algorithm for ACS optimization

Algoritem ACS, ki je prikazan na sliki 2, lahko zapišemo: m mravelj je postavljenih na n mest izbranih glede na poljubno inicializacijsko pravilo. Vsaka mravlja zgradi pot (možna rešitev TSP-ja) s ponavljajočim dodajanjem stohastičnega požrešnega pravila (pravilo prehoda stanj). Med konstrukcijo svoje poti mravlja tudi spreminja količino feromona na obiskujočih povezavah z dodajanjem lokalnega posodobitvenega pravila. Ko vse mravlje zaključijo svojo pot, je količina feromona na povezavah spet posodobljena – z dodajanjem globalnega posodobitvenega pravila. Enako kot v AS tudi tu mravlje uporabljajo za gradnjo svojih poti obe, hevristično informacijo (raje izbirajo krajše povezave) ter informacijo o količini feromona.

2.2.1 Pravilo prehoda stanj

V ACS je pravilo prehoda stanj naslednje: mravlja postavljena v mesto r izbere mesto s, kamor se bo premaknila s pravilom:

$$s = \begin{cases} \arg\ max_{u \in J_k(r)}\{[\tau(r,u)] \cdot [\eta(r,u)]^\beta\}, \\ \qquad če\ q \leq q_0\ (izkoriščanje) \\ S,\ sicer\ (pristransko\ raziskovanje) \end{cases} \tag{3}$$

kjer je q naključno število uniformno porazdeljeno v [0...1], q_0 je parameter $(0 \leq q_0 \leq 1)$ in S je naključna spremenljivka izbrana glede na verjetnost porazdelitve v (1).

Pravilo prehajanja stanj sestavljeno iz (3) in (1) se imenuje pseudo naključno proporcionalno pravilo (pseudo random proportional rule). To pravilo prehajanja stanj je prav tako kot prejšnjo naključno proporcijsko pravilo naklonjeno prehodom k mestom, ki so povezana s krajšimi povezavami in z veliko količino feromona. Parameter q_0 določa relativno pomembnost izkoriščanja napram raziskovanja: vsakič ko mora mravlja v mestu r izbrati mesto s za premik, naključno izbere število $0 \leq q \leq 1$. Če je $q \leq q_0$, potem je izbrana povezava glede na (3), sicer je izbrana povezava glede na (1).

2.2.2 Globalno posodobitveno pravilo

V ACS samo globalno najboljša mravlja (mravlja, ki je zgradila najkrajšo pot od začetka procesa) lahko položi feromon. Ta izbira skupaj z uporabo pseudo naključnega proporcionalnega pravila naredi iskanje bolj direktno: mravlje iščejo v soseščini trenutno najboljše najdene poti. Globalno posodabljanje je izvedeno, ko vse mravlje končajo svoje poti. Stopnja feromona je posodobljena z uporabo globalnega posodobitvenega pravila:

$$\tau(r,s) \leftarrow (1-\alpha) \cdot \tau(r,s) + \alpha \Delta\tau(r,s) \tag{4}$$

kjer je:

$$\Delta\tau(r,s) = \begin{cases} (L_{gb})^{-1},\ če\ (r,s) \in & globalno \\ & najboljše\ poti \\ 0,\ sicer & \end{cases}$$

$0 < \alpha < 1$ je parameter izhlapevanja feromona in L_{gb} je dolžina globalno najboljše poti od začetka procesa.

Namen globalnega posodabljanja je zagotovitev večje količine feromona na kratkih poteh. Enačba (4) narekuje, da samo povezave, ki pripadajo globalno najboljši poti,

dobijo ojačitev. Preizkušen je tudi drugi tip globalno posodobitvenega pravila, imenovan ponovitveno najboljši, ki uporablja L_{ib} (dolžina najboljše poti trenutne iteracije) v enačbi (4). Pri ponovitveno najboljšem pravilu, povezave, ki dobijo ojačitev, pripadajo najboljši poti trenutne iteracije. Preizkusi so pokazali, da je razlika med tema dvema shemama minimalna, z manjšo prednostjo pri uporabi globalno najboljšega pravila.

