



UNIVERZITET SINGIDUNUM
DEPARTMAN ZA POSLEDIPLOMSKE STUDIJE

**-MASTER STUDIJSKI PROGRAM-
SAVREMENE INFORMACIONE TEHNOLOGIJE**

Inteligencija jata, Optimizacija pomoću mravlje kolonije

- Master rad -

Mentor:
prof. dr Milan Milosavljević

Student:
Ivan Šarenac
br. indeksa: 410383/2013



Beograd, 2014.

Inteligencija jata, Optimizacija pomoću mravlje kolonije

Sažetak: Inteligencija jata (Swarm Intelligence - SI) predstavlja novi koncept Veštačke inteligencije i postaje svakim danom sve popularniji. Priroda je najčešći izvor inspiracije za razvoj SI tehnika. Ove tehnike su pokazale izuzetne rezultate kod rešavanja problema koji su teško rešivi uobičajenim računarskim tehnikama. Optimizacija pomoću mravlje kolonije je optimizaciona tehnika koja se pojavila ranih devedesetih godina dvadesetog veka, a njen tvorac je profesor Marco Dorigo. Izvor inspiracije za nastanak ove tehnike je ponašanje pravih kolonija mrava u potrazi za hranom. U mravljim društvima, ili upštenije društvima insekata, ponašanje jedinki kao i celokupnog društva ne nalazi se pod nekom centralizovanom kontrolom. S druge strane, prilagodljiv i snažan odgovor koji prevazilazi repertoar ponašanja jedinke može se pojaviti na nivou grupe. Ovakva kompleksna globalna ponašanja su posledica samoorganizovanja koje je rezultat lokalne interakcije i komunikacije velikog broja relativno jednostavnih jedinki. Ovakvo ponašanje koriste kolonije veštačkih mrava kako bi pronašle približna rešenja za diskretne kao i kontinualne optimizacione probleme i za rešavanje važnih telekomunikacionih problema kao što su rutiranje i raspodela opterećenja. U ovom radu najpre je opisana biološka inspiracija za nastanak ACO algoritma, a zatim i kako je ona primenjena na rešavanje diskretnih optimizacionih problema. Nakon toga dat je detaljniji opis ACO algoritma, a navedene su i najuspešnije varijante ACO algoritma.

Ključne reči: Optimizacija pomoću mravlje kolonije, Veštačke mravlje kolonije, Algoritam za diskretnu optimizaciju, Veštačka inteligencija, Inteligencija jata

Swarm Intelligence, Ant Colony Optimization – ACO

Abstract: SI represents a new concept of Artificial Intelligence and is becoming increasingly popular in recent years. Nature provides many inspirations for the development of SI techniques. These SI techniques have shown remarkable capabilities on solving problems that are often difficult to handle by conventional computational techniques. Ant colony optimization is a technique for optimization that was introduced in the early 1990's by prof. Marco Dorigo. The inspiring source of ant colony optimization is the foraging behavior of real ant colonies. In ant societies, and, more in general, in insect societies, the activities of the individuals, as well as of the society as a whole, are not regulated by any explicit form of centralized control. On the other hand, adaptive and robust behaviors transcending the behavioral repertoire of the single individual can be easily observed at society level. These complex global behaviors are the result of self-organizing dynamics driven by local interactions and communications among a number of relatively simple individuals. This behavior is exploited in artificial ant colonies for the search of approximate solutions to discrete optimization problems, to continuous optimization problems, and to important problems in telecommunications, such as routing and load balancing. In this paper, first, we deal with the biological inspiration of ant colony optimization algorithms. We show how this biological inspiration can be transferred into an algorithm for discrete optimization. Then, we outline ant colony optimization in more general terms in the context of discrete optimization, and present some of the nowadays bestperforming ant colony optimization variants.

Keywords: Ant Colony Optimization, Artificial Ant Colonies, Algorithm for Discrete Optimization, Artificial Intelligence, Swarm Intelligence

SADRŽAJ

Spisak slika.....	3
1. Uvod.....	4
2. Pojam inteligencije jata.....	6
2.1 Predmet istraživanja	7
2.2 Cilj izučavanja i način na koji se dizajnira SI sistem	8
3. Podela oblasti SI	10
3.1 Prirodna/naučna - ponašanje mrava u potrazi za hranom.....	11
3.2 Veštačka/naučna - „sakupljačko“ ponašanje kod roja robota.....	11
3.3 Prirodna/inženjerska - kolektivno ponašanje životinjskih grupa.....	12
3.4 Veštačka/inženjerska - analiza podataka zasnovana na SI tehnikama	12
4. Svojstva SI sistema	13
5. Studije i primene SI tehnika	15
5.1 „Sakupljačko“ ponašanje kod mrava.....	15
5.2 Ponašanje osa i termita prilikom izgradnje gnezda	15
5.3 Formiranje jata kod ptica i riba.....	16
5.4 Ant Colony Optimization –ACO– Optimizacija pomoću mravlje kolonije	18
5.5 Particle Swarm Optimization –PSO– Optimizacija pomoću rojeva čestica	18
5.6 Upravljanje kompjuterskim mrežama zasnovano na SI tehnikama.....	19
5.7 Kolektivna robotika	20
6. Biološka inspiracija	21
6.1 Eksperiment sa dva mosta	22
7. Optimizacione tehnike.....	24
7.1 Metaheuristike	24
7.2 Objašnjenje ACO algoritma preko primera.....	25
7.3 Najpoznatiji ACO algoritmi	27
8. ACO Metaheuristika	28

8.1 Model problema kombinatorne optimizacije.....	28
8.2 Pseudokod.....	30
9. Osnovni ACO algoritam	36
9.1 Ant System (AS).....	36
9.2 MAX-MIN Ant System (MMAS).....	37
9.3 Ant Colony System (ACS)	38
10. Teoretski rezultati.....	40
10.1 ACO i njegovi parametri	40
10.2 Optimalna vrednost parametara za TSP	41
11. Primena ACO algoritma na rešavanje Problema trgovačkog putnika – demonstracija.....	43
12. Zaključak.....	55
Literatura	56

Spisak slika

Slika 1: Samoorganizovanje u simulaciji kolonije termita Mitchel Resnick-a iz 1994.....	6
Slika 2: Primeri iz prirode – jato ptica u V formaciji i jato riba	7
Slika 3: Kolonija mrava koja traži optimalan put krećući se od izvora hrane do mravinjaka	10
Slika 4: Roboti sakupljači	12
Slika 5: Osinjaci.....	16
Slika 6: Model ponašanja Boid-a.....	17
Slika 7: Eksperiment sa dva mosta (a) oba mosta su iste dužine (b) različitih su dužina...	22
Slika 8: Mrav koji se nalazi u gradu i bira sledeći grad koji će posetiti pomoću stohastičkog mehanizma. Ako grad j još uvek nije posećen, može biti odabran sa verovatnoćom koja je proporcionalna sa količinom feromona deponovanog na grani (i,j)	26
Slika 9: Nepotpuna lista najuspešnijih ACO algoritama po hronološkom redu pojavljivanja	27

1. Uvod

Inteligencija jata je relativno noviji pristup rešavanju optimizacionih problema koji je inspirisan socijalnim ponašanjem insekata i drugih životinja. Na primer, mravi su poslužili kao inspiracija za veliki broj metoda i tehnika rešavanja problema. Najčešće korišćena i najviše proučavana i uopšte najuspešnija je široko primenljiva optimizaciona tehnika poznata pod nazivom Optimizacija pomoću mravlje kolonije. Mravlje kolonije, ili uopštenije društva socijalnih insekata, su distribuirani sistemi koji uprkos jednostavnosti jedinki koje ih čine predstavljaju visoko struktuirane i visoko organizovane socijalne grupe. Kao rezultat njihove organizovanosti, mravlje kolonije mogu obavljati složene zadatke koji u nekim slučajevima mogu daleko prevazići sposobnosti mrava kao jedinke.

Polje izučavanja “mravljih algoritama” razvilo se na posmatranju ponašanja pravih mrava i koristi ih kao inspiraciju za dizajniranje algoritama kojim se rešavaju problemi optimizacije i problemi kontrole distribuiranih procesa. Osnovna ideja je da se princip samoorganizovanja koji omogućava visoko koordinisano ponašanje pravih mrava iskoristi kako bi se koordinisalo ponašanje veštačkih mrava koji zajedno rade na rešavanju nekog problema.

Različiti aspekti ponašanja mrava inspirisali su različite mravlje algoritme, na primer: ponašanje mrava u potrazi za hranom, podela rada u mravljjoj koloniji, sortiranje larvi ili saradnja kod transporta tereta. Najuspešniji primer mravljeg algoritma poznat je kao Optimizacija pomoću mravlje kolonije, tj. Ant Colony Optimization, a otuda i skraćenica ACO. Optimizacija pomoću mravlje kolonije (ACO) inspirisana je ponašanjem jedne vrste mrava prilikom potrage za hranom. Dok se kreću ka izvoru hrane ovi mravi ostavljaju trag feromona kako bi označili put koji će do hrane pratiti ostali članovi kolonije. ACO koristi sličan mehanizam za rešavanje problema optimizacije.

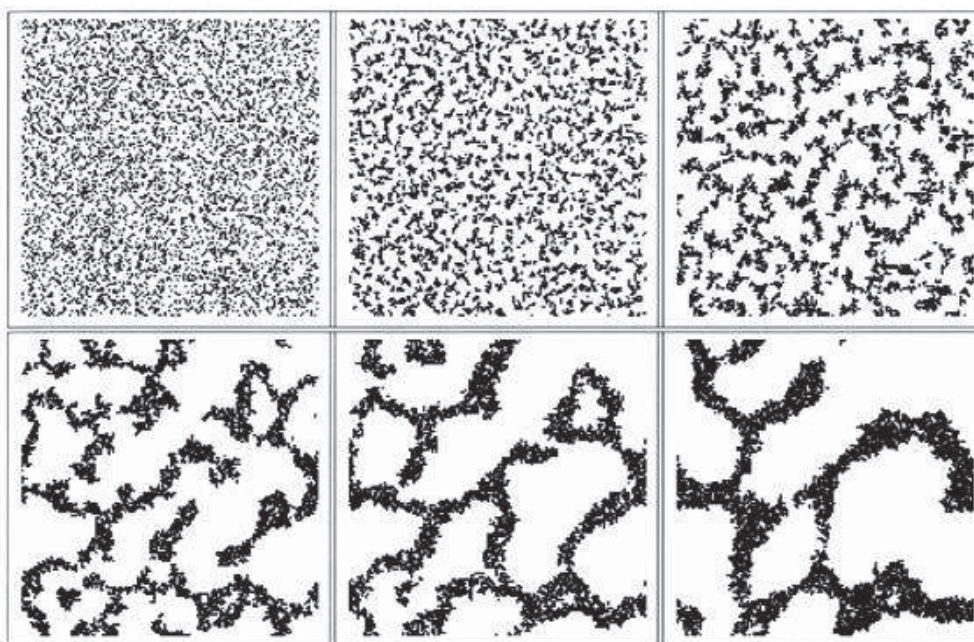
Sve od ranih devedesetih godina prošloga veka kada je predstavljen prvi ACO algoritam, ACO privlači pažnju velikog broja istraživača i do danas je napravljen veliki broj uspešnih aplikacija u kojima je algoritam primenjen. Štaviše, danas je istraživačima dostupna velika količina teoretskih rezultata vezanih za rad i primenu algoritma koji se mogu koristiti kao uputstva kod izrade aplikacija koje koriste ACO algoritam.

Cilj ovoga rada je najpre da predstavi naučnu oblast zvanu Inteligencija jata (Swarm Intelligence – SI), discipline koja se bavi veštačkim i prirodnim sistemima zasnovanim na

kolektivnom ponašanju društvenih insekata ili životinja, kao i da prikaže osnove ACO algoritma i da prati njegove najznačajnije primene. Nakon uvoda, u odeljcima dva, tri, četiri i pet dat je kraći prikaz oblasti Inteligencije jata, predmet istraživanja, podela i svojstva SI sistema, kao i najznačajnije studije i primene. U odeljku šest je prikazana biološka inspiracija za nastanak algoritma, zatim je u sedmom uprošćeno opisan ACO algoritam i navedene njegove osnovne varijante. Osmi deo daje opis ACO metaheuristike kao i njen pseudokod, dok su u devetom dati opisi najuspešnijih varijanti ovoga algoritma. U desetom delu su ukratko navedeni najznačajniji teoretski rezultati vezani za ACO, a u jedanaestom je dat primer primene ACO algoritma na rešavanje Problema trgovačkog putnika. Na kraju rada je dat zaključak.

2. Pojam inteligencije jata

Jato se može posmatrati kao skup većeg broja prostorno i vremenski distribuiranih, autonomnih jedinki koje međusobno komuniciraju (na pr. mravi, komunikacioni uređaji, računarski procesi) pri čemu svaka jedinka obavlja jedan ili više zadataka, uz malo ili bez ikakve centralizovane kontrole. Ako iz lokalne interakcije među jedinkama rezultira proces samoorganizovanja, to dovodi do interesantnih, često i korisnih ponašanja na nivou sistema, pa se može reći da tada posmatramo fenomen Inteligencije jata. Ako se posmatra kao naučna oblast, za Inteligenciju jata se može reći da je skorija računarska i bihevioristička metafora za oblast koja se bavi prirodnim i veštačkim sistemima sastavljenim od velikog broja jedinki koje sadejstvuju koristeći decentralizovanu kontrolu i proces samoorganizovanja. Ova oblast se bavi rešavanjem distribuiranih problema koji su inspirisani biološkim primerima zasnovanim na ponašanju insekata koji žive u društvima – mravi, termiti, pčele ili ose.



