

Memetski algoritam i njegove primene

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Bogićević Marica
marica.bog8@gmail.com

Karanović Boris
boskonet@gmail.com

Košanin Petar
kosaninp@gmail.com

Šašić Filip
sasicf@gmail.com

1. april 2020

Sažetak

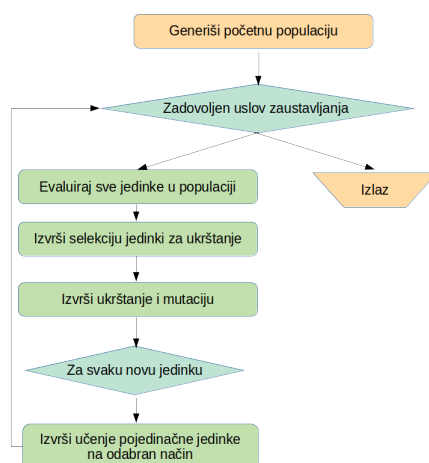
Kroz ovaj rad čitalac će se upoznati sa memetskim algoritmom i njegovim primenama. Prikazan je postupak primene algoritma na dva NP problema: *problem bojenja grafa* i *problem trgovačkog putnika*. Takođe je naveden i *problem detektovanja zajednica* koji ilustruje i primenu memetskog algoritma na realan problem. Primene memetskog algoritma su dosta raznolike, što svrstava celu familiju evolutivnih algoritama kojoj i sam pripada, među popularnim temama u akademskim i industrijskim krugovima.

Sadržaj

1	Uvod	2
2	Primene memetskog algoritma	2
2.1	Problem bojenja grafa	3
2.1.1	HCA	3
2.1.2	MACOL	4
2.1.3	HEAD	4
2.2	Problem trgovačkog putnika	5
2.2.1	Prethodni radovi	5
2.2.2	Metodološki pristup	5
2.2.3	Memetski algoritam primenjen na problem trgovačkog putnika	6
2.2.4	Opis Memetskog algoritma	6
2.2.5	Eksperimentalne metode i rezultati za TSP	8
2.3	Problem detektovanja zajednica	8
2.3.1	Gustina modularnosti	9
2.3.2	Opis memetskog algoritama za detekciju zajednica	9
2.3.3	Eksperimentalni rezultati za detekciju zajednica	10
3	Zaključak	10
	Literatura	10

1 Uvod

Memetski algoritam je prvi put uveden u računarstvo 1989. godine od strane Pabla Moscata [17]. Predstavlja hibridizaciju tradicionalnog, populacionog, genetskog algoritma sa metodama lokalne pretrage kako bi umanjio verovatnoću prevremene konvergencije. Danas, memetski algoritmi su poznati i pod nazivima: *hibridni evolutivni algoritmi* i *kulturalni algoritmi*. U nastavku je dat prikaz memetskog algoritma u njegovoj osnovnoj formi.



Memetski algoritam

Prilikom dizajniranja samog memetskog algoritma, osim parametara koji su neophodni i za osnovni genetski algoritam, potrebno je odrediti još nekoliko parametara specifičnih za memetski algoritam.

- Koliko često trebamo vršiti poboljšanje pojedinačnih jedinki ?
- Na kojim jedinkama ćemo vršiti poboljšanje ?
- Koje metode ćemo koristiti za jedinke ?
- Koliko vremena ćemo potrošiti na poboljšanja ?

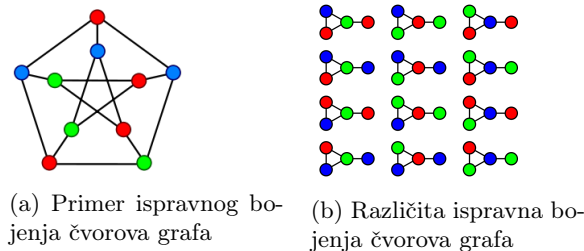
Određivanje ovih parametara predstavlja izazov prilikom kreiranja memetskog algoritma, i direktno utiču na njegovu efikasnost.

