SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 518

PRIMJENA ALGORITAMA OPTIMIZACIJE KOLONIJOM MRAVA ZA RJEŠAVANJE PROBLEMA RASPOREDA

Dino Klemen

Zagreb, lipanj 2009.

Sažetak

U ovom se završnom radu proučava problem rasporeda ispita iz klase problema odabira podskupa, na koji se primjenjuje optimizacija kolonijom mrava, odnosno njezina inačica max-min mravlji sustav koji se svrstava u skupinu evolucijskih algoritama. Naglasak se stavlja na programsku implementaciju rješenja koja se testira na stvarnoj datoteci Fakulteta elektrotehnike i računarstva u Zagrebu. Na temelju dobivenih rezultata preporučuju se najbolje vrijednosti za potrebne parametre i strategije.

Ključne riječi: algoritam optimizacije kolonijom mrava, problem rasporeda ispita, evolucijski algoritmi

Abstract

This work explains examination timetabling problem from the subset selection problem class with the use of ant colony optimization or, more precisely, its max-min version from the subset of evolutionary algorithms. Main focus is put on software implementation which enables this method to be used for dealing with actual timetabling problem. The results are analised, and based on them the conclusion which enables selection of best parameters and strategies for given problem is made.

Keywords: ant colony optimization, examination timetabling problem, evolutionary algorithms

Sadržaj

1. Uvod

Ovaj rad se nadovezuje na prethodna dva obavljena u sklopu FER-ovog preddiplomskog studija. Seminar "Izranjajuća inteligencija" te rad "Optimizacija kolonijom mrava" u sklopu projekta "Evolucijski algoritmi" su bili uvod i priprema za tematiku ovog završnog rada.

Iza pojma izranjajuće inteligencije krije se jednostavno ponašanje velikog broja jedinki koje rezultira nečim značajnijim od običnog zbroja doprinosa. Dolazi do izranjanja inteligencije kada cjelina postaje veća od sume svojih jedinki. Naglasak u tom radu bio je na ponašanju mrava.

Optimizacija kolonijom mrava u projektu je promatrana na primjeru problema bojanja grafa, koji je jako blizak ovdje promatranom problemu rasporeda ispita. Čak se i prvi pokušaji automatiziranja izrade rasporeda temelje na rješenjima iz problema bojanja grafa.

Tako se u ovom radu usredotočuje na zadani problem – problem izrade rasporeda i na zadano sredstvo rješavanja problema – optimizaciju kolonijom mrava. U poglavlju [2] definira se zadani problem, u poglavlju [3] se daje teorijska podloga rješavanju problema, koja se nadopunjava implementacijskim detaljima u poglavlju [4]. Na kraju poglavlje [5] izvodi mjerenja i donosi zaključke o odabiru odgovarajućih parametara, opcija i strategija.

2. Opis problema

Problem rasporeda ispita (Examination Timetabling Problem) jest zanimljiv problem s kojim se susreću edukacijske ustanove, s posebnim naglaskom na one sveučilišne. Jasno je kako je ovaj problem konkretan i nimalo apstraktan jer ima direktne primjene u našem životu. Zato mu treba pristupiti s posebnom odgovornošću jer dobiveni rezultati mogu biti od velike važnosti.

Automatiziranje izrade rasporeda počelo je privlačiti pozornost istraživačke zajednice šezdesetih godina prošloga stoljeća. [1] Prvi od značajnijih radova bio je onaj Gotlieba iz 1963, iako su se pristupi rješavanju problema u to vrijeme temeljili na ljudskom pristupu kojeg zovemo direktnom heuristikom. [2] Ručni pristup rješavanju problema zna potrajati od tjedan dana do par mjeseci na većim fakultetima, a problem nije uvijek statičan, pa se zbog raznih izmjena proces treba ponavljati više puta. Zanimljivo je spomenuti da je na početku ovog tisućljeća u samo 21% britanskih sveučilišta korišteno računalo za stvaranje rasporeda ispita. [3]

Problem nije ograničen samo na edukacijske ustanove, nego se može primijeniti i na niz drugih, poput bolnica, transportnih sustava ili sportskih događanja. Pronalaženje dobrog rasporeda je preduvjet za osiguravanje pružanja dobre usluge.

2.1. Problem odabira podskupa

Promatramo li problem rasporeda ispita s veće udaljenosti, mogli bismo ga svrstati u veliku skupinu problema odabira podskupa (subset selection problem). SSP traži optimalan dozvoljen podskup početnog skupa objekata, pazeći na ciljnu funkciju i određene restrikcije.

Slijedi formalna definicija problema odabira podskupa:

- S je skup objekata;
- $S_{prihvatljivo} \subseteq P(S)$ je skup koji sadrži prihvatljive podskupove od S;
- $f: S_{prihvatljivo} \rightarrow R$ je ciljna funkcija.

Zadatak SSP-a (S, $S_{prihvatljivo}$, f) je pronaći $S^* \subseteq S$ takav da je $S^* \in S_{prihvatljivo}$ i f(S*) maksimalan.

2.2. Definicija problema rasporeda ispita

Iako je promatrani problem dovoljno blizak problemu maksimalnog ispunjenja ograničenja (maximum constraint satisfaction) [4, 5], ovdje će ipak biti opisan konkretni problem rasporeda ispita s njemu specifičnim oznakama.

Definicija problema je u nastavku:

- zadan je skup ispita $E = e_1, e_2, ..., e_e$;
- zadan je skup termina $T = t_1, t_2, ..., t_t$;
- zadan je skup studenata $S = s_1, s_2, ..., s_s$ koji pohađaju ispite;
- zadan je skup kapaciteta vezan uz svaki termin $C = C_1, C_2, ..., C_t$.

Zadatak je za svaki ispit pronaći odgovarajući termin tako da se zadovolji skup osnovnih ograničenja:

- konfliktni ispiti (koji imaju zajedničke studente) ne mogu biti održani istovremeno;
- za svaki termin, broj studenata koji se trebaju pojaviti na ispitima ne smije premašivati zadani kapacitet.

Navedene zahtjeve možemo zapisati formalnije:

$$x_i \neq x_j, \forall i, j \in E, i \neq j \ \& \ D_{ij} > 0 \tag{1}$$

$$\sum_{i \in E} s_i \leq C_t, x_i = t, t \in T \tag{2}$$

gdje D_{ij} predstavlja broj konfliktnih studenata za E_i i E_j, a x_i je termin kojem je E_i pridružen, dok je s_i broj studenata vezanih za E_i. [6] Ukoliko postoje neka dodatna ograničenja, zadatak je odabrati ono rješenje koje kršenje tih novih zahtjeva smanjuje što je više moguće.

2.3. Definicija FER-ovog problema rasporeda ispita

Raspoređivanje ispita na Fakultetu elektrotehnike i računarstva u Zagrebu ima zadanih pet elemenata:

- popis predmeta, te njima pripadajućih studenata;
- popis dozvoljenih termina, te njima pripadajućih kapaciteta;
- popis predmeta koji moraju biti vezani uz jedan određeni termin;
- popis predmeta koji moraju biti vezani uz jedan od određenih termina;
- popis clustera predmeta koji moraju biti održani u istom terminu.

Problem zahtjeva da se uvaže spomenute restrikcije, uz one podrazumijevane, da jedan student ne može istovremeno biti na dva ispita (zahtjev nultog reda), da studenti ne bi trebali imati više ispita u istom danu, ili dva dana za redom (zahtjev prvog reda), te da kapacitet termina ne može biti premašen (zahtjev drugog reda).

2.4. Kompleksnost

Za ETP jest poznato kako je riječ o NP-teškom problemu [7]. U teoriji kompleksnosti kratica NP označava nedeterminističko polinomno vrijeme (non-deterministic polynomial time), a odnosi se na skup problema rješivih pomoću nedeterminističkog Turingovog stroja u polinomnom vremenu.

NP-potpuni problemi su oni koji su u skupu NP, a ujedno se svi drugi problemi iz NP skupa mogu svesti na njega. Za razliku od NP-potpunih, NP-teški zadržavaju svojstvo svođenja drugih problema iz NP na taj problem, dok NP-teški problemi ne moraju nužno biti iz skupine NP. [8]

Kao rezultat činjenice da je promatrani problem NP-težak, zaključujemo kako izravne metode pretraživanja prostora poput postupka vraćanja unatrag neće biti efikasne. Upravo zato treba ovom problemu pristupiti na stohastički način koji reducira prostor pretraživanja, a time i vrijeme izvršavanja.

2.5. Gustoća konflikata

Težina rješavanja problema obično se ilustrira pomoću stupnjeva čvorova u neusmjerenom težinskom grafu koji predstavlja zadani problem. Čvorovi grafa se vežu uz ispite, a lukovi uz konflikte među ispitima. Dva su ispita međusobno konfliktna ukoliko postoje zajednički studenti koji trebaju položiti oba ispita. Lukovima se dodaju težine koje odgovaraju broju konfliktnih studenata između dva ispita (predmeta).

Tako se često u literaturi i u raznim instancama problema na Internetu definira gustoća konflikata (conflict density). Gustoća konflikata definira se preko matrice konflikata C čije elemente c_{ij} za svaki par ispita definiramo na sljedeći način:

$$c_{ij} = \begin{cases} 1 \text{ ako uKonfliktu(i, j)} \\ 0 \text{ inace} \end{cases} \tag{3}$$

Na temelju dobivene matrice konflikata računamo gustoću konflikata kao omjer jedinica naprema ukupnom broju elemenata u matrici. Uobičajene vrijednosti na koje se nailazi kreću se od 0,03 do 0,42 [9] dok je vrijednost FER-ovog problema ove godine iznosio 0,24.

2.6. Ograničenja

Uobičajeno se ograničenja dijele u dvije osnovne skupine, jaka i slaba ograničenja. Ako se rigorozno držimo definicije, jaka ograničenja su ona koja se ni pod kojim uvjetima ne mogu prekršiti. Rješenje koje ne krši jaka ograničenja nazvat ćemo prihvatljivim. Tek nakon što je rješenje prihvatljivo, možemo uopće razgovarati o kršenju slabih ograničenja. Rješenje koje minimalno krši slaba ograničenja nazvat ćemo optimalnim.

