Optimizacija kolonijom mrava za kombinatorne probleme

Janković, Stjepan

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Stjepan Janković

Optimizacija kolonijom mrava za kombinatorne probleme

ZAVRŠNI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Stjepan Janković

Matični broj: 46011/17–R

Studij: Informacijski sustavi

Optimizacija kolonijom mrava za kombinatorne probleme

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Ivković Nikola

Varaždin, rujan 2021.

Stjepan Janković

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema ovog rada je optimizacija kolonijom mrava (eng. Ant Colony Optimization – ACO) i njezina primjena na kombinatorne probleme. Na temelju općih smjernica napravljen je algoritam ACO za rješavanje problema paralelne dostave dronovima i trgovačkog putnika (eng. Parallel Drone Scheduling Traveling Salesman Problem – PDSTSP)

Cilj rada je saznati kakav efekt imaju različite instance, broj iteracija i mrava na ponašanje algoritma optimizacije kolonijom mrava. Rad algoritma testiran je na primjercima PDSTSP-a veličine od 20 do 100 lokacija koje su generirane za potrebe ovog istraživanja. Algoritam je implementiran u jeziku C#, a za razvoj je korišten Visual Studio 2019. Rezultati eksperimenata prikazani su tablično i grafički. Razvijen algoritam pokazao se primjerenim za rješavanje PDSTP-a. Programsko rješenje unutar kojeg je zapisan algoritam će biti objašnjeno u odjeljku 4.

Unutar eksperimenata možemo vidjeti razliku između povećanja broja mrava i povećavanja broja iteracija, njihov efekt na dobiveno rješenje i rad algoritma. Možemo vidjeti da se povećava trajanje algoritma unutar programskog rješenja, no uz to se poboljšava i rezultat. Uz sve veće povećavanje mrava i iteracija trajanje algoritma postaje sve veće, ali se dobitak u poboljšanom rezultatu sve više smanjuje.

Ključne riječi: algoritam, algoritam kolonije mrava, optimizacija, inteligencija rojeva, računalna inteligencija

Sadržaj

1. Uvod

Vrijeme je novac (*eng. Time is Money* [1]) je već poznati izraz korišten stoljećima i on nikad nije bio ispravniji nego u našem modernom svijetu. Svaka se osoba u današnje vrijeme koristi online trgovinom od roditelja koji naručuje potrepštine svoje djece za školu do velikih poduzeća koja se bave prodajom ogromnog asortimana robe. Sama eksplozija E-trgovine odnosno online trgovine se prijašnjih godina dogodila zbog dostupnosti i jednostavnosti online naručivanja, dok u zadnjim godinama se događa još veća eksplozija zbog potrebe za izolacijom.

Uz potražnju koja samo raste mnogim poduzećima je potreban bolji sistem dostave kako bi mogli maksimizirati dobit i minimizirali gubitak, kako bilo na uštedi kroz kraće trajanje dostave do povećanja potražnje kupca zbog brze usluge. Pregledom rezultata surveja od DALLAS--(BUSINESS WIRE)--Omnitracs, LLC, a Solera company iz 2021 godine [2] vidimo da 53% generalnih kupaca unutar USA se koristi online trgovinom. Od njih 65% bi platilo više za bržu dostavu i 30% bi platilo više za dostavu istog dana narudžbe.

Kako bi poduzeća mogla iskoristiti ovu mogućnost za povećanje svoje zarade mogu koristiti različite načine poput korištenja različitih algoritama za potražnju najboljeg puta dostave ili korištenja različitih vozila za samu dostavu, kao naprimjer dronove. Mnoga poduzeća već koriste dronove za dostavu kao što su Amazon [3] ili FedEx [4].

Ovaj problem je u radu prikazan kao problem paralelne dostave dronovima i trgovačkog putnika (*eng. Parallel Drone Scheduling Traveling Salesman Problem – PDSTSP*). Postoje različiti modeli i algoritmi korišteni za rješavanje takvog problema. Unutar rada je korišten MMAS sistem kolonije mrava uz pomoć dronova kako bi se riješio problem, odnosno pronašao što bolji put za dostavu.

Na početku rada je generalno opisan problem trgovačkog putnika, algoritam optimizacije kolonijom mrava i njegove vrste. Nakon toga su opisane instance i način rada programsko koda pomoću kojeg se rješava problem. Na kraju se nalaze rezultati sa zaključkom unutar kojeg se objašnjava utjecaj različitih broja iteracija i mravi na određenu instancu.

2. Optimizacija kolonijom mrava

Mravi kao grupa insekata postoji već milijunima godinama. Oni se gledaju kao socijalni kukci; provode većinu svojeg vremena u kolonijama, imaju sistem padajućeg autoriteta i sposobni su ponovno izgraditi svoje kolonije iz malih populacija, u usporedbi sa općenitom veličinom jedne kolonije. Mravi nisu jedina vrsta kukca koja živi u kolonijama, ali oni su jedni od glavnih razloga za nastajanje algoritma inteligencije roja (eng. Swarm Intelligence) i kreiranja algoritama kolonije mrava (eng. Ant Colony Algorithms). Kao što je Andries P. Engelbrecht napisao u svojoj knjizi „Computational Intelligence“ [5] mravi i njihovo ponašanje se istraživa već stoljećima.