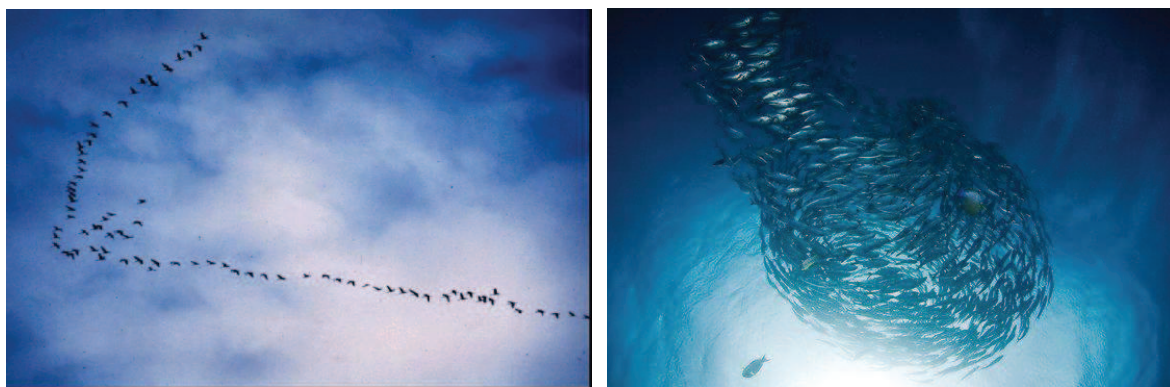
Slika 1: Samoorganizovanje u simulaciji kolonije termita Mitchel Resnick-a iz 1994.

Strukture koje proističu iz interakcije pojedinaca razvijaju se kroz proces samoorganizovanja. Samoorganizovanje je spontano, naizgled svrsishodno formiranje prostornih, vremenskih, prostorno-vremenskih struktura ili funkcija u sistemu sastavljenom od nekoliko ili od velikog broja komponenti (slika 1). Pri tome, strukture se pojavljuju na globalnom nivou kao rezultat interakcije među komponentama nižeg nivoa i interakcije

među komponentama nižeg nivoa i okruženja. Za postizanje samoorganizovanja ključni aspekt je komunikacija i to komunikacija koja se dešava na lokalnom nivou između jedinki – direktna komunikacija i između jedinki i njihovog okruženja – indirektna komunikacija ili stigmergija. Dve jedinke komuniciraju indirektno kada jedna od njih menja okruženje, tj. sredinu, a druga reaguje na novu sredinu u nekom od narednih trenutaka.

2.1 Predmet istraživanja

Može se reći da je predmet istraživanja oblasti Inteligencije jata generisanje složenosti iz jednostavnosti. Ideja je zasnovana na primećenim pojavama u prirodi, recimo na posmatranju načina na koji termiti prave mravinjake, ili kako se organizuju pčelinja društva, ili na koji način kolonije mrava tragaju za hranom, ili kako se ptice ili ribe grupišu u jata (slika 2). Postavlja se pitanje šta ova interesantna ponašanja, uprkos različitosti, imaju zajedničko?



Slika 2: Primeri iz prirode – jato ptica u V formaciji i jato riba

Pokazalo se najpre da ne postoji centralizovana kontrola, već da je kontrola potpuno distribuirana među brojnim jedinkama, odnosno pojedincima. Zatim da se komunikacija među pojedincima dešava isključivo na lokalnom nivou, a da ponašanje na nivou sistema prevazilazi oblike ponašanja pojedinaca i da je opšti odgovor sistema prilično snažan i prilagođen promenama u okruženju. Centralno pitanje vezano za funkcionisanje ovakvih sistema je kako jedinke organizovane u snažna društva, recimo insekti ili neke životinje,

recimo ptice ili ribe, koordinišu svoje akcije da bi se ispoljili neverovatni vidovi ponašanja na nivou grupe, tj. celokupnog posmatranog sistema.

2.2 Cilj izučavanja i način na koji se dizajnira SI sistem

Opštiji cilj izučavanja je razumevanje inteligentnog ponašanja bez obzira da li se ono ostvaruje kod ljudi, mašina ili životinja. Nešto uži je proučavanje kolektivnih ponašanja koja su rezultat lokalnih interakcija između velikog broja jedinki u jatu i između jedinki i njihovog okruženja, kao i primena tih saznanja na različite oblasti ljudskog delovanja, na primer za rešavanje optimizacionih problema i problema analize podataka, dizajn multi-robotskih sistema itd.

Prirodni rojevi ili jata su skupovi pokretnih jedinki koje se sve zajedno kreću i to na prvi pogled haotično i u svim smerovima, ali uprkos tome uspevaju ostati na okupu. Osnovna svojstva prirodnog roja/jata koja se mogu preneti i na apstraktni roj/jato su [9]:

- bliskost: roj može provoditi jednostavne signale,
- mogućnost odgovora: roj može odgovarati na različite kvalitativne faktore iz okoline,
- različitost odgovora: aktivnosti roja nisu ograničene samo na uske, ustaljene kanale delovanja,
- stabilnost: roj ne menja svoje ponašanje pri svakoj promeni okoline i
- adaptivnost: roj mora biti sposoban da promeni ponašanje kada se pokaže da je to od koristi.

Kod problema koji se rešavaju pomoću tehnika Inteligencije jata, dizajn se svodi na dodeljivanje računarskih resursa brojnim relativno jednostavnim jedinicama koje su bez ikakve centralizovane kontrole i koje komuniciraju na relativno jednostavan i lokalizovan način. Pri tome, cilj nam je postizanje nekog globalno korisnog ponašanja. Ustvari, modeliraju se apstraktni rojevi jedinki i opažaju se obrasci ponašanja koji se javljaju na kolektivnom nivou. Apstraktni rojevi su metafora za slobodno struktuirane, dinamične skupove „apstraktnih čestica“ u „apstraktnim prostorima“, koji pokazuju slična zanimljiva

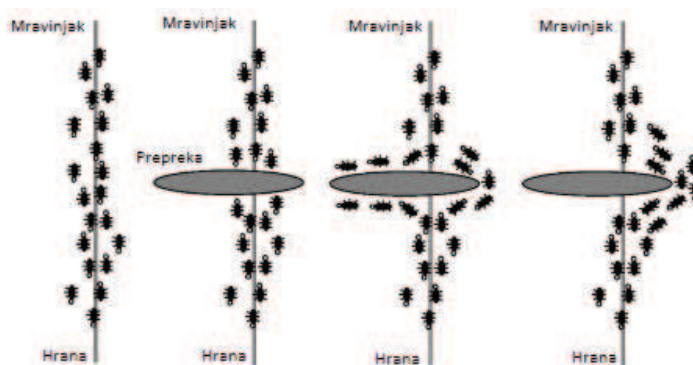
kolektivna svojstva kao i prirodna jata. Agenti, tj. učesnici su najčešće softverski entiteti koji međusobno interakuju u najčešće virtuelnom okruženju. Oni predstavljaju model za „čestice jata“. Kod modeliranja se kreće od specifikacije mikro-nivoa, tj. pravila ponašanja i interakcije pojedinaca. Zatim sledi izvršavanje računarske simulacije sa većim brojem jedinki i njihovih interakcija, kao i istraživanje emergencije na makro-nivou, tj. opažanje obrazaca ponašanja koji se javljaju na kolektivnom nivou. Za neki fenomen se kaže da je emergentan ako su za opis fenomena kao celine potrebne nove kategorije koje nisu potrebne za opis komponenti tog fenomena [5]. Fenomen roja je emergentan, što znači da nema smisla na nivou jedinke, već samo celokupnog roja i nastaje kroz međusobne interakcije članova roja i članova roja sa okolinom. Emergentna svojstva su često iznenađujuća zato što je teško predvideti sve posledice čak i najjednostavnijih oblika interakcije [2]

3. Podela oblasti SI

Multidisciplinarni karakter SI je potvrđen u činjenici da se sistemi sa navedenim karakteristikama mogu naći u različitim oblastima ljudskog delovanja. Moguća je klasifikacija SI istraživanja prema sledećim kriterijumima:

Prirodna ili veštačka - Uobičajeno je podeliti SI na dve oblasti istraživanja u skladu sa prirodom sistema koji se analizira. Stoga govorimo o prirodnim SI istraživanjima gde se proučavaju biološki sistemi i o veštačkoj SI koja se bavi ljudskim izumima.

Naučna ili inženjerska - Alternativna i može se reći više informativna klasifikacija SI istraživanja data je na osnovu ciljeva koji se žele postići tako da se može govoriti o naučnom i inženjerskom pravcu. Cilj naučnog pravca je modelovanje SI sistema kao i izdvajanje i razumevanje mehanizama koji omogućavaju sistemu u celini da se ponaša na koordiniran način, a sve kao rezultat lokalnih jedinka-jedinka i jedinka-okruženje interakcija. Sa druge strane, cilj inženjerskog pravca je korišćenje modela razvijenih u naučnom pravcu sa ciljem da se osmisle i razviju sistemi koji su u stanju da rešavaju probleme od praktičnog značaja.



Slika 3: Kolonija mrava koja traži optimalan put krećući se od izvora hrane do mravinjaka

Iako se tipično naučno istraživanje tiče prirodnih sistema, a tipično inženjerska primena odnosi na razvoj veštačkih sistema, veliki broj SI studija su sprovedene na rojevima robota radi potvrđivanja matematičkih modela bioloških sistema. Ove studije su samo spekulativne prirode i pripadaju naučnom pravcu SI. Sa druge strane, moglo bi se uticati na ponašanje ili izmeniti ponašanje jedinki u biološkom jatu tako da se ponašanje na nivou jata

koje odatle proizilazi može koristiti za rešavanje zadataka od praktičnog značaja. U ovom slučaju, iako je sistem kao takav prirodan, ciljevi koje treba postići su iz oblasti primene inženjeringa. U narednom tekstu dati su primeri za svaku od četiri moguće kombinacije.

3.1 Prirodna/naučna - ponašanje mrava u potrazi za hranom

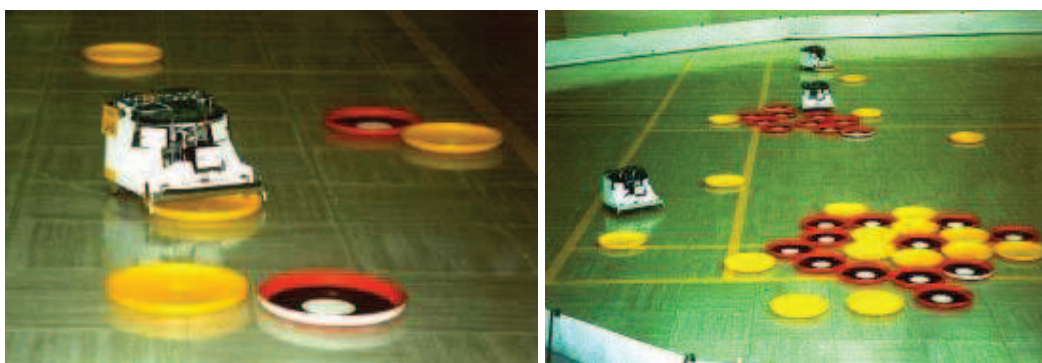
U ovom sada već klasičnom eksperimentu urađenom 1990, Deneubourg i njegovi saradnici su pokazali da kada im se da mogućnost da izaberu između dve putanje različitih dužina koje spajaju gnezdo - mravinjak sa izvorom hrane, kolonija mrava će sa velikom verovatnoćom kolektivno izabrati onaj kraći (slika 3.). Deneubourg je pokazao da se ovakvo ponašanje može objasniti preko jednostavnog modela verovatnoće u kome svaki mrav odlučuje kuda će ići na osnovu slučajnog izbora zasnovanog na intenzitetu feromona opaženog na tlu. Taj feromon duž putanje ostavljaju mravi dok se kreću od mravinjaka ka izvoru hrane i nazad.

3.2 Veštačka/naučna - „sakupljačko“ ponašanje kod roja robota

Neke vrste mrava sakupljaju mrtva tela kako bi formirali groblja. Deneubourg i njegovi saradnici 1991. bili su među prvima koji su predložili distribuirani model verovatnoće kojim bi objasnili ovakvo „sakupljačko“ ponašanje. U ovom modelu mrav podiže i ispušta mrtva tela sa verovatnoćama koje zavise od informacije o gustini mrtvih tela koja je lokalno dostupna mravima. Beckers i njegovi saradnici su 1994. programirali grupu robota koji su ispoljili slično „sakupljačko“ ponašanje i na taj način demonstrirali jednu od prvih naučno orijentisanih SI studija u kojoj su korišćeni veštački agenti (slika 4).

3.3 Prirodna/inženjerska - ispitivanje kolektivnog ponašanja životinjskih grupa

Mogući razvoj SI je u kontrolisanoj eksploataciji kolektivnog ponašanja u životinjskim grupama. Nema primera iz ove oblasti SI iako su u toku neka obećavajuća istraživanja. Na primer, u projektu Leurre (<http://leurre.ulb.ac.be/index2.html>), mali insekti-roboti se koriste kao mamac kojim se utiče na ponašanje grupe bubašvaba. Tehnologija razvijena u ovom projektu može se primeniti u različitim oblastima ljudske delatnosti uključujući poljoprivredu i stočarstvo.



Slika 4: Roboti sakupljači

3.4 Veštačka/inženjerska - analiza podataka zasnovana na SI tehnikama

Inženjeri su iskoristili model sakupljačkog ponašanja kod mrava kao inspiraciju za dizajniranje data mining algoritama (algoritama za rudarenje/pretragu podataka). Pionirski rad u ovom pravcu dali su Lumer i Faieta 1994. Oni su osmislili veštačko okruženje u kome veštački mravi kupe i ostavljaju podatke sa verovatnoćama koje su određene sličnošću sa ostalim podacima koji postoje u njihovom susedstvu. Isti algoritam je korišćen za rešavanje problema kombinatorne optimizacije ili kako se drugačije formulišu, problema grupisanja (Bonabeau i njegovi saradnici 1999.).

