SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIO TEHNIČKE DOKUMENTACIJE U OKVIRU PROJEKTA „EVOLUCIJSKI ALGORITMI“

Optimizacija kolonijom mrava

Dino Klemen

Voditelj: *Marin Golub*

Zagreb, prosinac, 2008.

Sadržaj

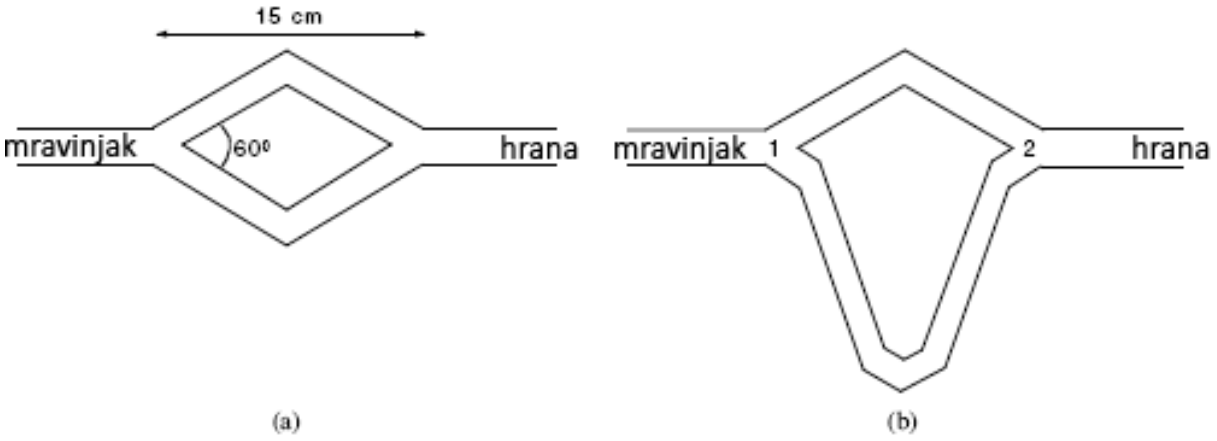
1. Uvod

Algoritam optimizacije kolonijom mrava proučava umjetne sustave inspirirane ponašanjem pravih mravljih kolonija, koji se koriste za rješavanje optimizacijskih problema, najčešće pronalaženja najkraćeg puta u zadanom grafu.

Pojam ACO (*ant colony optimization*) uvodi Marco Dorigo u svome doktoratu iz 1992., kao rezultat istraživanja pristupa kombinatoričkoj optimizaciji. U početku se algoritam primjenjivao na problem putujućeg trgovca i na problem kvadratičnog pridruživanja. Od 1995. Dorigo i ostali rade na raznim proširenim verzijama početne ideje.

Tom je događaju prethodio značajni eksperiment Deneubourga iz 1990. gdje se koristio dvostruki most koji je povezivao mravinjak i izvor hrane. Eksperiment je uključivao variranje omjera duljina grana mosta $r = l_1 / l_2$. Početno je postavljeno $r = 1$. Obzirom da na početku ne postoje nikakvi tragovi feromona, mravi nasumično biraju jednu od grana mosta. Čim je veći broj mrava, može se pretpostaviti da neće jednak broj mrava krenuti na obje grane, nego će slučajno jedna grana imati nešto više mrava. Nešto više mrava povlači nešto više feromona, dok nešto više feromona povlači nešto više novih mrava koji se privlače. Pojam "nešto više" se tako u nekoliko iteracija pretvori u "puno više" sve dok se nakon nekog vremena ne odluče svi mravi za jednu granu, i to upravo za onu na koju je slučajno prešlo nešto više mrava na samom početku eksperimenta.

Dalje se omjer postavio na $r = 2$, s time da mravi ne mogu unaprijed znati koja je grana kraća. Situacija počinje identično prethodnom slučaju – mravi se nasumično odlučuju za neku od grana. No oni koji su krenuli kraćom granom, ti će prvi stići do hrane i početi ostavljati trag feromona i time pozivati nove mrave. U ovom će se slučaju ponovno svi mravi odlučiti nakon nekog vremena za jednu granu, i to onu kraću, ali ovog puta ne zbog nasumičnosti, već zbog bržeg obnavljanja traga feromona na kraćoj grani. [1] Tim se eksperimentom intuitivno može objasniti glavna ideja algoritma optimizacije kolonijom mrava, koji će se razmatrati u ostatku rada.



Slika 1-1 Postavke eksperimenta dvostrukog mosta (a) s jednakim granama (b) s različitim granama

2. Teorijska podloga optimizacije kolonijom mrava

Insekti međudjeluju bez nadzora, te se njihove akcije temelje na interakcijama među pojedincima. Iako su njihove interakcije primitivne na nižoj razini, na onoj višoj – globalnoj razini često daju impresivne rezultate. Takvo kolektivno ponašanje koje izranja iz grupe naziva se inteligencija roja. Glavne prednosti su fleksibilnost (brza prilagodljivost promjenjivoj okolini), robusnost (otpornost na manja odstupanja) te samo-organizacija (sposobnost funkcioniranja bez nadziranja). [2] ACO algoritam upravo pripada klasi inteligencije roja (*Swarm Intelligence*).

2.1 Primjer iz prirode

Kako je algoritam optimizacije kolonijom mrava nastao po uzoru na ponašanje iz prirode, najbolje je princip rada objasniti upravo na stvarnom ponašanju. Stavimo li mrave u nepoznatu okolinu da pronađu izvor hrane, prvo će se kretati u potpunosti nasumično. U trenutku kada mrav pronađe hranu, tada će pri povratku u koloniju ostavljati trag feromona koji će privlačiti druge mrave upravo u smjeru hrane. Tako će se postepeno "neodlučni" mravi koji se kreću nasumično počet kretat u smjeru traga feromona, te će se pojam "slučajno" pretvoriti u "optimalno". Svaki novi mrav koji se priključi pronalasku hrane, pojačavat će trag feromona i tako privlačiti još više novih mrava. Riječ je o strategiji novačenja koji radi na principu povratne veze.

Kroz vrijeme, trag feromona počinje slabiti, što je dobra stvar, obzirom da je pozicija hrane u okolišu dinamična. Zasigurno ne bi bilo poželjno imati trag koji navodi ostale mrave u smjeru gdje je prije dva dana bila hrana, ali sada više nije. No, činjenica slabljenja traga ukazuje na drugu zanimljivu činjenicu. Ako pretpostavimo da postoje 2 izvora hrane, jedan bliži i jedan udaljeniji od kolonije i ukoliko se svi mravi u početku kreću nasumično, u jednom će trenutku oba izvora biti pronađena, te će mravi početi ostavljati trag feromona koji usmjerava ostale mrave u tom smjeru. No, kraći put povlači kraće vrijeme prolaska, što znači da će trag manje slabiti jer će se brže obnavljati. Za duži put treba duže vrijeme prolaska, odnosno više će feromona oslabiti. Kako mrave privlači jači trag feromona, tako će se više mrava odlučiti upravo za kraći put. Također je bitno istaknuti da isparavanje feromona ima prednost u smislu da se izbjegava konvergencija lokalno optimalnog rješenja, pa se ostavlja mjesto i za istraživanje širih prostora, na kojima se možda nalaze bolji izvori hrane.

Ideja algoritma upravo oponaša spomenuto ponašanje mrava. Uvodimo simulaciju "mrava" koji će se kretati po grafu tražeći najkraći put. Spomenuti se algoritam najčešće opisuje primjerom rješavanja problema putujućeg trgovca. Prednost ACO algoritma jest upravo ta što se može prilagoditi dinamičkoj situaciji (npr. promjeni grafa u vremenu).

Osim navedenih sličnosti, postoje i bitne razlike između stvarnog svijeta i simulacije umjetnih mrava. Tako primjerice umjetni mravi žive u diskretnom svijetu, dok se njihovi pokreti sastoje od prijelaza iz diskretnog u diskretno stanje. Također imaju interno stanje koje sadrži informacije o svojim prethodnim akcijama. ACO se često unaprjeđuje dodatnim sposobnostima poput lokalne optimizacije ili gledanja unaprijed, čime se ne mogu pohvaliti mravi iz stvarnog svijeta.