U idealnom ćemo slučaju u prvom prolasku razdvojiti prihvatljiva od neprihvatljivih rješenja, a zatim ćemo pokušati smanjiti kršenje slabih ograničenja – idealno se približavajući nuli. Realno, takve su situacije rijetke i većina postojećih problema će u nekoj mjeri kršiti slaba ograničenja. Primjeri uobičajenih ograničenja mogu se vidjeti u tablicama [Tablica 2.1] i [Tablica 2.2]. Raznolikost ograničenja dobro ilustrira istraživanje koje je proveo Burke 1996 nad 56 britanskih sveučilišta gdje je pronađeno ukupno 32 različitih vrsta ograničenja.

U manjem broju slučaja može se vidjeti i uvođenje treće kategorije, nužnih ograničenja. [10] Nužna ograničenja su ekstremna jaka ograničenja koja se obično implementiraju unutar programskog rješenja, tako da nikada ne mogu biti prekršena. Za razliku od jakih ograničenja, koja mogu biti prekršena, ali će onda biti odbačena.

Zanimljivo je da nije toliko uobičajeno raspravljati o dobivanju prihvatljivog rješenja, već se ono na neki način podrazumijeva. Tako da se glavni naglasak stavlja u istraživačkom smislu na udovoljavanju slabih ograničenja, koja ponekad znaju biti i međusobno kontradiktorna.

Tablica 2.1. Uobičajeni primjeri jakih ograničenja

1	Student ne može istovremeno pohađati dva (ili više) ispita.
2	Mora biti dovoljno mjesta u učionicama za sve studente.
3	Ispiti s posebnim zahtjevima (dozvoljeni termini, zahtjevi učionice) trebaju biti pridruženi odgovarajućim terminima ili učionicama.
4	Ispiti koji zahtijevaju istovremeno izvođenje.

Tablica 2.2. Uobičajeni primjeri slabih ograničenja

1	Ispite u konfliktima razdvojiti što ravnomjernije.
2	Održati ispite s većim brojem studenata što ranije.
3	Ravnomjerno rasporediti "teške" ispite.
4	Uvoditi trajanja ispita u obzir te grupirati slična trajanja.

2.7. Usporedba s problemom bojanja grafa

Kada bismo problemu rasporeda ispita ukinuli slaba ograničenja, dobili bismo problem bojanja grafa (graph coloring problem). Dva su spomenuta problema isprepletena i zato nije ni neobično kako se rani pristup ETP-u temeljio na prethodnim razmatranjima nad GCP-om. Welsh i Powell su 1967. povezali ta dva problema i otvorili prostor daljnjem razvoju ETP-a u dobrom smjeru.

ETP bez slabih ograničenja se svodi na pridruživanja termina (boja) čvorovima zadanog grafa (ispitima) tako da dva susjedna čvora (ispita) nisu pridružena istom terminu (obojana istom bojom). Opisana ideja je upravo temelj problema bojanja

grafa. Uvođenjem slabih granica, ograničenja se počinju ispreplitati i ne možemo više poistovjećivati dva problema.

2.8. Pozitivan trend u svijetu vezan za ETP

Zanimljivo je kako se u posljednjih nekoliko godina stvorila kompetitivna atmosfera vezana za rješavanje problema rasporeda ispita. Ne samo da je zanimljivo naći kvalitetna rješenja, već je i bitno pronaći brza rješenja, koja će biti primjenjiva na što više postojećih problema, koji mogu biti temeljeni na stvarnim podacima ili kompjuterski generirani.

Tako je 2002. godine održano prvo internacionalno natjecanje u izradi rasporeda, dok je drugo održano pet godina kasnije. Primjerice, pobjednik posljednjeg natjecanja koristio je simulirano kaljenje kome su prethodile dvije faze lokalnih pretraga, čime se naglasak stavlja na hibridizaciju algoritama o čemu će biti još govora u nastavku rada.

Također postoji niz alata i jezika nastalih za rješavanje ili definiranje problema rasporeda ispita. Tsang, Mills i Williams razvili su jezik za specifikaciju ETP-a [11] dok su Di Gaspero i Schaerf razvili alat EASYLOCAL++ za implementaciju algoritama lokalne pretrage nad problemima rasporeda. [12] Dodatno je De Causmaecker 2002 proučavao kako se može primijeniti semantički Internet kod raspoređivanja. [13]

Pozitivan trend potvrđuje i niz internacionalnih konferencija PATAT (Practice and Theory on Automated Timetabling) te osnivanje radne grupe na problemu automatiziranja rasporeda WATT unutar organizacije EURO (European Association of Operational Research Societies).

3. Koncept rješenja

Rješenju se u radu pristupa optimizacijom kolonijom mrava (ant colony optimization). Riječ je o metaheurističkom pristupu kojeg je uveo Marco Dorigo 1996 kako bi riješio teške optimizacijske probleme. Iako je još 1992, također Marco Dorigo uveo preteču ACO-u, mravlje sustave (ant system). Metaheuristike općenito traže optimalna (ili skoro pa optimalna) rješenja na aproksimativan i nedeterministički način.

Samo ime kaže da je algoritam inspiriran ponašanjem mravlje kolonije, i to specifično na temelju mravlje strategije skupljanja hrane gdje glavnu ulogu igra supstanca feromon kojeg mravi ostavljaju za sobom nakon što pronađu hranu. Na taj način indirektno signaliziraju drugim mravima da prate taj trag kako bi i oni pronašli isti izvor hrane. Takav oblik komunikacije poznat je pod nazivom stigmergije (stigmergy) gdje agenti komuniciraju preko okoline koju su modificirali drugi agenti.

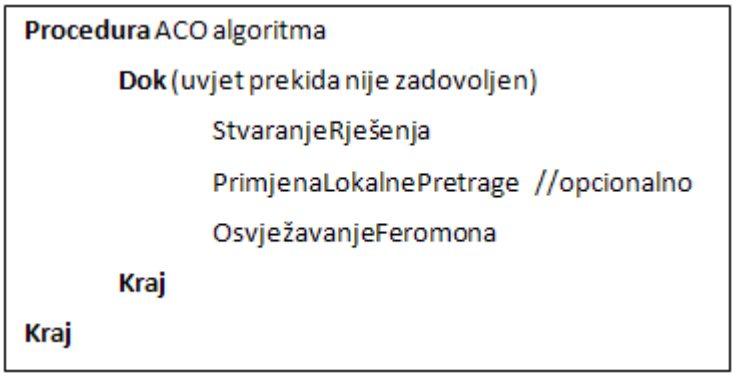
Dodatnom klasifikacijom, ACO možemo smjestiti u skup evolucijskih algoritama čiji temelj leži na manipulaciji i evoluciji populacije jedinki.

3.1. Osnovna ideja optimizacije kolonijom mrava

Kod algoritma optimizacije kolonijom mrava, kolonija mrava ograničene veličine kolektivno pretražuje svoju okolinu kako bi pronašla dobra rješenja zadanog optimizacijskog problema. Svaki mrav gradi rješenje u konačnom broju koraka gdje svaki korak predstavlja parcijalnu gradnju rješenja koje se pridružuje trenutno izgrađenom od strane tog istog mrava. Odabir parcijalnog rješenja mrav temelji na dva faktora. Kao jedan faktor uzima se u obzir heuristika specifična za problem, koju nazivamo vidljivošću. Dok drugi faktor predstavlja povjerenje u prethodno građena rješenja cijele kolonije, što se krije iza pojma feromona.

Nakon koraka stvaranja rješenja može uslijediti primjena lokalne pretrage o kojoj će biti nešto više riječi u nastavku, dok se postupak zaključava osvježavanjem feromona gdje u stvarnosti svaki mrav koji je pronašao kakvo-takvo rješenje ostavlja za sobom trag feromona, ali se zbog nepraktičnosti kod implementacije primjenjuje elitistička strategija gdje feromone ostavlja samo onaj

mrav koji je generirao najbolje rješenje. Slika [Slika 3.1] prikazuje osnovne korake opisanog postupka.



Slika 3.1. Kostur algoritma optimizacije kolonijom mrava

3.2. Donošenje odluke

Već je spomenuto kako mrav koristi dva pristupa kod donošenja odluke o konstrukciji rješenja, heurističkom i feromonskom. Sada treba definirati spomenute pristupe te dobiti zajedničku funkciju. Treba istaknuti da je jedan od temelja ACO algoritma (a tako i stvarnih mrava) određena doza nasumičnosti koja se potiče zbog potrage za boljim rješenjima. Taj će se princip primijeniti i ovdje.

Interno se pohranjuju podaci o doprinosima heuristike, feromona, te kombinacije u matricama dimenzija (ispiti , termini). Promjena vrijednosti feromona poziva se u odnosu na prethodnu vrijednost, pa se rekurzivno definira funkcija:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \Delta\tau_{ij} \qquad (4)$$

gdje ρ predstavlja faktor isparavanja, a $\Delta\tau_{ij}$ promjenu koja će se obaviti nad tim poljem. U slučaju da je taj potez dio kvalitetnog rješenja, vrijednost će se povećati za neku vrijednost, dok će u suprotnom izraz poprimiti vrijednost 0.

Drugu ćemo komponentu računati na nešto teži način, a i funkcija će se pozivati puno češće nego ona za feromone:

$$\eta_{et}(A_{i-1}) = 1/(1+Ver(Ai-1)) \qquad (5)$$

gdje $V_{et}(A_{i-1})$ odgovara broju novih kršenja ograničenja koje bi uzrokovalo dodavanje ispita e u termin t u parcijalno konstruirano rješenje A_{i-1}. Vidimo da će

ovdje mrav morati gledati "unaprijed" da bi vidio što bi se dogodilo kada bi probao neko rješenje.