4. Svojstva SI sistema

Tipičan SI sistem ima sledeće osobine:

- sastoji se od velikog broja jedinki,
- jedinke su relativno homogene, tj. one su ili identične ili se mogu svrstati u nekoliko različitih tipova,
- interakcije među jedinkama su zasnovane na jednostavnim pravilima ponašanja koja koriste samo lokalne informacije koje jedinke razmenjuju direktno ili indirektno, preko okruženja i
- sveukupno ponašanje sistema je rezultat interakcije između jedinki i interakcije jedinki sa okruženjem, tj. samoorganizovanog ponašanje grupe.

Karakteristično svojstvo SI sistema je sposobnost da reaguje na koordinisan način bez prisustva koordinatora ili spoljašnjeg kontrolera. Mnogi primeri se mogu naći u prirodi gde jata ili rojevi pokazuju interesantne vidove kolektivnog ponašanja bez jedinke koja kontroliše grupu ili koja je svesna celokupnog ponašanja grupe. Bez obzira na nepostojanje jedinke koja kontroliše grupu, jato kao celina može pokazati inteligentno ponašanje, što je isključivo posledica interakcije prostorno susednih jedinki koje se ponašaju prema jednostavnim pravilima. Najčešće, ponašanje svake jedinke u jatu je opisano probabilističkim terminima, tj. ponašanje svake jedinke je stohastičko i zavisno od individualne percepcije najbliže okoline, odnosno susedstva.

Na osnovu navedenih osobina SI sistema moguće je projektovati sistem koji je skalabilan, paralelan i toleriše greške.

Skalabilnost znači da sistem može da održava svoju funkciju dok povećava veličinu bez potrebe da se redefiniše način na koji njegovi delovi međusobno interaguju. Zbog toga što u SI sistemima interakcija podrazumeva samo susedne jedinke, broj interakcija ne teži da raste sa povećanjem broja jedinki u jatu, tj. individualno ponašanje ne zavisi mnogo od dimenzija jata. Kod veštačkih sistema skalabilnost je interesantno svojstvo, jer skalabilan sistem može da poveća učinak jednostavnim povećanjem veličine bez ikakvog reprogramiranja.

Paralelne akcije su moguće u SI sistemima zato što jedinke koje čine jato mogu da obavljaju različite akcije na različitim mestima u isto vreme. U veštačkim sistemima paralelne akcije su poželjne, jer to čini sistem fleksibilnijim, tj. sposobnim za

samoorganizovanje u timove koji istovremeno rade na različitim aspektima nekog složenog zadatka.

Tolerantnost na greške je inherentno svojstvo SI sistema usled decentralizovane, samoorganizujuće prirode njihovih kontrolnih struktura. Pošto se sistem sastoji od velikog broja izmenjivih jedinki i nijedna od njih ne kontroliše ponašanje sistema kao celine, jedinka koja greši se može lako „otpisati“ i zameniti drugom koja u potpunosti funkcioniše.

5. Studije i primene SI tehnika

U daljem tekstu je dat kratak prikaz nekoliko primera naučnih i inženjerskih studija SI sistema. Od naučnih opisano je ponašanje karakteristično za koloniju mrava, kao i kolonije osa i termita. Kao primer inženjerskih studija ukratko je opisana jedna od prvih računarskih simulacija koja imitira ponašanje karakteristično za jata ptica i riba, kao i dve najpoznatije metaheuristike koje u svojim kodovima imitiraju ove vidove ponašanja.

5.1 „Sakupljačko“ ponašanje kod mrava

Mravi grade groblja sakupljanjem mrtvih tela na jednom mestu u mravinjaku. Oni takođe organizuju prostorno raspoređivanje larvi u odeljke sa manjim larvama u centru odeljka i većim, tj. starijim, na periferiji. Ovakvo ponašanje kod mrava je inspirisalo niz naučnih studija. Naučnici su formulisali jednostavne modele verovatnoće ovih ponašanja i testirali ih kroz simulacije (Bonabeau i njegovi saradnici 1999.). Osnovni modeli kažu da će neopterećeni mrav pokupiti mrtvo telo ili larvu sa verovatnoćom koja je obrnuto proporcionalna lokalno uočenoj gustini, dok je verovatnoća da opterećeni mrav ispusti teret proporcionalna lokalnoj gustini sličnih objekata (mrtvo telo ili larva). Ovaj model je potvrđen eksperimentalnim podacima dobijenim na osnovu ponašanja stvarnih mrava. U taksonomiji ovo je primer prirodno/naučnog SI sistema.

5.2 Ponašanje osa i termita prilikom izgradnje gnezda

Ose prave gnezda - osinjake sa veoma složenom unutrašnjom strukturom koja je daleko izvan kognitivnih sposobnosti jedne ose (slika 5). Termiti grade gnezda - mravinjake čije dimenzije mogu dostići više metara u visinu ili u prečniku i koji su ogromni u odnosu na jedinku termita koja meri tek nekoliko milimetara. Naučnici su proučavali mehanizme koordinacije koji omogućavaju izgradnju ovih objekata i formulisali modele verovatnoće

koji koriste stigmergijsku komunikaciju kako bi objasnili ponašanje ovih insekata. Neki od modela su implementirani u kompjuterske programe i iskorišćeni za izgradnju simuliranih struktura koje podsećaju na morfologiju pravog gnezda (Bonabeau i njegovi saradnici 1999.). U taksonomiji ovo je primer prirodno/naučnog SI sistema.



Slika 5: Osinjaci

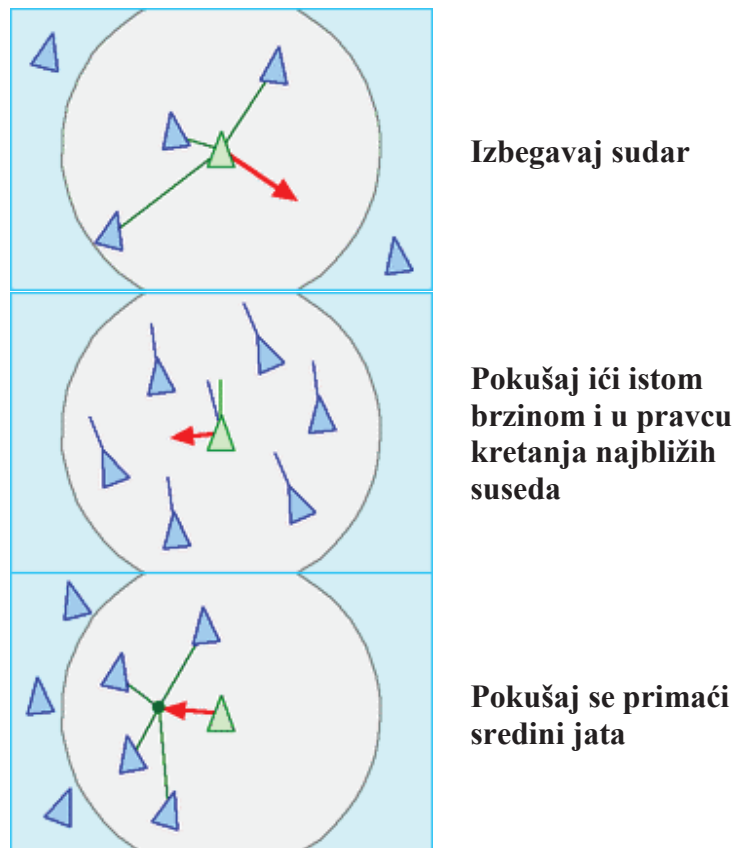
5.3 Formiranje jata kod ptica i riba

Formiranje jata je primer visoko koordinisanog grupnog ponašanja velikog broja ptica ili riba. Naučnici su pokazali da se ovakva „elegantna“ ponašanja na nivou jata mogu razumeti kao rezultat procesa somoorganizovanja u slučaju kada ne postoji lider, tj. jedinka-vođa i gde svaka jedinka zasniva svoje odluke o kretanju isključivo na lokalno dostupnim informacijama: daljina, procenjena brzina i pravac kretanja najbližih suseda. Ove studije su inspirisale veliki broj kompjuterskih simulacija koje se koriste u kompjuterskoj grafičkoj industriji, filmskoj industriji i kompjuterskim igrama. Prva je bila simulacija Craig Reynolda iz 1986. nazvana Boids (slika 6). Objekti koji formiraju jato nazvani su Boid-ima. To su jednostavni agenti koji ne poseduju nikakvu internu reprezentaciju okoline, nemaju ciljeve, ne planiraju delovanje već sprovode određene akcije kada su zadovoljeni određeni uslovi. Osnovni model njihovog ponašanja je zasnovao na tri jednostavna pravila:

- izbegavati sudar,
- pokušati ići brzinom kojom idu najbliži susedi i u prosečnom pravcu kretanja najbližih suseda i
- pokušati da se primakne središtu jata.

Implementacijom na računaru dobijeni su vrlo realistični prikazi kretanja jata. Osnovni model je kasnije proširen uvođenjem kompleksnijih pravila kao na primer izbegavanje prepreka i traženje cilja.

U taksonomiji ovo su primeri prirodno/naučnog i veštačko/inženjerskog SI sistema respektivno.



Slika 6: Model ponašanja Boid-a

5.4 Ant Colony Optimization –ACO– Optimizacija pomoću mravlje kolonije

Stigmergija je dovela do razvoja algoritma Optimizacije pomoću mravlje kolonije, tj. za njega je bitan ovaj vid komunikacije. Optimizacija pomoću mravlje kolonije (Dorigo, Meniezzo i Colorni 1991.; Dorigo i Stütze 2004.) je metaheuristika zasnovana na populaciji koja se može koristiti za pronalaženje približnih rešenja teških optimizacionih problema. Algoritam je inspirisan ranije opisanim „sakupljačkim“ ponašanjem kod mrava. U ACO algoritmu skup softverskih agenata nazvanih „veštački mravi“ traga za dobrim rešenjem za dati optimizacioni problem preveden u problem nalaženja minimalne cene puta u težinskom grafu. Veštački mravi postepeno nalaze rešenje kretanjem po grafu. Proces nalaženja rešenja je stohastički i zasnovan je na feromon modelu, odnosno skupu parametara povezanih sa komponentama grafa (bilo čvorovima ili granama) vrednostima koje mravi modifikuju u toku rada. ACO je uspešno primenjen na mnoge klasične kombinatorne probleme optimizacije, kao i diskretne optimizacione probleme koji imaju stohastičke i/ili dinamičke komponente.

Primeri primene ACO algoritama su aplikacije za rutiranje u komunikacionim mrežama (videti odeljak upravljanje mrežama zasnovano na jatima) i stohastičke verzije dobro poznatih kombinatornih problema kao što je Problem trgovačkog putnika (Traveling Salesman Problem - TSP). ACO algoritam je proširen tako da se može koristiti za rešavanje kontinualnih i optimizacionih problema mešovitenih promenljivih (Socha i Dorigo u štampi).

ACO je verovatno najuspešniji primer veštačko/inženjerskog SI sistema sa velikim brojem primena u realnim problemima.

5.5 Particle Swarm Optimization –PSO– Optimizacija pomoću rojeva čestica

Optimizacija pomoću rojeva čestica (Kennedy i Eberhart 1995.; Kennedy i Shi, 2001.) je stohastička optimizaciona tehnika zasnovana na populaciji za rešavanje kontinualnih optimizacionih problema. Inspirisan je socijalnim ponašanjem u jatima ptica i riba. U PSO algoritmu skup softverskih agenata, nazvanih čestice, traga za dobrim rešenjem za dati kontinualni problem optimizacije. Svaka čestica je potencijalno rešenje razmatranog

problema i koristi sopstveno iskustvo i iskustvo susednih čestica da bi odabrala kako da se kreće u prostoru pretrage. U fazi inicijalizacije svakoj čestici je dat slučajno odabrani položaj i početna brzina. Pozicija čestice predstavlja rešenje problema i zato ima vrednost datu funkcijom cilja. Dok se kreću u prostoru pretrage čestice pamte položaj, tj. poziciju najboljeg rešenja koje su našle. U svakoj iteraciji algoritma svaka čestica se kreće brzinom koja je suma tri komponente: stare brzine, brzine koja pokreće česticu ka poziciji u prostoru pretrage gde je prethodno pronašla najbolje rešenje do tada i brzine koja pokreće česticu prema lokaciji u prostoru pretrage gde su susedne čestice našle najbolje rešenje do tada. PSO algoritam je primenjen na veliki broj različitih problema i još je jedan primer veštačko/inženjerskog SI sistema.

5.6 Upravljanje kompjuterskim mrežama zasnovano na SI tehnikama

Prvi pristup upravljanju mrežama zasnovan na SI tehnikama predložili su Schoonderwoerd i njegovi saradnici 1996. i Di Caro i Dorigo 1998. Schoonderwoerd i saradnici predložili su Ant-Based Control (ABC) – Kontrolu zasnovanu na mravima, algoritam za rutiranje i podelu opterećenja u mrežama kod kojih se koristi prenos podataka sa komutacijom veza. Di Caro i Dorigo predložili su AntNet algoritam za rutiranje u mrežama gde se podaci prenose sa komutacijom paketa. Dok je ABC bio dokaz koncepta, AntNet (koji je ustvari ACO algoritam) je u poređenju sa mnogim vrhunskim algoritmima i njihovim performansama konkurentan, posebno u situacijama visoko dinamičnog i stohastičkog protoka podataka kakav se može naći u mrežama poput Interneta. Nadgradnja AntNet-a je uspešno primenjena na ad-hoc mreže (Di Caro, Ducattelle i Ganbardella, 2005.) na primer za rutiranje u mobilnim ad hoc mrežama gde je svaki čvor domaćin i ruter, mreže su bez infrastrukture ili centralizovane kontrole, čvorovi se mogu kretati, udružiti ili napustiti mrežu u bilo koje vreme, kratkog su dometa i bučne transmisije. Ovi algoritmi su još jedan primer uspešnog veštačko/inženjerskog SI sistema.

