SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4370

Simulacija i vizualizacija umjetne kolonije mrava prilikom rješavanja problema labirinta

Filip Dujmušić

Zagreb, lipanj 2016.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

SADRŽAJ

1. Uvod

U ovom radu obrađen je algoritam optimizacije kolonijom mrava (engl. *ant colony optimization*, ACO). ACO je zapravo podvrsta veće grupe algoritama kojima je zajedničko da su zasnovani na inteligenciji roja čestica (engl. *swarm intelligence*), a koriste se pri rješavanju nekih poznatih NP-teških problema o čemu će biti više govora u 3. poglavlju.

Osim toga, u 2. poglavlju opisan je princip ACO algoritma te njegovih podvrsta. Ponašanje algoritma je analizirano na jednom konkretnom problemu pronalaska najkraćeg puta unutar labirinta.

Opis domene te prilagodba samog algoritma kako bi se ostvarila računalna simulacija pronalaska rješenja izloženi su u poglavljima 4 i 5.

Konačno, programski sustav je ispitan na nekoliko primjera različite složenosti a rezultati su komentirani u poglavlju 5.

2. Mravlji algoritmi

Marco Dorigo, jedan od pionira u ovoj domeni, je 1992. godine u svojoj doktorskoj disertaciji predložio varijantu ACO algoritma kao metodu pronalaska optimalnog puta na grafu [6]. Pri tome je pokušao modelirati stvarno kretanje mrava od mravinjaka prema izvoru hrane.

U sljedećim potpoglavljima analizirano je stvarno ponašanje mrava u prirodi te matematički model koji bi takvo ponašanje mogao opisati.

2.1. Inspiracija

Fascinantna je činjenica da su mravi slijepi ili imaju jako loš vid, a gledajući ih kao sustav primjećujemo visoku organiziranost što upućuje na to da ipak postoji neki oblik komunikacije.

Osnovne dvije činjenice koje omogućuju indirektnu komunikaciju jedinki su mogućnost ispuštanja kemijskog feromonskog traga te razvijen osjet feromona. Što se događa u pozadini i kako se ove dvije stvari koriste?

U početnom stanju kada je okolina nepoznata kretanje je nasumično, mravi se kaotično kreću u nadi da će pronaći izvor hrane. U trenutku kad neki mrav pronađe hranu, vraća se u mravinjak te putem ispušta feromonski trag. Ukoliko se drugi mravi nađu u blizini ovog traga šanse su veće da će i sami krenuti tim tragom nego da će nastaviti lutati. Novi mravi koji budu išli tim putem će i sami dodatno ispustiti feromon te na taj način podebljati trag što će uzrokovati privlačenje još više jedinki.

Vrlo važna stvar koja čini sustav dinamičnim je isparavanje feromonskog traga. Svojstvo te kemikalije je da intenzitet na nekoj lokaciji na kojoj je trag ispušten slabi proporcionalno vremenu koje je proteklo od tog trenutka. Prema tome, što je više vremena trebalo mravu da dođe do izvora hrane i natrag to će više vremena feromonski trag isparavati. S druge strane, što je put kraći, feromonski trag na njemu će se češće osvježavati jer će ga mravi učestalije prolaziti. Direktna posljedica svega ovoga je da će najkraći put u konačnici biti preferirani put za veliku većinu mrava. Da nema ovog

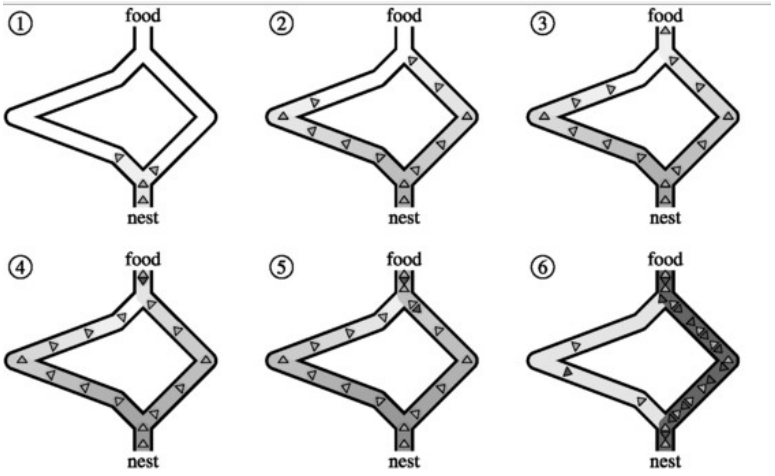
mehanizma prvi mrav koji nađe bilo kakav izvor hrane odredio bi odmah put koji se više ne bi mogao mijenjati, a potencijalne bliže ciljne točke ostale bi neotkrivene. Kada se izvor hrane iscrpi, mravi koji se s tog odredišta vraćaju neće više osvježavati trag koji će zbog toga nakon određenog vremena ispariti i "natjerati" koloniju da traži nove izvore.

Na slici 2.1 prikazano je stanje kolonije mrava koji se usmjereno kreću usmjereno zbog prethodno opisanog mehanizma.

Slika 2.1: Kolonija safari mrava[10].

Zanimljivo je da se osim pronalaska hrane feromon koristi kada je mrav u opasnosti ili ranjen. U tom slučaju se ispušta feromon koji signalizira uzbunu, a na kojeg drugi mravi reagiraju tako da se kreću prema njemu i postaju agresivni. Međutim ovakvo ponašanje nije našlo primjenu u evolucijskim algoritmima.

Eksperiment koji je uistinu potvrdio do sad analizirano ponašanje mrava je poznati "dvokraki most" kojeg su proveli Jean-Louis Deneubourg i suradnici[8]. Hrana i mravinjak su međusobno povezani dvama putevima od kojih je jedan vidljivo duži od drugog. Putevi se račvaju na mjestu bliskom mravinjaku i to pod kutem od 30°u odnosu na okomicu kako bi se postigla nepristranost u odluci svakog mrava koji mora izabrati jednu ili drugu opciju. Ideju prikazuje slika 2.2.



Slika 2.2: Eksperiment dvokrakog mosta prikazan kroz vrijeme.[9]

U početku bi mravi išli slučajnim izborom u obje grane ali nakon nekog vremena uočilo se da počinju češće ići kraćim putem. Priroda nas je još jednom fascinirala. Mrav kao jedinka je slab i izgubljen u prostoru ali promatrano na skali cjelokupne kolonije, zbog indirektne komunikacije jedinki dolazi do ponašanja kojeg nazivamo *izranjajuća inteligencija* (engl. *emerging intelligence*)[4].

Neke od ideja iznesenih u ovom poglavlju će upravo biti iskorištene u klasičnom ACO algoritmu opisanom u sljedećem poglavlju.

2.2. Algoritam Ant System

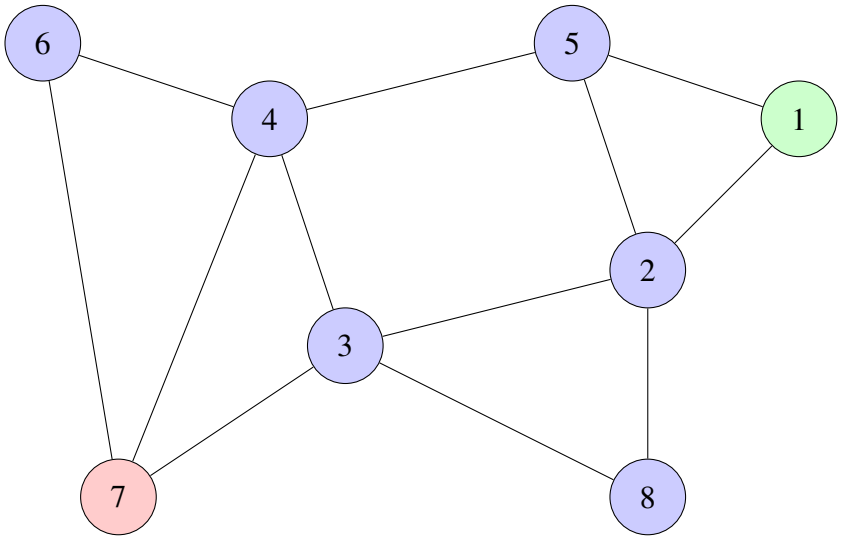
Općenito gledano, prostor pretrage modelira se jednim povezanim grafom i težinama prijelaza između povezanih čvorova. Primjerice takav jedan graf dan je na slici 2.3.

Zelenom bojom označen je početni čvor a crvenom ciljno stanje. U ovisnosti o problemu početna pozicija se može birati na različite načine i to ne mora nužno biti jedan čvor. Cijene prijelaza bit će određene konkretnim problem koji se rješava a na grafu su izostavljene.

Algoritam koristi skup elementarnih koraka pa počnimo redom.

