Klasterovanje nepotpunih podataka pomoću $\mathcal{MAX}-\mathcal{MIN}$ mravljeg sistema

Magdalena Tarajić

Matematički fakultet, Univerzitet u Beogradu
magdalenat098@gmail.com

Sažetak

Max-Min mravlji sistem (MMAS) pripada kategoriji algoritama Optimizacije kolonijom mrava (ACO) i najčešću primenu nalazi u problemima nalaženja optimalnih puteva na grafovima, kao što su Problem trgovačkog putnika. U ovom radu je, ipak, posmatran jedan negrafovski problem (klasterovanje) i predstavljena nova adaptacija MMAS-a na problemu klasterovanja, u cilju poređenja metaheurističke metode sa neuralnim mrežama. U obzir je uzet i problem nedostajućih podataka (rešen uvođenjem nove funkcije udaljenosti, preuzete iz literature), pa su baze na kojima su vršena testiranja sadržale određeni procenat nedostajućih podataka. Krajnje poređenje je izvršeno sa šest metoda neuralnih mreža.

Ključne reči: Mravlji algoritmi · Baze podataka · Klasifikacija objekata · Nedostajući podaci

Sadržaj

1 Uvod

Grupisanje podataka na osnovu nekih zajedničkih osobina (klasterovanje) predstavlja veoma bazičan problem i intenzivnose izučava u računarskim naukama već izvesno vreme. Kao jedan od popularnijih problema u oblasti data mining-a, klasterovanje postaje sve relevantnija tema i nalazi sve šire primene u različitim tehnologijima. Sa ovom povećanom potražnjom raste i potreba za efikasnim algoritmima za klasterovanje. Ono se može obaviti pomoću klasičnih algoritama kao što je K-means, ali i pomoću novijih metoda kao što su neuralne mreže. Ipak, metaheuristički algoritmi za klasterovanje inspirisani prirodom [11] će biti od najvećeg interesa u ovom radu i to pre svega oni koji imitiraju ponašanje mrava, a koji takođe pripadaju algoritmima inteligencije rojeva (*Swarm Intelligence* algoritmi).

Još jedan metaheuristički algoritam iz kategorije inteligencije rojeva je i Optimizacija kolonijom pčela (*Bee Colony Optimization*, BCO) [3], zasnovan na ponašanju pčela. Jedna od adaptacija BCO algoritma se može naći i u [2], sa čijim će se rezultatima u ovom radu izvršiti poređenje i čijih je par ideja preuzeto. U [2] se BCO pokazao kao superiorniji u odnosu na algoritme iz kategorije neuralnih mreža kao što su *Selective Neural Network Ensemble* (SNNE) i *Multi-Granulation Ensemble Method* (MGNE), iako u implementaciju nisu ugrađeni elementi učenja. Jedan od ciljeva ovog rada je da se proveri da li još jedna metaheuristička metoda kao što je Optimizacija kolonijom mrava (*Ant Colony Optimisation*, ACO) može dati bolje rezultate od neuralnih mreža.

Velike količine informacija često mogu dovesti do pojave da određeni procenat podataka u bazama fali. Ovaj problem u radu nije zanemaren i baze na kojima su rađena testiranja sadrže određeni procenat nedostajućih podataka.

Rad je organizovan na sledeći način. U narednoj glavi je data definicija problema klasterovanja, kao i nova funkcija udaljenosti koja će se koristiti, a koja je (i ovde i u [2]) preuzeta iz [6]. U trećoj glavi su date opšte crte ACO algoritma, kao i Max-Min mravljeg sistema (*Max-Min* Ant System, MMAS) koji predstavlja jednu od prvih varijanti ACO algoritma. Četvrta glava je posvećena podešavanju MMAS-a za klasterovanje, što podrazumeva nameštanje svih neophodnih parametara. U četvrtoj glavi je takođe data i funkcija preciznosti pomoću koje se meri uspešnost klasterovanja. Krajnji rezultati testiranja su dati u petoj glavi, dok su zaključak i ideje za budući rad date u šestoj glavi.