5.7 Kolektivna robotika

Postoji veliki broj oblika ponašanja jata posmatranih u prirodnim sistemima koji su inspirisali inovativne načine rešavanja problema pomoću grupe ili roja robota. Kooperativno ponašanje kod grupe robota je oblast Kolektivne robotike (Swarm Robotics - SR) koja privlači sve više interesovanja, a tiče se novijeg pristupa koordinaciji multi-robotskih sistema. Drugim rečima, Kolektivna robotika je primena principa Inteligencije jata za kontrolu grupe robota. Kolektivno ponašanje kod robota nastaje kroz međurobotske interakcije i interakcije robota sa okolinom, slično kao kod prirodnih rojeva. Individualna kontrola pojedinih robota zamenjuje se refleksnim i reaktivnim uticajem jednog robota na drugog. Roj čini veći broj relativno jednostavnih robota, agenata, kod kojih je kretanje često implementirano kao varijanta Reynold-ovog Boids algoritma za kretanje jata.

Osnovno svojstvo roja robota je distribucija posla, tj. podela opterećenja među robotima prilikom izvođenja nekog zadatka, kao i deljenje rezultata izvršavanja individualnih zadataka. Neka od bitnijih svojstava su i jednostavniji dizajn robota, manji i jeftiniji procesori i zamenjivost individualnih robota.

Budućnost Kolektivne robotike bi mogla biti u razvoju rojeva kognitivnih robota, tj. rojeva sa složenim agentima sposobnim za pojačano učenje kod kojih će bitna svojstva biti interna reprezentacija okoline, znanje o vlastitom delovanju i njegovim posledicama, planiranje ostvarivanja ciljeva i sposobnost improvizacije. Na primer, mogao bi se razviti robotski tim za misije traganja i spašavanja u slučajevima prirodnih i tehnoloških katastrofa.

Kao i kod SI sistema uopšte, SR sistemi imaju i naučni i inženjerski pravac. „Sakupljačko“ ponašanje u roju robota je ranije pomenuto kao primer veštačko/naučnog sistema. Primer veštačko/inženjerskog SR sistema je kolektivni prevoz objekta preteškog za jednog robota, ponašanje takođe često viđeno u koloniji mrava.

6. Biološka inspiracija

Čulo vida kod većine mravljih vrsta je rudimentalno razvijeno, a postoje i vrste koje su potpuno slepe. U stvari, veći deo komunikacije između jedinki i između jedinki i okruženja zasnovan je na korišćenju hemikalija koje mravi proizvode. Ove hemikalije nazivaju se **feromoni**. Posebno važno ponašanje za društveni život mrava je polaganje i praćenje traga feromona. Ovo **kolektivno ponašanje gde na jedinku mrava utiče trag feromona koje je ostavio drugi mrav je osnovna inspiracija za nastanak mravljih algoritama**.

Četrdesetih i pedesetih godina dvadesetog veka francuski entomolog Pierre-Paul Grasse je zapazio da neke vrste termita reaguju na kako ih je on nazvao “značajni podsticaj” ili “značajni stimulans”. Primetio je da efekti ove reakcije mogu biti novi značajni stimulans i za insekta koji ih je izazvao i za ostale insekte u koloniji. Gresse je koristio termin **stigmergija** kako bi opisao ovaj način komunikacije u kojem su “radnici stimulisani učinkom koji su postigli”.

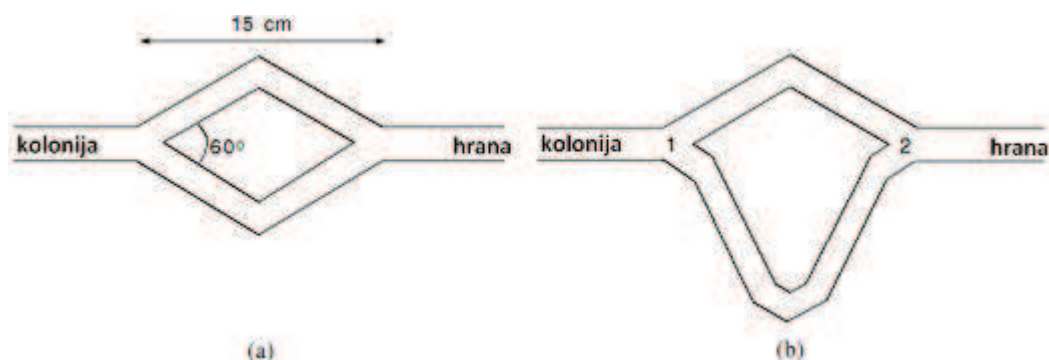
Dve najznačajnije karakteristike stigmergije koje je razlikuju od ostalih oblika komunikacije su sledeće:

- stigmergija je indirektan, ne-simbolički vid komunikacije koji koristi okruženje - insekti razmenjuju informacije menjajući okruženje i
- stigmergijska informacija je lokalno dostupna - dostupna je samo onim insektima koji posete mesto gde je ona deponovana ili njegovu najbližu okolinu.

Primere stigmergije možemo videti kod kolonije mrava. Kod mnogih vrsta mrava, mravi koji idu ka i od izvora hrane deponuju na tlu supstancu zvanu feromon. Drugi mravi osećaju prisustvo feromona i teže da prate putanje na kojima je koncentracija feromona veća. Pomoću ovog mehanizma mravi veoma efikasno transportuju hranu od izvora do gnezda (mravinjaka).

6.1 Eksperiment sa dva mosta

Deneubourg i njegovi saradnici su detaljno ispitali deponovanje i praćenje traga feromona kod mrava. U eksperimentu poznatom kao “eksperiment sa dva mosta”, gnezdo kolonije argentinskih mrava povezano je sa izvorom hrane preko dva mosta jednake dužine. Duž putanje između izvora hrane i gnezda, argentinski mravi deponuju feromon. Inicijalno, tj. na početku eksperimenta, svaki mrav nasumično bira jedan od dva mosta (slika 7a). Međutim, kao posledica slučajnog protoka mrava, posle nekog vremena na jednom od dva mosta pojaviće se veća koncentracija feromona koja će kao posledicu privući više mrava. Ovo dodatno povećava koncentraciju feromona na tom mostu i čini ga privlačnijim za mrave, a kao posledica toga nakon nekog vremena čitava kolonija će težiti da koristi isti most. Deneubourgh i njegovi saradnici su ponovili eksperiment više puta i zapazili da će mravi odabrati jedan od dva mosta sa verovatnoćom od 50%. Ovakvo ponašanje na nivou kolonije, zasnovano na eksploataciji povratne informacije, mravi koriste kako bi pronašli najkraći put između izvora hrane i njihovog gnezda[1].



Slika 7: Eksperiment sa dva mosta (a) oba mosta su iste dužine (b) različitih su dužina

Goss i njegove kolege su razmatrali varijantu eksperimenta sa dva mosta u kojoj je jedan most značajno duži od drugog (slika 7b). U ovom slučaju, stohastički protok kod inicijalnog odabira mosta je značajno redukovan a drugi mehanizam igra značajnu ulogu. Naime, mravi koji slučajnim izborom odaberu kraći most prvi će stići nazad do gnezda. Kraći most će stoga primiti feromon ranije nego duži, a ova činjenica povećava verovatnoću da će mravi koji slede odabrati kraći pre nego duži most. Goss i njegovi saradnici su razvili matematički model posmatranog ponašanja[1].

Pretpostavimo da je u datom trenutku m_1 mrava koristilo prvi, a m_2 drugi most, tada će verovatnoća p_1 da će naredni mrav odabrati prvi most biti:

$$p_1 = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h}$$

gde su vrednosti parametara k i h određeni na osnovu eksperimenta, a očigledno verovatnoća da će naredni mrav odabrati drugi most iznosi $p_2 = 1 - p_1$. Monte Karlo simulacije pokazale su veoma dobre rezultate za $k \approx 20$ i $h \approx 5$.

7. Optimizacijske tehnike

Model koji su predstavili Goss i njegovi saradnici za ponašanje mrava pri sakupljanju hrane je bio inspiracija za razvoj Optimizacije pomoću mravlje kolonije (ACO). U ACO, veliki broj veštačkih mrava izgrađuju rešenje problema optimizacije i razmenjuju informacije o kvalitetu pronađenih rešenja na način preuzet od pravih mrava.

Optimizacija pomoću mravlje kolonije (ACO) je metaheuristika zasnovana na populaciji koja se može koristiti za rešavanje teških optimizacionih problema. Kod ACO algoritma set softverskih agenata, zvanih veštački mravi, traga za povoljnim rešenjem datog problema optimizacije. Da bi ACO mogao da se primeni dati optimizacioni problem se prevodi u problem pronalaženja najpovoljnijeg puta u težinskom grafu. Veštački mravi postepeno izgrađuju rešenje kretanjem po grafu. Izgradnja rešenja je stohastički proces zasnovan na feromon modelu, tj na setu parametara povezanih sa komponentama grafa, bilo čvorovima ili granama, čije vrednosti u toku pretraživanja grafa menjaju veštački mravi.

7.1 Metaheuristike

Prefiks *meta* na grčkom jeziku znači *iznad*, što ukazuje da metaheuristike kao metodi kombinatorne optimizacije rade na višem nivou od heuristika. Mogu biti predstavljene kao algoritmi koji su “unapređene heuristike u cilju što efikasnijeg i efektivnijeg pretraživanja prostora rešenja”. Reč je prvi put upotrebio Glover i od tad je prihvaćena kao naziv ove klase algoritama.

Postoje mnogobrojne vrste metaheuristika, a neke od njih su inspirisane ponašanjem određenih rojeva ili jata jedinki u prirodi. U ovu grupu spadaju *metoda Optimizacije pomoću mravlje kolonije*, *metoda Optimizacije pomoću roja pčela* (Bee Colony Optimization), *Genetski algoritmi* (Genetic Algorithm - GA). Ove metaheuristike polaze od nekog praznog parcijalnog rešenja, a zatim iterativno zamenjuju novom generacijom rešenja prethodnu sve dok kriterijum zaustavljanja ne bude zadovoljen. Razlikuju se dve podklase ove vrste algoritama - sa pamćenjem informacija o prethodnoj generaciji rešenja ili

bez pamćenja. U prvom slučaju, informacija o staroj populaciji se može iskoristiti da bi se generisala nova generacija rešenja.

7.2 Objašnjenje ACO algoritma preko primera

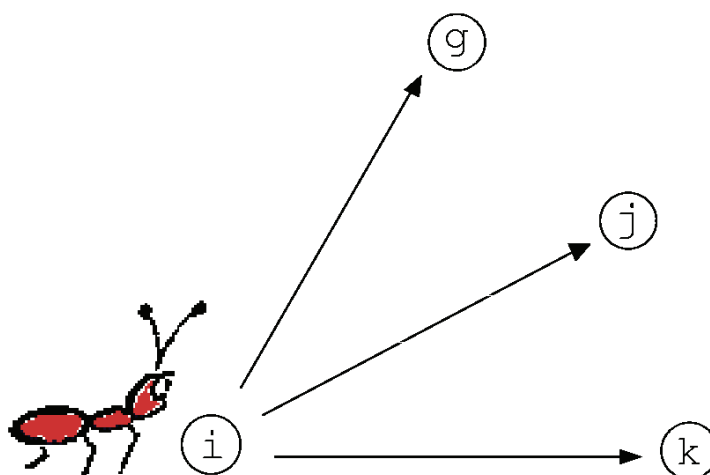
Najjednostavniji način za razumevanje načina rada ACO algoritma je preko primera. Razmatraćemo primenu algoritma na tzv. Problem trgovačkog putnika (Traveling Salesman Problem - TSP). U postavci problema dat je skup lokacija – gradova i rastojanja između njih. Cilj je naći najkraći put koji će svaki grad posetiti jednom i samo jednom. Formalno, treba naći Hamiltonov put minimalne dužine u potpuno povezanom grafu.

U ACO ovaj problem se rešava simuliranjem velikog broja veštačkih mrava koji se kreću po grafu na kojem je problem predstavljen na sledeći način: svaki čvor predstavlja grad, a svaka grana je veza između dva grada. Promenljiva zvana feromon povezana je sa svakom granom grafa i nju mogu da čitaju i menjaju veštački mravi.

Kako bismo primenili ACO na TSP posmatraćemo graf čiji će čvorovi biti povezani sa skupom gradova koje treba posetiti. Ovakav graf naziva se *graf izgradnje (rešenja)*. Kako je kod TSP moguće preći iz svakog grada u bilo koji drugi iz postojećeg skupa gradova, graf izgradnje je potpuno povezan i broj čvorova jednak je broju gradova. Dužina grana između dva čvora proporcionalna je rastojanju između dva odgovarajuća grada, a vrednost feromona i heuristike povezujemo takođe sa granama grafa. Vrednost feromona se menja tokom izvršavanja algoritma i predstavlja kumulativno iskustvo kolonije mrava, dok su vrednosti heuristike zavisne od posmatranog problema, a u slučaju TSP to su vrednosti inverzne dužinama grana.