2.2.1. Inicijalizacija

Stvara se početna populacija koja se sastoji od m agenata. Razine feromona svih prijelaza postavljaju se na neku malu početnu vrijednost τ_0. Zatim se trebaju odrediti



Slika 2.3: Prostor stanja nekog problema

parametri algoritma koji direktno utječu na njegovo izvođenje. Često je potrebno eksperimentalno odrediti vrijednosti ovih parametara kako bi se za konkretan problem u razumnom vremenu dobili prihvatljivi rezultati.

Postoji varijanta ovog algoritma koja dinamički optimizira svoje parametre uporabom genetskog algoritma (engl. *evolving ant colony optimization* - skraćeno EACO) i spada u domenu hiperheuristika ali to se u okviru rada neće obrađivati.

Tablica 2.1: Parametri modela

Oznaka parametra	**Kratki opis**
m	Broj agenata
α	Relativni doprinos feromonskog traga
β	Relativni doprinos heuristike
τ_0	Početna razina feromona na prijelazima
ρ	Brzina isparavanja feromona

Tablica 2.1 prikazuje osnovni skup parametara koji će biti korišteni u daljnjoj analizi.

Treba napomenuti da je izbor parametra τ_0 karakterističan za Ant System varijantu. Parametar se određuje kao:

$$\tau_0 = \frac{m}{C} \tag{2.1}$$

gdje C predstavlja najkraću udaljenost od početnog do ciljnog stanja. Ovo je važan

korak jer ukoliko se početni τ_0 postavi na premalu vrijednost posljedica je da će agenti jako konvergirati već nakon prvog ostavljanja feromonskog traga jer će razlika u razinama feromona na poljima traga i bez traga biti jako velika[4].

2.2.2. Strategija prijelaza

U ovom koraku definirana je osnovna strategija kojom se služi svaka jedinka kako bi napravila prijelaz iz trenutnog čvora na susjedni.

Proces prijelaza modeliran je pomoću vjerojatnosti a opisan je sljedećim izrazom[4]:

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{l \in N_i^k} \tau_{il}^\alpha \cdot \eta_{il}^\beta}, & \text{ako je } j \in N_i^k \\ 0, & \text{ako je } j \notin N_i^k \end{cases} \tag{2.2}$$

Iako djeluje složeno izraz je zapravo jednostavan. Izraz predstavlja vjerojatnost prelaska iz čvora *i* u *j* za *k*-tog mrava.

Brojnik daje mjeru dobrote prelaska u čvor *j* i može se vidjeti da ga određuje feromonski intenzitet τ na tom prijelazu te heuristika η koja govori koliko je dobro preći iz *i* u *j*. Ove heuristike su obično unaprijed poznate[4]. Feromonski utjecaj i heuristika su skalirani svojim koeficijentima α i β.

Nazivnik predstavlja sumu mjera dobrote svih mogućih prijelaza iz čvora *i* u susjede (pa tako i prijelaza koji je izračunat u brojniku). Stoga ovo doista predstavlja matematičku vjerojatnost.

U svrhu dodatnog pojašnjenja primjena ovog izraza prikazana je na konkretnom primjeru. Iz grafa 2.3 uzet je samo čvor 3 zajedno sa njegovim susjedima. Čvor 3 je trenutna pozicija agenta iz koje se računaju vjerojatnosti prijelaza.

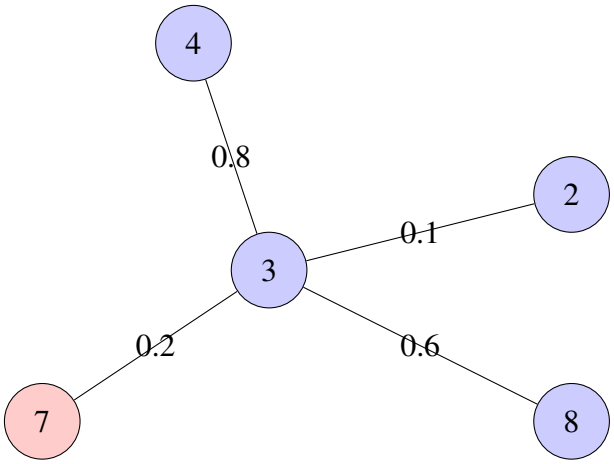
Pretpostavka je da je heuristika nepoznata, odnosno da je koeficijent $\beta = 0$. Prema tome, vjerojatnost prijelaza na susjede određivat će samo feromonske razine.

Dio grafa sa pretpostavljenim feromonskim razinama prikazan je na slici 2.4.

Kako je koeficijent $\beta = 0$ utjecaj heuristike η iščezava a od izraza 2.2 preostaje jednostavniji oblik:

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha}{\sum_{l \in N_i^k} \tau_{il}^\alpha} & \text{ako je } j \in N_i^k \\ 0, & \text{ako je } j \notin N_i^k \end{cases} \tag{2.3}$$

U tablici su prikazane raspodjele vjerojatnosti za tri različite vrijednosti parametra α.



Slika 2.4: Konkretna situacija za čvor 3 u *k*-toj iteraciji

Tablica 2.2: Parametri modela

$\boldsymbol{\alpha}$	$\boldsymbol{p^k_{3,4}}$	$\boldsymbol{p^k_{3,2}}$	$\boldsymbol{p^k_{3,8}}$	$\boldsymbol{p^k_{3,7}}$
0.5	0.37	0.13	0.32	0.18
1.0	0.47	0.06	0.35	0.12
2.0	0.61	0.01	0.34	0.04

Iz tablice 2.2 očigledne su dvije stvari. Suma vrijednosti po retcima iznosi točno 1.0 za svaki redak, još jedna potvrda da su izračunate brojke vjerojatnosti i da model funkcionira.

Druga stvar je utjecaj parametra α na raspodjele. Po grafu 2.4 prijelaz 3,4 je imao najveću razinu feromona a prijelaz 3,2 najmanju. Posljedica toga: 3,4 će imati najveću vjerojatnost a 3,2 najmanju u svakom slučaju. Međutim, ono što parametar α radi je mijenjanje odnosa tih vjerojatnosti.

Vidi se da kada je taj parametar $\alpha > 1.0$ razlika u feromonskim razinama se još jače ističe, postiže se povećanje vjerojatnosti prolaska putem najveće feromonske razine, odnosno smanjenje vjerojatnosti prolaska putem najmanje feromonske razine.

S druge strane uzimanjem parametra $\alpha < 1.0$ dolazi do suprotne situacije. Ujednačavaju se utjecaji pojedinih razina, odnosno smanjuje se značaj razine feromona.

2.2.3. Osvježavanje feromonskih tragova

Prilikom jedne iteracije algoritma čeka se da svaki mrav kretanjem po prostoru stanja (pomoću postupka opisanog u koraku 2) pronađe rješenje. Jednom kad se to dogodi kreće se u postupak inkrementacije feromonskih razina na određenim prijelazima.

Strategija koja određuje kako se ovaj korak čini dijeli opisani algoritam na neke specifične podvrste.

U obzir se uzimaju putevi od *n* najboljih agenata gdje je $n \leq m$. Parametar m je ukupan broj agenata (pogledati tablicu 2.1). Iako se mogu uzeti u obzir svi agenti u praksi se to pokazalo kao loša ideja.

Prije no što se uvećaju feromonske razine pristupa se procesu isparavanja koji je jednostavan. Prolazi se kroz sve prijelaze *j* i ažurira se razina feromona prema izrazu:

$$\tau_j = \tau_j \cdot (1 - \rho) \tag{2.4}$$

Na ovaj način se postiže eksponencijalno opadanje feromonskih razina kroz vrijeme i sprječava se opasnost od prevelikog povećanja razina te upadanja u lokalni optimum - drugim riječima sustavu se dodaje dinamika (kojoj dodatno pridonosi i faktor slučajnosti).

Nakon isparavanja slijedi uvećanje razina na putevima kojima je prošlo *n* najboljih mrava. Mravi se mogu rangirati po ukupnoj cijeni svih bridova kojima su prošli, gdje je manja suma znači bolji rezultat.

Obrati li se pažnja samo na jednog od najboljih agenata *i*, ažuriranje po njegovom putu će se obaviti na sljedeći način:

$$\tau_{jk} \leftarrow \tau_{jk} + \frac{1}{L^i} \tag{2.5}$$

gdje je L^i ukupna cijena prijeđenog puta mrava *i*. τ_{jk} predstavlja jedan djelić rute kojom je mrav prošao, a ažuriranje se mora obaviti po svim dijelovima. Očigledno je da će mravi koji su prošli boljim putevima ostaviti jači feromonski trag.

2.2.4. Pseudokod

Jednom kada su svi koraci algoritma objašnjeni može se napisati pseudokod.

Algoritam 1

```
1:  procedure ANT COLONY SYSTEM
2:      inicijalizacija(α, β, ρ, m, τ0)
3:      stvoriPopulaciju(m, position)
4:      postaviPočetneFeromone(τ0)
5:      while true do
6:          for all mrav_i ∈ populacija do
7:              while pozicija(mrav_i) ≠ cilj do
8:                  pozicija(mrav_i) ← tranzicija(pozicija(mrav_i))
9:          ispariFeromone()
10:         for all mrav_j ∈ najboljihM(populacija) do
11:             osvjeziFeromone(povijest(mrav_j))
```

Naravno dani pseudokod se pri rješavanju stvarnog problema neće vrtiti do u beskonačnost nego dok algoritam nakon dovoljnog broja iteracija ne proizvede prihvatljivo dobro rješenje.