2 Primene memetskog algoritma

Memetski algoritam nalazi uspešne primene u mnogim realnim problemima. Kao i genetski algoritam, može se koristiti i u rešavanju raznih NP-teških problema, poput problema trgovačkog putnika, minimalnog bojenja grafa, problem ranca i raznih drugih problema. Mnoge moderne primene podrazumevaju korišćenje memetskog algoritma u oblastima analize poslovanja biznisa, istraživanje podataka, prepoznavanje šablona, dizajniranja električnih kola, mašinskog učenja. Takođe, značajna je i primena u bioinformatiči.

2.1 Problem bojenja grafa

Neka je zadat neusmeren graf $G = (V, E)$, gde je V skup čvorova, a E skup grana tog grafa. Bojenje grafa G predstavlja problem dodeljivanja "boja" elementima grafa¹. Bojenje čvorova je pridruživanje po jedne boje svakom čvoru grafa G . Dodavanjem ograničenja maksimalnog broja boja, dobijamo problem k -bojenja grafa. *Ispravno k -bojenje grafa G* je ono za koje ne postoje dva susedna čvora² koji su obojeni istom bojom. Drugim rečima, ispravno k -bojenje grafa deli čvorove skupa V u k nezavisnih podskupova, pri čemu jedan taj podskup čine nesusedni čvorovi, obojeni istom bojom. Na slici 2 prikazan je primer ispravnog bojenja grafa.



Slika 2: Slika (a) prikazuje ispravno 3-bojenje Petersonovog grafa, dok na slici (b) prikazano je 12 različitih načina 3-bojenja grafa. Izvor slike [1]

Pored svoje teoretske značajnosti kao NP-kompletni problem [19], bojenje grafa nalazi primenu u praksi, a neki od problema koji se svode na bojenje grafa su alokacija registara, izrada rasporeda, raspoređivanje komunikacije satelita [14].

Rešavanje problema bojenja grafa može se predstaviti preko rešavanja serije problema k -bojenja grafa [7]. Naime, rešavanje problema počinje sa inicijalnom vrednošću $k \leq |V|$. Ako se nađe rešenje za k , vrednost k postavljamo na $k - 1$, i ponavljamo proces sve dok ne nađemo na k za koje ne postoji rešenje.

U nastavku biće opisane tri varijacije memetskog algoritma za rešavanje problema bojenja grafa. Prvenstveno biće reči o HCA, prvi HEA (eng, Hybrid Evolutionary Algorithm) primenjen nad bojenjem grafa, a zatim dve varijacije memetskog algoritma nastale po uzoru na HCA.

2.1.1 HCA

U radu [7], predložen je HCA (eng. Hybrid Coloring Algorithm) za rešavanje problema bojenja grafa. HCA čine 4 glavne komponente, *generator inicijalne populacije*, *operator ukrštanja*, *lokalna pretraga*, i *pravila ažuriranja populacije*.

Jednu jedinku populacije čini podela skupa V na k disjunktnih podskupova $\{V_1, V_2, \dots, V_k\}$. Takođe, dodaju se penali onim jedinkama koje sadrže susedne čvorove. Inicijalnu populaciju mogu činiti jedinke koje predstavljaju nedopustiva rešenja³. Lokalna pretraga se zatim koristi za unapređivanje svake jedinke, i eventualno ispravljanje nedopustivih

¹Problem isključivo zavisi od broja boja, a ne od konkretnih boja.

²Dva čvora su susedna, ako postoji grana koja ih povezuje.

³Rešenje je nedopustivo ako ne zadovoljava nametnuta ograničenja problema.

rešenja. HCA za lokalnu pretragu koristi tabu pretragu [18]. Operator ukrštanja konstruiše nova k-bojenja i zatim koristi lokalnu pretragu radi daljeg unapređivanja novih jedinki. Korišćen operator ukrštanja od strane HCA je GPX(eng. Greedy Partition Crossover)[7]. Konačno, pravilo ažuriranja populacije bira jedinke koje prelaze u novu generaciju, a koje će biti zamenjene.