2.2 Opći model problema

Umjetni mravi u ACO algoritmu predstavljaju procedure koje inkrementacijski grade rješenje dodavajući komponente rješenja partikularnom rješenju koje je trenutno u izgradnji. Tako u općem slučaju zadajemo minimalizacijski problem (S, f, Ω) s pripadajućim značenjima: S je skup potencijalnih rješenja, f je ciljna funkcija koja pridružuje f(s,t) svakom potencijalnom rješenju $s \in S$, dok je Ω(t) skup zadanih ograničenja. Parametar t pokazuje promjenjivost funkcija u vremenu. Cilj je pronaći globalno optimalno rješenje s*, koje predstavlja minimalnu funkciju f.

Takav se matematički model najčešće predstavlja pomicanjem po potpuno povezanom grafu G = (V,E), gdje su restrikcije Ω(t) ugrađene u ponašanje umjetnih mrava. Skup V predstavlja čvorove, a skup E rubove zadanog grafa G. Komponente $v_i \in V$ i veze $e_{ij} \in E$ mogu biti povezane s tragom feromona τ i vrijednošću η. Trag feromona τ predstavlja dugotrajnu memoriju o trenutnom stanju procesa pretrage koji obnavljaju sami mravi. S druge strane, vrijednost η predstavlja a priori informaciju o konkretnom problemu, čiji izvor nisu mravi. Riječ je najčešće o cijeni dodavanja određene komponente u trenutno rješenje, na temelju čijih vrijednosti će mravi donositi odluke o kretanju po grafu.

2.2.1 Ponašanje umjetnih mrava

Svaki od k mrava koristi konstruirani graf G = (V, E) kako bi pronašao optimalno rješenje $s^* \in S^*$, u čemu mu pomaže lokalna memorija M^k koju može koristiti za gradnju dopuštenih rješenja, obradu zadanih vrijednosti η, procjenu kvalitete pronađenog rješenja ili povratka unatrag. Također postoji početno stanje x_s^k, kao i jedno ili više uvjeta prekida e^k. Ako se mrav nađe u stanju $x_r = (x_{r-1},i)$ bez da je zadovoljen uvjet prekida, tada se pomiče na čvor j unutar svog susjedstva N^k, odnosno u stanje (x_r, j). Odabir pokreta se temelji na vjerojatnosti temeljenoj na tragovima feromona i vrijednostima η, memoriji mrava i ograničenjima problema. Kako bismo kontrolirali utjecaj τ, odnosno η, uvodimo α i β parametre. Često se kaže da se η faktor vodi načelom pohlepne sile, obzirom da prevladava onaj susjedni čvor koji nosi trenutno najveći profit, neovisno hoće li to biti optimalan izbor na kraju. Za odabir jednog od mogućih putova, najčešće se koristi sljedeća jednadžba:

$$p_{ij}^k(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^\alpha * [\eta_{ij}]^\beta}{\sum_{k \in \text{dopušteno}_k}[\tau_{ik}(t)]^\alpha * [\eta_{ik}]^\beta} & ako\ (j \in \text{dopušteno}_k) \\ 0 & inače \end{cases} \qquad (1)$$

Kada se dodaje komponenta c_j u trenutno stanje, može se osvježiti trag feromona τ. Kada se izgradi rješenje, može se povratkom unazad osvježiti trag feromona na svim korištenim komponentama za rješenje. Osvježavanje traga feromona provodi se općom formulom:

$$\tau_{ij} = p * \tau_{ij} + \Delta\tau_{ij} \qquad (2)$$

p predstavlja koeficijent isparavanja feromona, dok je $\Delta\tau_{ij}$ količina feromona koja se ostavlja u novom koraku, obično zadana sa:

$$\Delta\tau_{ij}{}^{k} = \begin{cases} 1/L_k & ako\ mrav\ k\ putuje\ po\ rubu\ i,j \\ 0 & inače \end{cases} \quad (3)$$

gdje je L_k cijena puta (obično duljina). [3]

2.2.2 Okvir pseudokoda

U ovom će se poglavlju dati okvirni pseudokod ACO algoritma, dok će se kasnije algoritam opisati detaljnije za specifične probleme. Neformalno, ACO algoritam se zamišlja kao kombinacija tri procedure: StvoriRješenja, OsvježiFeromone i DaemonAkcije.

```
Procedura ACOalgoritam
        StvoriRješenja
        OsvježiFeromone
        DaemonAkcije % opcionalno
Kraj procedure
```

Slika 2-1 ACO pseudokod

StvoriRješenja upravlja kolonijom mrava koji konkurentno i asinkrono posjećuju susjedne čvorove unutar zadanog grafa. Kretanje se temelji na prethodno opisanoj jednadžbi (1). Jednom kada mrav stvori rješenje ili dok ga stvara, on će ga ujedno i ocijeniti što će predati proceduri OsvježiFeromone zbog odluke o količini feromona koji će ostaviti.

U koraku OsvježiFeromone modificira se trag feromona. Ako mravi prolaze nekim rješenjem, vrijednost feromona raste, dok onim neposjećenim čvorovima količina feromona pada uslijed isparivanja. Tako pojačanje feromona osigurava da će ona rješenja koja produciraju dobar rezultat biti korištena i u sljedećoj iteraciji, dok isparavanje feromona predstavlja zaboravljanje, čime se izbjegava lokalna konvergencija.

Konačno, u proceduri DaemonAkcije koriste se centralizirane akcije koje ne može izvesti jedan mrav, već je potreban neki globalni gledatelj koji će po potrebi uključiti lokalno optimiziranje ili na temelju globalnih podataka odlučiti treba li na nekim područjima pojačati feromon, ako se gleda globalno rješenje. Upravo posljednja procedura nije obvezna u svim oblicima ACO algoritma, pa se ona koristi ovisno o zadanom problemu.

2.3 Problem rasporeda posla

Problem rasporeda posla (Job Shop Scheduling Problem) predstavljen je standardnim modelom m-strojeva i n-poslova koji se sastoje od sekvence operacija koje izvode strojevi. Pokušavamo dodijeliti operacije strojevima i vremenskim intervalima tako da ukupno vrijeme izvođenja bude minimalno, s time da u nekom intervalu jedan stroj može istovremeno raditi samo jedan posao. Kako bismo problem mogli riješiti ACO algoritmom, moramo zadani problem prevesti u graf G = (V, E) gdje se definira skup komponenti problema $V = \{v_1,...v_N\}$, skup veza između elemenata E, takav da je $|E| \leq |V|^2$. Za svaki $e_{ij} \in E$ postoji pridružena vrijednost d_{ij} koja predstavlja cijenu putovanja od komponente i do j uz dodatnu oznaku $\eta_{ij} = 1/d_{ij}$. Prema potrebi se dodaje skup restrikcija Ω. Skup V daje sve moguće sekvence $<v_i...v_k>$ tako da je $S' \subseteq S$. Opće rješenje koje tražimo jest $\psi \in S$, dok je cijena funkcije $\Phi_\psi(E, t)$ povezana uz svako rješenje ψ, ovisno o vremenu t.

2.3.1 Pristup rješenju

Osnovna ideja jest imati populaciju m umjetnih mrava koji iterativno grade rješenje primjenjujući odluke kretanja n puta dok rješenje nije pronađeno. Mravi koji pronađu dobro rješenje, označit će svoj pređeni put tragom feromona na rubove grafa kojim su prošli. Mravi također imaju memoriju (tzv. tabu lista) koja pamti posjećene komponente njihovog puta. Ovdje uvodimo tehniku istraživanja kročenja nogom (*FS, Foot Stepping*). Ključna karakteristika FS strategije jest da mijenja informaciju koju su prikupili mravi. Mravi se koriste kako bi unaprijedili rješenje koje je pronašla kolonija. Za razliku od mnogih tehnika, FS je fleksibilna i neovisna o problemu, a često se uspoređuje s MMAS tehnikom (*Max-Min Ant System*).

Kročenje nogom je inspirirano mogućnošću pravih mrava koji mogu modificirati svoje putove ukoliko iznenadno naiđu na neku prepreku. U našem slučaju, ako se to dogodi, pretpostavljamo da će svaki trag feromona na tom području nestati. Stoga su mravi primorani pronaći novo rješenje. Cilj kročenja nogom jest stvoriti oscilacije na postojećim dobrim rješenjima koje je pronašla kolonija, te se primjenjuje tek nakon ili blizu globalne stagnacije. Preciznije, FS tehnika uzrokuje promjene traga feromona nekih rubova. Obzirom da FS počinje kada je pronađeno dobro rješenje, ono će uzrokovati lokalnu pretragu u okolini trenutnog rješenja. Kročenje nogom spada u DaemonAkciju opisanu u [2.2.2] jer je pod utjecajem vanjskog čimbenika. U nastavku razmatramo podvrste FS tehnike.