2 Formalna defincija klasterovanja i nova funkcija udaljenosti

Pretpostavimo da imamo skup O koji se sastoji od N objekata $o^1, \ldots, o^N$ koje treba grupisati u K klastera. Svaki od ovih objekata će imati n-dimenzionu kooridantnu reprezentaciju $o^j = (a_1, \ldots, a_n)$ $(j = 1, \ldots, N)$ gde koordinata a_i $(i = 1, \ldots, n)$ predstavlja atribut objekta koji će, u slučaju da je to potrebno, biti pretvoren u numeričku vrednost.

Kako je zadatak klasterovanja da grupiše objekte na osnovu nekih istih osobina, potrebna nam je funkcija koja će pružati informaciju o tome koliko su dva objekta slična. Zato, označimo funkciju udaljenosti (tj. različitosti), čiju ćemo preciznu definiciju dati uskoro, sa $D : O \times O \to \mathbb{R}_0^+$.

Vrednost svakog atributa objekata ne mora uvek biti poznata. Razlozi za to mogu da budu razni, ali ono što je bitno je da se ovaj problem često javlja i da je neophodan određen pristup njegovom rešavanju. U ovom radu ćemo za uzor uzeti [2] gde se ovaj problem premostio uvođenjem nove funkcije udaljenosti preuzete iz [6].

Neka je $o = (o_1, \ldots, o_n)$ proizvoljni objekat kome možemo pridružiti iskaznu formulu

$\alpha = o_1 \wedge \cdots \wedge o_n$. Ukoliko neki atribut o_i nije poznat, zamenićemo ga sa disjunkcijom svih vrednosti koje može uzeti, odnosno $o_i = o_i^1 \vee \cdots \vee o_i^s$, gde je s broj svih mogućih vrednosti koje o_i može uzeti. Na ovaj način dobijamo skup $\mathcal{A}$ koji se sastoji od formula:

$$\alpha^1 = o_1 \wedge o_2 \wedge \cdots \wedge o_i^1 \wedge \cdots \wedge o_n;$$

$$\alpha^2 = o_1 \wedge o_2 \wedge \cdots \wedge o_i^2 \wedge \cdots \wedge o_n;$$

$$\vdots$$

$$\alpha^s = o_1 \wedge o_2 \wedge \cdots \wedge o_i^s \wedge \cdots \wedge o_n.$$

Naravno, može se desiti da objektu o fali više od jednog atributa. U tom slučaju će $\mathcal{A}$ činiti sve moguće kombinacije vrednosti koje mogu uzeti svi nedostajući atributi.

Najzad, dolazimo do definicije funkcije D. Neka su o i p dva objekta sa pridruženim skupovima formula $\mathcal{A}$ i $\mathcal{B}$. Udaljenost između o i p, koristeći Hamingovu udaljenost d, računamo na sledeći način:

$$D(o,p) = D(\mathcal{A},\mathcal{B}) = \frac{\max\limits_{\alpha\in\mathcal{A}} \min\limits_{\beta\in\mathcal{B}} d(\alpha,\beta) + \max\limits_{\beta\in\mathcal{B}} \min\limits_{\alpha\in\mathcal{A}} d(\alpha,\beta)}{2}. \tag{1}$$

Hamingova udaljenost dva n-dimenziona objekta predstavlja broj koordinata na kojima se oni razlikuju.

Sada kada imamo definiciju funkcije udaljenosti, možemo se vratiti na samo klasterovanje. Kao što smo rekli, N objekata je potrebno smestiti u K klastera. Te klastere možemo reprezentovati na više načina. U ovom radu, nećemo odmah određivati klastere u celosti, već će se nalaziti K objekata koji će predstavljati takozvane centroide i onda ostale objekte rasporediti u zavisnosti od toga koji objekat je kom centroidu najbliži.