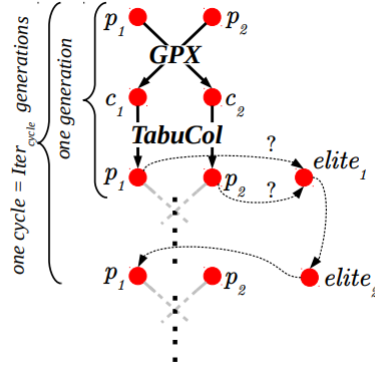
2.1.2 MACOL

MACOL [14] predstavlja još jednu verziju memetskog algoritma za rešavanje problema bojenja grafa. Za inicijalizaciju populacije, MACOL koristi randomizovanu verziju DANGER [8] metode. Za razliku od HCA pristupa, MACOL predlaže novu metodu ukrštanja AMPaX(eng. Adaptive Multi-Parent crossover) [14], koju autori opisuju kao proširenu verziju GPX metode. Jedna od razlika je u tome što u ukrštanju učestvuju dve ili više jedinki. Slično kao i u HCA pristupu, MACOL koristi tabu pretragu.

2.1.3 HEAD

HCA je do 2012. godine davao najbolje rezultate nad DIMACS [10] grafovima. HEAD (eng. Hybrid Evolutionary Algorithm in Duet) [15] predstavlja unapređenje HCA algoritma. Osnovno svojstvo HEAD algoritma je veličina populacije, tj populaciju čine *dve* jedinke. Nakon nasumične inicijalizacije jedinki, vrši se ukrštanje korišćenjem GPX metode. Zatim, HEAD koristi tabu pretragu kako bi unapredio jedinke, i konačno se vrši izbor jedinki koje će činiti sledeću generaciju.

Kako populaciju čine samo dve jedinke, veliki problem HEAD algoritma je prevremena konvergencija. Da bi se zaobišao navedeni problem, HEAD uvodi novo pravilo ažuriranja populacije, tj, koriste se dve pomoćne jedinke $elita_1$ i $elita_2$. Naime, $elita_1$ predstavlja najbolje rešenje trenutne



Slika 3: Dijagram HEAD algoritma

generacije, dok $elita_2$ najbolje rešenje prethodne. U svakoj iteraciji algoritma, jedna od novonastalih jedinki ukrštanjem zamenjuje se sa $elita_2$ jedinkom 3. Pretpostavka je da $elita_2$ se dovoljno razlikuje od jedinki trenutne generacije, i samim tim da doprinosi diverzifikaciji populacije.

2.2 Problem trgovačkog putnika

Problem trgovačkog putnika (eng. Travelling Salesman Problem - TSP) je klasičan problem u polju diskretne i kombinatorne optimizacije. Spada u grupu NP-teških problema čija je složenost $O(n!)$. Matematičke probleme slične njemu prvi je razmatrao Euler koji se bavio pitanjem kako će skakač na šahovskoj tabli posetiti sva 64 mesta samo jednom. Početkom dvadesetog veka matematičar William Rowan Hamilton i Thomas Kirkman su razmatrali probleme koji se svode na problem trgovačkog putnika.

Opšta forma problema TSP se javlja tridesetih godina dvadesetog veka, a pojam "trgovački putnik" prvi put je upotrebljen 1932 godine od strane Karla Mengera koji je u svom radu spomenuo brute-force algoritam i definisao TSP onako kako ga danas poznajemo. Problem je vremenom postajao sve više popularan i izučavan od strane mnogih naučnika.

Neformalna formulacija problema: Dato je n gradova i poznate su sve udaljenosti između njih. Trgovački putnik treba da obide sve gradove i da se na kraju vrati u grad odakle je krenuo, ali da pri tome razdaljina koju pređe bude najkraća.

2.2.1 Prethodni radovi

MTSP (Minimalni problem trgovačkog putnika) je jedan najviše proučavanih kombinatorno optimizacionih problema. U [11] je prikazan kratak pregled početnih Memetskih algoritama (MA) za TSP i predstavljali su približna optimalna rešenja za mali broj gradova. Iako rezultati nisu bili konačni, bili su ohrabrujući i naredna primena Memetskih algoritama na TSP (i na druge NP-Optimizacione probleme) je bila inspirisana tim ranijim radovima.