2.3. Ostale podvrste algoritama

Ant System koji je u prethodnim poglavljima objašnjen je najstarija varijanta koju je, kao što je već spomenuto, predložio Dorigo[6].

Iako uspješna, povijesnim razvojem je došlo do pojave modificiranih verzija a dvije najpopularnije su obrađene u sljedećim potpoglavljima.

2.3.1. Ant Colony System

Ant Colony System algoritam (skraćeno ACS) je unaprjeđenje originalnog Ant System algoritma.

Prva važna razlika je način odluke kretanja agenata pri konstrukciji rješenja. U ACS algoritmu agenti koriste pravilo *pseudoslučajne proporcionalnosti*: vjerojatnost da mrav prijeđe iz čvora *i* u *j* ovisi o slučajnoj varijabli *q* koja je jednoliko distribuirana na intervalu $[0, 1]$, te o dodatnom parametru q_0; Ako je $q \leq q_0$ onda se od mogućih

prijelaza bira onaj kod kojeg je maksimalna vrijednost $\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}$. U suprotnom se taj prijelaz određuje kao kod AS[7][1].

Ovo pohlepno pravilo, koje mnogo više favorizira iskorištavanje informacije o fermonskim razinama, uravnoteženo je sa komponentom koja donosi dinamiku algoritmu: *lokalno osvježavanje feromona*.

Lokalno osvježavanje provodi svaki mrav nakon svakog pomaka i to na posljednji brid kojim je prošao:

$$\tau_{ij} = (1 - \varphi) \cdot \tau_{ij} + \varphi \cdot \tau_0 \tag{2.6}$$

gdje $\varphi \in (0, 1]$ predstavlja koeficijent *feromonskog raspadanja*.

Cilj lokalnog osvježavanja je da se pretraga što više rasprši. Ukoliko se bolje pogleda, izrazom 2.6 smanjuju se feromonske razine na bridovima *ij* a shodno tome i vjerojatnost da drugi mravi prođu tim istim bridom. Direktna posljedica ovoga je manja šansa da više mrava proizvede isto rješenje tokom jedne iteracije[7].

Naravno, lokalno osvježavanje treba razlikovati od općenitog postupka osvježavanja feromona, koji se provodi nakon svake iteracije na jednak način kao i kod AS algoritma, s razlikom da se u obzir uzimaju samo bridovi kojima je prošao najbolji mrav.[2]. Feromoni se uvećavaju na isti način a globalno isparavanje je zamijenjeno lokalnim.

Treba napomenuti da se većina novih stvari u algoritmu ACS već prije pojavila u Ant-Q algoritmu koji je razvijen od strane istog autora[5].

[1] Pogledati izraz 2.2

[2] Pogledati izraze 2.5 i 2.4

2.3.2. MAX-MIN Ant Aystem

MAX-MIN Ant System (skraćeno MMAS) je još jedno poboljšanje originalnog AS algoritma od kojeg se razlikuje u dvije stvari[7]

1. samo najbolji mrav u svakoj iteraciji može osvježavati feromonske razine
2. minimalne i maksimalne vrijednosti razina feromona na bridovima su eksplicitno ograničene (za razliku od AS i ACS algoritama kod kojih su ove granice bile implicitno određene samom procedurom)

Način na koji se osvježavaju bridovi je po već spomenutom izrazu 2.5 sa razlikom da se provodi samo za najboljeg mrava.

Razine feromona su ograničene sa τ_{min} i τ_{max}.Nakon uvećavanja razina na bridovima provjerava se da za svaki τ_{ij} vrijedi:

$$\tau_{min} \leq \tau_{ij} \leq \tau_{max} \tag{2.7}$$

Ako 2.7 ne vrijedi za neki brid, njegova se razina pomiče na bližu granicu (τ_{max} ili τ_{min} ovisno o tome u kojem je intervalu vrijednost bila).

Važno je za napomenuti da je postupak isparavanja globalni i provodi se po svim bridovima (baš kao kod klasičnog AS algoritma).

Vrijednost τ_{min} najčešće se eksperimentalno određuje iako postoji razvijena teorija o analitičkom računanju ovog parametra[11]. τ_{max} se može računati analitički ukoliko je unaprijed poznata cijena optimalnog puta L^*. U tom slučaju

$$\tau_{max} = \frac{1}{\rho \cdot L^*} \tag{2.8}$$

Također je bitno napomenuti da se inicijalne vrijednosti feromonskih razina na bridovima postavljaju na τ_{max} i algoritam se ponovno pokreće ako u određenom broju posljednjih iteracija nije došlo do poboljšanja rezultata[7].

3. Primjena mravljih algoritama

Postoje mnogi problemi u stvarnosti koji su dobro definirani ali su preteški za klasične algoritamske pristupe rješavanja problema kao što su potpuni prolazak kroz prostor stanja (engl. *brute force*) pa čak i neke naprednije tehnike poput dinamičkog programiranja.

U takvim situacijama pristupa se korištenju heurističkih odnosno metaheurističkih tehnika.

Metaheuristika je procedura ili heuristika višeg reda, osmišljena kako bi pronašla, generirala ili izabrala heuristiku (djelomično pretraživanje prostora stanja) koja može pružiti dovoljno dobro rješenje optimizacijskog problema, posebice u slučaju kada su podaci nepotpuni a računalni resursi ograničeni[1].

Dakle korištenje takvih tehnika ne može garantirati pronalazak optimalnog rješenja ali u razumnom vremenu može rezultirati dovoljno dobrim rješenjem.

Posebna grupa ovih tehnika su mravlji algoritmi koji bi se mogli svrstati u prirodom inspiriranu metaheuristiku zasnovanu na populaciji agenata.

U ovom poglavlju navedene su i neke konkretne primjene mravljih algoritama sa opisima problema koji se takvim algoritmima rješavaju.

3.1. Problem trgovačkog putnika

Problem ovog tipa je česta pojava u stvarnosti a definicija je jednostavna.

Problem opisuje trgovca koji mora proći kroz N gradova. Redoslijed obilaska je nebitan, uvjet je samo da se posjeti svaki grad, vraćajući se konačno u početni iz kojeg je krenuo. Svaki grad je povezan sa ostalim gradovima a cijene odlaska iz jednog u drugi su poznate. Trgovac nastoji minimizirati ukupnu cijenu svog putovanja.

Problem spada u grupu NP-teških problema a primjer je prikazan na slici ispod. 3.1.

[1] Bianchi, Leonora; Marco Dorigo; Luca Maria Gambardella; Walter J. Gutjahr (2009). "A survey on metaheuristics for stochastic combinatorial optimization". Natural Computing: an international journal

Slika 3.1: Optimalna ruta kroz 15 najvećih gradova u Njemačkoj[12]

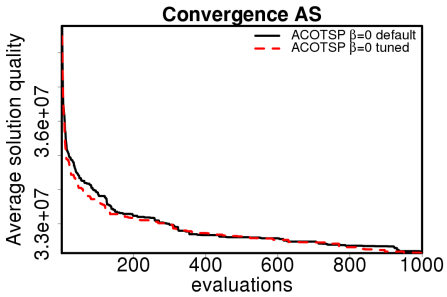
Već za problem veličine 15 gradova postoji 43.589.145.600 kombinacija a na slici je prikazana samo jedna od njih, koja je ujedno i rješenje problema.

Dodavanjem novih gradova ovaj broj se vrtoglavo povećava - pojava koju u matematici nazivamo kombinatornom eksplozijom.

Pokazalo se u praksi da su mravlji algoritmi jako dobri pri rješavanju ovog tipa problema.

U prvom koraku se populacija od m mrava postavlja nasumično na različite gradove. Mravi se po tim čvorovima kreću već opisanim postupcima a pri tome se koristi heuristika koja je funkcija cijene na nekom bridu. Svaki mrav ima ograničenu memoriju u kojoj pamti prethodna stanja kako bi se rješenje moglo rekonstruirati. Promjena feromonskih razina nakon jedne iteracije obavlja se na klasičan način kao kod algoritma AS i to tako da se prvo globalno reduciraju sve razine a potom se bridovi na putanji svakog mrava uvećavaju za iznos proporcionalan dobroti rješenja koje je pojedini mrav našao.

Slika 3.2 prikazuje kojom brzinom algoritam AS konvergira prema optimalnom rješenju kroz iteracije pri čemu su rješavani problemi veličine 50, 60, 70, 80, 90 i 100 gradova uz populacije od 5 do 100 agenata.



Slika 3.2: Konvergencija ACO heuristike kroz iteracije[3]

3.2. AntNet algoritam usmjeravanja

Telekomunikacijske mreže se također mogu predstaviti grafom gdje čvorovi predstavljaju usmjeritelje ili krajnje korisnike dok su bridovi komunikacijske veze koje pak mogu biti usmjerene, neusmjerene, sa ili bez fizičkog medija.