Da bismo problem klasterovanja definisali kao zadatak celobrojnog linearnog programiranja (CLP), uvodimo naredna dva skupa binarnih promenljivih odlučivanja (preuzeto iz [2]):

$$x_{jl} = \begin{cases} 1, & \text{ako je objekat } o^j \text{ dodeljen klasteru predstavljenom centroidom } o^l, \\ 0, & \text{inače.} \end{cases}$$

$$y_l = \begin{cases} 1, & \text{ako objekat } o^l \text{ predstavlja centroid,} \\ 0, & \text{inače.} \end{cases}$$

Klasterovanje je sada kao CLP formulisano kao optimizacioni zadatak p-medijane (p-median) na sledeći nacin:

$$\min \sum_{j=1}^{N} \sum_{l=1}^{N} x_{jl} D(o^j, o^l) \tag{2}$$

tako da važi

$$\sum_{l=1}^{N} x_{jl} = 1, \qquad 1 \leqslant j \leqslant N, \tag{3}$$

$$x_{jl} \leqslant y_l, \qquad 1 \leqslant j \leqslant N, \;\; 1 \leqslant l \leqslant N, \tag{4}$$

$$\sum_{l=1}^{N} y_l = K, \tag{5}$$

$$x_{jl}, y_l \in \{0,1\}, \qquad 1 \leqslant j \leqslant N, \;\; 1 \leqslant l \leqslant N. \tag{6}$$

Funkcija cilja (2), koju želimo da minimizujemo, predstavlja sumu udaljenosti svakog objekta od centroida klastera u kom se nalazi. Ograničenja data formulama od (3)-(6) ukazuju na to da želimo da se svaki objekat nađe u tačno jednom klasteru (3), da objekat može pripasti nekom klasteru predstavljenim objektom o^l samo ako je o^l zaista centroid (4), kao i da imamo tačno K objekata koji su centroidi (5). Ograničenja (6) definišu binarnu prirodu promenljivih odlučivanja.

3 ACO algoritam

Iako će se u ovom radu prikazati jedna druga varijanta, ACO je [9] algoritam koji vuče svoje korene u nalaženju optimalnih puteva na grafovima. Prvu varijaciju ovog algoritma M. Dorigo spominje u svojoj doktorskoj disertaciji [4] iz 1992, da bi se 1996. pojavio rad [5] posvećen prvom ACO algoritmu pod nazivom Mravlji sistem (*Ant System*, AS), sa namerom da grupa kooperativnih mrava nađe optimalno rešenje Problema trgovačkog putnika (*Traveling salesman problem*, TSP). Međutim, ACO algoritam je nastavio da se koristi i za mnoge druge optimizacione probleme, a za početak ćemo navesti opis njegovog korišćenja u opštem slučaju.

3.1 Opšti opis ACO algoritma

Kao što se može primetiti u samom nazivu, ACO je inspirisan ponašanjem mrava u prirodi. Postoji više aspekata njihovog ponašanja koji se mogu imitirati u vidu algoritma, ali svojstvo mrava koje je ključno za ACO je pre svega pojava da oni pri svom kretanju otpuštaju feromone koje drugi mravi mogu prepoznati kada se nađu na tom istom putu [7]. Ovo im omogućava da pri pronalasku hrane kooperativno nađu putanju koja će ih do nje odvesti, a koja će ujedno biti i najkraća. Razlog za to je što će mrav moći među mnoštvom puteva da izabere onaj koji je popularniji kod ostalih mrava, tj. koji sadrži više feromona. Kako će se sve veći broj mrava kretati takvim putanjama, na kraju se formira kolona mrava koja se kreće isključivo najkraćom putanjom (na svim ostalima feromoni su isparili i mravima više nisu zanimljivi).