2.2.3 Lokalno posodobitveno pravilo

Med gradnjo rešitve za TSP, mravlje obiskujejo povezave in spreminjajo njihovo stopnjo feromona z uporabo lokalnega posodobitvenega pravila:

$$\tau(r,s) \leftarrow (1-\rho) + \rho \cdot \Delta\tau(r,s) \tag{5}$$

kjer je $0 < \rho < 1$ parameter.

Vloga lokalnega posodabljanja v ACS je, da pomeša poti, tako da so zgodnja mesta v poti ene mravlje raziskana kasneje v poteh drugih mravelj. Z drugimi besedami, učinek lokalnega posodabljanja je to, da se zaželenost povezav dinamično spreminja: vsakič ko mravlja uporabi povezavo, ta postane malo manj zaželena (ker izgubi nekaj svojega feromona). V tej smeri bodo mravlje bolje izkoristile informacijo o feromonu: brez lokalnega posodabljanja bi vse mravlje iskale v ozki soseščini najboljše prejšnje poti.

2.3 MAX-MIN sistem mravlje - MMAS

MAX-MIN sistem mravlje [5] je direktna izboljšava AS. Rešitve v MMAS so zgrajene točno tako kot v AS, to pomeni, da so verjetnosti izbire izračunane enako kot v enačbi (1). Dodatno obstaja tudi vrsta MMAS, ki uporablja pseudo naključno proporcionalno izbirno pravilo povzeto po ACS. Z uporabo tega pravila so dobre rešitve najdene zelo hitro, medtem ko je kvaliteta končne rešitve slabša.

Glavne spremembe v MMAS glede na AS so naslednje:

- za izkoriščanje najboljše najdene rešitve lahko po vsaki iteraciji samo ena mravlja doda feromon (kot v ACS),
- da se izognemo zastajanju iskanja, je dovoljen obseg sledi feromona omejen z intervalom $[\tau_{min}, \tau_{max}]$, kar pomeni, da za $\forall\tau_{ij}$ velja $\tau_{min} \leq \tau_{ij} \leq \tau_{max}$, in
- sledi feromona so inicializirane na zgornjo mejo, kar povzroči večjo raziskovanje na začetku algoritma.

Ko vse mravlje zgradijo rešitev, se sledi feromona posodobijo glede na:

$$\tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t) + \Delta\tau_{ij}^{best} \tag{6}$$

kjer $\Delta\tau_{ij}^{best} = 1/L^{best}$. Mravlja, kateri je dovoljeno, da doda feromon, je lahko iteracijsko najboljša T^{ib} ali globalno najboljša T^{gb}. Če so določene povezave pogosto uporabljene v najboljših rešitvah, bodo dobile večjo količino feromona.

Sledi feromona so v MMAS inicializirane na maksimalno vrednost feromona. Zaradi tega je raziskovanje poti na začetku algoritma povečano, ker je relativna razlika med količino feromona manj razločna.

2.4 Sistem mravelj z rangiranjem - AS_{rank}

Naslednja izboljšava je sistem mravelj z rangiranjem *Rank-Based Version of Ant System* [7]. V AS_{rank} je vedno uporabljena globalno najboljša pot za posodobitev sledi feromona, podobno kot v AS. Dodatno pa lahko še nekaj najboljših mravelj trenutne iteracije doda feromon. Da to dosežemo, so mravlje urejene po njihovi dolžini poti ($L^1(t) \leq L^2(t) \leq ... \leq L^m(t)$), in količina feromona, ki ga lahko mravlja položi, je določena z njenim položajem. Samo $(w-1)$ najboljšim mravljam je dovoljeno, da dodajo feromon. Globalno najboljši rešitvi, ki da najmočnejši odziv, je dana utež w. r-ta najboljša mravlja trenutne iteracije prispeva feromon z utežjo, dano glede na $max\{0, w-r\}$. Modificirano pravilo posodobitve feromona je naslednje:

$$\tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t) + \sum_{r=1}^{w-1}(w-r)\cdot\Delta\tau_{ij}^{r}(t) + w\cdot\Delta\tau_{ij}^{gb}(t) \tag{7}$$

kjer je $\Delta\tau_{ij}^{r}(t) = 1/L^{r}(t)$ in $\Delta\tau_{ij}^{gb}(t) = 1/L^{gb}(t)$.