Zadatak usmjeritelja je promet koji njime prolazi što optimalnije distribuirati na svoje izlaze kako bi konačno put od izvora paketa do primaoca bio što optimalniji. Pri tom se ne misli samo na što manji broj skokova po čvorovima grafa, u obzir treba uzeti i dinamiku mreže te moguća zagušenja na nekim dijelovima. Uz to, dijelovi mreže mogu biti podložni promjenama kao što je iznenadno ispadanje nekog čvora uslijed greške.

Algoritam AS je našao svoju primjenu i u ovoj domeni. Algoritam je adaptivan jer mu faktor slučajnosti i mehanizam isparavanja omogućuju relativno brzu prilagodbu novonastalim promjenama u sustavu. Osim toga, jasno je da je ovaj algoritam kao i ostali iz grupe rojeva čestica skalabilan i otporan na pogreške zbog korištenja više agenata.

Algoritam AntNet, kojeg su razvili DiCario i Dorigo, je algoritam usmjeravanja za IP mreže bazirane na komutaciji paketa. Zadatak mrava u prirodi da pronađe najkraći put između mravinjaka i hrane je mapiran na domenu telekomunikacijske mreže i zadatka pronalaska najkraćeg puta između izvora paketa i odredišta. Agenti koji rješavaju ovaj problem su umjetni mravi koji putuju po mreži. Svaki čvor u mreži održava svoju tablicu usmjerenja baziranu na vjerojatnostima. Ove tablice imaju ulogu feromonskih razina u klasičnom algoritmu. Još se nazivaju i *feromonske tablice*. Tablica za svaku izlaznu vezu sadrži jedan broj τ_{jd} koji govori koliko je dobro paket čije je konačno odredište d poslati na vezu j.

Svaki čvor s u mreži šalje male kontrolne pakete (u ulozi mrava) u pravilnim vremenskim intervalima na nasumična odredišta. Cilj svakog takvog mrava je pronaći put do svog odredišta d i evaluirati ga. Put kojim se pojedini mrav kreće u svakom čvoru se

određuje izrazom 2.2 koji je osnova za tranzicije u algoritmu AS. Mrav pamti vrijeme koje mu je bilo potrebno od čvora do čvora te jednom kada stigne na svoje odredište d vraća se prema izvoru istim putem. Na svakom čvoru i pri povratku ažurira vrijednost zapisa u feromonskoj tablici koji se odnosi na odredište d. Korisni paketi koji prolaze mrežom usmjeravaju se na osnovu ovih tablica.

3.3. Detekcija crta na slici

Osnovna ideja pri rješavanju ovog problema je da se određen broj mrava nasumično rasporedi po slici dimenzija $M \times K$. Osim toga, inicijaliziraju se dodatno dvije matrice:

- feromonska matrica
- heuristička matrica

Feromonska matrica sadrži razine feromona na pojedinim slikovnim elementima. Ove se razine u početku inicijaliziraju na neku malu vrijednost τ_0.

Heuristička matrica gradi se prilikom incijializacije algoritma, a prije pokretanja iteracija. Računa se lokalno za svaki slikovni element (i, j) prema izrazu

$$\eta_{ij} = \frac{V_c(I_{i,j})}{V_{max}} \tag{3.1}$$

gdje $I_{i,j}$ predstavlja intenzitet slikovnog elementa na poziciji (i, j).

$V_c(I_{i,j})$ je funkcija koja djeluje lokalno na grupu slikovnih elemenata koji okružuju element (i, j) i računa se kao:

$$\begin{aligned} V_c(I_{i,j}) = |I_{i-1,j-1} - I_{i+1,j+1}| + |I_{i-1,j} - I_{i+1,j}| \\ +|I_{i-1,j+1} - I_{i+1,j-1}| + |I_{i,j-1} - I_{i,j+1}| \end{aligned} \tag{3.2}$$

V_{max} je maksimalni intenzitet u cjelokupnoj slici i služi samo kao faktor normalizacije.

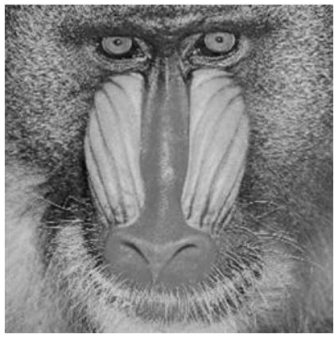
Nakon inicijalizacije slijedi iterativni postupak i inkrementalna konstrukcija rješenja. Kretanje mrava je opisano već dobro poznatim izrazom 2.2.

Mrav se može pomaknuti na bilo koji susjedni slikovni element osim onog na kojem je bio u prethodnom koraku. Svaki pomak mrava na novu poziciju rezultira lokalnim osvježavanjem razine feromona vezane za tu poziciju.

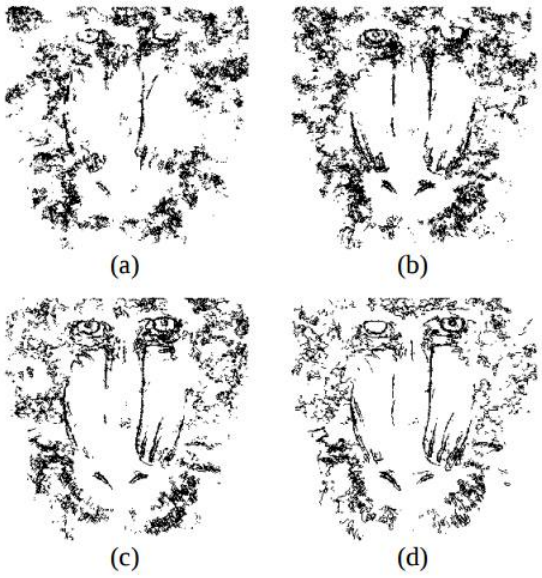
Nakon svake iteracije se također radi i globalno osvježavanje feromona. U problemima kao što su problem trgovačkog putnika imalo je smisla uzeti samo najboljeg mrava i njegovu putanju. Ovdje će se rješenje sastojati od više dijelova (više linija na

slici koje su pronađene) što znači da u obzir treba uzeti putanje svih mrava. Dakle, svi bridovi kojima je prošao barem jedan mrav podliježu feromonskom osvježavanju.

Konačno, nakon dovoljnog broja iteracija uzima se rezultantna feromonska matrica. Iz ove matrice generira se slika tako da se prvo odredi feromonski prag. Potom se svaki pixel klasificira u odnosu na prag te shodno tome poprimi crnu ili bijelu boju.



Slika 3.3: Babun



Slika 3.4: Usporedba AS i ACS algoritama pri detekciji bridova za različite vrijednosti q_0: (a) AS, (b) ACS 0.2, (c) ACS 0.4, (d) ACS 0.6

Sa slika se vidi da algoritam ACS bolje detektira bridove. Parametar q_0, kao što je opisano u poglavlju 2.3.1, kontrolira tendenciju mrava da istražuju nove prostore i važno ga je pravilno postaviti na vrijednost iz intervala $(0, 1)$. Veće vrijednosti q_0 (manje istraživanja) su općenito pogodnije za slike sa više detalja.

Postupak ovakvog načina detekcije bridova opisani su mnogo detaljnije u članku čiji su autori Anna Veronica Baterina i Carlos Oppus[1].

4. Problem labirinta

Za praktični programski sustav koji će simulirati koloniju mrava odabrana je domena labirinta. Domena je dvodimenzionalno polje lokacija od kojih svaka lokacija može biti ili zid ili slobodno polje. Od slobodnih polja jedno se bira kao početak, a drugo kao izlaz iz labirinta do kojeg je potrebno pronaći najkraći put.

Naravno, za ovakav tip problema već postoje mnogi algoritamski postupci poput A^* algoritma ili pretraživanja po širini (engl. *breadth-first search*) međutim cilj projekta je dobiti vizualizaciju primjene AS algoritma te usporedbe tih rezultata za domene različitih složenosti.

4.1. Formalizacija problema

Neka su w i h pozitivni cijeli brojevi koji predstavljaju redom širinu i visinu labirinta. Tada preslikavanje:

$$L : [0, w) \times [0, h) \to \{0, 1\}$$
$$w, h \in \mathbb{N}$$

predstavlja funkciju koja nad zadanom domenom gradi labirint. Po dogovoru 0 može predstavljati zid a 1 slobodno polje. Ukupan broj svih mogućih verzija labirinta je prema tome određen izrazom

$$N = 2^{w \cdot h} \tag{4.1}$$

U svrhe potpunosti, dodatno se definiraju i dva preslikavanja koja određuju početno stanje te cilj kao:

$$\begin{aligned} \textit{Početak} &: [0, w) \times [0, h) \to \{\top, \bot\} \\ \textit{Kraj} &: [0, w) \times [0, h) \to \{\top, \bot\} \end{aligned} \tag{4.2}$$

Ova preslikavanja će za točno jednu lokaciju vratiti istinitost, i to ako je ta lokacija početak u prvom slučaju, odnosno kraj u drugom.
Cjelokupan problem se može definirati kao uređena petorka

$$(w, h, L, \textit{Početak}, \textit{Kraj}) \tag{4.3}$$

Od svih mogućih problema zadanih na ovaj način bit će razmatran samo jedan specifični podskup za koji vrijedi ograničenje da izborom bilo koje dvije slobodne točke postoji put kroz labirint koji ih povezuje. Ovo će omogućiti da za svaku generiranu domenu sustav mora moći nakon određenog vremena pronaći rješenje. Jednom kada je problem definiran potrebno je definirati i cilj.