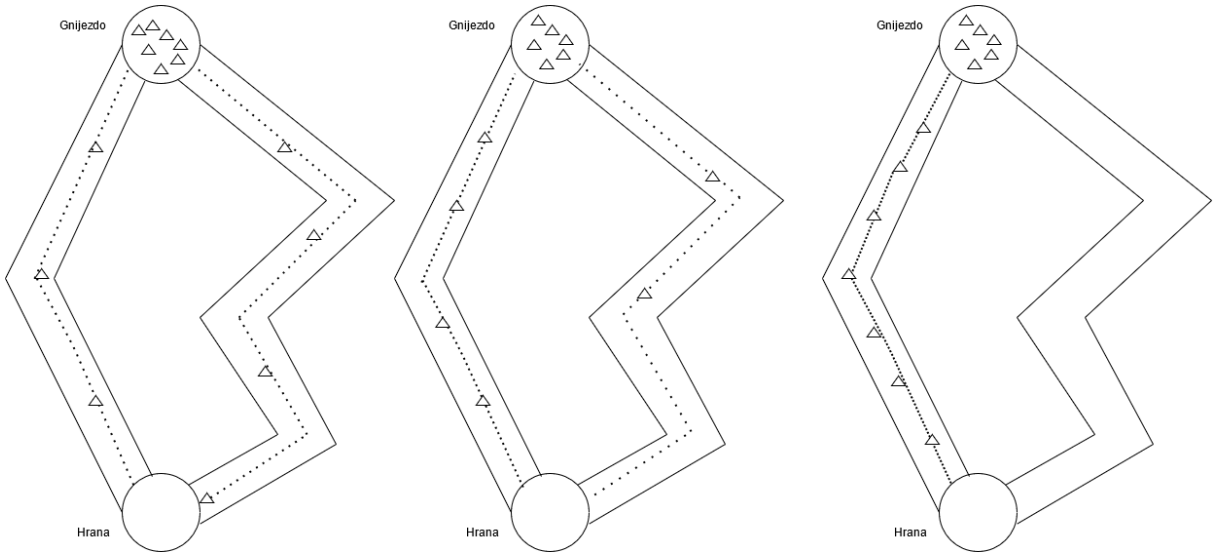
Tijekom njihovog vremena na ovom planetu, mravi su usavršiti sistem kreiranja najboljeg puta do svog odredišta. Sve je to od zasluge njihovim feromonima koje ostavlja svaki mrav na svom putu do odredišta. Kada mrav napusti svoje gnijezdo u istraživanje okoline, on ostavlja feromone na svojem putu. U slučaju da mrav naiđe na nešto što bi bilo pozitivno za koloniju, kao naprimjer hranu, on se vraća istim putem kojim je došao do te hrane i šalje signal ostalim mravima da je pronađena hrana. Nakon toga mravi prate feromone koje je ostavio bazični mrav i polako skupljaju hranu koji nose nazad u gnijezdo.

Da je to sve što mravi rade ne bih imali koristi od njihovog ponašanja. Naime, mravi ne prate feromone 1 za 1, već imaju odskakanja u njihovom putu. Neka od tih odskakanja rezultiraju pronalasku kraćeg puta, dok neka rezultiraju pronalasku dužeg puta. Kroz takva odskakanja mravi pronalazi sve brže i brže puteve do hrane. Glavni razlog za to su feromoni koje ostavlja svaki mrav, i njihovo isparavanje kroz vrijeme. Što je put duži, ima više vremena da se jačina njegovih feromona otopi. Zbog toga se svi putevi na koji su duži polako nestaju iz korištenja jer njihovi feromoni su se otopili prije kraćih puteva koje koriste sve više i više mravi.

Iz takvog ponašanja mrava su nastali algoritmi čiji je cilj optimizacija različitih dijelova našeg svakodnevnog života. Od jednostavne raspodjele rada među ljudima do raspodjele cesta, ulica i kuća za što bolje moguće rješenje. Ovo poglavlju će objašnjavati rad mrava, nastajanje algoritma i njegovih vrsta.

2.1. Ponašanje mrava

Ponašanje mrava možemo najlakše prikazati preko primjera sa dva puta. Ovakav primjer je prikazan u većini radova kojima je tema, ili dio teme algoritmi kolonijom mrava kao što možemo vidjeti u knjizi „Introduction to Evolutionary algorithms“ od Mitsuo Gen i Xinjie Yu [6]. Postoje dva puta za mrave od gnijezda do hrane. Nijedan mrav nije još prošao nijedan put, zbog toga nema feromona i vjerojatnost mrava da odabere jedan od dva puta je 50%. Nakon prolaska mravi ostavljaju svoje feromone kao što je na prvom prikazu slike 1. Sada na snagu stupa isparavanje feromona. Duži put polako gubi feromone, dok kraći put ih dobiva. Razlog za to je broj mravi koji prolazi putem nakon isparavanja.



Slika 1: Primjer ponašanja mrava (Izvor: Mitsuo Gen i Xinjie Yu (2010))

Zbog toga što na dužem putu treba više vremena da mrav prođe cijeli put i sa tim put provodi više vremena gubeći svoje feromone, manje mravi prolazi tim putem i odabire put sa više feromona. Možemo ovo vidjeti na drugom prikazu slike 1. Ovaj proces se ponavlja sve dok, u savršenom svijetu, svi mravi ne počnu koristi kraći put.

U realnom svijetu će još uvijek biti takozvanih lutalica (*eng. Straglers*) koji će odabrati duži put. Pojam vremena je isto bitan. U slučaju da postoji veći broj puteva, postoji mogućnost da će sva hrana već biti prebačena unutar gnijezda dok se svi mravi ne odluče preseliti na kraću put. Uz to je još moguće da će sve hrana biti prebačena unutar gnijezda prije nego je uopće pronađen apsolutno najkraći put.

Sam prikaz ponašanja mrava možemo zapisati u pseudo kodu, kao što je to već napravio Andries P. Engelbrecht napisao u svojoj knjizi „Computational Intelligence“ [5] u Algoritmu 17.1.

for svaki mogući put **{**

Izračunaj trenutne feromone na putu;

If nasumično odabrani broj se podudara sa izračunatom vrijednosti {

Prati odabrani put;

Break;

} }

Ovakav algoritam mrav obrađuje u glavi svaki put kada mora donesti nekakvu odluku. Prateći ponašanje mrava, M. Dorigo je u svojem znanstvenom radu [7][8] uspio kreirati algoritam poznat kao sustav mrava (*eng. Ant System – AS*) koja se i danas koristi. Algoritam za promjenu feromona:

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

Gdje τ_{xy} predstavlja količinu feromona na određenom putu xy, ρ predstavlja koeficijent isparavanja feromona, m je broj mravi koji se koristi i $\Delta\tau_{xy}^{k}$ je vrijednost feromona koje ostavlja k mrav na svojem putu. Ovakav algoritam se kasnije počeo koristiti i na raznim drugim problemima od kojih je najpoznatiji problem trgovačkog putnika. Iz AS je nastalo mnogo drugih algoritama sa različitim načinima rješavanja problema. Spomenuti algoritmi pripadaju skupini algoritama znani kao; optimizacija kolonijom mrava (*eng. Ant Colony Optimization - ACO*).