Nakon definirane obje komponente može se oblikovati vjerojatnost da će neki mrav pridružiti ispit e terminu t na temelju prethodno konstruiranog dijela rješenja i pripadajućih matrica. Treba napomenuti kako u nekom koraku mrav promatra samo red matrice, a ne cijelu zato što kod parcijalnog dodavanja rješenja mrav već zna koji ispit je na redu za smještaj u raspored. Formula je u nastavku:

$$p_{ei,t}(\tau(A_i-1),\eta(A_i-1)) = \frac{(\tau_{ei,t}(A_i-1)^{\alpha}\,\eta_{ei,t}(A_i-1)^{\beta})}{\sum_{v\in T}(\tau_{ei,v}(A_i-1)^{\alpha}\,\eta_{ei,v}(A_i-1)^{\beta})} \quad (6)$$

koja jednostavno kaže kako odluka da će mrav ispit e_i pridružiti uz termin t ovisi o matrici (redu) feromona i heuristike prethodnog koraka. Svaki se element u redu težinski potencira faktorima α i β a zatim se proporcionalno odrede vjerojatnosti za svaki termin.

3.3. Max-min mravlji sustav

Max-min mravlji sustav (max-min ant system) je vrsta ACO algoritma na koji se stavlja naglasak u ovom radu.

MMAS su razvili Stützle i Hoos 2000. godine kao nadogradnju AS-a. [14] Iako razlike ne djeluju spektakularno, ipak su se pojavili osjetne razlike uspješnosti kod dva spomenuta pristupa. Osnovna razlika leži u načinu na koji se koriste postojeće informacije (feromoni):

- osvježavanje feromona izvršava samo mrav s najboljim rješenjem;
- tragovi feromona ograničavaju se na interval [τ_{min} , τ_{max}];
- tragovi feromona se inicijaliziraju na gornju granicu, τ_{max}.

Valja istaknuti da se ograničavanje feromona ne vrši skaliranjem, već spuštanjem (odnosno podizanjem) svih vrijednosti izvan granica na odgovarajuće rubove intervala.

Spomenute promjene nastale su iz praktičnih razloga, ali su poduprte logikom. Primjerice, samo najbolji mrav osvježava feromone na temelju vlastitog rješenja kako bismo smanjili vrijeme izvođenja, ali ujedno i usmjerili ostatak kolonije prema

dobrom rješenju. Granice feromona su uvedene zbog prevelikih razlika koje su se stvarale, pa se gubio osnovni koncept nasumičnosti jer je dolazilo do stagnacije. Konačno, inicijalizacija na gornju granicu potiče viši doseg istraživanja mrava na početku izvođenja, te postepeno smanjenje kako se nalaze sve bolja rješenja. Osnovni algoritam prikazan je na slici [Slika 3.2]

```
Procedura ACO algoritma (problem I)
        InicijaliziranjeFeromona(Tmax)
        Dok (vremensko ograničenje nije pređeno)
                Za svakog mrava
                        A0 = {}
                        Za i = 1 do brojIspita
                                Odaberi t na temelju vjerojatnosti
                                Ai = Ai-1 U {(ei, t)}
                        Kraj
                        C ← rješenje
                        Citer ← bolji (C, Citer)
                Kraj
                Citer ← LokalnaPretraga(Citer)
                Cglobal ← bolji (Citer, Cglobal)
                OsvježavanjeFeromona (Cglobal)
        Kraj
        Vrati Cglobal
Kraj
```

Slika 3.2. Max-min mravlji sustav kroz algoritam

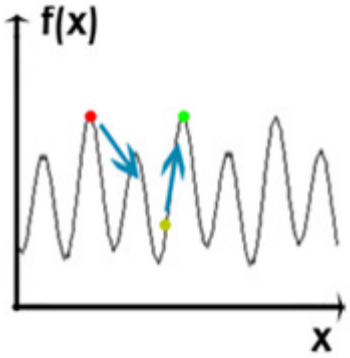
3.4. Memetički algoritmi

Na trenutak ćemo se odmaknuti od optimizacije kolonijom mrava, kako bi se prezentirala kompozicija dva pristupa, koja je posebno popularna u novije vrijeme. Iako i sam ACO predviđa opcionalni korak lokalne pretrage, ovdje se uvodi nova klasa algoritama kod spajanja različitih pristupa i iskorištavanja najboljih karakteristika s obje strane.

Memetički algoritmi su metaheuristika gdje se istražuje susjedstvo rješenja dobivenog nekim od genetskih algoritama. Kako se radi o istraživanju susjedstva,

taj se korak naziva lokalnom optimizacijom gdje se nastoji približiti lokalnom optimumu. Nakon optimizacije rješenja od kojeg se krenulo u taj postupak, vraća se poboljšano rješenje natrag u postupak genetskog algoritma, te se proces ciklički ponavlja. Kao primjeri takvih lokalnih optimizacija mogu poslužiti tabu pretraga (tabu search) ili algoritam velikog potopa (great deluge algorithm).

Važno je uočiti da će genetski algoritam raditi na što širem pretraživanju prostora, pod cijenu vlastite kvalitete, a baš suprotno, postupak lokalne optimizacije će se usredotočiti na uzak prostor rješenja, ali će zato unutar njega pronaći ono najbolje. Zato kažemo da se genetski algoritam bavi istraživanjem, a lokalna optimizacija eksploatacijom. Ilustrativni primjer može se vidjeti na slici [Slika 3.3] gdje crvena točka predstavlja početno rješenje, žuta točka se dobiva nakon genetskog algoritma, a zelena točka nakon lokalne optimizacije. Treba istaknuti da su na primjeru kvalitetnija rješenja veće vrijednosti f(x) što nije uobičajeno. Prednost takvog hibridnog pristupa nad pojedinačnim pokazana je u nekoliko novijih radova. [15]



Slika 3.3. Suradnja dviju faza memetičkih algoritama

3.5. Tabu pretraga

Kako memetički algoritam uvodi pojam lokalne optimizacije, tako valja naći dobar primjer takvog algoritma koji će dobro surađivati s početnom optimizacijom kolonijom mrava. Jedan od takvih koji se često navodi u literaturi i javlja uspješne rezultate jest upravo tabu pretraga.

Osnovni koncept leži u tome da se na temelju početnog rješenja određenim operatorima generira susjedstvo rješenja od kojih se bira ono najbolje. Odabrano rješenje postaje novo početno rješenje koje ulazi u novi krug proširenja. Dodatno,

posjećena stanja spremamo u tabu listu na neko određeno vrijeme. Razlog pamćenja tih podataka je izbjegavanje kruženja unutar postupka. Iznimka kada se može posjetiti stanje iz tabu liste jest kada bi se generiralo globalno najbolje rješenje u tom trenutku, što nazivamo aspiracijskom strategijom.

Kod definiranja susjedstva nekog rješenja, možemo koristiti dva načina. Prvi je jednostavan i definira susjedstvo kao zamjenu termina kod jednog ispita. Drugi se temelji na Kempovim lancima koje je uveo Morgenstern. [16] Takvo susjedstvo je skup svih rasporeda koji se od početnog razlikuju u jednoj zamjeni termina između dviju grupa ispita koji se nalaze unutar svojih termina. Kako bi se zadržala prihvatljivost rješenja, drugi tip susjedstva se obično proširuje zahtjevom da se zamjene svi ispiti unutar dva termina. Slike [Slika 3.4] i [Slika 3.5] prikazuju oba primjera susjedstva.

Treba istaknuti kako tabu pretraga samostalno primijenjena na problem rasporeda ispita daje također odlične rezultate. [1] U samostalnim varijantama, tabu pretraga ima kratkotrajno i dugotrajno pamćenje kako bi se u obzir uzimala komponenta globalnog, a ne samo lokalnog.

predmeti	MAT	FIZ	ELE	BAZE	OKOLIŠ	JAVA
termini	T1	T2	T3	T1	T2	T2

predmeti	MAT	FIZ	ELE	BAZE	OKOLIŠ	JAVA
termini	T1	T2	T1	T1	T2	T2

Slika 3.4. Primjer elementarnog susjedstva

predmeti	MAT	FIZ	ELE	BAZE	OKOLIŠ	JAVA
termini	T1	T2	T3	T1	T2	T2

predmeti	MAT	FIZ	ELE	BAZE	OKOLIŠ	JAVA
termini	T1	T1	T1	T2	T1	T2

Slika 3.5. Primjer složenog susjedstva

3.6. O kvaliteti i brzini rješenja

Dvije suprotstavljene strane kod ovakvih pristupa rješavanju problema su kvaliteta dobivenog rješenja i vrijeme potrebno da se dođe do konačnog rješenja. Čim smo odabrali koristiti metaheuristički postupak za razliku od direktnog pristupa, znali smo da se odričemo potencijalne kvalitete rješenja u zamjenu za brzinu izvođenja, jer bi direktan pristup s obzirom na veličinu prostora pretrage mogao potrajati neefikasno dugo.

No, iako već u početku dajemo prednost brzini nad kvalitetom, možemo još promatrati ima li smisla davati prednost kvaliteti ili brzini u okviru ACO algoritma na konkretnom problemu rasporeda ispita. Prvo što pada na pamet jest činjenica da se raspored ispita provodi jednom po semestru (ili nekoliko puta), pa možemo reći da definitivno ne postoji pritisak nužnosti brzine koji postoji recimo kod transportnih sustava. No, ipak treba paziti da postupak ne potraje predugo jer problem može biti i dinamički – uvjeti se mogu mijenjati, studenti se mogu ispisivati, učionice se mogu koristiti i za nešto drugo, stoga postoji i realna mogućnost da će se postupak provoditi više puta.

4. Implementacija rješenja

Glavna misao kojom je ovaj program vođen jest da se postigne što je veća moguća parametrizacija aplikacije, kako bi se mogle testirati razne opcije i pronaći ona najbolja. Obzirom da se na mnogo mjesta u literaturi spominje da je kod metaheurističkih pristupa vrlo nezahvalno direktno u kod unositi zahtjeve, onda se i o tome vodilo računa gdje se moglo.

4.1. Format ulaza

Kako je opisan problem rasporeda ispita na FER-u u poglavlju [2.3], ovdje se samo navodi specifičan format kojeg ulazna datoteka mora zadovoljavati. Popis sadržaja dan je slijedno u tablici [Tablica 4.1]. Popisi spomenuti u tablici ostvaruju se nizanjem elemenata odvojenih zarezom, bez dodatnih praznina. Iznimka su popis clustera, što je i naznačeno u tablici.