Put P po labirintu od točke A do točke B može se zapisati kao uređeni niz polja po kojima se potrebno kretati:

$$P = (A, p_1, p_2, \ldots, p_n, B) \tag{4.4}$$

uz sljedeće restrikcije

$$\begin{aligned} \textit{Početak}(A) &= \top \\ \textit{Kraj}(B) &= \top \\ ||p_i - p_{i+1}||_1 &= 1 \\ p_i \in [0, w) \times [0, h) &\wedge L(p_i) = 1 \end{aligned} \tag{4.5}$$

Dakle ono što je zahtijevano sa iznad navedenim pravilima je da se put može sastojati samo od slobodnih polja, s tim da prvo polje mora odgovarati početnoj poziciji a posljednje polje konačnom cilju. Također, udaljenost između susjednih polja puta mora biti 1 što efektivno znači da se u svako sljedeće polje iz prethodnog mora doći točno jednim horizontalnim ili vertikalnim pomakom. Tako definiranom putu duljina će biti jednaka broju polja koji ga opisuju

$$D(P) = |P| \tag{4.6}$$

a najkraći put od A do B je stoga:

$$D^* = min_{P_i} D(P_i) \tag{4.7}$$

Kako mravlji algoritmi spadaju u heuristike, cilj algoritma je da nakon određenog broja iteracija pronađe put koji je po duljini vrlo blizu optimalnog D^*.

4.2. Generiranje domene

Postoji nekoliko algoritama za generiranje strukture labirinta. Za ovaj konkretan problem pogodna je jedna varijanta takvog algoritma koja osigurava zahtjev da su svake dvije prohodne točke međusobno povezane nekim putem.

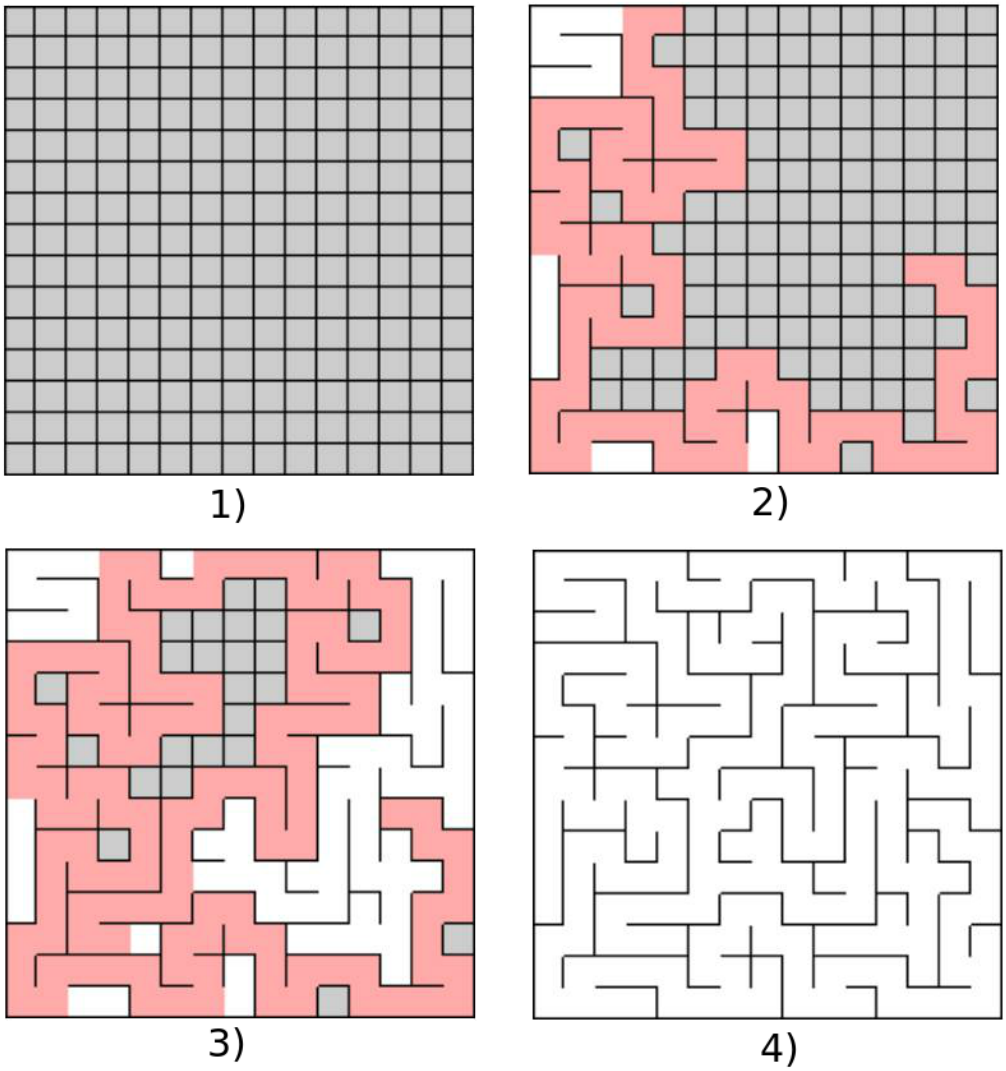
Pseudokod je dan u nastavku.

Algoritam 2

```
1:  procedure GENERATOR LABIRINTA
2:      postavi zidove između svih polja
3:      posjećenePozicije = ∅
4:      stog = ∅
5:      trenutnaPozicija ← nasumičnaPozicija
6:      posjećenePozicije ← trenutnaPozicija
7:      while (postoje neposjećene pozicije) do
8:          if neposjećeniSusjedi(trenutnaPozicija) ≠ ∅ then
9:              novaPozicija ← nasumičniIzbor(neposjećeniSusjedi)
10:             stog ← trenutnaPozicija
11:             ukloniZidIzmeđu(trenutnaPozicija, novaPozicija)
12:             trenutnaPozicija ← novaPozicija
13:             posjećenePozicije ← trenutnaPozicija
14:         else
15:             stog → trenutnaPozicija
```

Algoritam koristi metodu pretrage u dubinu (engl. *depth-first search* a prednost je, osim povezivosti svake dvije slobodne točke, ta što proizvodi labirinte sa niskim faktorom grananja. Osim toga labirint će sadržati mnogo dugačkih hodnika što je također povoljno za simulaciju.

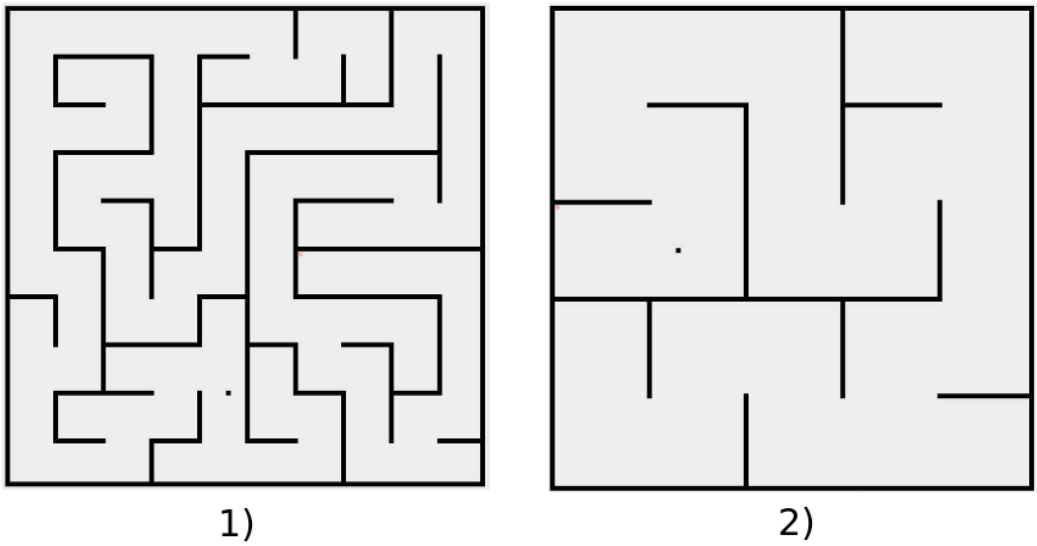
Početno stanje labirinta može se zamisliti kao velika mreža polja (poput šahovske ploče) koja su sa svih strana okružena zidovima. Počevši od nasumično odabranog polja, bira se slučajnim odabirom jedan od neposjećenih susjeda. Zatim se uklanja zid koji ih dijeli te se susjeda dodaje na vrh pomoćnog stoga. Algoritam se potom miče na njegovu poziciju čime se susjed efektivno dodaje u listu posjećenih stanja a cijeli proces se ponavlja iznova. Kada se u nekom trenutku algoritam nađe u slijepoj ulici (polje koje nema neposjećenih susjeda) koristi se stog pomoću kojeg se obavlja povratak u prvo stanje iz kojeg se može dalje raditi grananje.