2.2. MMAS (Max Min) optimizacija kolonijom mrava

MMAS algoritam su kreirali Thomas Stützle i Holger H Hoos pokušavajući poboljšati rad AS algoritma dodavajući Maksimum i Minimum na feromone, odnosno ograditi feromone u rasponu Max – Min, kako bi algoritam bolje iskorištavao najbolje rješenje svake iteracije. Prema njihovom radu (*eng. MAX-MIN Ant System* [9]) MMAS rješava zadane probleme dosljednije od AS algoritma, odnosno MMAS je u više slučajeva došao do optimalnog rješenja nego AS algoritam.

MMAS algoritam radi na drugačijem principu od AS algoritma. Unutar AS algoritma, svaki mrav ostavlja svoj feromonski trag koji onda čitaju ostali mravi, dok unutar MMAS

algoritma, feromonski trag ostavlja samo mrav koji je pronašao najbolje rješenje te iteracije. Raspon vrijednosti koje feromon može postati je zadan uz pomoć vrijednosti rho i vrijednosti optimalnog rješenja:

$$\frac{1}{1-\rho} * \frac{1}{f(s^{opt})}$$

Iznad je zapisana formula koja nam određuje najveću moguću feromonsku vrijednost. Ovo se postiže korištenjem vrijednošću optimalnog rješenja $f(s^{opt})$. Minimalnu vrijednost je najbolje odrediti u usporedbi sa maksimalnom vrijednošću kako bi se dobio optimalan raspon feromona. Glavni razlog za ovakav rad algoritma su ekstremne vrijednosti feromona koje mogu dovesti do stagnacije (eng. Stagnation) zbog koje postoji mogućnost ignoriranja najboljeg puta. Novija varijanta algoritma ACO, poznata pod nazivom sustav mrava s trima granicama (eng. Three Bound Ant System, TBAS) donosi dodatna poboljšanja u odnosu na MMAS kao što su brže izvođenje i općenitiji način upravljanja feromonima. [12]

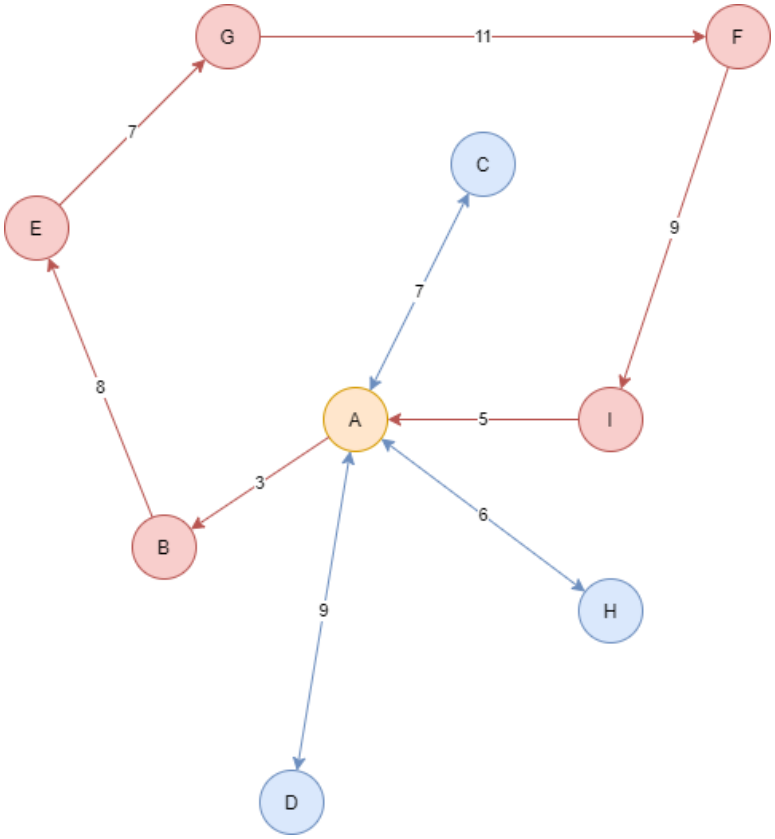
3. Problem paralelne dostave dronovima i trgovačkog putnika

Problem trgovačkog putnika (kraće TSP). je već daleko poznat problem koji se koristi za testiranje raznih optimizacijskih algoritama. Od velike je važnosti i za istraživanje ACO, jer se prva verzija ACO koristila za rješavanje TSP.[8] Uz to je TSP optimizacijski problem NP-težine. Preciznija specifikacija problema NP-težine je: problem H je NP-težak kada se svaki problem L u NP može svesti u polinomskom vremenu na H; to jest, pod pretpostavkom da rješenje za H traje 1 jedinicu vremena, H-ovo rješenje može se koristiti za rješavanje L u polinomskom vremenu. [10]

Sam TSP se može opisati kao problem pronalaska najboljeg puta između zadane liste gradova i njihove udaljenosti. Korištenjem ACO možemo doći do optimalnog rješenje i kraćem vremenu nego mnogi drugi algoritmi. Problem predstavljen unutar ovog rada je problem paralelne dostave dronovima i trgovačkog putnika (*eng. Parallel Drone Scheduling Traveling Salesman Problem – PDSTSP*). PDSTSP je sličan TSP sa tim da umjesto jednog kamiona imamo kamion i dron koji paralelno dostavljaju pakete na odredišta. Dok kamion se kraća od grada do grada, dron leti iz baze do svojeg odredišta i vraća se nazad u bazu gdje odabire sljedeće odredište i ponavlja postupak.

Problemi u ovom radu su zadani u formatu TSPLIB i biti će rješavani MMAS algoritmom. Za potrebe eksperimentalnog istraživanja generirano je 5 instanci paralelne dostave dronovima i trgovačkog putnika s brojem gradova 20, 40, 60, 80 i 100. Sve instance unutar rada su kreirani u TSPLIB[11] formatu.