ACO je iterativni algoritam. U svakoj iteraciji (koraku) posmatra se veći broj veštačkih mrava. Mravi izgrađuju rešenje na sledeći način. Svaki mrav polazi od slučajno izabranog grada – čvora grafa izgradnje. Tokom svakog sledećeg koraka kreće se duž grana grafa. Svaki mrav pamti putanju kojom je prošao, a u svakom narednom koraku bira granu koja vodi do čvora koji još nije posetio. Mrav je formirao rešenje problema kada poseti sve čvorove posmatranog grafa. U svakom koraku mrav bira granu do narednog čvora koji nije posetio na osnovu verovatnosnih pravila koja zavise od vrednosti feromona i heurističke informacije (slika 8). Što je veća vrednost feromona i heuristike za neki čvor, to je veća

verovatnoća da će mrav baš njega izabrati za sledeći. Na kraju koraka (iteracije), kada svi mravi formiraju svoje rešenje, vrednosti feromona vezanih za grane u grafu se menjaju – ažuriraju, kako bi naveli mrave u sledećim iteracijama da konstruišu rešenje slično najboljem prethodno konstruisanom. Svaka vrednost feromona se na kraju svakog koraka inicijalno smanjuje za određeni procenat a kao posledica procesa isparavanja ili evaporacije. Svaka grana zatim dobija dodatnu količinu feromona proporcionalnu kvalitetu rešenja kojima pripada (jedno je rešenje po mravu u svakom koraku – iteraciji algoritma). Procedura se zatim ponavlja sve dok se ne zadovolji kriterijum za zaustavljanje.



Slika 8: Mrav koji se nalazi u gradu i bira sledeći grad koji će posetiti pomoću stohastičkog mehanizma. Ako grad j još uvek nije posećen, može biti odabran sa verovatnoćom koja je proporcionalna sa količinom feromona deponovanog na grani (i,j) .

U eksperimentu sa dva mosta je pokazano da je nivo isparavanja feromona zanemarljiv u poređenju sa vremenom potrebnim da kolonija mrava konvergira ka kraćem putu, tako da se pri modeliranju problema može zanemariti. Kod veštačkih mrava situacija je drugačija. Eksperimentalni rezultati su pokazali da kod jednostavnijih grafova, recimo onih koji modeluju postavku eksperimenta sa dva mosta, evaporacija feromona takođe nije neophodna. Nasuprot tome, simuliranje evaporacije značajno poboljšava performanse algoritma u pronalaženju dobrog rešenja u TSP kod kompleksnijih grafova.

7.3 Najpoznatiji ACO algoritmi

Do danas je predstavljen veći broj različitih varijanti algoritma Optimizacije pomoću mravlje kolonije. Originalni, prvi ACO algoritam poznat kao Ant System (AS) – Sistem mrava, pojavio se početkom devedesetih godina dvadesetog veka, tačnije 1991. i prvi put su ga objavili Dorigo i njegovi saradnici. Od tada se pojavio veliki broj varijanti ACO algoritma (slika 9). Među poznatijim su Ant Colony System (ACS) – Sistem kolonije mrava, koji su objavili Dorigo i Gambardella 1996. i MAX-MIN Ant System (MMAS) Stützle i Hoos 1996. [10] Svi ovi algoritmi dele istu ideju koja se najbolje može pokazati na primeru gde se oni primenjuju. U sledećem poglavlju je dat više formalni opis ACO algoritma.

Naziv algoritma	Autor	Godina pojavljivanja
Ant System (AS)	Dorigo i saradnici	1991.
Elitist AS	Dorigo i saradnici	1992.
Ant-Q	Gambardella & Dorigo	1995.
Ant Colony System	Dorigo & Gambardella	1996.
MAX-MIN AS	Stützle & Hoos	1996.
Rank-based AS	Bullnheimer i saradnici	1997.
ANTS	Maniezzo	1999.
BWAS	Cordón i saradnici	2000.
Hyper-cube AS	Blum i saradnici	2001.

Slika 9: Nepotpuna lista najuspešnijih ACO algoritama po hronološkom redu pojavljivanja

8. ACO Metaheuristika

ACO je formalizovana u metaheuristiku za probleme kombinatorne optimizacije. Prvi su to učinili Dorigo i njegovi saradnici. Reč heuristika potiče od grčke reči heurisko i prevedena na srpski jezik ima značenje otkriti ili naći. Heuristika predstavlja kompromis između potrebe da algoritam bude jednostavan i želje da isti razlikuje dobre i loše odluke. Prefiks meta na grčkom jeziku znači iznad, što ukazuje na to da metaheuristike kao metodi kombinatorne optimizacije rade na višem nivou od heuristika. Mogu biti predstavljene kao algoritmi koji su „unapređene heuristike u cilju što efikasnijeg i efektivnijeg pretraživanja prostora rešenja“. Reč je prvi put upotrebio Glover i od tada je prihvaćena kao naziv ove klase algoritama. Drugačije se može reći da je metaheuristika skup algoritamskih koncepata koji se mogu koristiti za opisivanja heurističkih metoda koje koristimo kod rešavanja širokog spektra različitih problema. Drugim rečima, metaheuristika je opšti algoritam koji se može primeniti na različite optimizacione probleme uz relativno malo modifikacija. Kako bi se ACO mogao primeniti na dati problem kombinatorne optimizacije potrebno ga je prilagoditi adekvatnom modelu problema.

8.1 Model problema kombinatorne optimizacije

Model $P = (S, \Omega, f)$ problema kombinatorne optimizacije sastoji se od:

- prostora pretrage S definisanim nad konačnim skupom diskretnih varijabli, tj. promenljivih - $X_i, i = \overline{1, n}$
- skupa Ω ograničenja nad promenljivima
- i funkcije cilja $f: S \rightarrow R_0^+$ čiji minimum tražimo

Inače, svaki problem traženja maksimuma trivijalno se može prevesti u problem traženja minimuma, naime traženje maksimuma funkcije g ekvivalentno je traženju minimuma funkcije $f = -g$.

Promenljiva X_i uzima vrednosti iz skupa dopustivih vrednosti $D_i = \{v_i^1, v_i^2, \dots, v_i^{|D_i|}\}$. Prihvatljivo rešenje $s \in S$ dobijamo kada sve promenljive dobiju vrednosti koje zadovoljavaju sva ograničenja u Ω . Rešenje $s^* \in S$ je globalni optimum (minimum) akko $f(s^*) \leq f(s), \forall s \in S$.

Ovaj model problema kombinatorne optimizacije se koristi za definisanje feromon modela ACO algoritma. Vrednost (količina) feromona dodeljuje se svakoj mogućoj komponenti rešenja. Formalno, vrednost feromona τ_{ij} povezana je sa komponentom rešenja c_{ij} koja je vezana sa dodelom $X_i = v_i^j$ vrednosti diskretnoj promenljivoj. Skup svih mogućih komponenti rešenja obeležava se sa C .

U ACO algoritmu veštački mravi izgrađuju rešenje pretražujući potpuno povezan težinski graf $G_C(V, E)$, gde je V skup svih čvorova, a E skup svih grana ovoga grafa. Graf može biti povezan sa komponentama rešenja, skupom C , na dva načina: komponente mogu biti predstavljene ili čvorovima ili granama. Veštački mravi se kreću od čvora do čvora duž grana grafa, izgrađujući tako (inkrementalno) parcijalno rešenje. Neke metaheuristike, kao što je ACO, polaze od praznog parcijalnog rešenja. Uz to, mravi ostavljaju (deponuju) izvesnu količinu feromona duž komponenti preko kojih se kreću, bili to čvorovi ili grane grafa. Količina $\Delta\tau$ deponovanog feromona zavisi od kvaliteta rešenja koje su pronašli. Sledeći mrav koji dođe na taj čvor ili granu koristi informaciju o feromonu kako bi se nadalje kretao ka onim elementima prostora pretrage koji „obećavaju“ optimalnije rešenje.

Kod problema trgovačkog putnika (TSP) rešenje se može predstaviti kao skup od n promenljivih, gde je n broj gradova koje treba obići. Svaka od promenljivih povezana je sa tačno jednim gradom. Promenljiva X_i označava grad koji treba posetiti nakon grada i . Ovde su komponente rešenja parovi gradova koje treba posetiti jedan za drugim u datom redosledu. Komponenta rešenja $c_{ij} = (i, j)$ označava da rešenje koje se analizira preporučuje da grad j treba posetiti odmah nakon grada i . U ovom slučaju težinski graf izgradnje je graf u kome su čvorovi gradovi, a grane su komponente rešenja datog TSP problema. Kao posledica toga, mravi deponuju feromon duž grana težinskog grafa.

Treba znati da se u težinskom grafu komponente rešenja mogu predstaviti i čvorovima na kojima se feromon deponuje. Iako ovaj način deluje manje „prirodno“ za TSP i on je korektan.

8.2 Pseudokod

U ACO algoritmu skup veštačkih mrava – agenata, koji odgovara koloniji mrava u prirodnom okruženju, zajednički rešava zadati problem koristeći pri tome graf izgradnje. Informacija koju agenti – mravi sakupe tokom procesa pretrage se čuva u tragu feromona τ_{ij} povezanim sa granom grafa (i,j) . Trag feromona predstavlja dugoročnu memoriju u kojoj se nalaze podaci o čitavoj dotadašnjoj pretrazi. U zavisnosti od problema koji se rešava, trag feromona može biti povezan sa svim čvorovima grafa izgradnje ili samo sa pojedinim. Uz čvorove može biti i najčešće i jeste vezana heuristička vrednost η_{ij} koja predstavlja od ranije poznatu informaciju o problemu, ili informaciju čiji izvor nisu sami mravi.

Navešćemo neke osobine koje važe za koloniju veštačkih mrava, kao i za veštačke mrave kao jedinke.

- Iako je svaki mrav kao jedinka sposoban da zasebno izgradi rešenje za posmatrani problem, kvalitetna rešenja nastaju gotovo isključivo kao posledica interakcije i zajedničkog rada svih mrava.
- Svaki mrav koristi samo privatne informacije i one lokalne vezane za čvor koji posećuju i granu kojom se kreću.
- Mravi komuniciraju isključivo na indirektan način preko informacija koje čitaju, odnosno upisuju u promenljive koje čuvaju vrednosti feromona.
- Sami mravi se ne menjaju ali menjaju način na koji je problem predstavljen i posmatran od strane drugih mrava.
- Mrav k ima memoriju M^k u kojoj se nalazi informacija o putu koji je do tada prešao. Memorija se koristi kako bi se sačuvalo do tada izgrađeno rešenje, kako bi se ono procenilo i kako bi se kretanjem unazad duž pređenog puta ažurirale vrednosti feromona.
- Mravu se može zadati početno stanje – čvor od koga polazi i jedno ili više završnih stanja ili terminalnih uslova.
- Mrav polazi od početnog stanja – čvora, kreće se ka dozvoljenim susednim stanjima i tako iterativno (korak-po-korak) izgrađuju rešenje. Proces izgradnje se završava kada bar jedan mrav zadovolji bar jedan od zadatih terminalnih uslova.
- Krećući se od čvora i ka susednom čvoru j mrav može ažurirati trag feromona τ_{ij} na grani (i,j) . Ovo je tzv. „onlajn“ korak-po-korak ažuriranje feromona.

- Kada izgradi rešenje, koristeći memoriju mrav se može vratiti duž puta kojim je prošao i ažurirati trag feromona na pređenim granama. Ovo je tzv. „onlajn“ odloženo ažuriranje feromona.
- Nakon što izgradi rešenje i nakon što ažurira trag feromona duž grana koje je posetio, mrav umire i tako oslobađa zauzete resurse.

ACO metaheuristika je opisana sledećim pseudokodom:

Algoritam 1: Pseudokod – ACO metaheuristika

PROCEDURE ACO_Metaheuristika()

 Podesi_Parametre();

 Inicijalizuj_Vrednost_Trage_Feromona();

WHILE (Nije zadovoljen kriterijum za zaustavljanje) **DO**

 Raspored_aktivnosti

 Konstruiši_Mrave_I_Mravlja_Rešenja();

 Primeni_Lokalnu_Pretragu(); {opcionalno}

 Podesi_Vrednosti_Feromona();

END Raspored_aktivnosti

END WHILE

END PROCEDURE

Nakon početnih procedura inicijalizacije metaheuristika prolazi kroz tri koraka u okviru bloka Raspored_aktivnosti. U prvom koraku formiraju se mravi koji zatim izgrađuju više rešenja koja se unapređuju kroz proceduru lokalne pretrage koja je opcionalna i konačno se ponovo podešavaju (ažuriraju) vrednosti feromona. Važno je zapaziti da u bloku Raspored_aktivnosti nije eksplicitno određeno kako će se i kojim redosledom ove tri aktivnosti izvršavati i posebno da li će se izvršavati paralelno ili nezavisno, odnosno da li je potrebna nekakva sinhronizacija u njihovom izvršavanju. Ostavljeno je dizajneru algoritma da osmisli kako će ove tri procedure međusobno interagovati. Sledi detaljniji opis navedena tri koraka.