3 Stagnacija iskanja

Očitno je, da imajo predlagani algoritmi ACO veliko skupnih značilnosti. Sistem mravelj je lahko viden kot prva študija za demonstriranje sposobnosti algoritmov ACO za reševanje težkih kombinacijskih problemov, ampak je njegova učinkovitost slabša v primerjavi z drugimi pristopi. Zaradi tega je predlaganih nekaj algoritmov ACO, ki močno izboljšajo učinkovitost sistema mravelj. Glavna skupna značilnost med predlaganimi izboljšavami je ta, da bolj močno izkoriščajo najboljšo najdeno rešitev med raziskovanjem.

Problem, ki se pojavi pri močnejšem izkoriščanju najboljše rešitve, je stagnacija iskanja, to je situacija, kjer vse mravlje sledijo isti poti. Zaradi tega nekateri algoritmi ACO (ACS in MMAS) predstavijo dodatne značilnosti, da se izognejo stagnaciji. V ACS imamo lokalno posodabljanje, pri MMAS pa vpeljemo meje na dovoljeno količino feromona na povezavah.

Drugi zanimiv pristop za izboljšanje rešitev dobljenih z algoritmi ACO je uporaba lokalne optimizacije. V bistvu se že kar hitro pokaže, da je za gradnjo dobrih

rešitev optimizacija s pomočjo kolonije mravelj primeren pristop. Da pa takšno rešitev še popravimo, uporabimo na njej lokalne algoritme.

4 Lokalno iskanje

Ker se je pri preizkušanju pokazalo, da konstruktivni algoritmi pripeljejo do slabših rezultatov kot lokalni iskalni algoritmi, se poraja ideja, da oba pristopa združimo. V našem primeru uporabimo ACO za iskanje rešitev, ki jih potem optimiramo z lokalnim iskanjem in dobljeno rešitev nadalje uporabimo v ACO (za posodabljanje feromona).

```
do {
   for (vsako začetno točko i) {
      for (vsako končno točko j) {
         if ((δ(i,i+1)+δ(j,j+1)) > (δ(i,j)+δ(i+1,j+1))) {
            zamenjamo povezave, ter popravimo
            pripadajoče zaporedje
         } // if
      } // for j
   } // for i
} while (imamo izboljšavo)
```

Slika 3. Algoritem za 2-opt optimizacijo
Figure 3. 2-opt optimization algorithm

Izbira lokalnega iskalnega algoritma je odvisna od problema, ki ga rešujemo. Za TSP se uporabljata algoritma 2-opt (slika 3) in 3-opt, ki iterativno zamenjujeta po dve ali tri povezave, dokler ne najdeta boljše rešitve. Lokalno iskanje, ki zamenjuje več kot tri povezave, se zaradi velike časovne zahtevnosti ne uporablja. V tem članku uporabljamo samo algoritem 2-opt.

5 Eksperimentalni rezultati

Primerne nastavitve parametrov za algoritme ACO je potrebno dobiti s preliminarnimi eksperimenti. Za preizkuse smo uporabljali naslednje nastavitve:

- število mravelj m je enako kot število mest, tako da pride ena mravlja na vsako mesto,
- moč hevristične informacije β je nastavljena na 2,
- mero izhlapevanja ρ smo nastavili na 0.2, kar se je izkazalo za dobro izbrano vrednost (ne prehitro in ne prepočasno izhlapevanje),
- velikost liste kandidatov smo nastavili na 10% števila mest,
- pomembna je nastavitev količine feromona na povezavah in njegovih mej (τ_{min}, τ_{max}). Količina feromona je odvisna od dolžine najdene rešitve in je nastavljena na $\tau_{max} = 1/(\rho \cdot T^{gb})$, kjer je T^{gb} dolžina globalno najboljše najdene rešitve. Spodnja meja je izbrana glede na $\tau_{min} = \tau_{max}/(2 \cdot n)$.