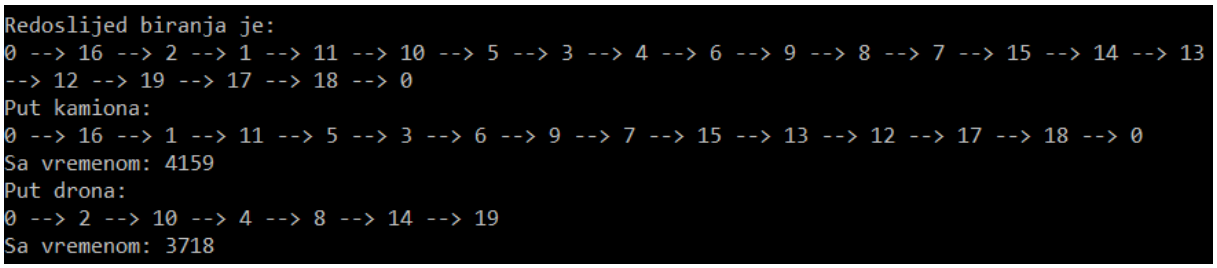
Svaka instanca je kreirana uz pomoć nasumičnog generatora brojeva, pomoću kojeg su generirane koordinate X i Y za gradove. Koordinate X i Y su odabrane u rasponu 0 – 1000. Koordinate X i Y od gradova iz instanci se pretvaraju u točke unutar dvodimenzionalne mape uz pomoć kojih računamo udaljenosti između gradova. Na slici 2 se nalazi primjer rada preko kojega će biti objašnjeno biranje puta.



Slika 2: Primjer rada

Na slici 2 možemo vidjeti primjer sa 9 gradova. Grad „A“ se tretira kao baza iz koje počinje put za dron i kamion. Kamion je birao put A > B > E > G > F > I > A iz kojih vrijeme puta bilo 3 > 8 > 7 > 11 > 9 > 5 sa ukupnim vremenom 43. Dron je birao put A > D > A > C > A > H > A iz kojih je vrijeme puta bilo 9 > 9 > 7 > 7 > 6 > 6 sa ukupnim vremenom 44.

Kao što je bilo prije spomenuto, dron putuje nazad u bazu nakon svakog odabranog grada i zbog toga se njegovo vrijeme putovanja dvaput broji. Nakon odabira puta se uspoređuje vrijeme puta kamiona i vrijeme puta drona kako bi se odabrao veći od njih. U ovom slučaju to je put drona sa ukupnim vremenom 44 i on se postavlja kao ukupno trajanje iteracije koje se koristi u kasnijem rješavanju. Na slici 3 se nalazi primjer jednog prikaza rješenja direktno iz programskog koda.



Slika 3: Primjer jednog prikaza rješenja

Na slici možemo vidjeti prikaz jednog rješenja. Kamion i dron polaze iz bazičnog grada „0“. Kroz računanje vjerojatnosti puteva kamiona i odabira nasumičnog broja je program odlučio da kamion kreće na grad „15“, nakon toga je program, kroz računanje vjerojatnosti puteva drona odlučio da dron kreće u grad „12“. Ponavlja se odabir gradova, kamion je za sljedeću destinaciju odabrao grad „11“, dok dron ne putuje u drugi grad već se vraća nazad u bazični grad „0“.

Možemo vidjeti vrijeme koje je potrebno kamionu, 4159, i dronu, 3718, da prođu odabrane gradove. Njihova vremena se uspoređuju kako bi se odabrao veći od njih jer oni istovremeno prolaze gradove. U ovom slučaju je 4159 veće vrijeme i ono se koristi u daljnjem radu programskog koda.

4. Algoritam MMAS za problem paralelne dostave dronovima i trgovačkog putnika

Problem dostave proizvoda je prikazan kao problem paralelne dostave dronovima i trgovačkog putnika (*eng. Parallel Drone Scheduling Traveling Salesman Problem – PDSTSP*) koji se rješava uz pomoć MMAS (*eng. Max-Min ant system*) algoritma kolonije mrava. Unutar algoritma se koristi 1 kamion i 1 dron zbog jednostavnosti. Koristi se MMAS algoritam kolonije mrava kako bi simulirali različiti broj agenata koji nasumično putuju iz grada u grada tražeći put koji obilazi sve gradove prateći tablicu feromona (*eng. pheromones*). Kako bi prikazali razliku između puteva drona i kamiona, put kamiona se množi sa 130% dok se put drona množi sa 80%. Unutar programskog koda se udaljenost drona množi sa 2 kako bi simulirali putovanje drona od baze do grada i nazad u bazu. Prikazani kod predstavlja računanje vremena putovanja između gradova i dodjeljivanje feromona za kamion korištenjem formule $\sqrt{(X_a - X_b) * (X_a - X_b) + (Y_a - Y_b) * (Y_a - Y_B)} * 1.3$:

```
for (int i = 0; i < x.Count; i++) {
 for (int j = 0; j < y.Count; j++) {
  if (i != j) {
   kamion[i, j] = (int)(Math.Sqrt(((x[i] - x[j]) * (x[i] - x[j])) +
((y[i] - y[j]) * (y[i] - y[j]))) * 1.3);
   pheromone[i, j] = gornjaGranica;
  }else{
   kamion[i, j] = 0;
   pheromone[i, j] = 0;
  }}}
```

Vremena putovanja između gradova i dodjeljivanje feromona za dron se računaju na isti način kroz formulu $\sqrt{(X_a - X_b) * (X_a - X_b) + (Y_a - Y_b) * (Y_a - Y_B)} * 0.8$. Vjerojatnost odabira sljedećeg grada se računa pomoću formule $(\tau_j^{i^\alpha} * \eta_j^{i^\beta}) / (\sum(\tau_j^{i^\alpha} * \eta_j^{i^\beta}))$. Zbog jednostavnosti dijelimo feromone sa 1 umjesto zbroja svih feromona.

```
for (int i = 0; i < Math.Sqrt(pheromone.Length); i++) {
  if (pheromone[pozicija, i] != 0 && !putovanje.Contains(i)) {
   probability[i] = (Math.Pow(pheromone[pozicija, i], alpha) *
Math.Pow(1 / (kamion[pozicija, i]), beta)) / 1;
  }}
```