U [5], MA su korišćeni sa nekoliko nestandardnih karakteristika. U [5], lokalna pretraga je bazirana na moćnoj Navodenoj lokalnoj pretrazi (GLS), Voudorisovoj i Tsangovoj metaheuristici. Ovaj algoritam je poreden sa Lokalnom pretragom sa više početaka (MSLS), Navodenom lokalnom pretragom i sa Memetskim algoritmom. Lokalna pretraga kod MA je bila ista kao i kod GLS-a, ali bez navođenja. U našem eksperimentu smo koristili instance preuzete sa TSPLIB[2]. Ni u jednom slučaju MLSL nije mogao da pronadje optimalnu turu, dok su ostala tri pristupa mogla da je pronađu. Od 31 testirane instance, MA je rešio optimalnih 24, MLSL 0, MA sa jednostavnom lokalnom pretragom 10 i GLSL 16.

Merz i Freisleben u [3] pokazuju različite kombinacije lokalne pretrage i genetske pretrage za TSP (i asimetričnu verziju ATSP) i pokazuju specifičnu ulogu ukrštanja i mutacije. U njihovom pristupu, inicijalna populacija je skup lokalnih optimuma po Lin-Kernighan heuristici, koji je osnova za GA sa skoro optimalnim rešenjima pri inicijalizaciji. Važna stavka njihovog pristupa je da selekcija nije $(\mu + \lambda)$, ni (μ, λ) već mešavina ta dva.

Markov Chains i iterativni Lin-Kernighan tehnike su specijalni slučajevi tehnike Merz-a i Freisleben-a. Njihova mutacija i selekcija je tradicionalnija. Važno je naglasiti da je Memetski algoritam Merz-a i Freisleben-a možda najuspešnija metaheuristika za TSP i ATSP a prethodno opisane šeme su bile pobednički algoritam na Prvom internacionalnom takmičenju u evolutivnoj optimizaciji.

2.2.2 Metodološki pristup

U nastavku ćemo opisati našu metodologiju i MA za TSP. Naša svrha je da prikazemo potencijal pretrage i različitost našeg pristupa. Nama nije

cilj da napravimo specijalizovani TSP rešavač. Mi smo koristili naivne i generičke genetske operatore (ukrštanje i mutacija). Lokalna pretraga ne koristi nikakvo znanje o instanci koja je rešena osim onih koje su obezbedile fitness funkcije. Na ovaj način možemo da garantujemo da će svaki pronađeni benefit biti svojstvo pristupa, a ne posledica korišćenja moćnih operatora. U svim prethodno navedenim Memetskim algoritmima za TSP, inteligentni operatori su korišćeni u formi specijano dizajnirane mutacije, ukrštanja i lokalne pretrage. Štaviše, lokalna pretraga je korišćena u navedenim radovima koristeći znanje o instancama (to jest listu najbližih suseda, kd-drвета...). Osim toga, ne postoji nigde u literaturi nešto poput "standardnog" MA za TSP s kojim bismo mogli da pravimo poređenje.

2.2.3 Memetski algoritam primenjen na problem trgovačkog putnika

U ovom odeljku ćemo opisati novu prilagođavajuću šemu. Pokazaćemo kako nelinearna interakcija između osnovnog genetskog algoritma i lokalne pretrage/diverzifikovani proces (koja je regulisana posmatranjem populacije genetskih algoritama) daje poboljšanje u metaheuristici globalne pretrage.