Vratimo se sada za trenutak na TSP. Ovaj poznati problem je definisan sledećim pitanjem: Na koji način trgovački putnik može obići svaki grad sa svog spiska tako da svaki poseti tačno jednom i vrati se u početni grad, a da pri tome put koji je prešao bude najkraći? Kada problem prevedemo na jezik grafova tako što posmatramo gradove kao čvorove, a puteve između njih kao grane, dobijamo preformulisano pitanje: Ukoliko imamo kompletan težinski graf, koji je njegov Hamiltonov ciklus najmanje težine? Svakoj grani grafa možemo pridružiti njenu težinsku vrednost koja će zapravo predstavljati udaljenost između naša dva grada koji su krajevi te grane. Ukratko, algoritam će na ovakvom problemu funkcionisati tako što će kolonija mrava određene veličine pri svakoj iteraciji generisati jedno rešenje (Hamiltonov ciklus) koje će na osnovu svog kvaliteta povećati količinu feromona na granama koje su komponente tog rešenja. TSP predstavlja dobar uvod za objašnjenje opštih crta ACO algoritma pri rešavanju problema kombinatorne optimizacije.

Za početak, potrebno je reći da će kod svakog problema koji ACO rešava konstrukcija rešenja uslediti iz biranja njegovih komponenti. Ono što je važno je da se svakoj komponenti koja može ući u sastav nekog rešenja dodeli određeni feromonski faktor koji će smanjivati/povećavati verovatnoću da se ona izabere. Pored feromonskog faktora, bitnu ulogu može igrati i takozvani heuristički faktor koji bi trebalo da da prednost određenim komponentama rešenja.

Formalizujmo prethodnu priču. Neka je $(\mathscr{S},\ \Omega,\ f)$ zadat problem kombinatorne optimizacije gde je diskretan skup $\mathscr{S}$ prostor pretrage, $X \subset \mathscr{S}$ skup dopustivih rešenja čije komponente zadovoljavaju uslove iz skupa restrikcija Ω i $f : \mathscr{S} \to \mathbb{R}$ funkcija cilja.

Dakle, rešenje bi bilo element skupa X koji minimizuje/maksimizuje funkciju cilja f, a pošto govorimo o problemu *kombinatorne* optimizacije, možemo ga predstaviti i preko (najviše prebrojivog) skupa komponenti $\mathscr{C} = \{c_1, c_2, \dots\}$ koje mogu ući u sastav istog. Svakoj mogućoj komponenti rešenja možemo pridružiti određenu numeričku vrednost koja će predstavljati njen feromonski faktor. Tačnije, $\forall c \in \mathscr{C}$, $\exists \tau_c \in \mathscr{T}$ gde je $\mathscr{T}$ skup feromonskih faktora. Na isti način definišemo i skup heurističkih faktora $\mathscr{H}$ sa jedinom razlikom u tome što će njihova vrednost tokom izvršavanja programa ostati konstantna. Dakle, $\forall c \in \mathscr{C}$, $\exists \eta_c \in \mathscr{H}$.

Kod TSP-a npr, jasno je da su rešenja Hamiltonovi ciklusi, a komponente grane tog ciklusa. Neophodna funkcija cilja pomoću koje bismo merili kvalitet rešenja je svakako dužina konsturisanog ciklusa koju bi trebalo minimizovati. Budući da zvuči logično dati malu prednost kraćim granama, mogli bismo svakoj grani dodeliti za heuristički faktor recipročnu vrednost njene dužine.

Svaki mrav će rešenje konstruisati probabilističkim dodavanjem jedne po jedne grane, tj. komponente. Verovatnoća da u k-tom koraku konstrukcije (nalazeći se u i-tom čvoru grafa) mrav izabere granu $c_{i,j}$ (granu koja spaja i-ti i j-ti čvor) se često zadaje sledećom formulom koju ćemo i mi koristiti u ovom radu:

$$P(c_{i,j}) = \frac{\tau_{i,j}^{\alpha} \cdot \eta_{i,j}^{\beta}}{\sum_{c_{i,l} \in N_k} \tau_{i,l}^{\alpha} \cdot \eta_{i,l}^{\beta}}. \tag{7}$$

Ovde N_k predstavlja skup mogućih grana koje mrav može izabrati u k-tom koraku, a realni brojevi α i β parametre koji će konstrolisati koliki uticaj feromonskog/heurističkog faktora želimo da postignemo.