Vjerojatnost pretvaramo u roulette raspona 0 – najveće vrijednosti vjerojatnosti kako bi kamion mogao odabrati put iz nasumično odabranog broja.

```
int zastavica = 0;
 for (int i = 0; i < probability.Length; i++) {
  if (probability[i] != 0) {
   double Zbroj = 0;
   for (int j = zastavica; j < probability.Length; j++) {
    Zbroj += probability[j];
   }
   probability[i] = Zbroj;
  }
  zastavica++; }
```

Odabire se nasumičan broj u rasponu 0-1 koji se množi sa najvećom vjerojatnosti, jer nismo dijelili vjerojatnosti sa ukupnom zbrojem vjerojatnosti. Kroz dobiveni broj se odabire sljedeća pozicija kamiona.

```
for (int i = 0; i < probability.Length; i++) {
 if (probability[i] != 0) {
  randomNum *= probability[i];
  break;
 }}
for (int i = 0; i < probability.Length; i++) {
 if (probability[i] != 0 && randomNum <= probability[i] && randomNum >
probability[i + 1]) {
  return i;
 }}
return 0;
```

Računanje vjerojatnosti za dron se računa na isti način, osim što se umjesto varijabli za kamion koriste varijable za dron i prije određivanja sljedeće pozicije se provjerava jeli dron trenutno u bazi ili na putu. Nakon što su kamion i dron poslani na sljedeću poziciju vjerojatnosti za oboje se resetira.

Kod se ponavlja u do-while petlji sve dok agent ne prođe sve gradove, nakon čega se vraća nazad u bazu, odnosno početni grad. Nakon što petlja prođe kroz sve gradove se uspoređuje trenutno najbolji put ove iteracije sa putem ovog agenta. Petlja za određivanje puta trenutnog agenta i biranje najboljeg puta se ponavlja za svakog novog agenta u drugoj do-while petlji

koja se ponavlja sve dok se ne prođe cijela populacija agenata. Kada su svi agenti proši sve gradove se određuje novi maksimum za feromone:

```
gornjaGranica = (1 / (1 - rho)) * (1 / minTotalCost);
donjaGranica = gornjaGranica / maxMinRation;
```

Feromoni isparavaju kroz formulu $\tau_j^i = (1-\rho) * \tau_j^i$ gdje ρ predstavlja stupanja isparavanja feromona i τ_j^i predstavlja feromone τ između gradova „i“ i „j“. Nakon isparavanja se provjerava jeli vrijednost feromona prešla granicu tijekom čega se, u slučaju da je, feromoni resetiraju na zadanu granicu. Isparavanje feromona:

```
for (int i = 0; i < Math.Sqrt(pheromone.Length); i++) {
 for (int j = 0; j < Math.Sqrt(pheromone.Length); j++) {
  if (i != j) {
   pheromone[i, j] = (1 - rho) * pheromone[i, j];
   if (pheromone[i, j] > gornjaGranica) {
    pheromone[i, j] = gornjaGranica;
  }else if (pheromone[i, j] < donjaGranica) {
    pheromone[i, j] = donjaGranica; }}}}
```

Nakon Isparavanja se pregledava odabrani najbolji put i samo za taj put se povećavaju feromoni kroz formulu $\tau_j^i = \tau_j^i + \frac{1}{f(s^{best})}$ gdje $f(s^{best})$ predstavlja najbolje vrijeme da se poslužе svi klijenti. Nakon računanja novih vrijednosti feromona se provjerava jeli vrijednost feromona prešla granicu tijekom čega se, u slučaju da je, feromoni resetiraju na zadanu granicu. Povećavanje feromona:

```
for (int i = 0; i < path.Count; i++) {
 if (i != path.Count - 1) {
  pheromone[path[i], path[i + 1]] += (1 / min);
  if (pheromone[path[i], path[i + 1]] > gornjaGranica ||
pheromone[path[i + 1], path[i]] > gornjaGranica) {
   pheromone[path[i], path[i + 1]] = gornjaGranica;
```

Na kraju, nakon što su svi agenti prošli sve gradove i feromoni su se otopili i povećali za najbolji put je prošla jedna iteracija MMAS algoritma. Algoritam se ponavlja sve dok nije dostignut zadani broj iteracija.

4.1. Pseudokod programskog rješenja

Ovdje opisan cijeli programski kod koji se koristio unutar programskog okružja Visual Studio 2019. Programski kod je zapisan kao pseudo kod zbog lakšeg razumijevanja. Opisane su sve funkcije koje se koriste unutar programskog koda.

```
do{
 do{
  do{
   for (i = 0 do korijena od dužine matrice feromona za kamion) {
    izračunaj vjerojatnost;
   }
   for (i = 0 do dužine niza vjerojatnosti kamiona) {
    izračunaj vjerojatnost kamiona;
   }
   randomNum = nasumično odabrani decimalni broj u rasponu 0 – 1 * najveća vjerojatnost kamiona;
   for (i = 0 do dužine niza vjerojatnosti kamiona) {
    odredi sljedeću poziciju kamiona;
   }
   ukupna udaljenost kamiona += udaljenost kamiona između trenutnog i sljedećeg grada;
   for (i = 0 do dužine matrice feromona za dron) {
    izračunaj vjerojatnost;
   }
   for (i = 0 do dužine niza vjerojatnosti drona) {
    izračunaj vjerojatnost drona;
   }
   randomNum = nasumično odabrani decimalni broj u rasponu 0 – 1 * najveća vjerojatnost drona;
   if (dron se nalazi u bazi) {
    for (i = 0 do dužine niza vjerojatnosti drona) {
     odredi sljedeću poziciju drona;
    } } else if (dron je poslan) {
    vrati dron nazad u bazu;
   }
   for (i = 0 do dužine niza vjerojatnosti kamiona) {
    resetiraj vrijednost vjerojatnosti kamiona [ i ] na 0;
```

```
  }
  for (i = 0 do dužine niza vjerojatnosti drona) {
   resetiraj vrijednost vjerojatnosti drona [ i ] na 0;
  } } while (Mrav nije prošao svaki grad)
 ukupna udaljenost = Max (ukupna udaljenost kamiona, ukupna udaljenost drona);
 if (ukupna udaljenost trenutnog mrava < najmanja udaljenost ove populacije mrava) {
  najmanja udaljenost = trenutna udaljenost;
 }
} while (Svi mravi nisu prošli sve gradove)
Gornja granica = (1 / (1 - rho)) * (1 / minTotalCost);
Donja granica = gornja granica / maxMinRatio;
for (i = 0 do dužine matrice feromona za kamion) {
 otopi feromone za kamion;
}
for (i = 0 do dužine niza feromona za dron) {
 otopi feromone za dron;
}
for (i = 0 do dužine niza putovanja za kamion) {
 dodaj feromone za najbolji put kamiona;
}
for (i = 0 do dužine niza putovanja za dron) {
 dodaj feromone za najbolji put drona;
}
}While (nije prošao broj zadanih iteracija)
```