2.2.4 Opis Memetskog algoritma

Korišćeni pseudokod MA je:

Algorithm 1: Pseudocode MA

```

1 begin
2   Initialize population Parents;
3   repeat
4     Local_Search(Parents,Pls);
5     mating_pool := Select_mating(Parents);
6     Offsprings:= Cross(mating_pool);
7     Mutate(Offsprings);
8     Parents := Select(Parents,Offsprings);
9   until (Finalization_criteria_met);
10 end
```

U ovoj osnovnoj šemi, Select(...) je $(\mu + \lambda)$ ili (μ, λ) strategije, predstavljajući dva ekstrema selektivnog pritiska, sa $+$ i $-$ strategijom imajući najveći pritisak i - najnižu strategiju. Select_mating(...) je turnirska selekcija. U slučaju $+$ i $-$ strategije, data jedinka može biti modifikovana više puta tokom svog "života" od strane lokalne pretrage ili mutacije zato što nam strategija dopušta da jedinka opstaje. Najbolja jedinka se nikada ne modifikuje lokalnom pretragom, tako da u funkciji lokalne pretrage pozivamo funkciju Apply_Move(indip):

Algorithm 2: Algorithm Apply_Move(indip)

```
1 begin
2   /* This is a minimizing process */
3   prevFitness = fitness(indip);
4   Modify(indip);
5   nFitness = fitness(indip);
6   if (prevFitness > nFitness) then
7     | Accept configuration;
8   end
9   else
10    deltaE=nFitness-prevFitness;
11    threshold =  $e^{-k \cdot \frac{\text{deltaE}}{\text{temperature}}}$ ;
12    if (random(0,1)<threshold) then
13      | Accept configuration;
14      /* even if worse than the previous one */
15    end
16    else
17      | Reject changes;
18    end
19  end
20 end
```

Mora biti naznačeno da Modify(...) može biti bilo koji potez lokalne pretrage (to jest 2swap, ubacivanje gradova...). Samoadaptacija lokalne pretrage eksploracijskom ili eksploatacijskom ponašanju je regulisana parametrom temperature. U slučaju predstavljenom iznad, cela populacija deli istu temperaturu. Ta temperatura određuje stepen kojim će kretanje uzbrdo biti dozvoljeno. Pošto je temperatura inverzno proporcionalna širenju fitnesa u populaciji, fitnes konvergira dok temperatura raste. Posledica ovoga je da će svaka jedinka u populaciji biti više "nervozna" i pokušaće da se udalji od inicijalne pozicije, istražujući prostor pretrage. U nekom trenutku fitnes će se širiti, spuštajući temperaturu populacije. Mi obezbeđujemo modifikaciju najboljih jedinki lokalnom pretragom, stoga je uslov najboljeg fitnesa održan.

Primenili smo MA(...), Localsearch(...) i ApplyMove(...) na TSP sa modifikacijom definicije temperature koja je bila postavljena na

$\frac{1}{|maxFitness - averageFitness|}$ da bi bila bolja dinamičnost. Modify(...) procedura je koristila two_swap (TS) pomeranje. TS selektuje podturu i invertuje gradove. To dovodi do 4 promene putanje u datoj turi. Nazvana je two_swap zato što pravi dve promene putanja u originalnoj turi zarad dve nove putanje. Modify(...) bira random broj (između 1 i 10% od instancne veličine) koji određuje koliko vezanih primena two_swap(...) će biti primenjeno na jedinku. Posle primene pomeraja, nova jedinka će biti prihvaćena na osnovu Boltzmann-ove distribucije bazirane na temperaturi trenutne populacije.

Slede detalji MA: Koristili smo populaciju od 50 jedinki. Ukrštanje, mutacija i lokalna pretraga su primenjeni sa verovatnoćom od 0.8, 0.05 i 1.0 redom. Ove verovatnoće su bile fiksne sve vreme. Nije rađena iscrpna optimizacija parametara, već su određivani empirijski zavisno od broja pokretanja. Vrednost k je 0.01 u (μ, λ) strategiji, a 0.001 u $(\mu + \lambda)$. Svaka jedinka radi lokalnu pretragu/diverzifikovanu fazu u svakoj generaciji, osim najboljih jedinki. Selekcija je Turnirska veličine 2. GA je u jednom slučaju mirnog stanja sa $(50 + 50)$ selekcionom strategijom, a u

drugom slučaju (50,50) selektivne strategije je bilo korišćeno. Za kodiranje smo koristili niz intidžera koji imaju sledeće značenje: ako pozicija i ima intidžer j , onda putanja povezuje gradove i i j , (i,j) postoji u turi. Ukrštanje prihvata dva roditelja i generiše potomke počevši od random grada. Daje detetu najkraću putanju od bilo kojeg roditelja. Ako ni jedna putanja od oba roditelja nije slobodna, onda se dodaje random putanja. Mutacija je primena `two_swap` operatora. Inicijalizacija populacije je random.