Iako ACO sam po sebi može postići uspeh, uglavnom se nakon svake iteracije biraju mravi (koji mogu biti i svi) nad kojima će se primeniti dodatna popravka njihovog rešenja, što se najčešće čini pomoću lokalne pretrage koja se zbog toga uračunava u sastavni deo ACO algoritma.

Još jednu ideju koju preuzimamo od pravih mrava je to da feromoni, koliko god moćno sredstvo predstavljali, posle nekog vremena isparavaju. Međutim, to nam može pomoći da onemogućimo preveliki porast feromona na pojedinim mestima, što bi uticalo na to da neke komponente nemaju skoro nikakvu šansu da budu izabrane kao deo rešenja, tj. da algoritam prerano iskonvergira i to ka nekom relativno lošem rešenju. Zbog toga uvodimo koeficijent isparavanja ρ koji će dejstvovati na svako $\tau \in \mathscr{T}$ posle svake iteracije, po formuli

$$\tau = (1 - \rho)\tau.$$

Nekoliko puta smo već rekli da bi tokom vremena komponente kvalitetnijih rešenja trebalo da imaju veću verovatnoću da budu izabrane, tj. da se količina njihovih feromona povećava. Postoji više načina na koje možemo ovo učiniti. Moguće je npr. da posle svake iteracije povećamo vrednost feromona samo na komponentama koje su ušle u sastav rešenja globalno najboljeg mrava (onog koji je konstruisao najbolje rešenje na nivou celog programa) ili pak da to učinimo nad komponentama najboljeg mrava u konkretnoj iteraciji koja se desila neposredno pre. Prvi način poznat je kao globalno ažuriranje vrednosti feromona (*global pheromone update*), a drugi kao lokalno (*local pheromone update*).

Iz svega navedenog, možemo na kraju ovog odeljka dati pseudokod (Algoritam 1) koji opisuje opšte ponašanje ACO algoritma.

Algorithm 1 Pseudocode of ACO algorithm

```
repeat
    for all ant do
        repeat
            ExtendPartialSolutionProbabilistically()
        until solution is complete
    end for
    for all ant ∈ SelectAntsForLocalSearch() do
        ApplyLocalSearch(ant)
    end for
    EvaporatePheromones()
    DepositPheromones()
until termination criteria met
```

3.2 $\mathcal{MAX}-\mathcal{MIN}$ mravlji sistem

Kao jednu od mnogobrojnih adaptacija Ant System algoritma, Hoos i Stützle 1996. godine predlažu MMAS [12] koji uvodi nekoliko značajnih novina u odnosu na originalan algoritam, s namernom da se potencira što veće istraživanje prostora pretrage.

Tri bitna svojstva koja izdvajaju MMAS od ostalih ACO algoritama su: za povećavanje feromona se koristi samo globalno najbolji mrav, feromoni su ograničeni odozgo vrednošću parametara τ_{max} i odozdo sa τ_{min} i još, feromoni se inicijalizuju pomoću τ_{max}.

Običaj je da se kod MMAS-a povremeno dešava reinicijalizacija feromona, najčešće u situacijama kada algoritam iskonvergira ka određenom rešenju.

4 MMAS na problemu klasterovanja

Ideja da se ACO algoritam iskoristi za klasterovanje nije nova. U radovima kao što su [8] i [10] možemo videti dva ACO algoritma (*Ant Colony Optimization for Clustering* - ACOC i *Medoid-Based* ACO *Clustering Algorithm* - MACOC) koja oba imaju grafovski pristup problemu i za koje se ispostavlja da daju bolje rezultate od algoritama kao što su *Partition Around Medoids* (PAM) i K-means.