5. Eksperimentalna istraživanja

Programski kod je pisan u C# programskom jeziku u okružju Visual Studio 2019. Programski kod je izvršavan na računalu sa Intel(R) Core(TM) i5-7500 procesorom koji je radio brzinom 3.40 GHz. Korišteni brojevi agenata je 20, 50 i 80. Korišteni broj iteracija je 200, 500 i 800. Svaka kombinacija agenata i iteracija je ponovljena 21 put za računanje medijana. Stupanja isparavanja feromona rho je postavljen na 0.05, alpha i beta su postavljeni na 1. Razlike između gornje i donje granice je postavljena na 1000.

Broj agenata predstavlja broj mravi korišten unutar programa, broj iteracija predstavlja koliko je puta ponovljena najveća petlja unutar programa. Prosječno trajanje puta je izračunata aritmetička sredina svih odabranih puteva iteracija. Medijan trajanja puta je izračunat medijan svih odabranih puteva iteracija. Apsolutno min vrijeme predstavlja apsolutno najbolje pronađeno vrijeme puta kroz sve iteracije. U tablici 1 se nalaze rezultati za instancu za 20 gradova.

Tablica 1: Rezultati za instancu sa 20 gradova

Broj agenata	Broj iteracija	Prosječno trajanje puta	Medijan trajanja puta	Najbolje vrijeme zadnje iteracije	Apsolutno min vrijeme
20	200	6839	6855	5094	4208
20	500	6756	6759	4944	4095
20	800	6737	6754	4833	4078
50	200	6755	6699	4766	4159
50	500	6655	6682	4530	4080
50	800	6539	6554	4417	3938
80	200	6800	6795	4632	4139
80	500	6633	6575	4585	3866
80	800	6435	6438	4159	3866

Pregledavajući rezultate možemo jednostavno reći da što veći broj agenata sa što većim brojem iteracija nam daje najbolje rezultate. Dublje gledajući, veći broj agenata omogućuje algoritmu da ima veći broj pokušaja pronaći što bolji put, dok veći broj iteracija omogućuje algoritmu bolju elastičnost feromona.

Unutar tablice možemo vidjeti da je bilo koji odabrani put na zadnjoj iteracije bolji od bilo kojeg medijana zadnje iteracije, što nam govori da su ovo sve dobra rješenja. Možemo

vidjeti kako se sva apsolutno minimalna vremena blizu u vrijednosti jedno drugomu, što nam govori da se približavamo najboljem rješenju. Uz veći broj agenata i iteracija bi uspjeli pronaći to najbolje rješenje.

Uz to je važno napomenuti potrebno vrijeme rješavanja, odnosno razliku između vremena potrebno za dobivanje rješenja sa brojem agenata 20 i brojem iteracija 200 u usporedbi sa dobivanjem rješenja sa brojem agenata 80 i brojem iteracija 800. Smanjenje udaljenosti optimalnog puta za 17.3% je potrebno otprilike 14 puta više vremena. U tablici 2 se nalaze rezultati za instancu sa 60 gradova.

Tablica 2: Rezultati za instancu sa 60 gradova

Broj agenata	Broj iteracija	Prosječno trajanje puta	Medijan trajanja puta	Najbolje vrijeme zadnje iteracije	Apsolutno min vrijeme
20	200	22455	22533	20275	16936
20	500	22310	22153	18933	16218
20	800	21680	21846	18402	15950
50	200	22043	22000	18409	16691
50	500	21907	21855	17304	16284
50	800	21631	21272	16396	16044
80	200	21611	21650	18902	16250
80	500	21204	20850	17228	15724
80	800	20187	20487	16196	15120

Instanca sa 60 gradova pokazuje veće poboljšanje vremena potrebnog za cijeli put u usporedbi na instancu sa 20 gradova. Dronovi predstavljaju veliku uštedu u vremenu jer omogućuju kamionu da izbjegava gradove koji bi ga usporavali u njegovom okolnom putu tako što dron dostavlja do tog grada bez usporavanja od kojih kamion pati, kao što su ceste i gužva u prometu.

Zbog sposobnosti drona da radi paralelno uz kamion možemo reći da se svaki grad u koji dron putuje izbacuje iz ukupnog brojenja vremenskog trajanja što predstavlja daljnje poboljšanje unutar ukupnog vremena potrebnog za putovanje u sve gradove. Gledajući tablicu vidimo da najbolja vremena za broj agenata 50 i 80 imaju bliske rezultate, dok najbolje vrijeme sa brojem agenata 20 ima uočljivu razliku. Možemo zaključiti da je broj agenata 20 jednostavno pre mali za dobro rješavanje instance sa 60 gradova.

Svaka kombinacija agenata i iteracija je uspjela pronaći barem jedan put sa manjim vremenom putovanja (ispod 17000). Sa svakim povećanjem broja agenata i iteracija bi uspjeli pronalaziti bolja rješenje, no uz to bi sve više povećavali vrijeme potrebno za rješavanje za koje bi dobivali sve manje bolji rezultat od prijašnjeg. Na tablici 3 se nalaze rezultati za instancu sa 100 gradova.