Tablica 4.1. Sadržaj ulazne datoteke za FER-ov problem

rbr	ponavljanje	podaci
1		brojPredmeta
2	∀ predmet	idPredmeta#nazivPredmeta#popisStudenata
3		brojTermina
4	∀ termin	datum vrijeme#kapacitet#oznakaDana#idTermina
5		brojPredefiniranih
6	∀ predefinirani	datum vrijeme#idPredmeta
7		brojClustera
8	∀ cluster	popisPredmeta (idPredmeta odvojeni s "#")
9		brojRestrikcija
10	∀ restrikcija	idPredmeta#popisTermina (idTermina)

4.2. Format izlaza i zapis rješenja unutar programa

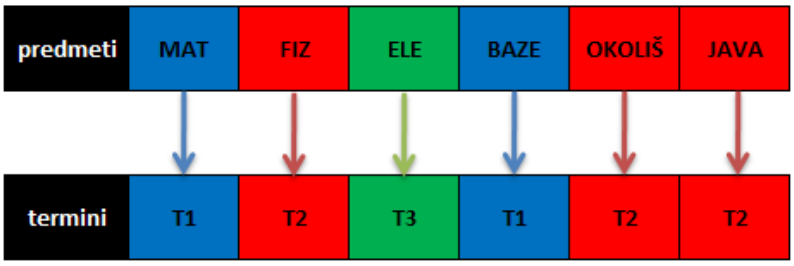
Format FER izlaza jest nešto jednostavniji od ulaza jer se sastoji od redaka koliko ima ispita, a svaki se definira na sljedeći način: datum vrijeme konstanta nazivPredmeta idPredmeta, gdje se za konstantu uzima 2. Što se tiče prikaza unutar programa, rješenje se pohranjuje unutar klase "Chromosome" koja sadrži

polje predmeta _cCourses i polje termina _cTerms. Valja napomenuti da se spomenute klase sastoje od cjelobrojnih podataka, pa su pogodni za rad na velikom broju iteracija.

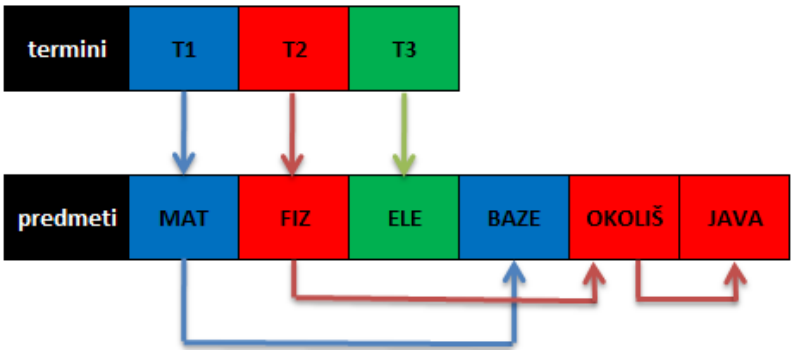
CTerm tip podatka sastoji se od četiri broja. _firstCourse i _lastCourse su indeksi prvog, odnosno posljednjeg predmeta kojeg smo vezali uz ovaj termin. _numberOfCourses broji predmete, dok _numberOfStudents broji studente pridružene terminu. Lista termina u _cTerms uvijek će biti u jednakom poretku jer nam to nije bitno. Ono što je bitno jest da mi preko termina dohvaćamo polje predmeta koje smo pridružili tom terminu.

Ccourse sadrži _index, indeks svog predmeta, zatim _previous i _next, indekse prethodnog, odnosno sljedećeg predmeta koji s ovime dijeli termin, te _term koji pohranjuje indeks termina koji je pridružen predmetu. Dakle, zanima li nas trenutna popunjenost nekog termina, jednostavno ćemo dohvatiti željeni termin (poredak je uvijek jednak), a zatim ćemo iz tog termina pratiti _firstCourse i naći odgovarajući prvi predmet. Do ostalih predmeta dolazimo praćenjem indeksa _next od trenutno promatranog predmeta.

Želimo li pročitati rješenje koje kromosom predstavlja, treba samo proći kroz sve elemente polja predmeta i pročitati njima pridružene termine zapisane u _term varijabli. Ilustrativni primjer o obostranom funkcioniranju može se vidjeti na slikama [Slika 4.1] i [Slika 4.2]. Strelice simboliziraju pokazivače, iako je u stvarnosti rješenje implementirano preko cjelobrojnih indeksa koji se pohranjuju.



Slika 4.1. Princip povezivanja termina uz predmete



Slika 4.2. Princip dohvaćanja predmeta po terminima

4.3. Priprema problema

Prije nego li se krene u ACO postupak, potrebno je napraviti neke korake kako bi se osigurao što efikasniji nastavak izvođenja, bez nepotrebnih višestrukih izračuna. Također od pet elemenata FER-ovog problema opisanih u poglavlju [2.3], dva se zbog sličnosti mogu spojiti s ostalima, uz dodatne modifikacije.

Konkretno, predmeti unutar clustera koji se moraju održati istovremeno su istovjetni jednom zajedničkom predmetu koji sadrži sve uključene studente koji se treba smjestiti u jedan termin. Nakon što se taj postupak, JoinClusters() obavi, više se ne mora paziti na to da se ti predmeti smjeste u isti termin. Logična pretpostavka da bi ovo funkcioniralo jest da clusteri predmeta nemaju interne presjeke studenata, inače zahtjev korisnika za stvaranjem takvog clustera ne bi imao smisla.

Dodatno, predefinirani uvjeti kažu da neki predmet treba ići u određeni (jedan) termin. S druge strane, restrikcije kažu da neki predmet treba ući u jedan od ponuđenih termina. Sličnost ta dva zahtjeva je očita pa se u postupku ConvertPredefinedToRestrictions() predefinirani uvjeti stavljaju u restrikcije i tako dobivamo zajednički popis ograničenja.

U ovom trenutku algoritam treba raditi samo s tri liste početnog problema – popis predmeta, popis termina i popis restrikcija. Konačno, kao najveću uštedu vremena postići ćemo kod stvaranja matrice konflikata prije nego li se uopće uđe u ACO postupak. Kako je ta matrica statična, ne bi imalo nikakvog smisla unutar postupka svaki put računati tko i s kime ima konflikte. A obzirom da postupak stvaranja matrice konflikata može potrajati i do nekoliko sekundi, onda ćemo

jednom dobivenu matricu pohraniti na računalu, a kod sljedećih pokretanja aplikacije čitati iz te datoteke. U slučaju modifikacija ulazne datoteke, korisnik mora paziti da se prisili stvaranje nove matrice, kako se ne bi koristila ona stara.

4.4. Strategija poretka ispita

Uobičajeno je u postupak pripreme problema dodati i izračun poretka ispita prema kojem će mravi konstruirati rješenja, no ovdje to nije učinjeno iz jednostavnog razloga. Promatrat će se pet načina poretka, od kojih su četiri statične i jedna dinamička koja se mijenja ovisno o izvođenju algoritma. Stoga, da se ne rade prevelike razlike u vremenu izvođenja ili implementaciji, napravljen je zajednički okvir za svih pet strategija poretka koji se može vidjeti na slici [Slika 4.3]. Pregled ostvarenih strategija može se naći u tablici [Tablica 4.2].

```
Procedura GradnjaRješenja
        Radi
                sljedećiIspit = DohvatiIspit (strategija, preostaliIspiti)
                IzbaciIspit(sljedećiIspit, preostaliIspiti)
                trenutnoRješenje.GradiParcijalno(sljedećiIspit, podaci)
        Dok (preostaliIspiti.Dužina > 0)
Kraj
```

Slika 4.3. Okvir gradnje rješenja za jednog mrava za sve strategije poretka

Tablica 4.2. Popis implementiranih strategija poretka

ozn	strategija	opis
A	predefinirani poredak	po redu kako su navedeni u ulaznoj datoteci
B	nasumični poredak	korištenjem random funkcije
C	prema broju studenata	od predmeta s većim brojem studenata
D	prema broju konflikata	od predmeta koji imaju više konflikata s drugim predmetima(statički)
E	prema broju slobodnih termina	od predmeta koji imaju najmanje termina na raspolaganju (dinamički)

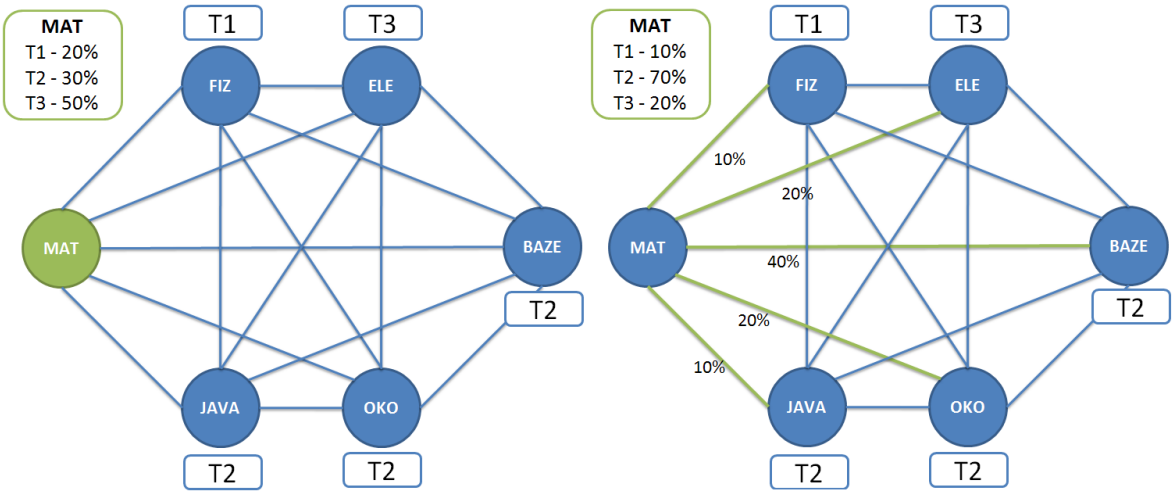
4.5. Matrice feromona i heuristike

Kod implementacije matrica prvo treba donijeti odluku hoće li se računati vrijednost nad čvorovima ili lukovima grafa. Podsjetimo se da u grafu čvorove predstavljaju ispiti, lukove konflikti među ispitima, a zadatak nam je pridružiti vrijednosti (termine) čvorovima (ispitima). Imajući to na umu, odaberemo li strategiju pamćenja vrijednosti nad čvorovima, morat ćemo za svaki čvor definirati vrijednosti za svaki ispit. Vrijednost nad nekim čvorom za neki termin govori kolike su šanse da će taj ispit biti pridružen baš tom terminu.