Međutim, u ovom radu je predstavljena jedna prilično jednostavnija varijanta koja zapravo ne zahteva prisustvo grafovske strukture, već sav uticaj feromona koristi za verovatnoću da se određeni objekat izabere kao centroid klastera.

Osnovni deo programa je inspirisan radom klasičnog ACO algoritma, a radi što veće sličnosti sa istim, potrebno je definisati sve neophodne parametre u kontekstu problema klasterovanja.

Neka broj A označava broj mrava koji generišu rešenja u svakoj iteraciji programa, N broj objekata koje je potrebno klasterovati i K broj klastera. Zadatak svakog mrava u našem programu je da klasteruje datih N objekata u K klastera, tj. da izabere K objekata koji će predstavljati centroide. Kada to učini, kvalitet (fitnes) njegovog rešenja će se procenjivati kao recipročna vrednost funkcije cilja njegovog klasterovanja, a to je (kao što je već rečeno) zbir udaljenosti svakog objekta od pripadajućeg centroida.

Da bismo ovaj problem zaista rešili koristeći MMAS, neophodno je pored feromonskog faktora definisati i heuristički. Ovaj faktor, za razliku od feromonskog, ostaje konstantan od početka do kraja programa i trebalo bi da na neki način omogući da budu favorizovani objekti koji po nekoj osobini već na početku predstavljaju bolji izbor za centroid. Zbog toga heuristički faktor svakog objekta predstavlja recipročnu vrednost njegove prosečne udaljenosti od svih ostalih objekata. Što je heuristički faktor veći, odnosno prosečna udaljenost manja, objekat će imati veću šansu da bude izabran kao centroid.

Označimo sa $\tau(t)$ i η vektore feromonskih tj. heurističkih faktora u iteraciji t:

$$\tau(t) = \begin{bmatrix}\tau_1(t) & \tau_2(t) & \dots & \tau_N(t)\end{bmatrix}, \ \eta = \begin{bmatrix}\eta_1 & \eta_2 & \dots & \eta_N\end{bmatrix}.$$

Pošto nam je često bitno da znamo u kojoj konkretnoj iteraciji programa se nalazimo, τ_i shvatamo kao funkcije od broja iteracije, u smislu da $\tau_i(t)$ predstavlja količinu feromona na i-tom objektu u iteraciji t.

Kao što smo malopre napomenuli, η računamo po sledećoj formuli:

$$\eta_i = \frac{1}{\frac{1}{N}\sum_{j=1}^{N} D(o^i, o^j)},$$

gde je D funkcija udaljenosti definisana formulom (1).

Po uzoru na formulu (7), verovatnoću da objekat bude izabran kao centroid (u iteraciji t) ćemo računati na sledeći način:

$$P\left\{o^j \textit{ je centroid u iteraciji } t\right\} = \frac{\tau_j(t)^\alpha \cdot \eta_j^\beta}{\sum_{i=1}^{N} \tau_i(t)^\alpha \cdot \eta_i^\beta}. \tag{8}$$

Realni brojevi α i β se mogu podešavati u zavisnosti od toga da li je poželjniji uticaj feromonskog ili heurističkog faktora.

U svakoj od s iteracija programa se inicijalizuje nova familija A mrava i pronalazi onaj čije rešenje ima najbolji fitnes. Tog mrava nazivamo lokalno najboljim, a ukoliko je njegov fitnes bolji od onog koje ima trenutni globalno najbolji mrav, on će postati novi globalno najbolji.

Pomoću vrednosti funkcije cilja f^{gb} globalno najboljeg mrava, vektor feromona ažuriramo na sledeći način

$$\tau_i(t+1) = \begin{cases} \tau_i(t)\cdot(1-\rho), & (\nexists j)\ i = k_j \\ \tau_i(t)\cdot(1-\rho) + 1/f^{gb}, & i = k_j \end{cases}, \tag{9}$$

gde konstanta ρ predstavlja već spomenuti koeficijent isparavanja.