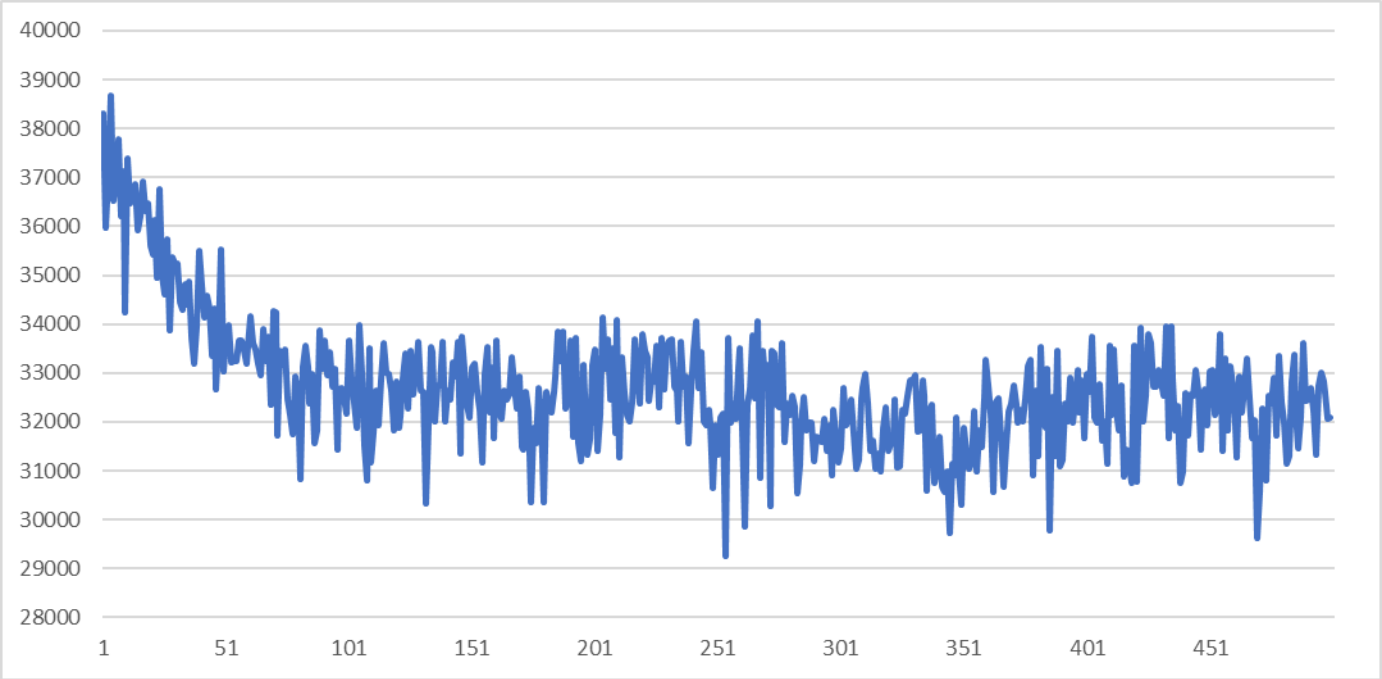
Tablica 3: Rezultati za instancu sa 100 gradova

Broj agenata	Broj iteracija	Prosječno trajanje puta	Medijan trajanja puta	Najbolje vrijeme zadnje iteracije	Apsolutno min vrijeme
20	200	38607	38928	34946	31780
20	500	37496	37552	33604	31674
20	800	37054	37215	33180	29311
50	200	37629	37837	34844	31474
50	500	37420	37361	33241	29702
50	800	36527	36233	33127	29632
80	200	37444	37196	32323	30316
80	500	36282	36953	31138	29517
80	800	35161	34684	30910	29374

Dobit između zadnjeg rješenja sa 80 agenata i 800 iteracije i prvog rješenja sa 20 agenata i 200 iteracije je oko 8.79%. Bilo bi potrebno više agenata i više iteracija kako bi došli do optimalnog rezultata u instancama sa većim brojem gradova.

Možemo zaključiti da u ovome slučaju je povećavanje broja agenata bolji izbor od povećanja broja iteracija. Razlog za to je sam broj gradova unutar instance. Nakon što broj agenata postane veći od broja gradova unutar instance bi povećanje broja iteracija dalo veću dobit nego daljnje povećanje broja agenata.

Sa većim brojem agenata i iteracija postoji mogućnost da bi veći broj svih odabranih puteva u pojedinoj iteraciji mogao dostići najbolju udaljenost puta. Treba napomenuti da sa sve većinom brojem agenata i iteracija će se postići sve bolje rješenje, no to će isto povećati trajanje rada programa. Slika 4 pokazuje napredovanje algoritma za instancu sa 100 gradova sa brojem agenata 50 i brojem iteracija 500. Osi X reprezentira broj iteracije, dok osi Y reprezentira vrijeme putovanja.



Slika 4: Prikaz napredovanja algoritma za instancu sa 100 gradova

Pregledom grafa prikazanog na slici 4, možemo vidjeti kako kroz vrlo mali broj iteracija se rezultate naglo poboljšao, dok sve većim brojem iteracija nakon toga je poboljšavanje rezultata usporilo. Kao što je prije spomenuto, sa sve većim brojem iteracija će se dobit polako smanjivati po iteraciji. Možemo vidjeti da imamo slučajeve unutar rješenje gdje se kvaliteta naglo poboljšala i onda vratila nazad prema prosjeku. Možemo zaključiti da se feromoni ne otapaju dovoljno brzo. Povećanjem vrijednosti rho bi se poboljšali rezultati, no isto bi se povećalo skakanje vrijednosti. U tablicama 4 i 5 se nalaze rezultati za instancu sa 40 gradova i instancu sa 80 gradova.