S druge strane, pridruživanje lukovima znači da stvaramo parove čvorova (ispita), pa će nam zapisane vrijednosti govoriti koliko je poželjno staviti dva ispita u isti termin (nebitno koji). Nedostatak ovog pristupa je očit, obzirom da ovdje ne uzimamo u obzir slaba ograničenja. Mi možemo reći da je dobro staviti dva ispita zajedno jer oni nemaju međusobnih konflikata, ali nam to ništa ne govori o pitanju jesu li ta dva ispita raspoređena blizu nekih drugih koji bi stvorili kršenje slabih ograničenja.

Stoga je odluka jednostavna i bira se prvi pristup, gdje ćemo stvoriti matricu koja se sastoji od liste redova dužine koja odgovara broju ispita, dok će svaki red biti dužine broja termina. Iako je matrica definirana preko niza nizova, možemo reći da su njene dimenzije (broj_ispita , broj_termina). Osnovne zamjerke odabranom pristupu temelje se na problemu ukoliko se rano kod izvođenja algoritma pronađe vrlo loše rješenje, pa se ostatak mrava usmjerava na to pogrešno i usporava se napredak prema boljim rješenjima. Nedostatke takvog pristupa obično poništi postupak lokalne pretrage. [17]

Ilustracija primjera za oba slučaja može se vidjeti na slici [Slika 4.4]. Vjerojatnosti lijevog slučaja se očitaju iz matrice, dok se vjerojatnosti za desni slučaj zbrajaju ovisno o vrijednostima na lukovima i pripadajućim terminima kojima su susjedi pridruženi.



Slika 4.4. Primjer prikaza izračuna vjerojatnosti za pristup čvorova (lijevo) i pristup lukova (desno)

4.6. Osvježavanje feromona

Može se postaviti pitanje zašto se osvježavanje feromona ne bi vršilo tijekom parcijalne konstrukcije rješenja, no to na primjeru zadanog problema ne bi imalo smisla zato što kvaliteta dijela rješenja ne mora biti indikator kvalitete konačnog rješenja. Pogotovo iz razloga što se kompletna evaluacija može provesti tek nakon što su svi ispiti raspoređeni.

Na pitanje trenutka osvježavanja feromona upravo je odgovoreno, izvršavat će se na kraju svake iteracije, a na pitanje tko će izvršiti osvježavanje, koristi se elitistička strategija koju preporučuje MMAS. O pitanju hoće li osvježenje izvršiti globalno najbolji ili iteracijsko najbolji mrav, odlučuje korisnik. Postoje još dvije kombinacije – da se osvježenje izvrši nad oba mrava (i globalno i iteracijsko najbolji) ili da se prepusti vjerojatnosti koji će od njih ostaviti trag. Za taj je slučaj korištena sljedeća formula koja određuje vjerojatnost da će biti odabran iteracijski najbolji:

$$p(iteracijski) = \gamma * q(c_{globa\ln i}) / q(c_{iteracijski}) \qquad (7)$$

gdje γ predstavlja faktor istraživanja, a q kvalitetu rješenja. Obzirom da je cilj što manja vrijednost kvalitete rješenja, vjerojatnost će se nalaziti u intervalu [0 , γ]. [18]

Osvježavanje se provodi u dva koraka. Prvi je identičan za sva polja matrice i odgovara isparavanju, gdje će svaki element biti pomnožen (1-ρ) faktorom, a zatim

će se u drugom koraku određena polja povećati za određeni faktor $\Delta\tau$ koji ponovno odlučuje korisnik. Skupljena su četiri najčešća povećanja iz literature i implementirana u programskom rješenju.

4.7. Inicijalno rješenje

Iako u genetskim algoritmima po definiciji ne bi trebalo biti bliska manipulacija inicijalnog rješenja, to je ipak čest slučaj obzirom da se dobiva na uštedi vremena, a i često se i dobivaju bolja rješenja. Ipak, u ovom radu se inicijalno rješenje ne generira nekim posebnim postupkom, ali je dopuštena opcija korisniku da algoritam započne s već postojećim rješenjem. Tako se može nastaviti poboljšavati neko prethodno dobiveno rješenje, ili pak usmjeriti nova rješenja u dobrom pravcu.

4.8. Evaluacija rješenja

Carter je 1996. Predložio kako vrednovati zahtjev prvog reda. Ako definiramo da će se kažnjavati bliskost do N termina, onda se faktor kazne za dva termina koja su udaljena za n računa prema: $P_n = 2^{N-n}$. Preporučeno je uzimati N=5. [9]

Za razliku od toga, FER koristi udaljenosti na razini dana, a ne termina. Tako se kažnjavaju udaljenosti od 0 (faktor 4) i 1 (faktor 1) dan, svaki pomnožen svojim faktorom. Naravno, faktori se u konačnici množe s brojem studenata koji sudjeluju u konfliktu.

Evaluacija za konflikte nultog reda prolazi kroz sve termine i unutar termina za svaki par pridruženih predmeta traži zajedničke studente. Evaluacija prvog reda mora još ući u petlju dubljeg reda, gdje za svaki termin mora proći sve susjedne termine (udaljene do jedan dan) i brojati njihove konflikte. Evaluacija drugog reda slično nultom, za svaki termin i pripadajuće predmete broji studente i provjerava je li kapacitet pređen.

Treba napomenuti da evaluacija podrazumijeva da su ostali zahtjevi (predefinirani, restrikcije i clusteri) uvaženi te ne postoji provjera jesu li ta pravila zadovoljena jer su prihvaćanja tih zahtjeva implementirana unutar koda putem zajedničke liste restrikcija.

U praksi se često javlja situacija da je u jednom trenutku nemoguće proizvesti prihvatljivo rješenje. Tada se proširuje skup resursa (npr. dodaje se novi termin) u koji se rezervira problematični ispit, a zatim to evaluacijska funkcija oštro kazni. [10] Za razliku od tog pristupa, algoritam implementiran u ovom radu u slučaju neizbježne neprihvatljivosti nastavi normalnim putem. Kao rezultat dobit će se neprihvatljivo rješenje, ali će biti kažnjeno velikim faktorom, u nadi da će se u narednim koracima pronaći bolje rješenje.

4.9. Delta evaluacija

Delta evaluacija se koristi kada se želi uzeti u obzir mnogo sličnih (susjednih) rješenja koja trebamo evaluirati. Umjesto da ispočetka računamo sva kršenja za neko rješenje, iskoristit ćemo činjenicu da nam je poznata evaluacija sličnog rješenja.

Delta evaluacija se konkretno koristi u trenutku izračuna vrijednosti za matricu heuristike, prema formuli (5). Obzirom da se svakim novim dodavanjem ispita mora proći svaki termin i pogledati što bi se dogodilo kada bismo taj novi ispit dodali baš tom terminu, ovdje ćemo dobiti značajnu uštedu vremena. Nakon svih provjera, vratimo se do mrava, kažemo mu kakve su kvalitete potencijalnih rješenja i onda se donosi odluka. Upravo ovdje koristimo tu činjenicu da imamo izračunatu evaluaciju sličnog rješenja – i to onog dobivenog do tog trenutka (prije dodavanja trenutnog ispita).

Delta evaluacija će umjesto da prolazi kroz sve ispite i sve termine proći samo kroz posljednji dodani termin i provjeravati koje je konflikte uzrokovao. U toj se situaciji vrijednost funkcije ne može smanjiti, može se samo povećati i to nam olakšava situaciju.

Osim na spomenutom mjestu, delta evaluacija može se koristiti i kod lokalne pretrage, obzirom da se ona temelji na pretraživanju susjedstva.

5. Analize i mjerenja

Iako se u ovom radu ne rade usporedbe s drugim algoritmima, ipak se može spomenuti da je u literaturi pokazano kako se ACO može do neke granice mjeriti s najboljim algoritmima na ETP-u, ovisno o složenosti problema. [19]

Često se ističe da se podešavanje parametara ne bi trebalo prepuštati krajnjem korisniku, jer se ne može očekivati da će svaka osoba koja izrađuje rasporede za neki fakultet imati dovoljno znanja o faktorima poput α, β ili koliko mrava treba staviti u koloniju. Ipak, postoje neki parametri koje je poželjno da korisnik postavlja obzirom da je on taj koji definira zahtjeve koji se traže od rješenja. Primjer takvih parametara može biti odabir ograničenja, faktori kazni za kršenja ograničenja ili vrijeme izvođenja.

Kako se u ovom radu usredotočuje na analizu ACO algoritma i na njegovu što uspješniju provedbu, tako će se ovdje promatrati odgovarajuća podešavanja parametara. Na temelju tog znanja možda će biti lakše konstruirati dinamičke funkcije za određivanje istih parametara, pa će krajnji korisnik biti do kraja oslobođen potrebe da posjeduje znanja iz ove domene.

Bitno je napomenuti da su se sva mjerenja u nastavku provodila 5 puta iz čega je izračunat prosjek globalno najboljeg rješenja nakon 100 iteracija, osim ako to nije navedeno drugačije. Sva su mjerenja provedena za problem FER-a iz školske godine 2008./2009. ljetnog semestra.

5.1. O utjecaju broja mrava

Logično je bilo prvo proučiti baš utjecaj broja mrava na kvalitetu rješenja, iz jednostavnog razloga što taj parametar najviše utječe na duljinu izvođenja postupka. Zato će se odmah na početku kvalitetno odrediti odgovarajući broj mrava, kako bi se dobilo na uštedi vremena kod ostalih mjerenja.