4.1 Opis baza

Pre bilo kakvog podešavanja parametara i analiziranja, navodimo opise za baze podataka nad kojima su izvršena sva testiranja. Opis baza koji uključuje broj objekata, klastera, kao i procenat nedostajućih podataka je dat u Tabeli 1. Svih 9 baza je preuzeto sa UCI repozitorijuma za mašinsko učenje [1].

4.2 Lokalna pretraga i reinicijalizacija vektora feromona

Kao što je već spomenuto, ACO algoritam sam po sebi može dostići nekakve rezultate, ali ih lokalna pretraga značajno poboljšava. Međutim, veliku manu lokalne pretrage predstavlja njeno vreme izvršanja koje raste sa povećanjem broja objekata/klastera. Njen uticaj, ali i podugačko vreme izvršavanja se može videti u Tabeli 2. Rezultati koji su prikazani su dobijeni korišćenjem lokalne pretrage (po *first improvment* strategiji pretraživanja) u punoj snazi, u smislu da se popravljanje rešenja koristilo nad lokalno najboljim mravom u svakoj iteraciji.

Zbog spomenutog dugačkog vremena izvršavanja, neophodno je koristiti lokalnu pretragu u umerenim dozama i u oslabljenoj varijanti, koja je prikazana Algoritmom 2.

Da bismo maksimalno iskoristili snagu MMAS-a, uvodimo par indikatora koji će nam ukazivati na to kada je najbolje koristiti lokalnu pretragu. Strategija koja je opisana dobijena

Tabela 1: Opis baza podataka: naziv, broj objekata, broj klastera, tip i broj atributa, procenat nedostajućih podataka

naziv	#objekata	#klastera	tip atributa	#atributa	% nedostajućih
B. cancer	699	2	cat.	10	0.03
CVR	435	2	cat.	16	1.05
Dermatology	366	6	cat.+int.	34	0.02
Heart - h	294	2	cat.+int.+real	13	0.34
Heart - c	303	5	cat.+int.+real	13	0.08
Hepatitis	155	2	cat.+int.+real	19	0.71
H.colic	368	2	cat.+int.+real	27	1.78
L.cancer	32	2	int.	56	0.17
MM	961	2	int.	5	0.21

Tabela 2: Poređenje fitnesa dobijenog sa i bez korišćenja lokalne pretrage. Brojevi u zagradi predstavljaju broj pojavljivanja najboljeg fitnesa u 30 pozivanja. Prosečno vreme je izraženo u milisekundama.

baza	najbolji fitnes bez lok.pretrage	najbolji fitnes sa lok. pretragom	prosečno vreme (ms) bez lok. pretrage	prosečno vreme (ms) sa lok. pretragom
Heart-h	**0.011707506** (1)	**0.011707506** (**30**)	925.5208	61,320.3125
Hepatitis	0.01288889176 (4)	**0.01288902431** (30)	485.9375	11,582.8125
L.cancer	**0.2057650321** (3)	**0.2057650321** (**30**)	0.125	0.51302083

je eksperimentalnom analizom raznih varijanti, a ovde opisujemo onu koja je dala najbolje rezultate. U toku programa ćemo uvek pamtiti, pored globalno i lokalno najboljeg mrava, prethodnog mrava iz iteracije, kao i njegovu verziju popravljenu pomoću (možda višestruke primene) lokalne pretrage. Ako je naš mrav isti kao globalno najbolji, nećemo menjati prethodnog mrava iz iteracije sa njim. Detaljniji proces odlučivanja kada se primenjuje lokalna pretraga prikazuje Algoritam 3. $Indicator1$ ukazuje na to da li smo popravili globalno najboljeg mrava u prethodnom pokušaju, dok $indicator2$ označava da li smo popravljenu verziju prethodnog mrava iz iteracije uspeli da popravimo kada smo to poslednji put probali. Pored spomenutih, imamo još i $indicator3$, tj. $indicator4$ koji ukazuju da li najbolji mrav u iteraciji ima jednako dobro rešenje kao globalno najbolji mrav, tj. onaj iz prethodne iteracije.