Tablica 4: Rezultati za instancu sa 40 gradova

Broj agenata	Broj iteracija	Prosječno trajanje puta	Medijan trajanja puta	Najbolje vrijeme zadnje iteracije	Apsolutno min vrijeme
20	200	15356	15369	12397	10438
20	500	15087	15345	12329	10284
20	800	14643	14982	11256	9688
50	200	14982	15122	11675	9979
50	500	14737	14803	11287	9625
50	800	14633	14502	11137	9496
80	200	14964	15124	11571	9407
80	500	14733	14567	10618	9237
80	800	14579	14440	10239	9090

Tablica 5: Rezultati za instancu sa 80 gradova

Broj agenata	Broj iteracija	Prosječno trajanje puta	Medijan trajanja puta	Najbolje vrijeme zadnje iteracije	Apsolutno min vrijeme
20	200	29140	28548	25743	22344
20	500	28462	28311	25155	21976
20	800	28014	28267	24289	21792
50	200	28828	28774	24692	22162
50	500	27687	28361	24341	21462
50	800	27223	28238	23735	21239
80	200	27724	28707	25136	22026
80	500	27413	28588	24188	21351
80	800	26892	28230	22505	20909

6. Zaključak

Optimizacija kolonijom mrava je nastala iz algoritama mrava koji su nastali iz promatranja ponašanja mrava i njihovog traženja hrane. Sama optimizacija se bazira na sposobnosti mravi da ostavljaju i prate feromone koje ostavljaju kretanjem po raznim putovima. Razlog zbog čega je optimizacija kolonijom mrava dobar način optimiziranja je isparavanje feromona. Svi feromoni isparavaju istim stupnjem isparavanja, no dužim putevima će se feromoni brže otopiti jer će mravima trebati duže da prođu taj put i sa tim će ostavljati manji broj tragova kroz vrijeme.

Optimizacija kolonijom mrava se koristi za rješavanje mnogih kombinatornih problema od kojih je najpoznatiji problem trgovačkog putnika. U TSP obliku su postavljene i sve instance koje su rješavane unutar ovog rada. Za rješavanje se koristio Max Min Ant System koji je nastao iz optimizacije kolonijom mrava. Korištenjem različitih brojeva agenta i iteracija je bilo moguće vidjeti njihov utjecaj na dobiveno rješenje. Mijenjanje broja gradova, broja agenata i broja iteracija imalo linearni utjecaj na količinu potrebnog vremena za rješavanje pojedine instance. Dobiveno rješenje je sa povećanjem samo gradova postajalo sve dalje od optimalnog iz čega možemo zaključiti da je uz veće instance potrebno postaviti veći broj agenata i veći broj iteracija. Dobit iz povećanja broja agenata i iteracija postaje sve manja što se njihov broj više povećava. Povećanjem broja agenata će se dobiti bolji rezultat sve dok broj agenata se ne izjednači sa brojem gradova unutar instance, nakon toga će povećanje broja iteracija dati bolje rezultate.

Korištenje optimizacije kolonijom mrava je brz i dosljedan način pronalaska najboljeg rješenja za određene probleme, no on još uvijek ima ovisnost o nasumičnom odabiru puta zbog čega dobiveno rješenje može odskakati u vrijednosti. Postoji mogućnost da će u jednom rješenju problema za problem biti potrebno 1000 iteracija kako bi se pronašao optimalni put, dok bi u drugom rješenju istog problema bilo potrebno 3000 iteracija kako bi program došao do istog optimalnog rješenja. Sa novijim algoritmima rješavanja poput MMAS se smanjuje ovisnost o nasumičnom odabiru puta i smanjuje se mogućnost velikog odskakanja rezultata.

Popis literature

1. Franklin, Benjamin. *Advice to a Young Tradesman*, (21 July 1748). Founders Online. National Archives and Records Administration/University of Virginia Press. Preuzeto 19.08.2021. sa https://founders.archives.gov/documents/Franklin/01-03-02-0130
2. DALLAS--(BUSINESS WIRE)--Omnitracs, LLC, a Solera company, *Delivering on Demand: Consumer 2021 Insights Survey.* (2021) Preuzeto 19.08.2021. sa https://www.omnitracs.com/sites/default/files/files/2021-06/Consumer%20Survey%20Infographic.pdf
3. Amazon *PrimeAir* (2019) Preuzeto 19.08.2021 sa https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011
4. FedEx *Winged drone delivery* (2019) Preuzeto 19.08.2021. sa https://www.fedex.com/en-us/sustainability/wing-drones-transport-fedex-deliveries-directly-to-homes.html
5. Andries P. Engelbrecht (2007). *Computational Intelligence an introduction, Second edition.* 359-383. University of Pretoria, South Africa: John Wiley & Sons, Ltd
6. Mitsuo Gen i Xinjie Yu (2010). *Introduction to Evolutionary Algorithms*. 327-351. Tsinghua University, Beijing, China: Springer Longon Dordrecht Heidelberg New York
7. Marco Dorigo, (1992). *Optimization, Learning and Natural Algorithms*, PhD thesis, Politecnico di Milano, Italy.
8. Marco Dorigo i Thomas Stützle (2004). *Ant Colony Optimization*. Massachusetts Institute of Technology, Massachusetts, USA: A Bradford Book, The MIT Press
9. Thomas Stützle, Holger H. Hoos (2000). *MAX–MIN Ant System. Future Generation Computer Systems.* Vol.16, Issue 8, Pages 889-914, https://doi.org/10.1016/S0167-739X(00)00043-1.
10. NP-hardness. (bez dat.). U Wikipedia. Preuzeto 21.08.2021. s https://en.wikipedia.org/wiki/NP-hardness
11. Universitat Heidelberg (bez dat.) TSPLIB. Preuzeto 18.08.2021. s http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
12. Ivković, Nikola; Golub, Marin (2014). A New Ant Colony Optimization Algorithm: Three Bound Ant System // Lecture Notes in Computer Science, 8667 280-281
13. Ivkovic, Nikola; Jakobovic, Domagoj; Golub, Marin (2016). Measuring performance of optimization algorithms in evolutionary computation // International Journal of Machine Learning and Computing, 6 3; 167-171 doi:10.18178/ijmlc.2016.6.3.593

14. Dell’Amico, M., Montemanni, R. & Novellani, S. Matheuristic algorithms for the parallel drone scheduling traveling salesman problem. Ann Oper Res 289, 211–226 (2020). https://doi.org/10.1007/s10479-020-03562-3

Popis slika

Popis tablica

Prilozi

1. Programski kod korišten u radu
2. Instanca Rand20.tsp
3. Instanca Rand40.tsp
4. Instanca Rand60.tsp
5. Instanca Rand80.tsp
6. Instanca Rand100.tsp
7. Potpisana izjava o izvornosti