2.3.2 Nepropagacijsko kročenje nogom

Nepropagacijsko kročenje nogom (Non-Propagation based Foot Spepping) počinje nasumičnim odabirom skupom čvora $V' \subseteq V$, gdje je veličina |V'| = brojKoraka. Za svaki $v_i \in V'$ se smanjuje trag feromona susjednih rubova pomoću PRF funkcije (*pheromone reduction function*), osim onog s maksimalnim tragom feromona. Vrijednost feromona ne može pasti na nulu (ili ispod nule), jer onda to rješenje ne bi bilo obuhvaćeno u konačnom rješenju. Preostali rub pojačava vrijednost feromona preko PAF funkcije (*pheromone amplification function*).

```
Procedura NPFS
Ako (MožeStati( ) )
        Mem = dohvatiMatricaFeromona( );
        Za i = 1 do brojKoraka
                v_i = odaberi nasumični čvor
                max = pronađi maksimalni feromon iz vi
                Za j = 1 do broj rubova iz vi
                        Ako nije (mem[v_i][j] = max)
                                mem[v_i][j]=PAF(mem[v_i][j])
                        Inače
                                mem[v_i][j]=PRF(mem[v_i][j])
                Kraj za
        Kraj za
Kraj procedure
```

Slika 2-2 Nepropagacijsko kročenje nogom

Metoda MožeStati( ) provjerava jesu li zadovoljeni uvjeti za izvođenje još jednog koraka (obično zadana granica ili uvjet stagnacije). Treba istaknuti kako višestruki koraci u malim intervalima mogu uzrokovati prevelike smetnje te rezultirati nasumičnoj pretrazi. Upravo je zbog pozitivne povratne veze tragova feromona bitno dati vremena koloniji da se prilagodi novoj raspodjeli feromona.

2.3.3 Propagacijsko kročenje nogom

Propagacijsko kročenje nogom (*Propagation based Foot Stepping*) razlikuje se od nepropagacijskog u smislu da se efekt promjene traga feromona propagira na svaki susjedni rub nasumično odabranog vrha. To uzrokuje da mravi intenzivnije pretražuju svoju okolinu u potrazi za jačim tragom feromona.

```
Procedura PFS
Ako (MožeStati( ) )
        Mem = dohvatiMatricaFeromona( );
        Za i = 1 do brojKoraka
                v_i = odaberi nasumični čvor
                brojProp = min {N-v_i, propagacija}
                za k = 1 do brojProp
                        max = pronađi maksimalni feromon iz v_i
                        podStaza = pronađi sve čvorove nakon v_i
                        Za j = 1 do broj rubova iz vi
                                Ako nije (mem[v_i][j] = max)
                                        mem[v_i][j]=PF(mem[v_i][j])
                                Inače ako (i = 1)
                                        mem[v_i][j]=PRF(mem[v_i][j])
                        Kraj za
                        v_i = v_{i+1}
                Kraj za
        Kraj za
Kraj procedure
```

Slika 2-3 Propagacijsko kročenje nogom

Razlika NPFS-a i PFS-a vidi se u k-petlji koja se izvodi brojProp puta, što je ograničeno zadanom vrijednošću propagacija ili krajnjim točkama zadanog grafa. PodStaza tako sadrži sve dozvoljene čvorove za propagaciju. Također se vidi da se

redukcija feromona provodi samo u prvom koraku propagacije, što rezultira pojačano istraživanje od propagacijskog čvora, bez pojačavanja tragova feromona okolnih čvorova.

I u tehnici propagacijskog i nepropagacijskog FS-a vidjeli smo neke predefinirane vrijednosti poput brojProp, no postoji i proširenje tih osnovnih modela na dinamički FS gdje se takvim konstantama pridjeljuju vrijednosti ovisno o tijeku izvođenja programa. Premalen broj uzrokuje premale pomake, stoga neka globalna rješenja mogu biti izostavljena. S druge strane, preveliki brojevi otežavaju mravima mala poboljšanja nad rješenjima.

2.3.4 Max-Min sustav mrava

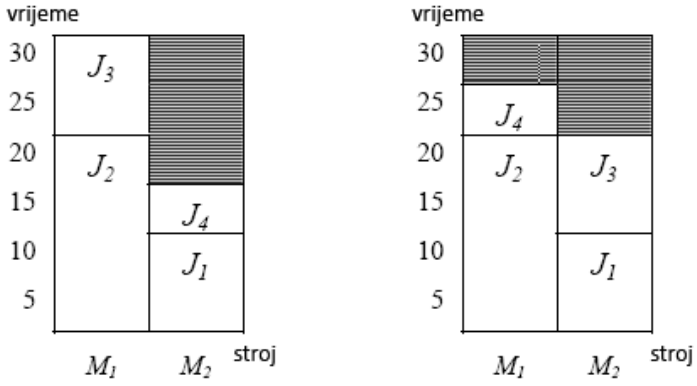
Max-Min sustav mrava (*MMAS, Max-Min Ant System*) razvili su Stutzle i Hoos kako bi se više iskoristila najbolja rješenja u procesu pretrage, a istovremeno izbjegla preuranjena stagnacija. Tako je MMAS za razliku od običnog Ant Systema, elitistički model koji ne ističe samo globalno najbolje rješenje, nego i najbolje rješenje trenutne iteracije. Koncept minimuma i maksimuma vrijednosti feromona na rubovima se uveo zbog preuranjene stagnacije, dok se inicijalno svi postave na maksimalne vrijednosti.

Jedna od bitnih karakteristika MMAS-a jest elitizam koji potiče rješenja koja se često pojavljuju kao iteracijski najbolja. Eksperimentalno su utvrđene i preporučene granice:

$$\tau_{max} = \frac{1}{1-p} * \frac{1}{\Phi_{najbolje}(E,\tau)} \qquad (4)$$

$$\tau_{min} = \frac{\tau_{max} * (1 - p_{dec})}{(avg - 1) * p_{dec}} \; uz \; p_{dec} = \sqrt[n]{p_{najbolje}} \qquad (5)$$

$p_{najbolje}$ predstavlja vjerojatnost odabira ispravnog odabira na svakoj točki odluke, gdje je za izračun potreban n = veličina puta. Obzirom da jednadžba za τ_{max} ovisi o trenutno najboljem rješenju, jasno je da će se kroz iteracije mijenjati vrijednosti τ_{max} i τ_{min}. [4]



Slika 2-4 Potencijalna rješenja problema rasporeda posla

2.4 Problem bojanja grafa

Problem bojanja grafa (*Graph Coloring Problem*) definiramo grafom G = (V, E) sa skupom čvorova V i skupom rubova E. Zadan je prirodni broj k, kako bismo predstavili k-bojanje kao funkciju boj : V → {1, 2, ... k}. Vrijednost boj(x) čvora x se zove boja čvora x. Čvorovi s jednakom bojom definiraju klasu boje. Ako dva susjedna čvora x i y imaju jednaku boju, tada x i y nazivamo konfliktnim čvorovima, dok je rub koji ih spaja konfliktni rub. Klasa boje bez konfliktnih rubova jest stabilan set, dok se k-bojanje bez konfliktnih rubova zove dozvoljeno rješenje i predstavlja pridruživanje čvorova u k stabilnih setova. Problem bojanja grafa traži minimalni broj k za koji postoji dozvoljeno rješenje nad zadanim grafom.

Ukoliko zadamo prirodni broj k, tada rješavamo optimizacijski problem k-GCP s ciljem otkrivanja k-bojanja grafa G s minimalnim brojem konflikata (u idealnom slučaju tražimo nula konflikata).

2.4.1 Pristup rješenju

Razlika između klasičnog ACO algoritma leži u tome što ćemo za ovaj pristup koristiti pojedinog mrava ne zato da izgradi cijelo rješenje, već da pridonese davanjem boje jednom čvoru. Također, stvorit će se samo jedno rješenje koje će evoluirati do optimalnog. Ideja se ostvaruje kreiranjem k-bojanja koje ne mora biti dozvoljeno rješenje, no svakim ćemo sljedećim korakom pokušati minimalizirati broj konflikata.