Slika 4.1: Neki koraci pri generiranju labirinta pretragom po dubini[2]

Slika 4.1 vizualizira proces generiranja labirinta na opisani način. U koraku jedan početno stanje je obična mreža a kroz korake dva i tri prikazuje se napredak. Rozom bojom je označen put kojim se algoritam kretao a čiji elementi su još uvijek na stogu, a bijelom bojom put kojim se algoritam vraćao nakon što je završio u slijepoj ulici. Na kraju svako pokretanje ovakvog algoritma mora završiti na način da se stog potpuno isprazni, pa je stoga u posljednjem koraku potpuno bijela slika. Povratkom u početnu točku labirint je gotov i proces završava.

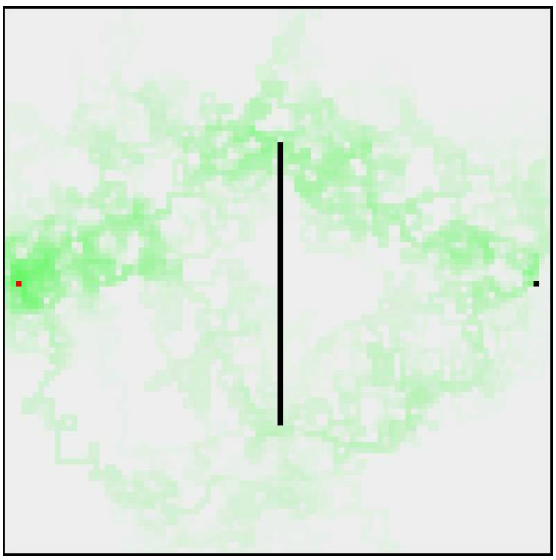
Korisno je u implementaciji imati mogućnost zadavanja složenosti labirinta prije njegova generiranja. Ovo se može jednostavno postići malom izmjenom procedure 2. Algoritam će biti proširen ulaznim parametrom koji određuje razmake između horizontalnih i vertikalnih linija u početnoj mreži. Na ovaj način, što veći parametar pošaljemo to će proizvedeni labirint biti jednostavniji.



Slika 4.2: Dva labirinta različitih složenosti

Ilustracija utjecaja ovog parametra prikazana je na slici 4.2. Lijevi labirint je izgrađen sa parametrom širine iznosa 50 slikovnih elemenata, a desni sa širinom 100 slikovnih elemenata. Očito je desni značajno jednostavniji za riješiti.

Još jedna stvar kojom se domena mora obogatiti je mogućnost pamćenja feromonskih razina koje će biti definirane za svako polje. Implementacijski je ovo riješeno tako da se poljima u strukturu doda atribut koji poprima vrijednost iz $\mathbb{R}^+$. Ako se tako postave stvari, trag se može vizualizirati mijenjanjem boje svakog polja na način da polja sa jačim feromonom ujedno imaju veći intenzitet korištene boje.



Slika 4.3: Intenzitet feromonskog traga na domeni

5. Vizualizacija rješavanja problema labirinta

5.1. Prilagodba AS algoritma

Kako bi se algoritam mogao primijeniti na opisani problem potrebno je izvršiti neke preinake.

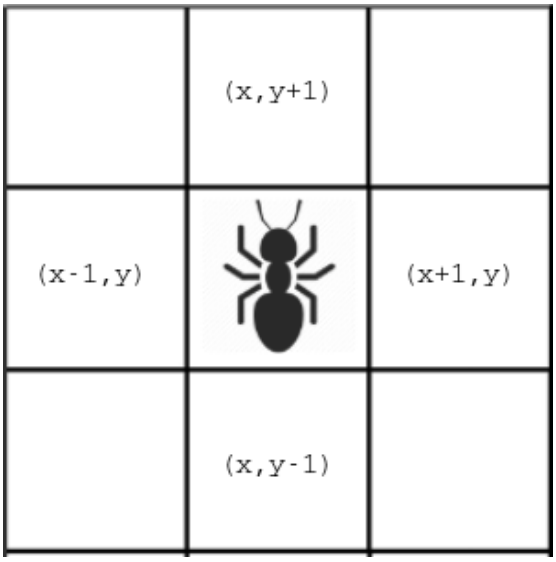
U originalnom AS algoritmu bilo je potrebno pričekati da svi mravi pronađu rješenje prije nego se evaluiraju njihovi putevi i odaberu najbolji. Takav pristup na ovoj domeni nije pogodan. Zbog veličine domene i nasumičnog kretanja mrava u prvoj iteraciji čekanje na pronalazak rješenja trajalo bi predugo.

Stoga se algoritmu kao parametar zadaje broj agenata n koji moraju naći rješenje prije nego što se započne s novom iteracijom. Taj broj bi općenito za što bolju simulaciju trebao biti znatno manji od ukupnog broja agenata.

Osim toga dodan je još jedan parametar m koji određuje koliko će najboljih mrava od ukupno n njih koji su pronašli rješenje dobiti dozvolu za ostavljanje feromonskog traga svojim putem.

Graf po kojem se agenti kreću je pretvoren u dvodimenzijsko polje gdje svaki element predstavlja jedan čvor a feromonske razine koje su u originalnom algoritmu svojstva prijelaza sada postaju svojstvom čvorova. Pokazalo se da ova promjena ne utječe previše na simulaciju algoritma.

Broj agenata se može zadati kao parametar, a za što ljepšu simulaciju je potrebno imati više agenata kako bi na velikoj skali došlo do efekta izranjajuće inteligencije.



Slika 5.1: Agent na polju labirinta

Svaki agent u domeni zna za svoju trenutnu poziciju a vidljiva su mu najviše četiri susjedna polja (kao na slici 5.1).

Izbor jednog od susjednih polja u koje će se pomaknuti vrši po algoritmu 2.2 s tim da je heuristički parametar $\beta = 0$ a kretanje određuju faktor slučajnosti i feromonske razine na susjednim poljima.

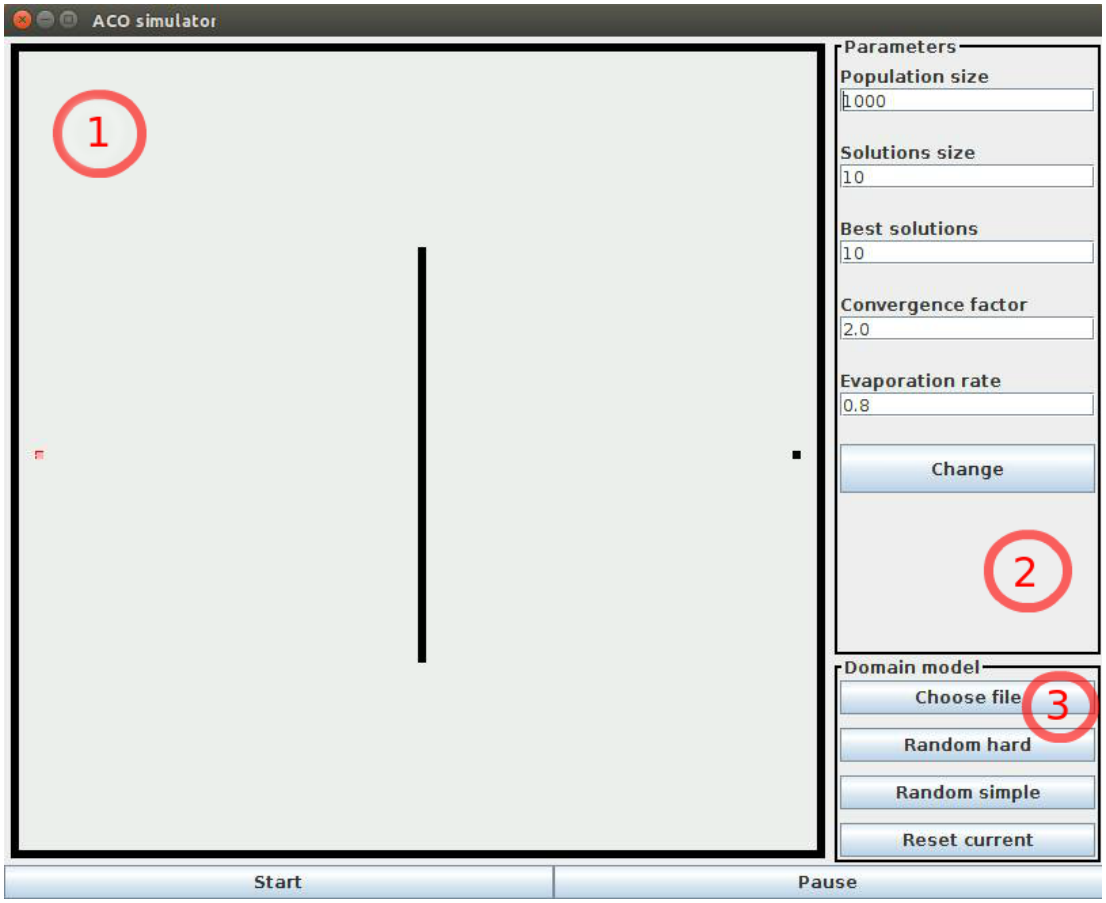
5.2. Programski sustav

Programski sustav predloženog algoritma simulacije ostvaren je u Java SE razvojnom okruženju. Pri tome su za vizualizaciju korištene komponente iz grupe alata koji pripadaju JFC/Swing biblioteci.