Kod ovog se parametra zapravo pitamo koliko je mrava dovoljno da dobivamo kvalitetna rješenja. Jer sigurno kako povećavamo broj mrava, kvaliteta neće padati, samo može rasti. Ali nas opet zanima je li taj rast kvalitete dovoljan u odnosu na vrijeme koje se produžuje s porastom broja mrava.

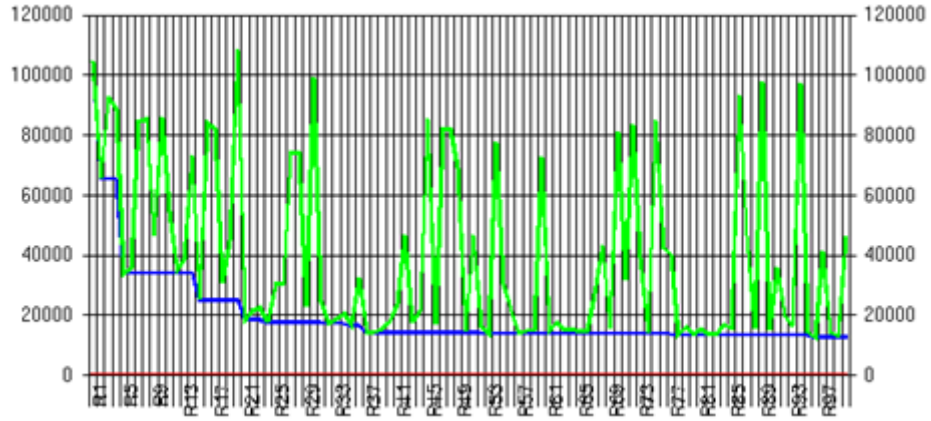
Mjerenja u tablici [Tablica 5.1] su pokazala da se već s pet mrava postižu dovoljno kvalitetna rješenja. Do te granice smo se osigurali da je kolonija dovoljno velika da ignorira neuspjele pokušaje pojedinaca, a svako daljnje dodavanje novog mrava ne utječe bitno na poboljšanje rješenja. Zato ćemo u nastavku mjerenja koristiti pet mrava.

Tablica 5.1. Utjecaj broja mrava na kvalitetu rješenja

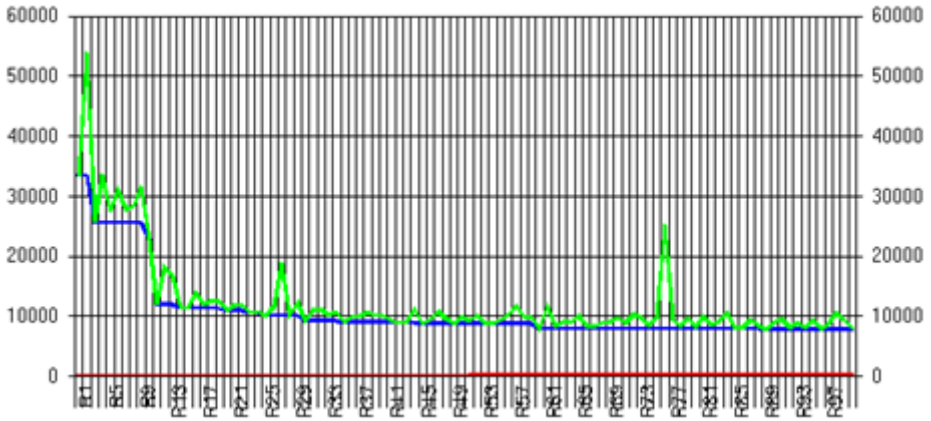
Mravi	1	2	3	4	5	6	7	8	9	10	15
1	14489	12711	12081	10261	12575	12264	9051	9977	10489	11695	10392
2	16353	16857	10645	13091	9524	10763	9013	11119	9439	9899	9892
3	15122	13042	11114	11845	10951	11147	9908	9739	11754	10883	9009
4	15206	12089	14307	11053	9601	9657	12682	11790	8820	11237	9721
5	14309	14152	13092	10170	10052	9739	9001	11202	9873	9766	8910
AVG	15096	13770	12248	11284	10541	10714	9931	10765	10075	10696	9585
BEST	14309	12089	10645	10170	9524	9657	9001	9739	8820	9766	8910

Dodatno se možemo osvrnuti na tijek događanja s malim brojem mrava (jednim) i velikim brojem (deset), što prikazuju slike [Slika 5.1] i [Slika 5.2]. Manje kolonije će tako više oscilirati po iteracijama, značajno se udaljavajući na trenutke od globalno najboljeg rješenja. S druge strane, veće će kolonije oscilirati samo u prvih nekoliko iteracija zbog visokog početnog feromona, dok će se kasnije uglavnom koncentrirati oko globalnog maksimuma. Iz toga bi se moglo dati naslutiti kako manje kolonije ostavljaju više mjesta istraživanju, no znamo da to nije točno, obzirom da svaki od deset mrava konstruira svoja rješenja, pa se dakle deset puta više rješenja prođe nego u prvom slučaju. Također se može vidjeti da je veća kolonija i postigla ukupno bolji rezultat (8000 naprema 15000).

Još treba istaknuti kako određivanje broja mrava može ovisiti o zadanom problemu, pa se ponekad predlažu i funkcije za broj korištenih mrava umjesto konstanti – primjerice da bude onoliko mrava koliko ima ispita ili termina. [15]



Slika 5.1. Pokrenuti primjer za koloniju s jednim mravom



Slika 5.2. Pokrenuti primjer za koloniju s deset mrava

5.2. O utjecaju broja iteracija

Ova će mjerenja biti nešto preciznija zbog drugačijeg pristupa. Naime, umjesto da mijenjamo broj iteracija i pokrećemo zasebno postupke, ovdje ćemo provesti pet mjerenja, svaku do 200 iteracija, te ćemo na kontrolnim točkama provjeriti vrijednost globalne funkcije. Ukoliko u nekom trenutku rješenje počne stagnirati, to nam daje naslutiti da daljnje izvođenje neće previše pomoći kvaliteti rješenja. U tom su trenutku vjerojatno mravi već potrošili feromone za nova istraživanja, dok ih svako novo istraživanje usmjerava k već postojećem najboljem rješenju.

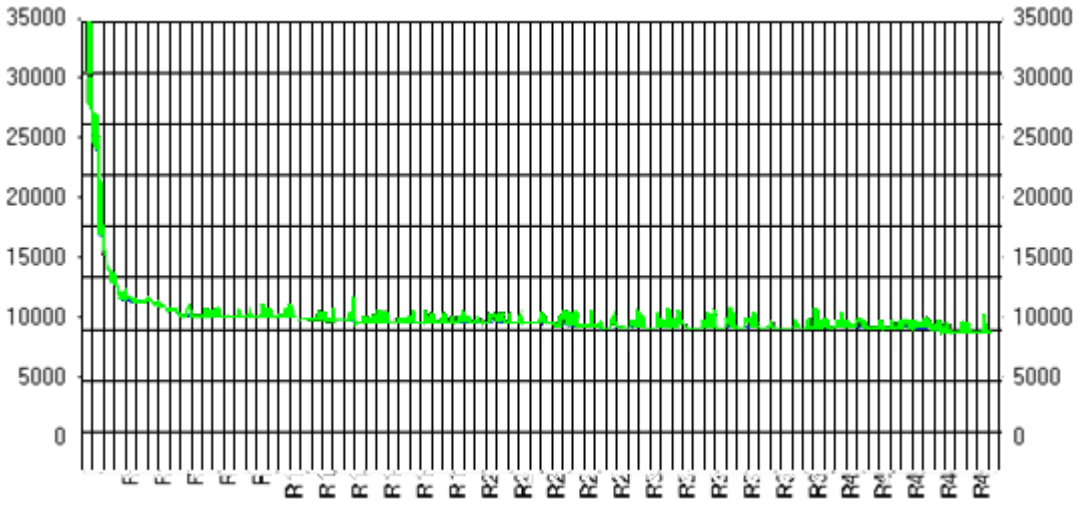
Tablica [Tablica 5.2] prikazuje dobivene rezultate iz kojih se zaključuje da iako neke promjene postoje i u kasnijim intervalima, ipak možemo reći da će se do 100. iteracije iskristalizirati rješenje. Taj je broj inače i često spominjan u literaturi, pa će se zato i koristiti u nastavku.

Obzirom da su u tablici [Tablica 5.2] crvenom bojom označena najbolja rješenja u stupcu, možda bi bilo za očekivati da će biti označena polja istog retka, no ipak uočavamo taj element nepredvidljivosti gdje jedno izvođenje može krenuti s boljim rješenjima, ali da ga u kasnijim koracima neko drugo izvođenje prestigne.

Tablica 5.2. Utjecaj broja iteracija na kvalitetu rješenja

Iter.	10	20	30	40	50	65	80	100	125	150	200
1	22913	15099	13178	11071	10613	9701	8424	8358	8292	8292	8292
2	19649	17362	14309	13587	13543	13520	11174	8873	8838	8838	8720
3	20930	15445	13984	12318	11982	11943	11799	10059	9995	9710	9692
4	23964	11599	9699	9649	9649	9441	9303	9085	9016	9016	9016
5	21646	14304	12716	12716	12676	11809	11503	10620	9638	8993	8598
AVG	21820	14762	12777	11868	11693	11283	10441	9399	9156	8970	8864
BEST	19649	11599	9699	9649	9649	9441	8424	8358	8292	8292	8292

Slika [Slika 5.3] daje primjer izvođenja kroz 500 iteracija, da se naglasi očitost stagnacije rješenja nakon određenog broja iteracija. Iako to ne mora nužno biti broj 100, ipak smo odredili da ćemo radije koristiti manji broj iteracija, ako su rješenja približno jednaka.



Slika 5.3. Pokrenuti primjer kroz 500 iteracija

5.3. O utjecaju poretka ispita

Popis i objašnjenje mogućih poredaka naveden je u tablici [Tablica 4.2] a u tablici [Tablica 5.3] dani su rezultati usporedbe spomenutih strategija. Oznake slova u dvjema tablicama se međusobno poklapaju.