Konstruiši_Mrave_I_Mravlja_Rešenja()

Pseudokod ove procedure bi mogao da izgleda ovako:

Algoritam2: Pseudokod

PROCEDURE *Konstruiši_Mrave_I_Mravlja_Rešenja()*

WHILE (*Ima raspoloživih resursa*) **DO**

Kreiraj_Novog_Mrava();

Aktiviraj_Novog_Mrava();

END WHILE

END PROCEDURE

Algoritam3: Pseudokod

PROCEDURE *Aktiviraj_Novog_Mrava()* {životni ciklus mrava}

Inicijalizuj_Mrava(); {početno stanje}

WHILE (*Trenutno_stanje* $\neq$ *Terminalno_stanje*) **DO**

A=Učitaj_Graf_Izgradnje();

P=Odredi_Moguće_Putanje(A,M, Ω);

Novo_stanje=Primeni_Pravilo_Odlučivanja(P, Ω);

Predi_Na_Novo_Stanje(Novo_stanje);

IF (*Onlajn korak-po-korak ažuriranje feromona*)

Ažuriraj_Trag_Feromona_Na_Posećenim_Granama();

Ažuriraj_Graf_Izgradnje();

ENDIF

M=Upiši_Novo_Stanje_U_Memoriju();

END WHILE

IF (*Onlajn odloženo ažuriranje feromona*)

FOREACH *Posećeni_čvor* **DO**

Ažuriraj_Trag_Feromona_Na_Posećenim_Granama();

Ažuriraj_Graf_Izgradnje();

END FOREACH

ENDIF

Mrav_Umire();

END PROCEDURE

Formira se skup od m veštačkih mrava koji zatim konstruišu rešenja od elemenata konačnog skupa dopustivih komponenata rešenja $C = \{c_{ij}\}, i = \overline{1, n}; j = 1, \dots, |D_i|$. Izgradnja rešenja kreće od praznog parcijalnog rešenja $s^p = \emptyset$. Pri svakom koraku izgradnje

rešenja, parcijalno rešenje s^p se proširuje dodavanjem dozvoljenih komponenti rešenja iz skupa $N(s^p) \subseteq C$ koji je definisan kao skup komponenti koje se mogu dodati parcijalnom rešenju s^p bez narušavanja ograničenja iz skupa Ω . Komponente rešenja se čuvaju u memoriji M mrava. Proces izgradnje rešenja može se posmatrati kao „šetnja“ po težinskom grafu $G_C(V, E)$.

Izbor komponente rešenja iz skupa $N(s^p)$ vođen je stohastičkim (verovatnosnim) mehanizmom koji je zasnovan na količini feromona povezanim sa svakim od elemenata skupa $N(s^p)$. Pravilo za izbor komponente rešenja varira u zavisnosti od varijante ACO algoritma, ali u svakom od njih inspirisan je modelom ponašanja koji daju pravi mravi u jednačini koje je data u ekserimentu sa dva mosta.

U proceduri Aktiviraj_Novog_Mrava, Terminalno_stanje se odnosi na kompletno rešenje koje mrav izgradi (svi čvorovi grafa izgradnje tačno jednom posećeni). Korak-po-korak i odloženo onlajn ažuriranje feromona su međusobno isključivi. Ako u proceduri ne navedemo ni jednu ni drugu, tada se ažuriranje feromona obavlja u proceduri Podesi_Vrednosti_Feromona. Ako u algoritmu ne koristimo korak-po-korak ili odloženo onlajn ažuriranje feromona, tada se vrednosti feromona ažuriraju na sledeći način: nakon svake iteracije algoritma izdvojimo mrava koji je pronašao najbolje rešenje i ažuriramo vrednosti feromona samo na granama puta kojim je on prošao. Ovo je tzv. „oflajn“ ažuriranje.

Primeni_Lokalnu_Pretragu()

Kada je rešenje konstruisano, a pre ažuriranja traga feromona, tj. podešavanja vrednosti feromona, uobičajeno je unaprediti rešenje koje su dali mravi pomoću lokalne pretrage. Ova faza, koja jako zavisi od samog posmatranog problema, je opcionalna iako je najčešće uključena u vrhunske ACO algoritme.

Lokalno pretraživanje je verovatno najstarija i najjednostavnija metaheuristika. Počinje od zadatog inicijalnog rešenja i u svakoj iteraciji menja trenutno rešenje susedstvom koje poboljšava funkciju cilja. Pretraga staje kada svi kandidati jednog susedstva nisu bolji od trenutnog rešenja – lokalni optimum je dostignut.

Za velike prostore pretraživanja kandidati za rešenja mogu biti podskupovi tog prostora. Uzimanje podskupova prostora pretraživanja ubrzava pretragu. Dopunske verzije lokalnog pretraživanja se karakterišu po načinu dobijanja rešenja (determinističke ili stohastičke) i načinu pretrage prostora rešenja.

Postoji više načina na koje se pretraga prostora rešenja može vršiti:

- **Najbolje poboljšanje** - Bira se susedstvo koje najbolje poboljšava funkciju cilja. Pretraga se vrši deterministički, gde se pretražuje celi prostor rešenja dok se ne dodje do traženog susedstva. Za velike prostore pretrage metoda je spora.
- **Prvo poboljšanje** - Uzima se prvo susedstvo koje je bolje od trenutnog rešenja i zamenjuje se starim. Podrazumeva parcijalnu evaluaciju prostora. U najgorem slučaju pretražiće se celi prostor rešenja.
- **Nasumičan izbor** - Slučajno se bira rešenje među svim onim koja su bolja od trenutnog.

Kompromis između kvalitetnog rešenja i kratkog vremena pretraživanja se može postići upotrebom strategije prvog poboljšanja kada je početno rešenje slučajno izabrano, a strategije najboljeg poboljšanja kada to nije slučaj. U mnogim slučajevima u praksi pokazalo se da strategija prvog poboljšanja dovodi do istih rezultata kao i strategija najboljeg poboljšanja, ali za manje vremena.

Sledeći algoritam predstavlja pseudokod lokalnog pretraživanja. Prvo se izabere početno rešenje, a nakon toga iterativno se ispituje susedstvo trenutnog rešenja. Ukoliko neka tačka susedstva poboljšava trenutno rešenje, ono se tom tačkom zamenjuje.

Algoritam: Pseudokod – lokalno pretraživanje

PROCEDURE Lokalno_Petraživanje()

Inicijalizuj_početno_rešenje(); { Izaberi neko rešenje iz prostora rešenja }

REPEAT

FOREACH korak **DO**

Izaberi_okolinu_trenutnog_rešenja();

U_njoj_nađi_sledeće_rešenje_prema_nekom_kriterijumu_izbora();

Za_izabrano_rešenje_proveri_da_li_je_bolje_od_trenutnog();

Ako_jeste_zameni_rešenje_nađenim();

END FOREACH

UNTIL (Kriterijum zaustavljanja zadovoljen)

Prikaži_najbolje_nađeno_rešenje();

END PROCEDURE

Metoda lokalnog pretraživanja je veoma laka za implementaciju i daje sasvim zadovoljavajuće rezultate za kratko vreme. Zato se često primenjuje u praksi na različitim problemima optimizacije. Mana ove metode je konvergencija prema *lokalnom optimumu*.

Detaljnije, algoritam je jako osetljiv prema početnom rešenju – velike razlike u kvalitetu rešenja mogu biti dobijene za neke probleme.

Iako je složenost ove metode u praksi prihvatljiva, ona u najgorem slučaju može biti eksponencijalna. Lokalno pretraživanje radi dobro ako u prostoru pretraživanja nema puno lokalnih optimuma ili ako je sličnost između raznih lokalnih optimuma velika.

Radi izbegavanja glavnog nedostatka ove metode, *konvergenciji ka lokalnom minimumu*, predloženo je korišćenje raznih alternativnih algoritama. Oni postaju popularni oko 1980. godine. Razlikuju se četiri pristupa:

- Promena inicijalnog rešenja
- Prihvatanje susedstva koji ne poboljšavaju rešenje
- Menjanje susedstva
- Menjanje funkcije cilja ili ulaznih vrednosti problema.

Podesi_Vrednosti_Feromona()

Cilj ažuriranja, tj. podešavanja traga feromona je povećanje vrednosti feromona povezanih sa dobrim ili obećavajućim rešenjima i smanjenje vrednosti feromona povezanih sa onim lošim. Obično, ovo se postiže smanjenjem svih vrednosti feromona preko procesa evaporacije (isparavanja) i povećanjem nivoa feromona povezanih sa odabranim skupom dobrih rešenja. Evaporacija feromona je potrebna kako bi se izbegla prebrza konvergencija algoritma ka optimalnom rešenju i time favorizovalo pretraživanje novih oblasti prostora pretrage.

9. Osnovni ACO algoritam

Originalni ACO algoritam je poznat kao Ant System i pojavio se devedesetih godina prošlog veka. Od tada, pojavilo se više varijanti ACO algoritma i svi su opisani u literaturi. Ovde ćemo prikazati originalni Ant System i dve najuspešnije varijante MAX-MIN Ant System i Ant Colony System. Kako bismo prikazali razlike između njih koristićemo Problem trgovačkog putnika kao konkretan primer.

9.1 Ant System (AS)

To je prvi ACO algoritam prikazan u literaturi[5]. Osnovna karakteristika mu je da u svakom koraku vrednosti feromona ažurira svih m mrava koji su u tom koraku izgradili rešenje. Trag feromona τ_{ij} povezan sa granom koja spaja gradove i i j ažurira se na sledeći način:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^k$$

gde je ρ nivo isparavanja (evaporacije), m je broj mrava, a $\Delta\tau_{ij}^k$ količina feromona deponovanog na grani (i,j) od strane mrava k :

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k}, & \text{ako mrav } k \text{ koristi granu } (i,j) \text{ na svom putu} \\ 0, & \text{inače} \end{cases}$$

gde je Q konstanta, a L_k je dužina puta koji je prešao mrav k .

Pri konstruisanju rešenja mrav bira sledeći grad (čvor) koji će posetiti pomoću stohastičkog mehanizma koji je opisan u sledećoj formuli:

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \eta_{ij}^\beta}{\sum_{c_{il} \in N(s^p)} \tau_{il}^\alpha \eta_{il}^\beta}, & \text{ako je } c_{ij} \in N(s^p) \\ 0, & \text{inače} \end{cases}$$

Ona daje verovatnoću događaja da će mrav k koji je u gradu i i do sada je izgradio parcijalno rešenje s^p otići u grad j koji na svom putu još nije posetio. U navedenoj formuli, $N(s^p)$ je skup prihvatljivih komponenti rešenja, tj. grana (i, l) gde je l grad koji mrav još nije posetio. Parametri α i β kontrolišu uzajamnu težinu, tj. „bitnost“ informacije o feromonu nasuprot heurističke informacije η_{ij} koja je data sa:

$$\eta_{ij} = \frac{1}{d_{ij}}$$

gde je d_{ij} rastojanje između gradova i i j .

9.2 MAX-MIN Ant System (MMAS)

Ovaj algoritam predstavlja unapređeni Ant System. Osnovna karakteristika mu je da samo najbolji mravi ažuriraju trag feromona i da je vrednost feromona ograničena. Feromon se ažurira po sledećoj formuli:

$$\tau_{ij} \leftarrow [(1 - \rho)\tau_{ij} + \Delta\tau_{ij}^{best}]_{\tau_{min}}^{\tau_{max}}$$

gde su τ_{min} i τ_{max} respektivno donja i gornja granica količine feromona, a operator $[x]_b^a$ je definisan sa:

$$[x]_b^a = \begin{cases} a, & \text{ako } x > a \\ b, & \text{ako } x < b \\ x, & \text{inače} \end{cases}$$

a $\Delta\tau_{ij}^{best}$ je:

$$\Delta\tau_{ij}^{best} = \begin{cases} \frac{1}{L_{best}}, & \text{ako } (i, j) \text{ pripada najboljem putu} \\ 0, & \text{inače} \end{cases}$$

a L_{best} je dužina puta najboljeg mrava. Ovo može biti ili najbolji put u trenutnoj iteraciji algoritma (iteration-best, L_{ib}) ili najbolje rešenje nađeno od početka algoritma (best-so-far, L_{bs}) ili kombinacija ova dva, a izbor se ostavlja dizajneru algoritma.

Što se tiče donje i gornje granice vrednosti feromona τ_{min} i τ_{max} , određuju se empirijski i zavise od posmatranog problema, a u literaturi su data i mnoga uputstva za njihovo definisanje koja su zasnovana na analiziranju ponašanja algoritma u zavisnosti od njihove promene.

9.3 Ant Colony System (ACS)

Najznačajniji doprinos ACS algoritma je uvođenje lokalnog ažuriranja traga feromona umesto ažuriranja traga feromona na kraju procesa konstrukcije rešenja, zvanog oflajn ažuriranje feromona.

Lokalno ažuriranje traga feromona vrše svi mravi nakon svakog koraka pri konstrukciji rešenja. Svaki mrav primenjuje ovaj proces samo na poslednje posećenoj grani:

$$\tau_{ij} \leftarrow (1 - \varphi)\tau_{ij} + \varphi\tau_0$$

gde je $\varphi \in (0,1]$ koeficijent traga feromona, a τ_0 inicijalna vrednost feromona.

Glavni cilj lokalnog ažuriranja traga feromona je da se postigne veća raznovrsnost u potrazi kod mrava koji slede pri svakoj narednoj iteraciji algoritma. Naime, smanjujući koncentraciju feromona na grani kojom prolazi mrav ohrabruje svoje sledbenike da biraju druge grane i time da proizvedu različita rešenja. Treba izbeći brzu konvergenciju algoritma ka optimalnom rešenju. Ovim se smanjuje mogućnost da više mrava proizvede isto rešenje pri jednoj iteraciji algoritma.

Oflajn ažuriranje traga feromona, slično kao kod MMAS, izvodi na kraju svake iteracije samo jedan mrav i to može biti ili najbolji u koraku (iteration-best) ili najbolji do sada (best-so-far). Ipak, formula po kojoj se vrši ažuriranje je malo drugačija i glasi:

$$\tau_{ij} \leftarrow \begin{cases} (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij}, & \text{ako } (i,j) \text{ pripada najboljem putu} \\ \tau_{ij}, & \text{inače} \end{cases}$$

gde kao i u MMAS $\Delta\tau_{ij} = \frac{1}{L_{best}}$, gde L_{best} može biti ili L_{ib} ili L_{bs} .

Još jedna značajna razlika između ACS i AS algoritma je pravilo odlučivanja koje mravi koriste prilikom izgradnje rešenja. U ACS koristi se takozvano pseudoslučajno proporcionalno pravilo koje kaže: verovatnoća da mrav iz grada i pređe u grad j zavisi od slučajno izabrane promenljive q uniformno raspoređene na intervalu $[0,1]$ i od parametra q_0 , tako da ako je $q \leq q_0$ tada

$$j = \operatorname{argmax}_{c_{il} \in N(s^p)} \{\tau_{il} \eta_{il}^\beta\}$$

a u suprotnom se koristi isti mehanizam kao i kod AS.