Za testni prostor smo iz knjižnice TSPLIB izbrali naslednje primere:

- a280.tsp - Drilling problem (Ludwig),
- att48.tsp - 48 capitals of the US (Padberg/Rinaldi),
- lin318.tsp - 318 city problem (Lin/Kernighan),
- eil51.tsp - 51 city problem (Christofides/Eliton),
- kroA100.tsp - 100 city problem (Krolak/Felts/Nelson).

Knjižnica TSPLIB se nahaja na *http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/*.

Cilj raziskave je zraven predstavitve algoritmov ACO tudi preverjanje delovanje algoritma ACS ter vpliv lokalnega iskanja na končen rezultat. Končen rezultat primerjamo tudi z optimalnim rezultatom za določeni testni primer. Prav tako primerjamo vpliv lokalnega iskanja ter razlike z naključnimi sprehodi po grafu. Hitrosti algoritma z drugimi algoritmi nismo primerjali, ker je težko zagotoviti enake pogoje, kot so bili izvedeni v drugih raziskavah.

V tabeli 1 so prikazani rezultati reševanja problemov trgovskega potnika. V prvem stolpcu so imena problemov. V drugem je njihova optimalna rešitev oz. najboljša najdena rešitev. Tretji stolpec vsebuje rešitev dobljeno z algoritmom ACS brez lokalnega iskanja. Četrti stolpec vsebuje rešitev z uporabo algoritma ACS ter dodanega lokalnega iskanja. Za lokalni iskalni algoritem smo uporabili 2-opt, ki je uporabljen na vsakih 25 iteracij. V petem stolpcu prikazujemo rezultate dobljene samo z algoritmom 2-opt, ki je dodan rešitvi dobljeni z naključnim sprehajanjem po grafu. V zadnjem stolpcu so rešitve dobljene z naključnim sprehajanjem po grafu. Pri naključnem sprehajanju po grafu izbiramo samo najbližja mesta iz liste kandidatov, kar pomeni, da je to naključno sprehajanje rahlo vodeno. Rezultati so dobljeni s tridesetimi ponovitvami vsakega algoritma, ter so podani kot najslabša (Max), najboljša (Min) ter povprečna (Avg) vrednost.

Iz prikazanih rezultatov vidimo, da je reševanje optimizacijskih problemov z ACO učinkovito, še posebej kadar jih uporabimo skupaj z algoritmi za izboljšanje rešitev (primerjava ACS in ACS + 2-opt). Algoritem ACS gradi rešitev (konstrukcijski algoritem), 2-opt pa je po svoji naravi algoritem, ki izboljšuje rešitev. Rezultati algoritma ACS so primerljivi z rezultati algoritma 2-opt. Vidi se tudi razlika pri uporabi samo algoritmov ACS za gradnjo rešitev, kar potrjuje tezo, da se splača združiti algoritme, ki rešitev gradijo, z algoritmi, ki izboljšujejo

Problem	Optimalna	ACS			ACS + 2-opt			2-opt			Naključno		
	rešitev	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg
att48	33523	36588	35116	35764	33957	33523	33686	37323	34604	35552	77017	70614	74032
eil51	426	459	443	453	432	428	430	488	446	456	912	878	891
kroa100	21282	23245	22601	22968	21390	21294	21343	23762	22020	22871	75580	72374	74232
a280	2579	3016	2940	2969	2711	2625	2668	2919	2821	2873	13925	13286	13718
lin318	42029	47829	45365	46671	43325	42974	43202	46916	44159	45488	255012	243265	250330

Tabela 1. Rezultati algoritmov na TSPLIB testnih primerih

Table 1. Results of algorithms on the TSPLIB test instances

rešitev. To tezo nam dokazuje primerjava rezultatov algoritma ACS z uporabo algoritma 2-opt ter samo z algoritmom 2-opt ali ACS. Dobljeni rezultati se ne razlikujejo veliko od optimalnih, včasih jih celo dosežejo. Prednost algoritmov ACO je tudi ta, da so primerni za paralelno obdelavo [6] (vsaka mravlja lahko gradi pot na svojem procesorju, pomnilnik (feromon) pa je skupen).