2.2.5 Eksperimentalne metode i rezultati za TSP

U ovom delu ćemo predstaviti korišćene metode i dobijene rezultate. Kao što smo ranije spomenuli, koristimo instance iz baze TSPLIB [2] koja ima 76 gradova. Pokrenuli smo 30 simulacija za dve različite selektivne strategije, (50,50) i (50+50). Probali smo ova dva scenarija jer smo hteli da istražimo ne samo krajnju dužinu ture već i raznolikost populacije. Hteli smo da vidimo kako se ponaša naš samoadaptivni algoritam u ovim ekstremima. Testirali smo naš algoritam sa četiri druga algoritma: Standardni GA, Memetski algoritam za penjanje uz brdo (HC), Boltzmann-ov memetski algoritam za penjanje uz brdo, Memetski algoritam linearnog žarenja (LMA) i Samoadaptivni memetski algoritam (MA).

Primer 2.1 U tabeli 1 je prikazan kratak pregled statističke analize za dužinu ture ispod (50,50), strategija: + predstavlja algoritam čije ime u koloni postiže veću turo od imena u vrsti, - predstavlja algoritam čije ime u vrsti postiže manju turo od imena u koloni, a - ili + sa * predstavlja p -vrednost ≥ 0.01

Tabela 1: Poređenje algoritama

Algorithms	GA	HC	BHC	LMA	MA
GA		-	+	+	+*
HC	+		+	+	+*
BHC	-	-		+	+*
LMA	-	-	-		+*
MA	-*	-*	-*	-*	

2.3 Problem detektovanja zajednica

Oblast detektovanja zajednica (eng. community detection) ima za cilj pronalazak podskupa čvorova grafa čiji je skup međusobnih grana gušći nego ostatak grafa, takav skup nazivamo zajednicom. Pripadnost čvorova istoj zajednici povećava njihovu šansu da imaju slične karakteristike [9].

Upotreba algoritama iz ove oblasti je diverzifikovana jer se mnogi problemi iz stvarnog sveta mogu predstaviti grafovima. Pronalazak ovakvih grupa pomaže oglašivačima da preciznije pogode ciljne grupe ljudi, društvenim mrežama da lakše daju preporuke sadržaja, kompanijama da ustanove mišljenje ljudi za grupu proizvoda, ovo su samo neke od primena u ovoj brzo rastućoj oblasti.

Gervan-Njumanov [16] hijerarhijski algoritam klasterovanja i jedan od najpoznatijih algoritama detekcije zajednica uvodi *modularnost* (eng. modularity) kao meru kvaliteta. Ova mera je jedna od najkorišćenijih funkcija

kvaliteta. Međutim Fortunato i Bartolomej [6] pronalaze da optimizacija mere modularnosti ne uspeva da pronađe zajednice manje od određene veličine čak i kada su one jasno razdvojene. Ova veličina zavisi od veličine ukupne mreže i međupovezanosti zajednica. U nastavku predstavimo primenu memetskog algoritma na problem detektovanja zajednica koristeći *gustinu modularnosti* (eng. modularity density) kao alternativnu meru kvaliteta da bi prevazišli ovo ograničenje.

2.3.1 Gustina modularnosti

Li *et al.* [13] uvodi novu kvantitativnu meru kvaliteta pod nazivom *gustina modularnosti* koja je zasnovana na gustini podgrafova. Sto je gustina veća to je kvalitet izdvojene zajednice bolji.