Tako će svaka boja iz {1, ..., k} biti pridružena svakom mravu, dok će k mrava biti pridruženo svakom čvoru. Dakle, koristimo ukupno |V| mrava od svake boje. Koristit ćemo proceduru Oboji kako bismo se približili optimalnom rješenju.

Neka je A skup čvorova inicijaliziranih na V koje trebamo obojiti uz zadano k-bojanje nad podgrafom G' = (V' = V – A, E'). Svakim korakom, procedura Oboji bira čvor x iz A s najvećim stupnjem saturacije (broj različitih boja susjednih vrhu x). U slučaju višestrukog izbora, bira se onaj čvor s više susjednih čvorova. Zatim x-u pridružujemo boju koju predstavlja barem jedan mrav na x, ali biramo onu koja minimalizira broj konflikata. U slučaju višestrukog izbora, bira se ona boja koja se podudara s više mrava na tom čvoru. Na kraju izbacujemo x iz A.

Tako na početku stavimo po jednog mrava svake boje na svaki vrh, te primjenjujemo proceduru Oboji na A = V. Zatim iterativno modificiramo pozicije mrava te na kraju svake iteracije ponovno zovemo proceduru Oboji. Ako uzmemo dva mrava: a_i koji nosi boju i na čvoru x i mrava a_j koji nosi boju j na čvoru y, tada se pokret (x,i) ↔ (y,j) sastoji u razmjeni pozicija a_i i a_j. Na kraju svake iteracije t, takvim pokretima modificiramo distribuciju mrava na grafu i zatim raspodijelimo boje vrhovima koristeći Oboji proceduru. Na kraju svake iteracije stoga moramo promijeniti boju čvorovima koji su bili uključeni u pomicanje u toj iteraciji, te sve čvorove koji su bili u konfliktu u prošloj iteraciji.

Bitno je definirati skup pokreta koje treba obaviti u iteraciji t. Uzmimo da je $N_l(x,t-1)$ broj mrava boje l na x-u u iteraciji t-1. Cilj nam je sada promijeniti boju barem jednog konfliktnog vrha x. Recimo da je x bio i boje u iteraciji t-1. Iz toga slijedi Ni(x,t-1)>0

jer je postojao barem jedan mrav te boje u prethodnom koraku. Kako bismo bili sigurni da ćemo promijeniti boju čvora x koji je u konfliktu, moramo premjestiti sve mrave i boje sa čvora. Izvodimo $N_i(x, t-1)$ pomaka m iz skupa:

$$M(t) = \{m = (x,i) \leftrightarrow (y,j)\ uz\ y \neq x, N_j(y,t-1) > 0, j \neq i\} \quad (6)$$

Odabir pomaka m u iteraciji t se provodi po ACO principu, koji objedinjuje pohlepni kriterij i trag feromona, uz faktore α i β, s time da su τ i η normalizirani na [0,1]:

$$p(m,t) = \alpha * \eta(m,t) + \beta * \tau(m,t) \quad (7)$$

2.4.2 Pohlepna sila

Pohlepna sila predstavlja kratkotrajni profit pojedinog mrava ukoliko obavi akciju m. Svakom iteracijom, želimo ukloniti konflikt što postižemo premještanjem mrava koji su obojani jednakom bojom c, a nalaze se na susjednim čvorovima x i y. Recimo da želimo promijeniti trenutnu boju i čvora x. Tada trebamo pomaknuti sve mrave boje i sa čvora x. Tako definiramo pohlepnu silu:

$$\eta(m,t) = Plus(m,t) - Minus(m,t) \quad (8)$$

gdje su Plus i Minus normalizirane prednosti odnosno nedostaci izvođenja pokreta m u iteraciji t. Definiramo S(x, y, i, t) kao broj mrava boje i na čvorovima različitim od y koji su susjedi od x. Mrava a_i na čvoru x privlači čvor y ako postoje mravi boje i na y ili ako postoje mravi boje i na čvorovima različitim od y, a susjednim x. Analogno se primijeni za mrava a_j čime dobivamo:

$$Plus(m,t) = N_i^2(y,t-1) + S(x,y,i,t-1) + N_j^2(x,t-1) + S(y,x,j,t-1) \quad (9)$$

Bitno je primijetiti kako se uzimaju oba mrava u obzir, dok se pojačava utjecaj grupiranja mrava jednake boje kvadriranjem člana N. Podsjetimo da $N_i(x, t)$ predstavlja broj mrava boje i na čvoru x u iteraciji t.

S druge strane, mrav a_j na čvoru y je odbijen od čvora x ako postoje mravi boje j na čvoru y ili postoje mravi boje j na čvorovima različitim od y, a susjedni x. Kao što znamo, postavili smo za cilj sve mrave boje i maknuti sa x, pa tako čvor x nije privučen čvorom y samo ako postoje mravi boje i na čvorovima različitim od x, susjednim y. Stoga dobivamo formulu:

$$Minus(m,t) = N_j^2(y,t-1) + S(x,y,j,t-1) + S(y,x,i,t-1) \quad (10)$$

2.4.3 Trag feromona

Osvježenje traga feromona kojeg ostavljaju mravi boje c na čvoru v na kraju iteracije t odgovara:

$$tr(v,c,t) = p * tr(v,c,t-1) + \Delta tr(v,c,t) \qquad (11)$$

gdje je 0 < p < 1 parametar isparavanja feromona. Dobar sustav traga feromona, djelovat će prema sljedećim točkama:

- Ako mrav boje c napusti čvor v, tada tr(v, c, t) treba ispariti, a ne se pojačati.
- Ako nema mrava boje c koji napusti čvor v iz skupa P(x), tada tr(v, c, t) treba ispariti i lagano se pojačati.
- Ako mrav boje c dođe u čvor v, onda tr(v, c, t) treba ispariti i pojačati, s time da ga treba posebno pojačati ako je generirano rješenje s' bolje nego s*.
- Ako se ništa ne dogodi relativno za čvor v i boju c, tada tr(v, c, t) treba lagano ispariti, bez pojačanja.

Sada se eksperimentalnim putem mogu utvrditi parametri isparavanja i pojačavanja za razne slučajeve. Konačno definiramo za pomak m =(x, i) ↔ (y,j) :

$$tr(m) = tr(x,j,t) + tr(y,i,t) - tr(x,i,t) - tr(y,j,t) \qquad (12)$$

Inicijalno se postavi tr(x, j, 0) = 1 za svaki x iz skupa čvorova i j iz skupa boja.

2.4.4 Pseudokod

Na temelju prethodnih postavki, možemo definirati osnovni pseudokod za rješavanje problema bojanja grafa. Eksperimentalno se pokazalo kako je optimalno uzeti parametar pohlepne sile α=1, te utjecaj traga feromona β=5, čime se stavlja naglasak na prednost "mravljeg" pristupa rješavanju problema.

```
t=0
postavi po jednog mrava svake boje na svaki čvor
inicijaliziraj tragove svake boje na svakom čvoru na 1
primijeni proceduru Oboji na A=V, gdje je s trenutno rješenje
postavi t=1, t'=1 A=V f* = f(s) i s*=s
dok (t' < maxiteracija) i (f*>0) radi
        odredi skup M(t) mogućih pokreta za iteraciju t
        izračunaj pohlepnu silu i trag feromona za svaki pokret M(t)
        normaliziraj pohlepnu silu i trag feromona na intervalu [0,1]
        učini dopušteni pokret m =(x, i) ↔ (y,j)  iz skupa M(t) uz maksimalni p(M, t)
        dok je barem jedan mrav boje i na čvoru x
                izvedi m maksimalnog p(M, t) iz skupa M(t) uz y ≠ x i da je boja j na y
        osvježi tragove tr(v, c) za svaki čvor v i boju c, te ih normaliziraj
        osvježi dopuštene pokrete
        postavi A = {čvorovi uključeni u pokrete iteracije t} U {konfliktni čvorovi iz t-1}
        osvježi boje čvorova iz A koristeći proceduru Oboji, uz dobiveno rješenje s
        ako je f(s)<f*, postavi s*=s, f*=f(s) i t'=-1
        postavi t=t+1 i t'=t'+1
```

Slika 2-5 Prilagođeni pseudokod za rješavanje problema bojanja grafa [5]

2.5 Ostale primjene

ACO se primjenjuje u dvjema kategorijama problema: statičkim i dinamičkim. Kod statičkih problema se karakteristike zadaju jednom, i to na početku i situacija se ne mijenja tijekom traženja rješenja. Klasičan primjer je problem putujućeg trgovca gdje se lokacije gradova i njihove relativne udaljenosti ne mijenjaju. S druge strane, kod dinamičkih se problema stvari mijenjaju u stvarnom vremenu, čemu se ACO uspješno prilagođuje. Standardni primjer u dinamičkoj kategoriji bilo bi mrežno usmjeravanje (*network routing*), jer se u mrežama čvorovi mogu dodavati i uklanjati.