6 Tehnike za pohitritev algoritma ACS

Časovna zahtevnost predstavljenega algoritma ACS je $O(n^2)$, kjer je n število mest. Tudi zahtevnost lokalnega iskanja (2-opt) je $O(n^2)$, kar da skupaj časovno zahtevnost $O(n^4)$. Takšno časovno zahtevnost se da zmanjšati z uvedbo tehnike imenovane lista kandidatov [2]. Lista kandidatov vsebuje mesta, ki so najbližje trenutnemu mestu (urejena po naraščajočem vrstnem redu) in ko izbiramo naslednje mesto za obisk, vzamemo mesto iz te liste. Če smo obiskali že vsa mesta iz liste kandidatov, potem vzamemo prvo še ne obiskano mesto. Praktične izkušnje so pokazale, da lahko lista kandidatov vsebuje sorazmerno malo število mest za enako učinkovitost iskanja. Recimo za TSP s 500 mesti je dovolj od 6 do 10 mest. Enako tehniko lahko uporabimo tudi pri lokalnem iskanju [2].

Podobno situacijo imamo tudi pri posodabljanju feromona. Ker je feromon shranjen v matriki z $O(n^2)$ elementi (en za vsako povezavo), mora biti tudi feromon posodobljen za vsako povezavo v vsaki iteraciji. Ker je to draga operacija, posodobimo samo feromon, ki se nahaja v listi kandidatov vsakega mesta.

7 Sklep

V delu je predstavljena analiza algoritmov ACO ter njihova uporaba za reševanje problema trgovskega potnika. Največji poudarek je na algoritmu ACS, ki je med predstavljenimi algoritmi najuspešnejši, posebej kadar ga združimo z algoritmi, ki izboljšujejo rešitev. Dobljeni rezultati potrjujejo uporabnost takšnega pristopa za reševanje težkih optimizacijskih problemov.

8 Literatura

[1] Marco Dorigo, Thomas Stutzle: Ant Colony Optimization, *MIT Press*, 2004.

[2] Thomas Stutzle, Marco Dorigo: ACO Algorithms for the Traveling Salesman Problem, *Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications*, pp. 163-184, Belgium, 1999.

[3] Marco Dorigo, Luca Maria Gambardella: Ant colonies for Traveling Salesman Problem, *BioSystems*, pp. 73-81, 1997.

[4] Marco Dorigo, Luca Maria Gambardella: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Transaction on Evolutionary Computation, Vol. 1*, pp. 53-66, 1997.

[5] Thomas Stutzle, Holger Hoos: MAX-MIN Ant System and Local Search for Combinatorial Optimization Problems, *Meta-Heuristic, Advances and Trends in Local Search Paradigma for Optimization, Kluwer Academic Publishers*, pp. 313-329, 1998.

[6] Bernd Bullnheimer, Gabriele Kotsis, Christine Strauss: Parallelization Strategies for the Ant System, *SFB Adaptive Information Systems and Modelling in Economics and Management Science*, Austria, 1997.

[7] Bernd Bullnheimer, Richard F. Hartl, Christine Strauss: A New Rank Based Version of the Ant System, *SFB Adaptive Information Systems and Modelling in Economics and Management Science*, Austria, 1997.

Ivan Pešl je diplomiral leta 2002 na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Trenutno je študent podiplomskega študija na omenjeni fakulteti. Njegovo področje raziskovanja so evolucijski algoritmi. Od leta 2000 je zaposlen v podjetju Hermes Softlab d.d.

Viljem Žumer je redni profesor na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Vodi Inštitut za računalništvo in Laboratorij za računalniške arhitekture in jezike. Področja, s katerimi se ukvarja, so programski jeziki, paralelno in porazdeljeno procesiranje ter računalniške arhitekture.

Janez Brest je diplomiral leta 1995, magistriral leta 1998 in doktoriral leta 2001 na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Od leta 1994 je zaposlen v Laboratoriju za računalniške arhitekture in jezike, kjer se ukvarja s spletnim programiranjem, s paralelnim in porazdeljenim procesiranjem. Njegovo področje dela so tudi programski jeziki, ukvarja pa se tudi z optimizacijskimi raziskavami.