Iako se očekivalo da strategija poretka neće značajno doprinositi kvaliteti rješenja, ipak se pokazalo suprotno. Također, kada je već jedna strategija pokazala prednost, bilo je za očekivati da će to biti jedina dinamička strategija od navedenih koja se prilagođava trenutnoj situaciji (oznaka E). Ipak, najbolje rezultate postigla je strategija predefiniranog poretka (oznaka A), odnosno pridjeljivanja termina ispitima redom kako su navedeni u ulaznoj datoteci. Rezultat bi bio shvatljiv ukoliko je popis predmeta u FER-ovoj datoteci upravo onakav koji daje bolje rezultate, iako se u pravilu ovakav pristup nastoji izbjegavati.

Općenito se preporuča strategija prema broju slobodnih termina također poznat pod nazivom najvišeg stupnja zasićenja (highest degree of saturation) koji redom uzima čvorove koji imaju najmanje mogućnosti za odabir na temelju svojih susjeda i njihovih odabira.

Tablica 5.3. Utjecaj redoslijeda odabira ispita na kvalitetu rješenja

Red.	A	B	C	D	E
1	8617	9174	10229	9302	8410
2	7719	10316	9404	9465	11696
3	7998	10290	8417	10612	8961
4	8186	10438	9592	10070	8682
5	7596	9908	8278	9085	9184
AVG	8023	10025	9184	9707	9387
BEST	7596	9174	8278	9085	8410

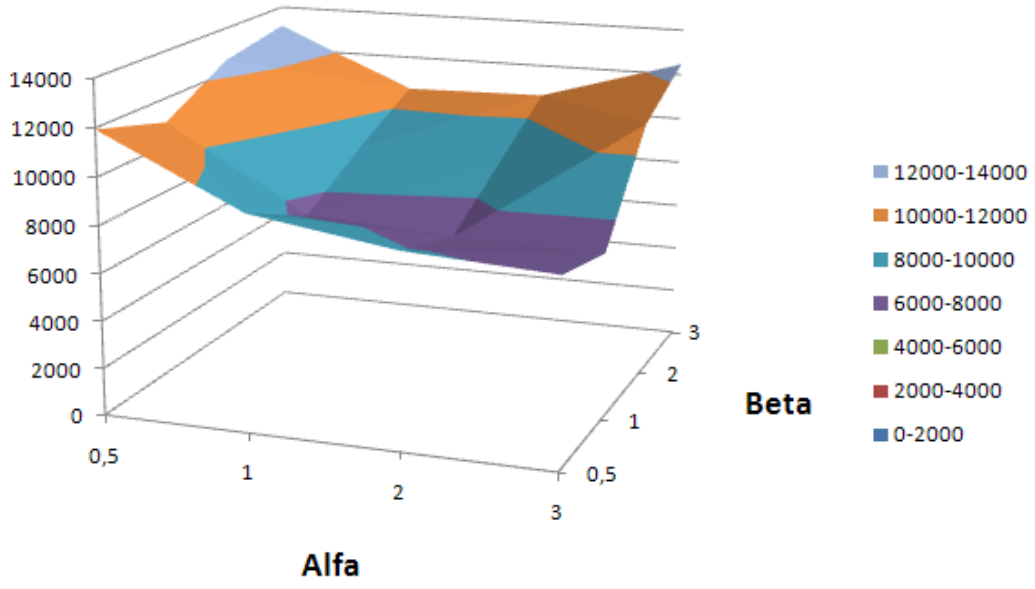
5.4. O utjecaju α i β

U ovim se mjerenjima inicijalno krenulo od fiksiranja parametra α na uobičajenu vrijednost 1, pa se promatrala ovisnost kvalitete rješenja o β, u kombinaciji sa i bez uključene opcije prisile ispravnog rješenja. Visoki β osigurava pridržavanje jakih ograničenja, obzirom da oni imaju vrlo visoke kazne. No, uključi li se opcija prisilne

ispravnosti, onda više nema potrebe za velikim vrijednostima parametra β, obzirom da će se sada usredotočiti samo na slaba ograničenja.

U takvom su se mjerenju rezultati pokazali najbolji za β = 1 (uz α = 1). U naprednijem mjerenju, koje je priloženo na slici [Slika 5.4], paralelno se mijenjaju faktori α i β kako bi se pronašla najbolja kombinacija doprinosa feromona i heuristike kod odlučivanja. U tim je mjerenjima opcija prisile ispravnosti rješenja uvijek bila uključena, pa je bilo za očekivati da neće biti potrebne velike vrijednosti za β.

Promatramo li rezultate primjećujemo udubljenje ljubičaste boje, odnosno područje koje daje bolje rezultate od drugih kombinacija α i β. Promatranjem po α vidimo da male vrijednosti ne daju dobre rezultate, dok promatranjem po β vidimo da velike vrijednosti ne daju dobre rezultate. Valja napomenuti da su mjerenja provedena na uskom rasponu α i β iz razloga što povećanjem raspona broj potrebnih mjerenja naglo raste, a u literaturi se obično uzimaju male vrijednosti za te parametre. Ova mjerenja daju najbolje rezultate za α = 3 i β = 1.



Slika 5.4. Utjecaj α i β faktora na kvalitetu rješenja

5.5. O utjecaju ρ, τ_{min} i τ_{max}

Kada su feromoni u pitanju, postoje dva pojma koja treba imati na umu, pojačanje (intensification) i raznolikost (diversification). [20] Pojačanje se ostvaruje dodavanjem nove vrijednosti od strane nekog mrava, dok se raznolikost ostvaruje isparavanjem. Pojačanje usmjerava mrave ka dobrom rješenju, dok raznolikost pomaže mravima da zaborave prethodna rješenja koja su bila lošija od novopronađenih. Zato je bitno pronaći dobru mjeru između pojačanja i raznolikosti koju ostvarujemo preko faktora $\Delta\tau_{ij}$ i ρ u formuli (4). Faktor ρ se promatra u ovom poglavlju, a $\Delta\tau_i$ u sljedećem.

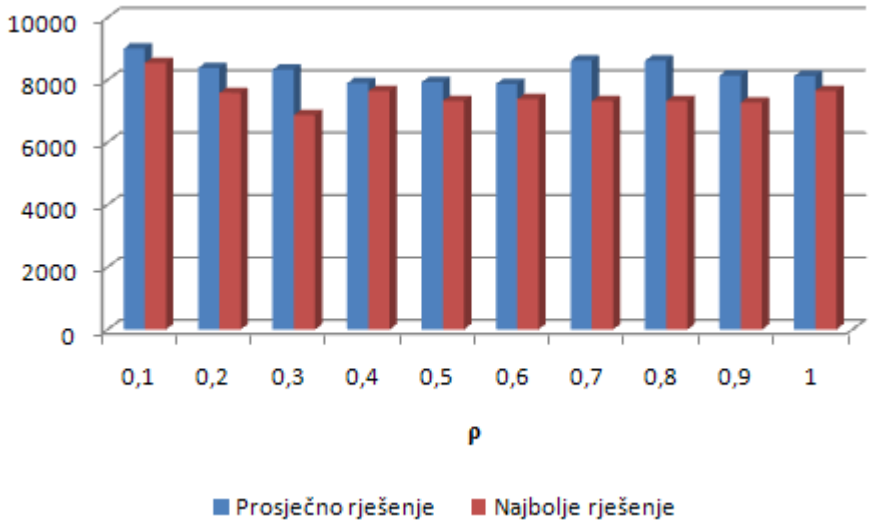
Valja istaknuti da se u formuli (4) koristi faktor 1-ρ, a ne ρ, ali je riječ o uobičajenom načinu zapisa. Ovo će promatranje zapravo mijenjati samo jedan faktor, obzirom da je τ_{min} statičan – ima ga jedino smisla postaviti blizu nule. Kada bi bio jednak nuli, tada bi se obračunavale vrijednosti koje su bile aktualne prije mnogo iteracija, a ovako to izbjegavamo. S druge strane, τ_{max} se prema osnovnoj ideji MMAS algoritma računa upravo na temelju faktora ρ i to formulom:

$$\tau_{\max} = 1/\rho \tag{8}$$

Sada nam samo preostaje mijenjati vrijednosti parametra ρ koji je ionako ograničen intervalom [0 , 1]. Tablica [Tablica 5.4] prikazuje dobivene rezultate, a slika [Slika 5.5] ih prikazuje grafički, s time da je uklonjeno jedno mjerenje kako bi se dobio jasniji prikaz ostalih. Promatramo li ekstreme, uočavamo da samo jedan od njih ima ekstremno ponašanje. Za vrijednost $\rho = 0$, odnosno (1-ρ) = 1 dobivamo osjetno loše vrijednosti. To je iz razloga što taj slučaj opisuje situaciju gdje se prethodni loši potezi ne zaboravljaju i nikada se neće smanjiti. U tom bi slučaju τ_{max} trebao težiti u beskonačnost, pa je korištena vrijednost 1000, u čemu opet leži razlog neuspjeha, jer je već istaknuto u prethodnim poglavljima da je poželjno držati feromone unutar međusobno mjerljivih granica. S druge strane, ekstrem ρ=1, odnosno (1-ρ) = 0 predstavlja zaboravljanje prethodnih koraka i usredotočenje samo na trenutno globalno najbolje rješenje koje će jedino imati vrijednost različitu od nule. Ostale vrijednosti unutar intervala ukazuju na optimalnost parametra u području [0.4 , 0.6], iako se u literaturi dosta često koristi parametar ρ = 0.3.