2.5.1 Problem putujućeg trgovca

Problem putujućeg trgovca (*Traveling Salesman Problem*) predstavlja se potpunim težinskim grafom G = (N, A) gdje je N skup čvorova koji predstavljaju gradove, dok A predstavlja skup putova između svaka 2 čvora. Svaki put (i, j) $\in$ A ima pridruženu vrijednost d_{ij} koja predstavlja udaljenost između gradova i i j, uz i,j $\in$ N. Možemo razlikovati asimetrični problem, gdje za barem jedan put (i, j) vrijedi $d_{ij} \neq d_{ji}$, odnosno udaljenost ovisi o smjeru kretanja. U simetričnom problemu vrijedi $d_{ij} = d_{ji}$ za svaki put (i, j) iz skupa A. Rješenje koje tražimo jest najkraći zatvoreni put kroz graf tako da se svakim čvorom n = |N| iz grafa G prođe točno jednom. Takav je put poznat pod nazivom Hamiltonovski put koji se može prikazati permutacijom π čvorova s indeksima {1, 2, ..., n} tako da je udaljenost f(π) minimalna: [1]

$$f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)} \qquad (13)$$

2.5.2 Problem kvadratičnog pridruživanja

Problem kvadratičnog pridruživanja (*Quadratic Assignment Problem*) reda n sastoji se od traženja najbolje raspodjele n aktivnosti na n lokacija, a zadali su ga Koopmans i Beckman 1957. Matematički se problem definira trima matricama dimenzija n*x*n: matrica D sadrži udaljenosti među lokacijama, matrica F sadrži protok između aktivnosti, te matrica C s cijenama pridruživanja aktivnosti lokaciji.

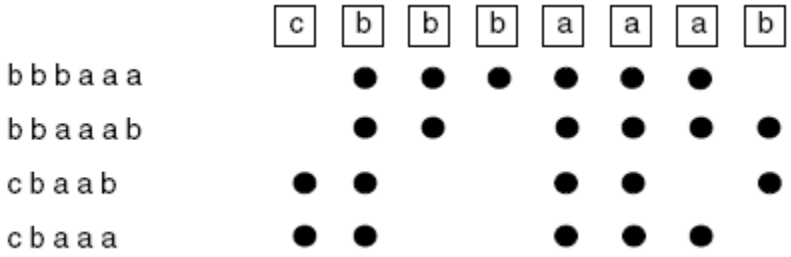
Pod tim pretpostavkama uzimamo permutaciju π: i -> π(i) kao pridruživanje aktivnosti j lokaciji i. Cijena prijenosa predstavljena je umnoškom udaljenosti lokacija kojima su aktivnosti pridružene i protokom između dviju aktivnosti. Tako se problem svodi na traženje funkcije:

$$\min z = \sum_{i,h=1}^{n} d_{ih} * f_{\pi_{(i)}\pi_{(h)}} \qquad (14)$$

Primjene rješavanja tog problema sežu do planiranja građevina na sveučilištima, odjela u bolnicama i minimalizaciju duljinu žica u električnim krugovima.[6]

2.5.3 Problem najkraće zajedničke supersekvence

Problem najkraće zajedničke supersekvence (*Shortest Common Supersequence Problem*) širi primjenu ACO algoritma na znakove. Zadan je skup L nizova slova nad abecedom Σ. Tražimo niz minimalne duljine tako da je supersekvenca svakog od nizova iz L. Niz B je supersekvenca niza A ukoliko se A može dobiti iz B brišući iz B nula ili više znakova. Rješavanje ovog problema ima primjene u DNK analizi, te dizajniranju pokretnih traka u automatiziranim procesima. [1]



Slika 2-6 Grafički prikaz pronalaženja supersekvence na temelju zadana 4 niza

2.5.4 Mrežno usmjeravanje

Mrežno usmjeravanje (*Network Routing*) se svodi na problem pretrage grafa koji predstavlja komunikacijsku mrežu. Iako je cilj pronaći najkraće udaljenosti između svih čvorova u grafu, ipak problem postaje kompliciran kada cijene puta ovise o vremenu. Zato je ovaj problem dinamičke prirode, u kojem se primjena ACO algoritma odlično snalazi. Preciznije, za taj specifični problem koristi se posebna inačica ACO algoritma – AntNet algoritam kojeg su predložili Di Caro i Dorigo 1998. godine. [7]

Kod AntNeta mravi donose odluke na temelju podataka Ai = $[a_{ind}(t)]_{|Ni|,|N|-1}$ čvora i koji se popunjavaju formulom:

$$a_{ind}(t) = \frac{\omega * \tau_{ind}(t) + (1-\omega) * \eta_n}{\omega + (1-\omega) * (|N_i| - 1)} \qquad (15)$$

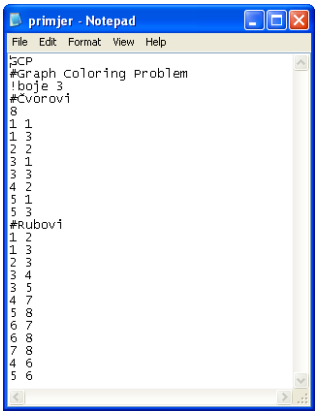
Gdje je N_i skup susjednih čvorova od i, n N_i, η_n je normalizirana unaprijed zadana vrijednost koja je obrnuto proporcionalna duljini poveznice prema čvoru n, dok ω predstavlja težinski faktor za utjecaj faktora. Zanimljivo je istaknuti da se mravi prilikom gradnje rješenja koriste istim redovima čekanja kao i pravi podaci koji bi prolazili kroz mrežu, kako bi iskusili uobičajena kašnjenja za mrežu, što će se uzeti u obzir kod evaluacije rješenja. Također se treba uzeti u obzir pod kakvim se uvjetima opterećenosti mreže provodi ispitivanje, pa se rezultati često iskazuju relativno u odnosu na stanje mreže. U trenutku kada mrav pronađe odgovarajuće rješenje, tada se vraća na početnu poziciju koju je zapamtio u svojoj memoriji, te ovisno o kvaliteti rješenja putem pojačava trag feromona.

3. Programska implementacija

Ovo poglavlje se usredotočuje na programsku realizaciju primjene optimizacije kolonije mrava na problemu bojanja grafa opisanog u poglavlju [2.4]. Program je ostvaren u Microsoft Visual C# 2008. Osnovna ideja i algoritam preuzeti su iz [5].

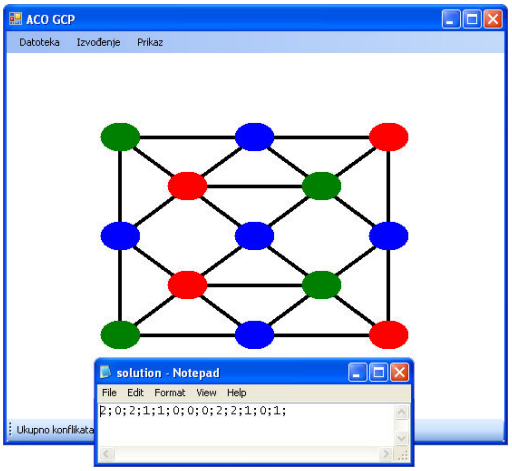
3.1 Ulazni i izlazni podaci

Ulazna datoteka se treba držati nekih pravila. Prvi red je rezerviran za ime problema ("GCP"). Svaki red koji počinje znakom "#" smatra se komentarom, dok početni znak "!" predstavlja definiranje parametra s ključnim riječima "alfa", "beta", "ro", "boje", "ciklusi" ili "iteracije", čije je značenje objašnjeno u [3.2]. Prvi od preostalih redova predstavlja broj čvorova u grafu, nakon čega se definiraju parovi koordinata za svaki čvor. Na kraju se popisuju rubovi grafa u obliku para povezanih čvorova.