10. Teoretski rezultati

Prvo teoretsko pitanje vezano za ACO metaheuristiku ticalo se konvergencije algoritma. Da li će posmatrani ACO algoritam u svakom slučaju naći optimalno rešenje? Prvi dokaz konvergencije dao je Gutjahr za varijantu ACO algoritma zvanu Graph-based Ant System (GBAS), ali ovaj rezultat se nije mogao primeniti na sve algoritme. Kasnije su se pojavili dokazi konvergencije i za ACO algoritme sa najboljim performansama, kao ACS i MMAS. Ipak, ni jedan od ovih dokaza ne predviđa brzinu konvergiranja ka optimalnom rešenju.

Dok nam dokazi konvergencije daju uvid u neke matematički bitne karakteristike algoritma, ne pružaju nam uputstva kako da efikasno implementiramo ACO algoritam. Bitnija za praktičnu primenu su istraživanja u oblasti boljeg razumevanja ponašanja samog algoritma. Ovakvih istraživanja ima dosta i dostupna su dizajnerima i istraživačima.

10.1 ACO i njegovi parametri

U svakom koraku t ACO algoritma mrav koji se nalazi na čvoru i mora da odabere naredni čvor j na koji će preći i to između onih čvorova koje na svom putu još nije posetio. Verovatnoća da neki čvor bude odabran određena je rastojanjem između čvorova i i j i količinom feromona na grani koja ih spaja. Promenljiva τ_{ij} je količina feromona, ili drugačije **trag** feromona na grani između čvorova i i j , a η_{ij} tzv. **vidljivost** čvora j iz čvora i i definisana je sa:

$$\eta_{ij} = \frac{1}{d_{ij}}$$

gde je d_{ij} rastojanje između gradova i i j .

Što je veći proizvod $\tau_{ij}\eta_{ij}$, veća je verovatnoća da će grad j biti odabran za naredni. U narednoj formuli koja predstavlja verovatnoću da će mrav k u narednoj iteraciji algoritma t odabrati da iz čvora i pređe u čvor j , trag feromona i vidljivost su podignuti na stepene α i

β respektivno. Parametri α i β u njoj kontrolišu uzajamnu bitnost informacije o tragu feromona τ_{ij} nasuprot heurističke informacije η_{ij} .

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \eta_{ij}^\beta}{\sum_{c_{il} \in N(s^p)} \tau_{il}^\alpha \eta_{il}^\beta}, & \text{ako je } c_{ij} \in N(s^p) \\ 0, & \text{inače} \end{cases}$$

Bolje performanse algoritma se dobijaju uvođenjem dodatnog parametra q_0 za koji važi $0 \leq q_0 \leq 1$ (kod ACS algoritma – Ant Colony System) kako bi se kontrolisao nivo slobode za odabir narednog čvora koji dajemo mravima. Ako je slučajno odabrana promenljiva q , $q \in [0,1]$, manja od q_0 tada za odabir narednog grada biramo gore navedeno verovatnosno pravilo, a ako je veća ili jednaka tada naredni čvor biramo deterministički i to onaj čvor j čiji je proizvod traga i vidljivosti iz čvora i najveći.

Nakon što svi mravi završe svoje kretanje kroz graf nivo traga feromona mora da se ažurira. Faktor isparavanja ρ obezbeđuje da se feromon ne akumulira beskonačno i određuje količinu „starog“ feromona koji će biti prenet u sledeću iteraciju algoritma.

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^k$$

U ovoj jednačini za ažuriranje traga feromona, m je broj mrava, a $\Delta\tau_{ij}^k$ je količina feromona koju je deponovao mrav k na grani (i,j) grafa. Kod AS algoritma $\Delta\tau_{ij}^k$ zavisi od neke konstante Q koju delimo sa dužinom puta koji je prešao mrav k .

10.2 Optimalna vrednost parametara za TSP

Dakle parametri koji se pojavljuju u ACO algoritmu (varijante AS i ACS) su α , β , p , q_0 , m i Q . Uporedna težina parametara τ_{ij} i η_{ij} u pravilu za odabir narednog čvora može se podesiti fiksiranjem vrednosti parametra α , a samo se vrednosti parametra β varira. Q je konstanta, a broj mrava m može se podesiti da bude jednak broju gradova u datom TSP problemu, pa za optimizaciju ostaju tri parametra β , p , q_0 .

Početni testovi su potvrdili tvrdnje M. Doriga da različite kombinacije vrednosti parametara znatno utiču na performanse algoritma. Optimalne vrednosti parametara variraju u zavisnosti od posmatranog problema i korišćenog algoritma, a dobijaju se testiranjem, tj. variranjem vrednosti posmatranih parametara i uporednim posmatranjem ponašanja algoritma. Standardne vrednosti parametara koje je dao Dorigo pri predstavljanju algoritma AS su $\beta=5$ i $p=0.5$, a pri predstavljanju algoritma ACS $\beta=2$, $p=0.1$ i $q_0=0.9$.

11. Primena ACO algoritma na rešavanje Problema trgovačkog putnika – demonstracija

U ovom poglavlju primenićemo Mravlji algoritam, tačnije ACO algoritam, na rešavanje kompleksnog optimizacionog problema. U suštini Optimizacija pomoću mravlje kolonije je algoritam koji koristimo za nalaženje optimalnih putanja u potpuno povezanom grafu. Ovaj algoritam oponaša ponašanje kolonije mrava u potrazi za hranom. U međusobnoj komunikaciji mravi koriste supstancu feromon, pa stoga količina feromona na svakoj od grana grafa određuje „privlačnost“ te grane za mravlje jedinice. Na taj način i određujemo optimalnu putanju kroz graf jer će mrav sa većom verovatnoćom odabrati granu koja sadrži veću količinu feromona. Koristeći ovaj koncept kreiramo agente koje nazivamo „veštački mravi“. Svaki mrav se postavlja na slučajno odabrani čvor grafa i dozvoljeno mu je da se kreće po granama grafa tako da treba da obiđe svaki od čvorova tačno jednom. Nakon što mrav obiđe sve čvorove grafa preračunavamo količinu feromona na svakoj grani putanje, a zatim joj dodeljujemo stohastičku vrednost koja zavisi od količine feromona i od vidljivosti čvora do koga grana vodi. Koristeći veliki broj mrava koji na ovaj način prolaze grafom pronalazimo optimalnu putanju. Na taj način imitiramo ponašanje mrava u prirodi i njihov kooperativni način rešavanja problema.

U nastavku ćemo prikazati način rešavanja jednog od najpoznatijih optimizacionih problema, Problema trgovačkog putnika, korišćenjem ACO algoritma. TSP problem se modelira na sledeći način:

Trgovački putnik treba da obiđe N gradova, u našem slučaju slučajno odabranih, tako da svaki od njih poseti tačno jedanput a da kretanje završi u gradu iz koga je na put pošao. Treba naći najkraći put da se ovo učini, a pri tome su nam data rastojanja između svaka dva grada.

Problem je prvi put formulisan 1930. i jedan je od najviše proučavanih optimizacionih problema. Inače, ovo je NP težak problem, a ACO algoritam je jedan od načina za njegovo rešavanje.

Poćemo sa kodom programa koji je napisan u programskom jeziku C++ tako što ćemo najpre definisati parametre potrebne za rad ACO algoritma.

```
#define MAX_GRADOVA 30
#define MAX_RASTOJANJE 100
#define MAX_DUŽINA (MAX_GRADOVA * MAX_RASTOJANJE)
#define MAX_MRAVA 30
```

MAX_GRADOVA se odnosi na maksimalan broj gradova koji možemo koristiti, a MAX_RASTOJANJE na maksimalno rastojanje između bilo koja dva grada. Stoga, najgori mogući scenario koji je definisan sa MAX_DUŽINA daje dužinu puta koju mrav mora da pređe jednaku MAX_GRADOVA * MAX_RASTOJANJE. Ovde definišemo i broj mrava koji je jednak maksimalnom broju gradova. Eksperimentalno je dokazano da kada je broj mrava jednak maksimalnom broju gradova algoritam brže konvergira ka optimalnom rešenju nego kada koristimo veći broj mrava.

Zatim kreiramo okruženje za TSP i to na sledeći način:

```
//Inicijalizacija problema
struct cityType{
    int x,y;
};

struct antType{
    int curCity, nextCity, pathIndex;
    int tabu[MAX_GRADOVA];
    int path[MAX_GRADOVA];
    double tourLength;
};
```

Ovde smo definisali strukturu cityType koja sadrži elemente x i y, koji predstavljaju koordinate grada, tj. njegovu lokaciju na mapi. Zatim je definisana struktura antType koja sadrži elemente potrebne za funkcionisanje mrava. Intedžeri curCity i nextCity predstavljaju grad u kojem se trgovac trenutno nalazi i sledeći koji će biti posećen na grafu. Niz tabu odnosi se na niz gradova koje je mrav već prošao, dok se u nizu path čuva putanja kojom je mrav prošao kroz celokupni graf. Promenljiva tourLength čuva dužinu putanje kojom je mrav prošao.

Pošto smo definisali osnovu za predstavljanje problema, kao i mrave koji učestvuju u simulaciji, prelazimo na definisanje parametara potrebnih za konvergenciju algoritma ka optimalnom rešenju.

```
#define ALPHA 1.0
#define BETA 5.0 //Ovaj parametar određuje važnost rastojanja nasuprot
vrednosti feromona
#define RHO 0.5 //Nivo evaporacije
#define QVAL 100
#define MAX_DUŽINE 20
#define MAX_VREME (MAX_DUŽINE * MAX_GRADOVA)
#define INIT_PHER (1.0/MAX_GRADOVA)
```

Parametri ALPHA i BETA su eksponenti verovatnosne funkcije koja definiše deponovanje feromona na grane grafa. ALPHA predstavlja “važnost” putanje, dok BETA definiše relativnu “bitnost” vidljivosti. Vidljivost je ovde modelovana inverznom vrednošću rastojanja između čvorova. RHO je faktor koji definiše nivo evaporacije, tj. isparavanja feromona sa grana grafa. QVAL je konstanta koja se koristi u formuli:

$$\Delta\tau_{ij}^k(t) = \frac{Q}{L^k(t)}$$

Ograničili smo i maksimalan broj tura koje mrav može preći na 20. Deklarisali smo i konstantu MAX_VREME koju koristimo u slučaju da algoritam iz nekog razloga ne konvergira ka optimalnom rešenju. Na kraju svakom čvoru inicijalno dodeljujemo vrednost feromona (1/MAX_GRADOVA).

```
//Strukture i globalne promenljive

cityType cities[MAX_GRADOVA];
antType ants[MAX_MRAVA];
double dist[MAX_GRADOVA][MAX_GRADOVA];
double phero[MAX_GRADOVA][MAX_GRADOVA];
double best=(double)MAX_DUŽINA;
int bestIndex;
```

Sada smo deklarirali niz struktura `cityType` koji će sadržati čvorove našeg grafa. Deklarirali smo i niz mrava iste veličine kao i prethodni. Dvodimenzionalna matrica `dist` sadrži rastojanja između gradova u parovima, dok matrica `phero` sadrži količinu feromona na svakoj od grana grafa (između gradova). Promenljiva `best` se koristi za beleženje najbolje ture (najkraćeg puta) nakon svake iteracije algoritma.

```
void init()
{
    int from,to,ant;
    //kreiranje gradova
    for(from = 0; from < MAX_GRADOVA; from++)
    {
        //slučajno pozicioniranje gradova
        cities[from].x = rand()%MAX_RASTOJANJE;
        cities[from].y = rand()%MAX_RASTOJANJE;
        //printf("\n %d %d",cities[from].x, cities[from].y);
        for(to=0;to<MAX_GRADOVA;to++)
        {
            dist[from][to] = 0.0;
            phero[from][to] = INIT_PHER;
        }
    }

    //preračunavanje rastojanja
    for(from = 0; from < MAX_GRADOVA; from++)
    {
        for( to =0; to < MAX_GRADOVA; to++)
        {
            if(to!=from && dist[from][to]==0.0)
            {
                int xd = pow( abs(cities[from].x - cities[to].x), 2);
                int yd = pow( abs(cities[from].y - cities[to].y), 2);

                dist[from][to] = sqrt(xd + yd);
                dist[to][from] = dist[from][to];
            }
        }
    }
}
```

```
//inicijalizovanje veštačkih mrava
to = 0;
for( ant = 0; ant < MAX_MRAVA; ant++)
{
    if(to == MAX_GRADOVA)
        to=0;
    ants[ant].curCity = to++;
    for(from = 0; from < MAX_GRADOVA; from++)
    {
        ants[ant].tabu[from] = 0;
        ants[ant].path[from] = -1;
    }
    ants[ant].pathIndex = 1;
    ants[ant].path[0] = ants[ant].curCity;
    ants[ant].nextCity = -1;
    ants[ant].tourLength = 0;

    //učitavanje prvog grada u tabu niz
    ants[ant].tabu[ants[ant].curCity] =1;
}
}
```