Na slici 5.2 prikazan je izgled korisničkog sučelja sa označenim osnovnim grupama naredbi koje omogućuju rad sa simulatorom.

Pogled označen brojem 1 je glavni okvir u kojem se prikazuje odvijanje simulacije u vremenu. Neposredno ispod su smješteni gumbi za pokretanje/zaustavljanje izvođenja koji mogu biti veoma korisni ako se cijeli sustav želi zamrznuti u nekom trenutku kako bi se pobliže analizirala situacija.

U pogledu označenom brojem 2 smještene su osnovne grupe parametara algoritma sa već pretpostavljenim vrijednostima. Parametri će biti objašnjeni redoslijedom pojavljivanja počevši od vrha. Prvi broj određuje veličinu populacije odnosno broj agenata koji će se kretati po domeni. Sljedeći parametar određuje trajanje iteracije. Naime, on označava koliko agenata od početne populacije mora pronaći rješenje prije nego li



Slika 5.2: Izgled korisničkog sučelja

se iteracija prekine, a pronađena rješenja evaluiraju. Manji broj znači da će iteracije kraće trajati. Treći parametar je usko povezan sa prethodnim, a predstavlja broj najboljih rješenja po čijim će se putevima ostavljati feromonski trag. Konkretno se na slici može vidjeti da je taj broj jednak broju traženih rješenja zbog čega će se svaki pronađeni put uzeti u obzir. Preostaju još faktor konvergencije i brzina isparavanja feromona. Ovi parametri nisu ništa drugo nego već spomenuti α i ρ iz tablice 2.1. Važno je napomenuti da se parametri mogu dinamički mijenjati za vrijeme same simulacije što je od koristi jer se može promatrati utjecaj njihovih vrijednosti na ponašanje agenata. Nakon svake izmjene potrebno je potvrditi vrijednosti klikom na gumb u toj grupi.

Posljednja, treća grupa, nudi korisniku mogućnost generiranja nasumičnih labirinata različite složenosti. Osim toga omogućeno je i učitavanje definicije labirinta opisanog jednostavnom datotekom na disku koja u prvih 5 redova mora sadržati početne postavke parametara iz grupe označene brojem 2.

Nakon parametara slijedi n linija širine m stupaca gdje svaki znak može biti jedan od:

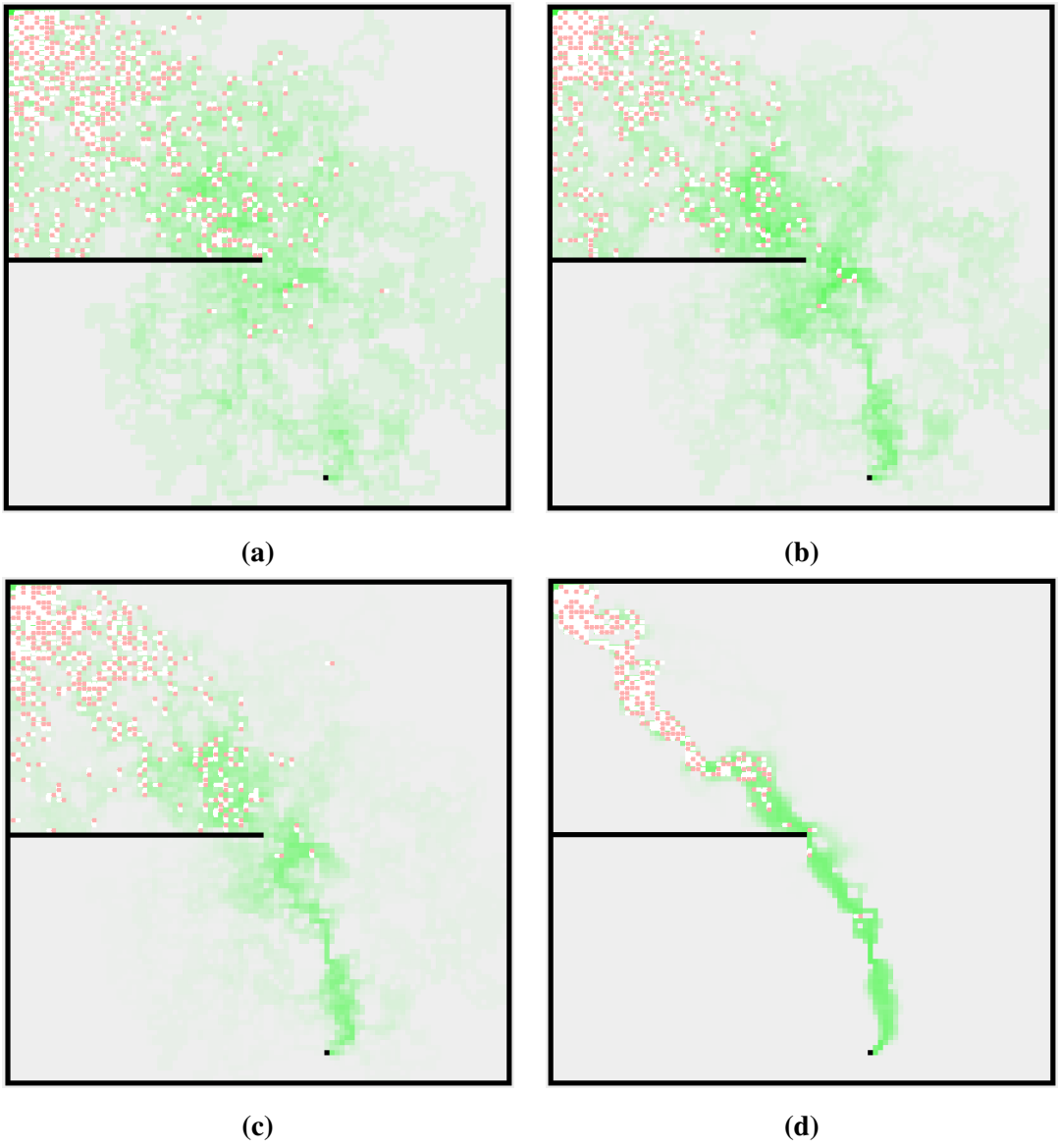
- '#' - označava zid
- ' ' - razmak označava prohodno polje
- 'f' - označava rješenje labirinta koje je potrebno pronaći
- 'n' - početno mjesto gdje će se populacija generirati u svakoj iteraciji

```
1000
10
10
2.0
0.8
#################
#         #       #
#    #    #   f    #
#    #             #
#    ####   ###    #
#                   #
#  n         #      #
#    #######        #
#                   #
#################
```