Slika 3-1 Ulazna datoteka

Program za izlaz daje niz oblika "indeks1;indeks2; ... ;indeksN;" gdje svaki indeks predstavlja boju pridruženu čvoru na pripadajućoj poziciji. Takvo se rješenje može pohraniti u podrazumijevanoj datoteci "solution.txt".



Slika 3-2 Izlazni podaci u grafičkom i tekstualnom prikazu

3.2 Parametri

Zbog lakšeg korištenja i jednostavnosti prilikom uzastopnog testiranja, korisniku je ponuđena opcija ručnog namještanja parametara bitnih za rad algoritma.

Slika 3-3 Parametri koje korisnik može definirati i njihove početne vrijednosti

Parametri alfa i beta se odnose na bitnu formulu (7) računanja ukupne težine nekog pomaka kojeg će obaviti dva mrava. Alfa tako povećava važnost pohlepne sile, dok beta naglašava važnost tragova koje ostavljaju mravi u pokretu. Vrijednost ro predstavlja varijablu p u formuli (11) te ima značenje faktora isparavanja traga koji se odnosi na feromone, ne na pohlepnu silu. Ciklusi uvode pitanje izbjegavanja cikličnosti, pa se tako definira koliko iteracija treba proći prije nego što se na određeni čvor mogu vratiti mravi upravo one boje koja je pri pokretanju zaštitnog ciklusa bila uklonjena s tog istog čvora. Boje definiraju broj boja pomoću kojih želimo obojiti graf. Konačno, max iteracije se odnosi na broj iteracija koje će proći od posljednjeg najboljeg pronađenog rješenja, prije nego što se odustane od daljnje potrage.

3.3 Izbornici

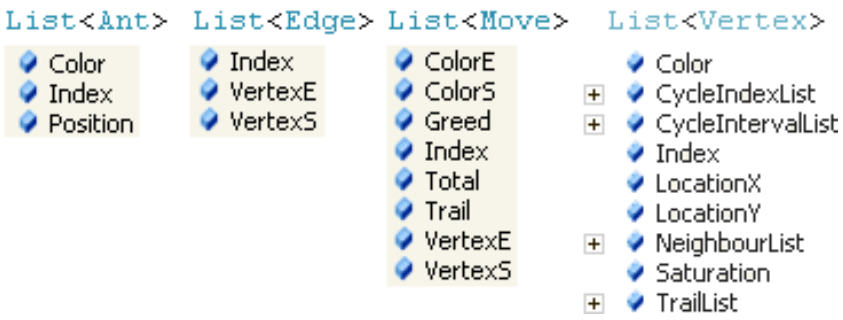
Postoje tri izbornika: "Datoteka", "Izvođenje" i "Prikaz". U prvom se izborniku redom može otvoriti proizvoljni par ulaznih datoteka, postaviti defaultne ulazne datoteke, spremiti dobiveno rješenje u defaultnu izlaznu datoteku ili izaći iz programa.

Drugi izbornik može pokrenuti algoritam na 2 načina: prvi preko parametra max iteracije koji je opisan u prethodnom poglavlju, a drugi preko parametra kojeg upisujemo u polje pored te opcije – kako bi se izveo određeni broj koraka algoritma. Dodatno postoje mogućnost povratka parametara na defaultne vrijednosti, kao i opcija "Reset" koja pokreće izvođenje trenutnog problema ispočetka.

Treći izbornik - "Prikaz" može dodatno prikazivati razne podatke, što će dodatno usporiti rad programa, pa su prema početnim postavkama svi isključeni. Tako prve tri opcije ispisuju određene informacije kod svake iteracije, dok posljednja opcija - "Liste podataka" ispisuje trenutno stanje svih četiriju lista struktura: "čvorovi", "rubovi", "mravi" i "pokreti".

3.4 Klase

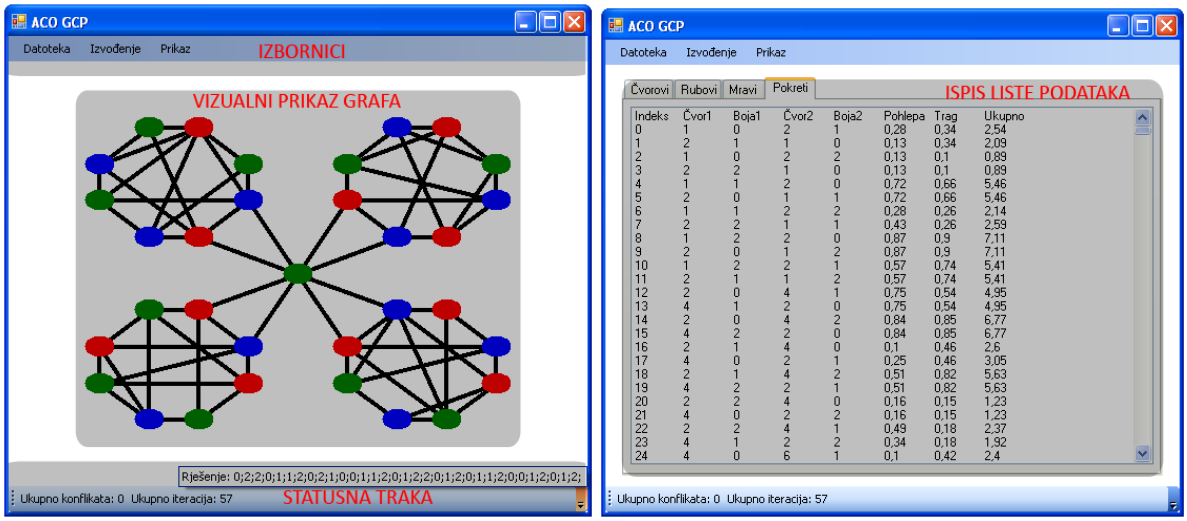
Programski je ostvareno četiri različitih klasa: "čvor", "rub", "mrav" i "pokret". Klasa mrav ("Ant") sastoji se od pozicije na kojoj se nalazi (čvoru) te boje koju predstavlja. Mravi ne mijenjaju boju, samo pozicije. Klasa rub ("Edge") pamti koje čvorove spaja. Klasa čvor ("Vertex") je nešto kompleksnija jer sadrži liste unutar sebe – tako svaki čvor mora pamtiti koje boje ne može poprimiti i koliko dugo (liste CycleIndexList i CycleIntervalList), zatim treba znati koji su mu čvorovi susjedi u listi NeighbourList, te dodatno sadržati popis tragova koji se pridružuju čvoru, na temelju kojih se računaju tragovi pridruženi pokretima. Tragovi pridruženi vrhovima koriste faktor isparavanja. Osim toga, svaki čvor ima svoju trenutnu boju, te faktor saturacije koji je opisan u [2.4.1]. Posljednja klasa "Move" sadrži informacije o čvoru i boji na obje strane ruba, vrijednostima pohlepne sile ("Greed"), traga feromona ("Trail") te njihovog zbroja.



Slika 3-4 Pregled korištenih klasa i njihovih svojstava

3.5 Korisničko sučelje

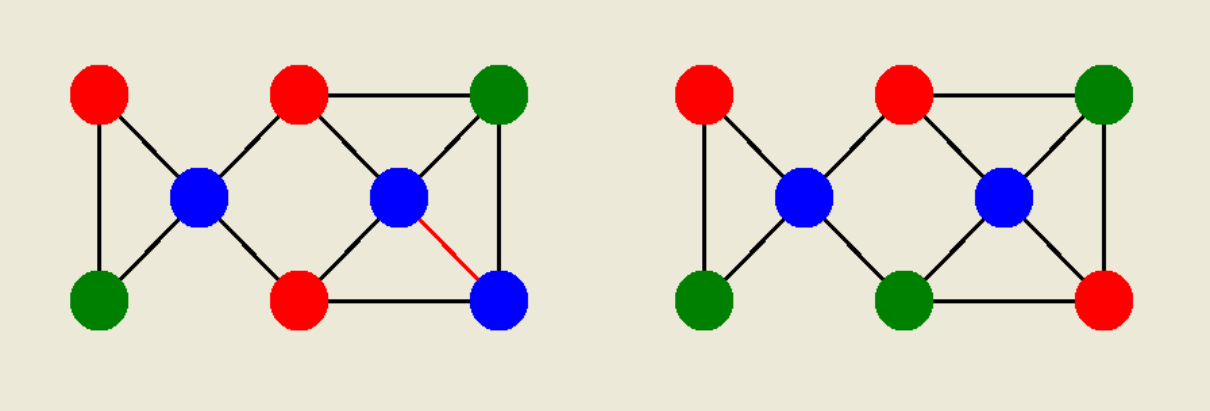
Korisničko sučelje objedinjuje prethodno opisane elemente.