Neka je zadat neusmeren graf $G = (V, E)$, gde je V skup čvorova, a E skup grana tog grafa. Neka je A adjungovana matrica grafa G . Neka su V_1 i V_2 dva disjunktna podskupa skupa V , definišemo $L(V_1, V_2) = \sum_{i \in V_1, j \in V_2} A_{ij}$, $L(V_1, V_1) = \sum_{i \in V_1, j \in V_1} A_{ij}$ i $L(V_1, \bar{V}_1) = \sum_{i \in V_1, j \in \bar{V}_1} A_{ij}$ gde je $\bar{V}_1 = V - V_1$. Za podelu $\Omega = \{V_1, V_2, \dots, V_m\}$ gde je V_1 skup čvorova podgrafova G_i za $i = 1, \dots, m$, gustina modularnosti D je definisana kao:

$$D_\lambda = \sum_{i=1}^m \frac{2\lambda L(V_i, V_i) - 2(1-\lambda) L(V_i, \bar{V}_i)}{|V_i|} \quad (1)$$

Pomoću parametra λ se kontroliše veličina zajednica koja se pronalazi i može imati vrednost između 0 i 1. Što je veća vrednost λ to su zajednice manje.

2.3.2 Opis memetskog algoritama za detekciju zajednica

Opisaćemo algoritam predstavljen u [9]:

Algorithm 3: Algorithm framework of Meme-Net

Input: Maximum number of generations: G_{max} ;
Population size: S_{pop} ; Size of mating pool: S_{pool} ;
Tournament size: S_{tour} ; Crossover probability:
 P_c ; Mutation probability: P_m .
1 $\mathbf{P} := \text{GenerateInitialPopulation}(S_{pop})$;
2 repeat
3 $\mathbf{P}_{parent} := \text{Selection}(P, S_{pool}, S_{tour})$;
4 $\mathbf{P}_{child} := \text{GeneticOperation}(\mathbf{P}_{parent}, P_c, P_m)$;
5 $\mathbf{P}_{new} := \text{LocalSearch}(\mathbf{P}_{child})$;
6 $\mathbf{P} := \text{UpdatePopulation}(P, \mathbf{P}_{new})$;
7 until $\text{TerminationCriterion}(G_{max})$;
Output: Convert the fittest chromosome in \mathbf{P} into a
partition solution and output.

Funkcija *Selection()* koristi turnirsku selekciju veličine 2 da izabere roditelje za dalju reprodukciju. Funkcija *GeneticOperation()* vrši dvopoziciono ukrštanje (eng. crossover) i mutaciju. Mutacija se vrši tako što se nasumično bira čvor u hromozomu i nasumično mu se dodeljuje neka susedna zajednica, šansa mutacije je 0.2. *LocalSearch()* maksimizuje gustinu modularnosti tako što se kreće od suseda do suseda u prostoru kandidata pretrage, pravi lokalne izmene dok optimalno rešenje nije pronađeno. *UpdatePopulation()* se koristi za ponovno sastavljanje populacije uzimajući

najbolje jedinke iz $P \cup P_{new}$. Kao uslov zaustavljanja koristi se maksimalni broj generacija G_{max} postavljen na 50.

2.3.3 Eksperimentalni rezultati za detekciju zajednica

Istraživanje se vrši na 11 različitih generisanih mreža dobijenih po principu predstavljenom u radu [12]. Različite mreže se dobijaju variranjem parametra μ između 0 i 0.5. Što je parametar μ veći to su zajednice teže uočljive. Kao mera sličnosti koristi se NMI (eng. normalized mutual information) predstavljena u [4]. NMI može imati vrednosti od 0 do 1, ako je vrednost jednaka 1 onda je algoritam pronašao prave zajednice koje smo generisali ako je 0, algoritam posmatra celu mrežu kao jednu veliku zajednicu. Za $\lambda = 0.5$ i $\mu = 0.25$ algoritam uvek pravilno pronalazi zajednice ($NMI = 1$). Kako povećavamo μ , preciznost je sve manja, za $\mu = 0.35$ dobija se da je $NMI = 0.9$. Već za $\mu = 0.4$ algoritam ne uspeva da podeli mrežu u zajednice, međutim ako se poveća λ sa 0.5 na 0.7 (povećanje λ omogućava pronalazak manjih zajednica) i za $\mu = 0.45$ algoritam uspeva da pronađe oko 80% zajednica. Sa druge strane, ako se smanji λ na 0.3 (omogućava pronalazak većih zajednica) i za $\mu = 0.3$ algoritam posmatra celu mrežu kao zajednicu ($NMI = 0$).