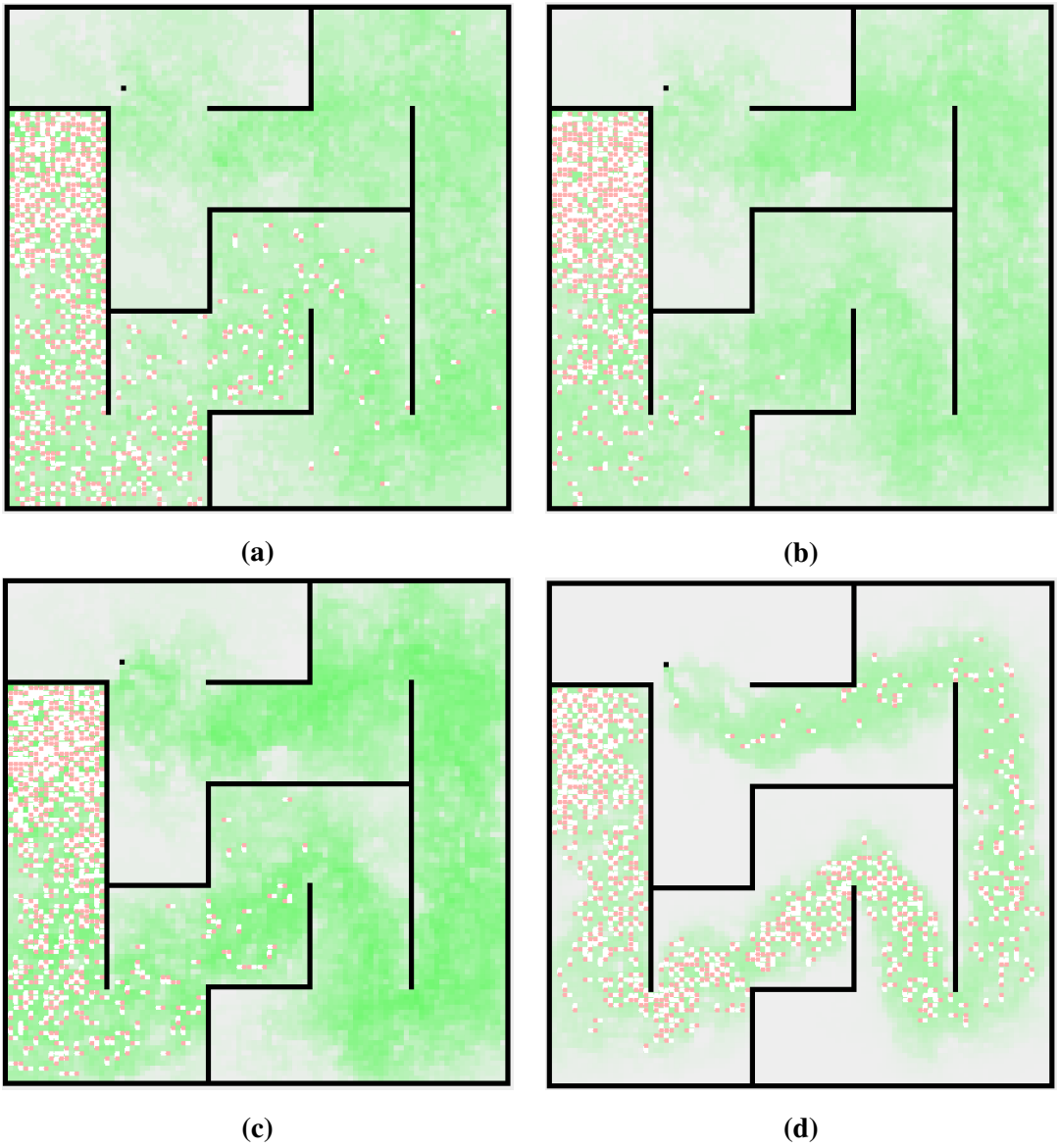
Listing 5.1: Primjer jednostavnog ASCII labirinta

5.3. Rezultati simulacije

Sam simulacijski sustav ispitan ja na više različitih labirinata. Dvije takve simulacije su prikazane kroz nekoliko koraka. Na slikama se može primijetiti kako u vremenu isparavaju manje efikasni putevi dok preživljava samo najkraći.

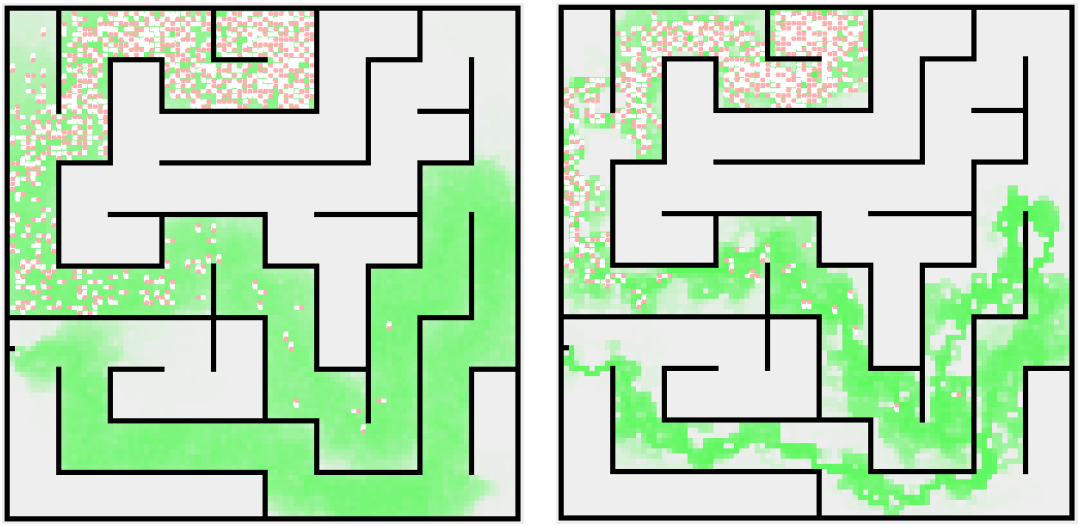


Slika 5.3: Simulacija ACO algoritma pri rješavanju jednostavne prepreke



Slika 5.4: Simulacija ACO algoritma pri rješavanju labirinta

Osim ovih primjera na slikama ispod je prikazan simulacija na još složenijem labirintu i to sa različitim parametrima.



(a) **(b)**

Slika 5.5: Simulacija ACO algoritma pri rješavanju složenijeg labirinta

Parametri korišteni u simulacijama A i B prikazani su tablicom 5.1.

Tablica 5.1: Parametri za simulacije

Parametar	**Simulacija A**	**Simulacija B**
Populacija	1000	1000
Broj rješenja	20	20
Najboljih n	10	3
Faktor konvergencije	2.0	4.0
Faktor isparavanja	0.8	0.8

U simulaciji B se povećao elitizam na način da se od 20 rješenja uzimaju 3 najbolja, dok se u simulaciji A uzima čak 10 najboljih. Faktor konvergencije α je povećan sa 2.0 na 4.0 a ostali parametri ostaju isti.

Na slici 5.5 put je mnogo uži a razlog je da se povećanjem konvergencije prvi izračunati putevi u svakoj sljedećoj iteraciji više preferiraju.

6. Zaključak

Prirodom inspirirani algoritmi predstavljaju područje sa velikim potencijalom koje još uvijek nije dovoljno istraženo. Porastom snage suvremenih računala ovakvi algoritmi dolaze do izražaja i pronalaze primjenu u najraznovrsnijim problemima. Jedan takav algoritam je i optimizacija kolonijom mrava.

Može se zaključiti da se mravlji algoritmi mogu primijeniti na zaista širok spektar problema, a kroz ovaj rad je analizirano ponašanje na problemu labirinta. Pokazalo se da za pronalazak najkraćeg puta nisu efikasni. Konvergencija je spora, a problem predstavlja nasumičnost domene te nedostatak smislene heuristike.

Moguće daljnje istraživanje svakako podrazumijeva pokušaj modeliranja heuristike koja bi na neki način mogla usmjeriti pretragu. Dodatno se algoritam može poboljšati paralelizacijom s obzirom da su agenti međusobno neovisni.

Unatoč svemu, simulacija omogućava lijep prikaz načina na koji algoritam funkcionira, a na posebno odabranim primjerima se može uočiti izranjajuća inteligencija.

LITERATURA

[1] A.V. Baterina i C. Oppus. Image edge detection using ant colony optimization. 2010.

[2] Jamis Buck. Maze generation: Recursive backtracking, 2010.

[3] Leslie Pérez Cáceres, Manuel López Ibáñez, i Thomas Stützle. Ant colony optimization on a limited budget of evaluations: Supplementary material. 2015.

[4] doc. dr. sc. Marko Čupić. *Prirodom inspirirani optimizacijski algoritmi. Metaheuristike.* 2013.

[5] M. Dorigo i L.M. Gambardella. Ant-q: A reinforcement learning approach to the traveling salesman problem. 1995.

[6] Marco Dorigo. *Optimizacija, učenje i prirododom inspirirani algoritmi*. Doktorska disertacija, Politehnički fakultet, Milano, 1992.

[7] Marco Dorigo. Ant colony optimization. 2007.

[8] S. Goss, S. Aron, J. L. Deneubourg, i J. M. Pasteels. Self-organized shortcuts in the argentine ant. 1989.

[9] P. Held i R. Kruse. Texts in computer science, 2013.

[10] Mehmet Karatay. Kolonija safari mrava uslikana u mount kenya, 2007.

[11] T. Stützle i H.H. Hoos. Max-min ant system. 2000.

[12] German Wikipedia, 2005.

Simulacija i vizualizacija umjetne kolonije mrava prilikom rješavanja problema labirinta

Sažetak

U ovom radu je detaljno opisan Ant System algoritam a površno su obrađeni Ant Colony System te MAX-MIN varijanta kao podvrste mravljih algoritama. Navedene su neke primjene heuristike u rješavanju stvarnih problema. Osim toga prikazani su rezultati simulacije algoritma pri rješavanju konkretnog problema labirinta.

Ključne riječi: prirodom inspirirani algoritmi, mravlji algoritmi, heuristike, NP-teški problemi, labirint, ACO

Simulation and visualization artificial ant colony during solving of maze

Abstract

This thesis provides detailed explanation of Ant System algorithm and less detailed cover of it's subgroups: Ant Colony Optimization and MAX-MIN Ant System. It also provides insight into few real world problems solvable with this heuristic. Finally, system is tested on labyrinth domain with results shown in last chapter.

Keywords: Nature-Inspired Optimization Algorithms, ant algorithms, heuristics, NP-hard problems, labyrinth, Ant Colony Optimization