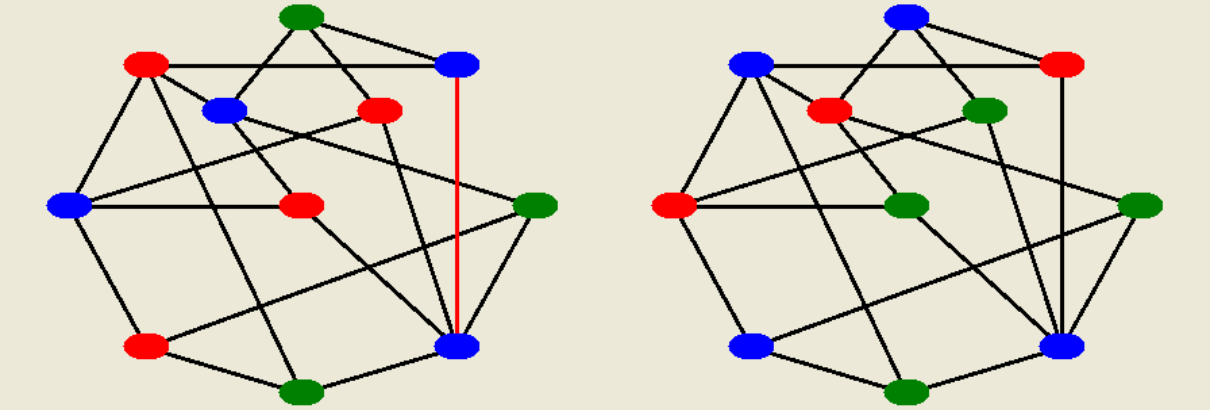
Slika 3-5 (lijevo) Glavno korisničko sučelje; (desno) Ispis liste podataka

3.6 Test primjeri

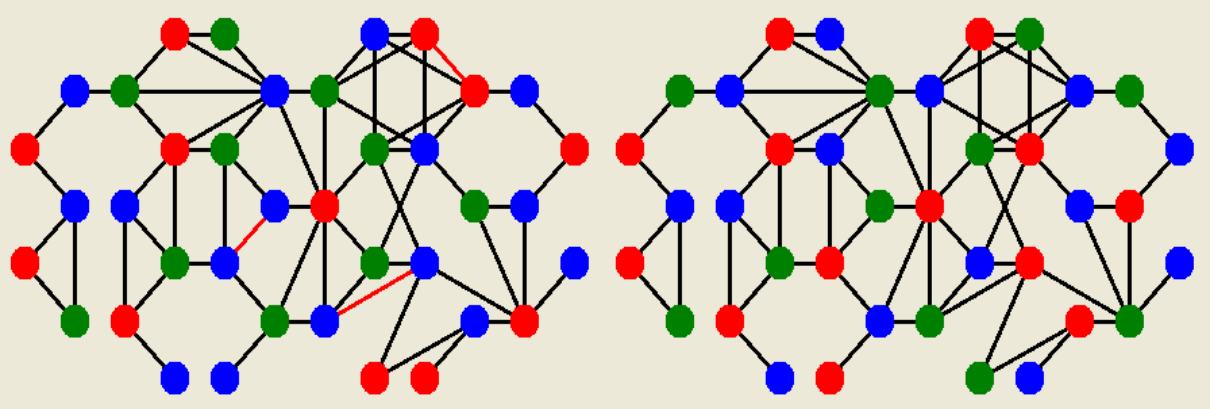
Test primjeri su pokazali jako dobro početno stanje koje se izračuna bez korištenja ACO algoritma. Algoritam dalje koristi dobiveno rješenje kroz iteracije pokušavajući ukloniti konflikte iz početnog. U test primjerima su korišteni defaultni parametri.



Slika 3-6 Prvi test primjer - (lijevo)početno stanje; (desno)rješenje nakon 1 iteracije



Slika 3-7 Drugi test primjer - (lijevo)početno stanje; (desno)rješenje nakon 7 iteracija



Slika 3-8Treći test primjer - (lijevo)početno stanje; (desno)rješenje nakon 39 iteracija

4. Rezultati i mjerenja

U provedenim mjerenjima će se mijenjati pojedini parametri, dok će ostali biti konstantne vrijednosti kako bi se pronašle optimalne vrijednosti određenih parametara. Promatrani parametri su navedeni u [3.2].

4.1 Odnos pohlepne sile i važnosti feromona

Očekivana teoretska vrijednost iznosi alfa = 1 i beta = 5, što znači da se veća prednost treba dati utjecaju feromona, nego pohlepnoj sili. Tablica 1 prikazuje rezultate dobivene za nekoliko primjera. Iako rezultati često znaju naglo varirati, treba se ipak uzeti neki prosjek, pa se tako najbolja rješenja dobivaju do omjera 1:5, dok je sve iznad manje prihvatljivo. Treba također istaknuti da se kod manje složenih problema mogu dobivati i dobri rezultati za veće omjere u korist alfe, iz razloga što se kod jednostavnih problema isplati koristiti pohlepni algoritam.

primjeri	1:1	1:2	1:3	1:4	1:5	1:6	1:7	1:8	1:9	1:10
1	115	291	1000+	86	76	1000+	1000+	1000+	1000+	324
2	577	227	654	29	29	766	31	732	100	632
3	23	62	1000+	150	38	38	38	260	316	174
4	148	19	143	43	44	286	223	236	247	518

Tablica 1 Utjecaj omjera alfe i bete na broj iteracija

4.2 Određivanje broja ciklusa za izbjegavanje cikličnosti

Čuvanjem alfe i bete u granicama normale, za neki jednostavniji primjer, dobiva se optimalni broj ciklusa između 2 i 5. Treba istaknuti da vrijednost broja ciklusa nije konstanta, te ovisi o složenosti zadanog grafa i broju konflikata koji se obično u njemu nalaze. Tako će za složenije zadatke trebati podesiti broj ciklusa na veći broj. U literaturi se spominje formula UNIFORM(0,9) + 0.6*NCV(s) gdje je NCV(s) broj konfliktnih rubova u trenutnom rješenju "s". [5]

ciklusi	9	8	7	6	5	4	3	2	1
alfa = 1 beta = 1	351	536	40	577	112	38	41	184	374
alfa = 1 beta = 2	130	1000+	160	227	49	45	47	21	103
alfa = 1 beta = 3	739	233	290	654	166	643	47	53	1000+

Tablica 2 Utjecaj broja ciklusa čekanja zbog cikličnosti na broj iteracija

4.3 Jakost isparavanja traga feromona

U literaturi se obično spominje optimalna vrijednost ro = 0.90, što su ova mjerenja i potvrdila. Naime, ako se držimo preporučenih omjera za alfu i betu, tada mijenjanjem parametra "ro" dobivamo najbolje rezultate za interval ro = [0.90 , 0.95].

ro	1,00	0,95	0,90	0,85	0,80	0,75	0,70	0,60	0,50	0,40	0,30	0,20	0,10
alfa = 1 beta = 1	163	41	41	189	100	39	268	203	57	135	78	81	182
alfa = 1 beta = 2	47	33	47	158	160	72	714	1000+	455	110	533	83	311
alfa = 1 beta = 3	47	47	47	957	1000+	179	626	134	1000+	296	169	151	123