Eksperimenti pokazuju da memetski algoritam daje bolje rezultate od uobičajenih genetskih algoritama za ovu vrstu problema. Povrh toga, gustina modularnosti omogućava dodatnu kontrolu rezolucije mreže u odnosu na tradicionalnu što je čini veoma primenljivom za problem detekcije zajednica. Podešavanje parametra λ omogućava promenu hijerarhijske strukture mreže i bolju analizu topološkog sastava. U ovom radu opisana je optimizacija jednog kriterijuma, sledeći korak u oblasti detektovanja zajednica je ispitivanje višekriterijumske optimizacije (eng. multi-objective optimization) [9].

3 Zaključak

Svi algoritmi koji pripadaju evolutivnim algoritmima pretražuju prostor rešenja uz pomoć heuristika. Memetski algoritam za razliku od većine ostalih algoritama poseduje dosta veći broj parametara koji mu služe za navođenje kroz prostor rešenja. Upravo na tom mestu se dolazi i do problema. Pogrešnim izborom možemo dosta sporije doći do rezultata, kao i da dobijemo rešenje koje nije blizu optimalnog. Takođe, dobrim izborom parametara, memetski algoritam se pokazuje dosta efikasnijim od standardnog genetskog algoritma.

Literatura

- [1] Graph coloring. https://en.wikipedia.org/wiki/Graph_coloring.
- [2] TSPLIB, 2013. on-line at: <http://comopt.ifl.uni-heidelberg.de/software/TSPLIB95>.
- [3] Peter Merz Bernd Freisleben. Asim. on-line at: https://www.researchgate.net/publication/3642891_A_genetic_local_search_algorithm_for_solving_symmetric_and_asymmetric_traveling_salesman_problem/.

- [4] Leon Danon, Albert Díaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005:P09008–P09008, 09 2005.
- [5] Pablo Moscato Ferrante Neri, Carlos Cotta. Memetic. on-line at: <https://www.worldcat.org/title/handbook-of-memetic-algorithms/oclc/1001250863>.
- [6] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104:36–41, 12 2006.
- [7] Philippe Galinier and Jin-Kao Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of combinatorial optimization*, 3(4):379–397, 1999.
- [8] Fred Glover, Mark Parker, and Jennifer Ryan. Coloring by tabu branch and bound. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 26:285–307, 1996.
- [9] Maoguo Gong, Bao Fu, Licheng Jiao, and Haifeng Du. Memetic algorithm for community detection in networks. *Physical Review E*, 84, 11 2011.
- [10] David J. Johnson and Michael A. Trick. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993*. American Mathematical Society, USA, 1996.
- [11] Hans-Paul Schwefel Kenneth De Jong, Lawrence Fogel. Handbook, 1997. on-line at: <https://doc.lagout.org/science/Artificial%20Intelligence/Evolutionary%20computation/The%20Handbook%20of%20Evolutionary%20Computation%20-%20Kenneth%20De%20Jong.pdf>.
- [12] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78, 10 2008.
- [13] Zhenping Li, Shihua Zhang, Rui-Sheng Wang, Xiang-Sun Zhang, and Luonan Chen. Quantitative function for community detection. *Physical Review E*, 77, 03 2008.
- [14] Zhipeng Lü and Jin-Kao Hao. A memetic algorithm for graph coloring. *European Journal of Operational Research*, 203(1):241–250, 2010.
- [15] Laurent Moalic and Alexandre Gondran. Variations on memetic algorithms for graph coloring problems. *Journal of Heuristics*, 24(1):1–24, 2018.
- [16] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69, 02 2004.
- [17] Moscato Pablo. On evolution, search, optimization, genetic algorithms and martial arts - towards memetic algorithms. 1989.
- [18] Mišković S. Tabu pretraga. <http://poincare.matf.bg.ac.rs/~stefan/opo/ts.pdf>.
- [19] Živković M. *Algoritmi*. Matematički Fakultet, Beograd.