Prethodnim delom koda inicijalizujemo kompletan graf kao i mrave koje ćemo koristiti za pretragu grafa. Slučajnim odabirom odredili smo koordinate gradova a zatim izračunali rastojanja među njima u parovima. Inicijalizovali smo i mrave i to tako što smo po jednog “spustili” na svaki od čvorova grafa, kao i ostala svojstva svakog od mrava, zatim tabo niz gradova i putanju. Sledeći deo koda restartuje mrave nakon svakog prolaska kroz graf tako što ih vraća na početnu poziciju na grafu.

```
//reinicijalizovanje svih mrava i njihova raspodela
void restartAnts()
{
    int ant,i,to=0;
    for(ant = 0; ant<MAX_MRAVA; ant++)
    {
        if(ants[ant].tourLength < best)
        {
            best = ants[ant].tourLength;
            bestIndex = ant;
        }
    }
}
```

```

    }
    ants[ant].nextCity = -1;
    ants[ant].tourLength = 0.0;
    for(i=0;i<MAX_GRADOVA;i++)
    {
        ants[ant].tabu[i] = 0;
        ants[ant].path[i] = -1;
    }
    if(to == MAX_GRADOVA)
        to=0;
    ants[ant].curCity = to++;
    ants[ant].pathIndex = 1;
    ants[ant].path[0] = ants[ant].curCity;
    ants[ant].tabu[ants[ant].curCity] = 1;
}
}

```

Najbolju putanju i vrednost najbolje putanje smestili smo u globalne promenljive koje smo ranije definisali.

```

double antProduct(int from, int to)
{
    return(( pow( phero[from][to], ALPHA) * pow( (1.0/ dist[from][to]), BETA)));
}

int selectNextCity( int ant )
{
    int from, to;
    double denom = 0.0;
    from=ants[ant].curCity;
    for(to=0;to<MAX_GRADOVA;to++) {
        if(ants[ant].tabu[to] == 0)
        {
            denom += antProduct( from, to );
        }
    }
    assert(denom != 0.0);
    do {
        double p;
        to++;
    }
}

```

```

    if(to >= MAX_CITIES) to=0;
    if(ants[ant].tabu[to] == 0)
    {
        p = antProduct(from,to)/denom;
        //printf("\n%lf %lf", (double)rand()/RAND_MAX,p);
        double x = ((double)rand()/RAND_MAX);
        if(x < p)
        {
            //printf("%lf %lf Yo!",p,x);
            break;
        }
    }
}while(1);
return to;
}

```

Prethodni skup funkcija se koristi kod odabira sledeće grane kojom će mrav proći. Svaki mrav bira narednu granu na osnovu verovatnosnih pravila koje smo opisali u prethodnim poglavljima ovoga rada. Ovde je korišćena jednostavna funkcija recipročne vrednosti kao mera vidljivosti, kao i identička funkcija koja predstavlja vrednost feromona. Postoji veliki broj radova u kojima se ove funkcije pokušavaju menjati nekim drugim, odgovarajućim heurističkim funkcijama. Za jednostavan TSP koji ovde rešavamo, funkcije koje smo naveli su dovoljno dobre.

Sledeći deo koda simulira koloniju mrava, a naredna funkcija je primer kako se simuliranje ponašanja kolonije mrava može koristiti za rešavanje TSP.

```

int simulateAnts()
{
    int k;
    int moving = 0;
    for(k=0; k<MAX_MRAVA; k++)
    {
        //proveravamo ima li još gradova koje nismo posetili
        if( ants[k].pathIndex < MAX_GRADOVA )
        {
            ants[k].nextCity = selectNextCity(k);
            ants[k].tabu[ants[k].nextCity] = 1;
            ants[k].path[ants[k].pathIndex++] = ants[k].nextCity;
            ants[k].tourLength += dist[ants[k].curCity][ants[k].nextCity];
        }
    }
}

```

```
        //handle last case->last city to first
        if(ants[k].pathIndex == MAX_GRADOVA)
        {
            ants[k].tourLength += dist[ants[k].path[MAX_GRADOVA -1]][ants[k].path[0]];
        }
        ants[k].curCity = ants[k].nextCity;
        moving++;
    }
}

return moving;
}
```

```
//ažuriranje tragova
void updateTrails()
{
    int from,to,i,ant;
    //evaporacija feromona
    for(from=0; from<MAX_GRADOVA;from++)
    {
        for(to=0;to<MAX_GRADOVA;to++)
        {
            if(from!=to)
            {
                phero[from][to] *= ( 1.0 - RHO);
                if(phero[from][to]<0.0)
                {
                    phero[from][to] = INIT_PHER;
                }
            }
        }
    }

    //dodaj novi feromon tragovima
    for(ant=0;ant<MAX_MRAVA;ant++)
    {
        for(i=0;i<MAX_GRADOVA;i++)
        {
            if( i < MAX_GRADOVA-1 )
            {
```

```

        from = ants[ant].path[i];
        to = ants[ant].path[i+1];
    }
    else
    {
        from = ants[ant].path[i];
        to = ants[ant].path[0];
    }
    phero[from][to] += (QVAL/ ants[ant].tourLength);
    phero[to][from] = phero[from][to];
}
}

for (from=0; from < MAX_GRADOVA; from++)
{
    for( to=0; to<MAX_GRADOVA; to++)
    {
        phero[from][to] *= RHO;
    }
}
}

```

Sledeći deo koda je pomoćna funkcija koja sve potrebne podatke prevodi u text fajl kako bi ga python script mogao predstaviti u boljem i razumljivijem obliku.

```

void emitDataFile(int bestIndex)
{
    ofstream f1;
    f1.open("Data.txt");
    antType antBest;
    antBest = ants[bestIndex];
    //f1<<antBest.curCity<<" "<<antBest.tourLength<<"\n";
    int i;
    for(i=0; i<MAX_GRADOVA; i++)
    {
        f1<<antBest.path[i]<<" ";
    }
    f1.close();
    f1.open("city_data.txt");
    for(i=0; i<MAX_GRADOVA; i++)

```

```

    {
        f1<<cities[i].x<<" "<<cities[i].y<<"\n";
    }
    f1.close();
}

```

Sada ćemo opisati main() metodu

```

int main()
{
    int curTime = 0;
    cout<<"MaxTime="<<MAX_VREME;
    srand(time(NULL));
    init();
    while( curTime++ < MAX_VREME)
    {
        if( simulateAnts() == 0)
        {
            updateTrails();
            if(curTime != MAX_VREME)
                restartAnts();
            cout<<"\nTime is "<<curTime<<" ("<<best<<") ";
        }
    }
    cout<<"\nBest tour = "<<best<<endl;
    emitDataFile(bestIndex);
    return 0;
}

```

Sada jednostavno pokrećemo simulaciju, zatim je restartujemo sve dok broj iteracija ne dostigne MAX_VREME. Ovim je obezbeđeno da se simulacija prekida bez obzira da li mravi-agenti konvergiraju ka optimalnom rešenju ili ne nakon određenog broja iteracija.

Ovo je osnovni kod ACO algoritma za TSP u programskom jeziku C++. Sledi jednostavni python script kojim iscrtavamo podatke koristeći matplotlib i tako dobijamo lepe i lako čitljive grafove.

```
from numpy import *
import matplotlib.pyplot as plt

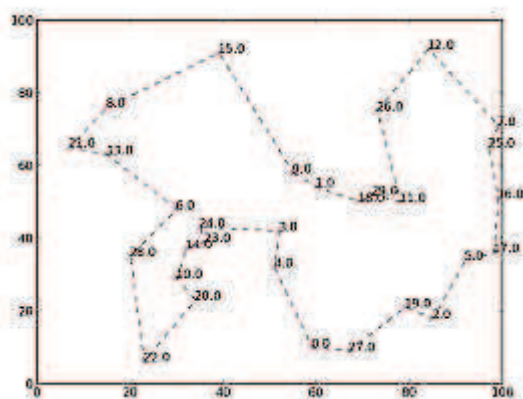
def city_read():
    x,y=loadtxt('city_data.txt', unpack = True)
    order=loadtxt('Data.txt')
    print order
    print x
    print y
    '''order1[]
    x1[]
    y1[]'''
    x1=[]
    y1=[]
    for i in order:
        x1.append(x[i])
        y1.append(y[i])
    x1.append(x[order[0]])
    y1.append(y[order[0]])
    plt.plot(x1,y1, marker='x', linestyle = '--', color = 'r')
    for i in range(len(order)):
        plt.text(x1[i],y1[i],order[i])
    plt.show()

def main():
    city_read()

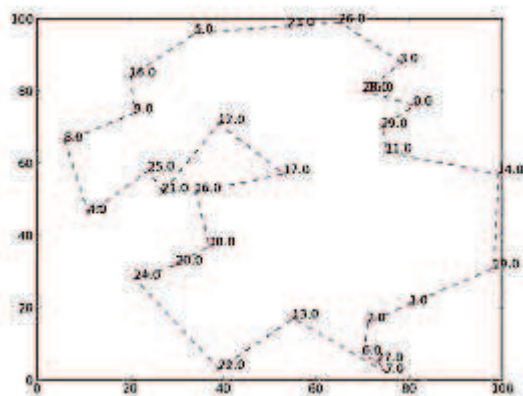
if __name__=="__main__":
    main()
```

Skript iscertava grafove koji se jednostavno čitaju i na kojima su označeni gradovi koordinatama, kao i optimalno rešenje koje daje algoritam.

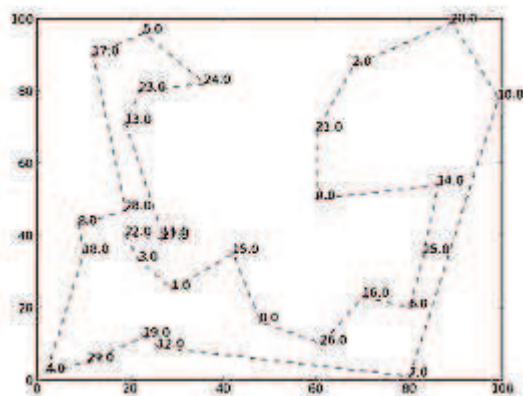
Sledi par konkretnih primera dobijenih grafova.



Rešen TSP. Veštački mravi su iscrtali putanju



Izlaz za slučajno generisanu mapu gradova



12. Zaključak

U ovom radu ukratko je opisan pojam Inteligencije jata kao multidisciplinarne naučne oblasti koja se bavi prirodnim i veštačkim sistemima sastavljenim od velikog broja jedinki. Opisan je spontani proces samoorganizovanja koji je neraskidivo povezan sa SI sistemima, kao i stigmergijski način komunikacije između jedinki u jatu. Zatim je data okvirna podela na prirodne ili veštačke i naučne ili inženjerske SI sisteme i dat je primer za svaku od četiri moguće kombinacije. Opisana su i osnovna svojstva a zatim navedene i neke od studija i primena vezanih za ovu oblast. Ukratko su opisane i najuspešnije primene Inteligencije jata.

Jedan od ciljeva ovoga rada bio je da sažeto iznese „opštu priču“ o Inteligenciji jata i da pokaže da je Inteligencija jata i generalno stigmetrijski model komunikacije interesantno polje za dalje izučavanje.

Poslednjih godina se pojavio veliki broj uspešnih primena ACO algoritma na širok spektar različitih problema optimizacije. Većina ovih aplikacija su primena ACO na *NP*-teške probleme, tj. na probleme kod kojih najbolji do sada poznati algoritam koji sigurno konvergira ka optimalnom rešenju to radi u najgorem slučaju u eksponencijalnom vremenu. Primena ovakvih algoritama je najčešće neizvodljiva u praksi, dok se ACO može koristiti za brzo pronalaženje kvalitetnog rešenja.

Druga oblast primene je na dinamičke probleme nalaženja najkraćeg puta koji proizilaze iz praktičnih problema kod telekomunikacionih mreža. Veliki broj uspešnih primena na akademske probleme motivisao je ljude da prihvate ACO i za rešavanje industrijskih problema. To je još jedna oblast primene kojom se pokazuje da su računarske tehnike primenljive i jako korisne u rešavanju svakodnevnih problema.

Literatura

- [1] Dorigo, M., Birattari, M., Stützle, T., „*Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique*“, IRIDIA – Technical Report Series, September 2006.
- [2] Bonabeau, E., Dorigo, M., Theraulaz, G., „*Swarm Intelligence: From Natural to Artificial Systems*“, NY: Oxford Univ. Press, 1999.
- [3] Dorigo, M., Di Caro, G., „*The Ant Colony Optimization Meta-Heuristic*“, IRIDIA – Technical Report Series, 2005.
- [4] M. Milosavljević, „*Veštačka inteligencija i neuronske mreže*“, Elektrotehnički fakultet, Beograd, 2003.
- [5] Dorigo, M., Stützle, T., „*Ant Colony Optimization*“, A Bradford Book, The MIT Press, Cambridge, Massachusetts, 2004.
- [6] Bonabeau, E. „*Swarm Intelligence: a presentation*“, 2005 IEEE Swarm Intelligence Symposium, June 8-10 2005.
- [7] Gaertner, D., Clark, K., „*On Optimal Parameters for Ant Colony Optimization algorithms*“, Dept of Computing, Imperial College London, 180 Queens Gate, London, UK
- [8] Liu, Y., Passino, K., **Swarm Intelligence: Literature Overview**, Dept. of Electrical Engineering The Ohio State University, 2015 Neil Ave. Columbus, OH 43210, March 30, 2000
- [9] Millonas, M. „Swarms, phase transitions and collective intelligence“ in **Artificial Life III**, Addison-Wesley, 1994.
- [10] Di Caro, G., „*An Introduction to Swarm Intelligence Issues*“, 2005 IEEE Swarm Intelligence Symposium, June 8-10 2005.
- [11] www.aco-metaheuristic.org, The official Web site of the ant colony metaheuristic.
- [12] Blum, C., „*Ant Colony Optimization: Introduction and recent trends*“, ALBCOM, LSI, Barcelona, Spain, October 2005.
- [13] Maniezzo, V., Carbonaro, A., „*Ant Colony Optimization: an Overview*“, University of Bologna, Italy, June 1999.
- [14] Dorigo, M., Blum, C., „*Ant Colony Optimization Theory: a Survey*“, IRIDIA, Brussels, Belgium, May 2005.