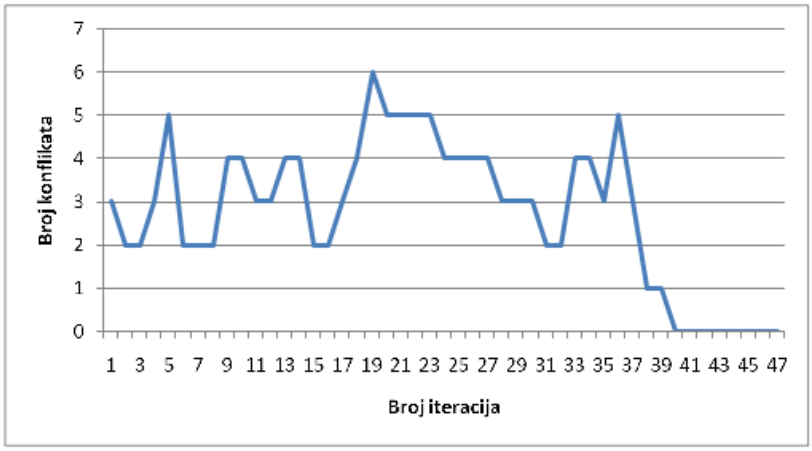
Tablica 3 Utjecaj jakosti isparavanja feromona na broj iteracija

Zanimljivo je još istaknuti da povećavanjem bete u odnosu na alfu izvan granica preporučenih omjera, dobivamo pomak optimalnog intervala za ro prema manjim vrijednostima. To se objašnjava činjenicom da se povećanjem bete povećava i doprinos feromona kod donošenja odluka. Zato se parametar ro treba smanjiti kako bi se smanjio i utjecaj iste sile i vratio u preporučene granice.

Ako idemo u drugi ekstrem – povećavanjem alfe u odnosu na betu, tada dobivamo rezultat da parametar ro ne utječe na kvalitetu rješenja, što se objašnjava činjenicom da se povećanjem alfe smanjuje utjecaj feromona i tako gubi važnost parametra ro, kada je doprinos te sile ionako već malen.

4.4 Broj konflikata kroz iteracije

Cilj je pokazati evolucijski proces. Razlog zašto broj konflikata ne pada konstantno leži u dobroj funkciji koja pronalazi početno rješenje od kojega algoritam započinje. No, iako početno rješenje ima mali broj konflikata, to ne znači da je rješenje kvalitetno u smislu da se malim brojem koraka može doći do željenog rješenja, već naprotiv, ovaj algoritam ima sposobnost odmicati se od lokalnih ekstrema kako bi se pronašlo optimalno rješenje, čak i ako to znači da treba povećati broj konflikata.



Slika 4-1 Evolucijski proces s dobrim startom

4.5 Usporedba s drugim algoritmima

Izrađeni algoritam se prema [5] naziva ANT-COL i ovdje se uspoređuje s drugim poznatim metodama za bojanje grafa. U obzir se uzima Dsatur kojeg je razvio Daniel Brelaz 1979, zatim mravlji algoritmi ANT-DSATUR i ANT-RLF, poznati tabu algoritam za pretraživanje Tabucol, te hibridni genetski algoritam GH, poznat kao najbolja metoda za bojanje grafova.

Metode su se uspoređivale na grafovima prosječnih gustoća d = 0.5, gdje je gustoća prosječan broj rubova između 2 čvora. Rezultati su prikazani kao parovi najvjerojatnijih boja koje će se koristiti za legalno bojanje grafa i minimalnih korištenih boja u najboljem slučaju.

|V|	ANT-COL	DSATUR	ANT-RLF	ANT-DSATUR	Tabucol	GH
100	16 (16)	19 (18)	16 (15)	16 (15)	14	14
300	39 (37)	43 (42)	36 (35)	39 (38)	33	33
500	59 (57)	66 (65)	56 (55)	68 (67)	49	48
1000	106 (105)	115 (114)	111 (111)	115 (114)	89	84

Tablica 4 Usporedba promatranog algoritma ANT-COL s ostalim algoritmima [5]

ANT-COL inače koristi ideje iz DSATUR i ANT-DSATUR, a iz tablice se vidi kako je dobivena metoda uspješnija od njihovih prethodnika. S druge strane, može se vidjeti kako je ANT-COL osjetno slabiji od Tabucola i GH-a koji koristi Tabucol. Prednost vjerojatno leži u tome što se za ANT-COL koristilo puno podataka na temelju kojih se donosi odluka (tragovi i pohlepne sile za svaki potez). No, osnovna ideja koju nosi ANT-COL jest ta da individualni mrav nosi malu ulogu, za razliku od ostalih gdje obično jedan mrav uz pomoć drugih generira konačno rješenje. Ovime se pokazalo da, iako sporiji, algoritam kod kojeg će mrav doprinositi samo dijelu rješenja, može se mjeriti s ostalim algoritmima.

5. Zaključak

ACO algoritam je relativno nov pojam u svijetu, pa je područje primjene i istraživanja dosta aktivno i aktualno u današnje vrijeme. Nove izazove predstavljaju višeciljni i dinamički problemi, za razliku od uglavnom statičkih koji su razmatrani u prošlosti. Upravo na takvim problemima dolazi moć lake prilagodbe novim situacijama ACO-a do izražaja. Također porastom interesa za paralelne arhitekture, pokušava se u budućnosti implementirati ACO u sklopu raspoloživog hardware-a.

Obzirom da je ACO algoritam samo inspiriran jednostavnošću individualnog mrava i kompleksnošću mravlje kolonije kao cjeline, ne možemo reći da je riječ o plagijatu. Od mrava je samo posuđena dobra ideja i prenesena na novu razinu. Tako se ACO postepeno bavio problemima najkraće udaljenosti i matematičkog pridruživanja, da bi kasnije počeo stvarati rasporede, bojati grafove, usmjeravati vozila, sve do naprednih primjena poput mrežnog usmjeravanja, gradskog planiranja ili DNK istraživanja. Vjerujem da bi i sami mravi bili ponosni na doseg njihovog započetog rada.

Implementirani ANT-COL algoritam pokazao je da se, iako mu je potrebno više vremena, može mjeriti s drugim postojećim algoritmima, unatoč tome što u ovom algoritmu jedan mrav doprinosi bojanju jednog čvora, što i nije velika uloga za pojedinog mrava. Tako se smanjio doprinos pojedinca i stavio naglasak na zajedničko pronalaženje rješenja među mravima kombinirajući pohlepnu silu i tragove feromona koje ostavljaju.

6. Literatura

[1] "Ant Colony Optimization", Marco Dorigo & Thomas Stützle, 2004.

[2] "Swarm Intelligence: A Whle New Way to Think About Business", Eric Bonabeau & Christopher Meyer, 2001.

[3] "The Ant System: Optimization by a colony of cooperating agents", Marco Dorigo & Vittorio Maniezzo & Alberto Colorni, 1996.

[4] "Ant Colony Optimization for Job Shop Scheduling Problem", M. Ventesca & B. M. Ombuki, 2004.

[5] "A New Ant Algorithm for Graph Coloring", Alain Hertz & Nicolas Zufferey, 2006.

[6] "The Ant System Applied to the Quadratic Assignment Problem", Maniezzo & Colorni & Dorigo

[7] "The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances", Marco Dorigo, 2000.

7. Sažetak

ACO algoritam inspiriran ponašanjem mrava u prirodi rješava optimizacijske probleme koji se svode na pronalaženje najkraćeg puta u grafu iskorištavajući dvije sile – pohlepnu silu i trag feromona. Uveo ga je Marco Dorigo u literaturu 1992., nakon čega se počeo širiti spektar primjena nad raznim problemima, poput statičkih (putujući trgovac, raspored posla) i dinamičkih (mrežno usmjeravanje). Pogledom u budućnost, istraživanja se nastavljaju u smjeru dinamičkih problema, te prilagodbe paralelnim sustavima.

Ovaj rad je nakon općeg modela problema opisao dvije odabrane primjene u kojima se koristi ACO – problem rasporeda posla u kojem tražimo optimalnu podjelu aktivnosti nad akterima u nekom vremenu i problem bojanja grafa gdje izbjegavamo susjedne vrhove istih boja. Dodatno su spomenute još neke primjene statičkih i dinamičkih problema koje ACO uspješno rješava.

Praktični dio obuhvaća implementaciju postojećeg algoritma ANT-COL za bojanje grafa, koji se nakon usporedbe s drugim algoritmima pokazao boljim od svojih prethodnika, ali ipak slabijim od nekih poznatijih algoritama za bojanje grafova.