Tablica 5.4. Utjecaj ρ, τ_{min} i τ_{max} faktora na kvalitetu rješenja

Ro	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
Tmin	2E-04	2E-04	2E-04	2E-04	2E-04	2E-04	2E-04	2E-04	2E-04	2E-04	2E-04
Tmax	1000	10	5	3,33	2,5	2	1,67	1,43	1,25	1,11	1
1	22416	8913	8879	6864	7858	8630	8285	9241	9241	8705	8625
2	22061	8544	9120	8430	7807	7558	7935	9996	9996	7274	7977
3	22257	8927	7582	8990	7688	7324	7604	7316	7316	8152	7713
4	22557	8795	8173	7884	8468	8062	7381	7469	7469	7782	8615
5	22012	9850	8146	9485	7636	8043	8135	9034	9034	8777	7643
AVG	22261	9006	8380	8331	7891	7923	7868	8611	8611	8138	8115
BEST	22012	8544	7582	6864	7636	7324	7381	7316	7316	7274	7643



Slika 5.5. Utjecaj ρ faktora na kvalitetu rješenja

5.6. O utjecaju subjekta i metode osvježavanja feromona

Slično kao i u poglavlju [5.3] očekivao se minimalan utjecaj pitanja tko i kako osvježava feromone na kvalitetu konačnog rješenja. To se pokazalo točnim, s dvije iznimke – jednim lošim i jednim dobrim pristupom. Kao dobar pristup se pokazao onaj koji se temelji na vjerojatnosti odabira između lokalnog i globalnog optimuma koji je opisan u formuli (7). Time se još jednom pokazalo da je dobro poticati nasumičnost na odgovarajućim mjestima jer ju ACO vrlo dobro iskorištava. Faktor γ koji je korišten jednak je 0.65. [18]

S druge strane, pristup formule B koji se odnosi na popunjenost soba u kombinaciji s iteracijski najboljim rješenjem ne prolazi najbolje. Ostali pristupi su međusobno skoro pa jednaki. Razlog tomu leži u sličnosti formula i specifičnosti velikih vrijednosti evaluacija (oko 10000) tako da onda većina formula teži u istu vrijednost. Spomenuti rezultati mogu se vidjeti u tablici [Tablica 5.5].

Treba istaknuti da se strategija ostavljanja feromona od strane svih mrava nije implementirala jer je to jedan od temelja MMAS algoritma. Razlog zašto nisu isprobane sve moguće kombinacije subjekta i metode leži u tome što C i D metode za globalni optimum postaju identične A metodi (izrazi postaju jednaki 1), a od dodatnog proučavanja kombinacija s B i P subjektima se odustalo nakon što se uočilo da ne postoje osjetne razlike na primjerima G i L u kombinaciji sa svim metodama. Sve oznake i njihova objašnjenja mogu se naći u tablicama [Tablica 5.6] i [

Tablica 5.7].

Tablica 5.5. Utjecaj subjekta i metode osvježavanja na kvalitetu rješenja

Osvj.	A+G	B+G	A+L	B+L	C+L	D+L	A+B	A+P
1	11602	9756	9907	12809	10834	11167	10477	8560
2	10330	13258	10730	15115	9568	10764	11927	9339
3	10397	9529	11081	13000	9705	10137	7537	10454
4	9723	9823	9857	14644	9328	9159	11684	9013
5	10758	10608	9826	13640	12026	10813	8851	9635
AVG	10562	10595	10280	13842	10292	10408	10095	9400
BEST	9723	9529	9826	12809	9328	9159	7537	8560

Tablica 5.6. Objašnjenje oznaka metoda osvježavanja feromona

ozn	formula	oznaka
A	$T = (1-\rho)T + 1$	(9)
B	$T = (1-\rho)(T + \tau_{\max} * popunjenost)$	(10)
C	$T = (1-\rho)T + {}^{1}/_{1+trenutni-najbolji}$	(11)
D	$T = (1-\rho)T + {}^{najbolji}/_{trenutni}$	(12)

Tablica 5.7. Objašnjenje oznaka subjekta osvježavanja feromona

ozn	subjekt	opis
G	globalno najbolje	najbolje pronađeno rješenje ostavlja trag
L	iteracijsko najbolje	najbolje rješenje u tom krugu ostavlja trag
B	i globalno i iteracijsko	oba prethodno opisana ostavljaju trag
P	globalno ili iteracijsko	vjerojatnost odabira na temelju formule (7)

5.7. O prisiljavanju ispravnog rješenja

Potvrđeno je ono što se spominje na više izvora, kako je sekvencijalni pristup uvažavanja ograničenja bolji od simultanog. [21] To se konkretno može vidjeti uključivanjem i isključivanjem opcije "prisiliti ispravnost". Kada je opcija isključena, mravi razmatraju sva moguća rješenja (čak i neprihvatljiva) koja zatim ocjenjuju i proporcionalno stohastički biraju termine. U toj opciji postoji određena vjerojatnost da će se odabrati neprihvatljivo rješenje. S druge strane, kada se opcija prisilne ispravnosti uključi, tada mrav prepozna neprihvatljivo rješenje, te postavi vjerojatnosti odabira tog rješenja na nulu. Stoga se takav izbor neće provesti.

Simultani pristup je dakle onaj koji i jakim i slabim ograničenjima pristupa istovremeno, i to preko težinske evaluacijske funkcije, dok sekvencijalni prvo zadovolji jaka ograničenja, a onda se bavi slabim. Zato je preporučeno držati opciju "prisiliti ispravnost" uključenom.

6. ZAKLJUČAK

U ovom radu su se primjenom max-min mravljeg sustava (MMAS), inačicom optimizacije kolonijom mrava (ACO) na problemu rasporeda ispita (ETP) izveli zaključci o optimalnim vrijednostima sedmero parametara, tri strategije i jedne dodatne opcije. Rezultati su se uglavnom poklopili s onima pronađenim u literaturi, iako su takvi parametri često bili zadavani nad računalno generiranim problemima. U ovom se radu primijenio algoritam na konkretan primjer izrada rasporeda ispita na FER-u koji spada u kategoriju srednje težine ETP-a.

Također se osvrtanjem na neke aktualnosti ili prethodne radove nastojao ukazati veliki porast interesa znanstvene zajednice za spomenutim problemom i algoritmom u posljednjih 15 godina, što je i rezultiralo velikim brojem novih ideja koje se mogu primijeniti.

Bitno je istaknuti da samostalni ACO nije idealan za rješavanje ETP-a, ali u kombinaciji s drugim metodama poput tabu pretrage ili genetskih algoritama može postići dobre rezultate koji su mjerljivi s onima koje su ostvarile najbolje metode u području ovog problema.

Razlog zašto se toliko inzistiralo na otkrivanju odgovarajućih vrijednosti parametara jest taj da se ima temelj za razvoj sustava u kojem krajnji korisnik neće morati zadavati složene parametre, već će ih računalo samostalno dinamički prilagoditi problemu.

7. Literatura

1. Djannaty, F ; Mirzaei, A. R: "Enhancing Max-Min Ant System for Examination Timetabling Problem", International Journal of Soft Computing 3 (3): 230-238, 2008
2. Schaerf, A: "Tabu Search Techniques for Large High-School Timetabling Problems", IEEE Transactions on Systems, Man, and Cybernetics, 1996
3. Zervoudakis, K; Stamatopoulos, P: "A Generic Object-Oriented Constraint Based Model for University Course Timetabling", Proc. of the 3rd International Conference on the Practice and Theory of Automated Timetabling PATAT 2000, Springer-Verlag London, UK, 2000
4. Nedjah, N; Mourelle, L: "Systems engineering using particle swarm optimization", Nova Science Publishers Inc, 2007
5. Chorbev, I. et al: "Solving the High School Scheduling Problem Modelled with Constraints Satisfaction using Hybrid Heuristic Algorithms", The International Conference on "Computer as a Tool", EUROCON, 2007
6. Qu, R. et al: "A survey of search methodologies and automated system development for examination timetabling", Journal of Scheduling archive, Kluwer Academic Publishers Hingham, MA, USA, 2009
7. Karp, R. M; Miller, R. E; Thatcher, J.W: "Reducibility Among Combinatorial Problems", The Journal of Symbolic Logic, Vol. 40, No. 4, UIUC Press, 1975
8. Garey, M; Johnson, D.S: "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H. Freeman and Company, 1979
9. Carter, M.W; Laporte, G; Lee, S.T: "Examination Timetabling: Algorithmic Strategies and Applications", Journal of the Operational Research Society, 47, 1996
10. McCollum, B; McMullan, P: "The Second International Timetabling Competition: Examination Timetabling Track", Technical Report, 2007
11. Tsang, E; Mills, P; Williams, R: "A computer aided constraint programming system", 1st PACLP International Conference, 1999
12. Di Gaspero, L; Schaerf, A: "Tabu search techniques for examination timetabling", Selected Papers from the 3rd PATAT International Conference, 2001
13. De Causmaecker, P. et al: "Using web standards for timetabling", Proceedings of the 4th PATAT International Conference, KaHo St.-Lieven, Gent, Belgium, 2002
14. Stützle, T; Hoos, H.H: "Max-min ant systems" Future Generation Computer System, 16, 2000
15. Azimi, Z. N: "Comparison of metaheuristic algorithms for Examination Timetabling Problem", Journal of Applied Mathematics and Computing, Volume 16, Numbers 1-2, Springer Berlin / Heidelberg 2004
16. Morgenstern, C: "Algorithms for general graph coloring", PhD thesis, Department of Computer Science, University of NewMexico, 1989
17. Socha, K; Knowles, J; Sampels, M: "A Max-Min Ant System For the University Timetabling Problem", 3rd International Workshop on Ant Algorithms, 2002
18. Socha, K: "MAX-MIN Ant System for International Timetabling Competition", Technical Report, 2003

19. Crauwels, H; Van Oudheusden, D: "Ant colony optimization and local improvement", Workshop of Real-Life Applications of Metaheuritics, Antwerp, 2003

20. Dorigo, M; Stützle, T: "Ant Colony Optimization", The MIT Press, Cambridge, MA, 2004

21. Ghaemi, S; Vakili, M.T; Aghagolzadeh, A: "Using a genetic algorithm optimizer tool to solve University timetable scheduling problem", Signal Processing and Its Applications, 2007

Dodatak A: Popis oznaka i kratica

ETP	problem rasporeda ispita (examination timetabling problem)
ACO	optimizacija kolonijom mrava (ant colony optimization)
GCP	problem bojanja grafa (graph coloring problem)
SSP	problem odabira podskupa (subset selection problem)
MCS	maksimalno ispunjavanje ograničenja (maximum constraint satisfaction)
GDA	algoritam velikog potopa (great deluge algorithm)
TS	tabu pretraga (tabu search)
AS	mravlji sustav (ant system)
MMAS	max-min mravlji sustav (max-min ant system)