Takođe, tu je i *counter* koji nam broji koliko puta za redom nismo uspeli da popravimo globalno najboljeg mrava. Posle svake iteracije ćemo proveravati da li je naš algoritam iskonvergirao i u slučaju da jeste, reinicijalizovaćemo vektor feromona. To ćemo činiti ako je *counter* dostigao vrednost 5, ali i u slučaju da su svi mravi u toj iteraciji zapravo dali isto rešenje kao $globalBestAnt$, na šta ukazuje $indicator5$.

Konačno, kada sumiramo sve, dobijamo Algoritam 4 koji predstavlja malo modifikovanu verziju Algoritma 1 i daje primenu MMAS algoritma na problem klasterovanja.

4.3 Određivanje kvaliteta rešenja (preciznost klasterovanja)

Mera uspešnosti programa bi trebalo da predstavlja koliko su dobijeni klasteri slični onima iz baze. Zbog toga je potrebno uspostaviti bijekciju između dobijenih klastera i onih "pravih"(baznih).

Neka su $k'_1, \ldots, k'_K$ bazni klasteri, a $k_1, \ldots, k_K$ oni koje je program izračunao. Definišimo matricu sličnosti S kao $K \times K$ matricu čiji element $s_{i,j}$ predstavlja broj zajedničkih objekata

Algorithm 2 Pseudocode of Local Search

```
function localSearch(ant)
    for all c ∈ ant.centroids do
        for all o ∈ objects do
            if o ∉ ant.centroids then
                temporarySolution = a.centroids
                temporarySolution.remove(c)
                temporarySolution.add(o)
                if fitness(temporarySolution) > a.fitness then
                    a.centroids = temporarySolution
                    a.fitness = fitness(temporarySolution)
                end if
            end if
        end for
    end for
    return ant
end function
```

klastera k_i i k_j':

$$S = \begin{bmatrix} s_{1,1} & \cdots & s_{1,K} \\ \vdots & \ddots & \vdots \\ s_{K,1} & \cdots & s_{K,K} \end{bmatrix}.$$

Naša tražena bijekcija između pravih i izračunatih klastera će zapravo biti ona permutacija $i_1, \ldots, i_K$ skupa $\{1, 2, \ldots, K\}$ koja će maksimizovati vrednost

$$s_{1,i_1} + s_{2,i_2} + \cdots + s_{K,i_K}.$$

Kada tu permutaciju $i_1, \ldots, i_K$ nađemo, za preciznost našeg klasterovanja uzimamo broj

$$\frac{s_{1,i_1} + s_{2,i_2} + \cdots + s_{K,i_K}}{N}.$$

4.4 Određivanje parametara

Na osnovu svega što je ranije rečeno, zaključujemo da je od parametara potrebno podesiti broj iteracija s, broj mrava A, parametre α i β, koeficijent isparavanja ρ, kao i donje i gornje ograničenje za feromone τ_{min} i τ_{max}. Kako ovde ima čak sedam parametara čiju je vrednost potrebno odrediti, zbog tehničkih nemogućnosti nije bilo izvodljivo obaviti detaljno testiranje koje bi podrazumevalo isprobavanje velikog broja kombinacija vrednosti ovih sedam parametara. Zbog toga je vrednost nekih parametara preuzeta iz relevantne literature (τ_{min} i τ_{max}), dok se za druge ipak obavilo manje (A, s) ili više (ρ, α, β) testiranja.

4.4.1 Donje i gornje ograničenje za feromone τ_{min} i τ_{max}

Još jedna od ideja koja je predstavljena u [12] je ažuriranje vrednosti τ_{min} i τ_{max} tokom izvršavanja programa. U tu svrhu, po uzoru na analognu verziju iz [12], dokazaćemo naredni stav. Označimo sa f^{opt} vrednost funkcije cilja (nama verovatno nepoznatnog) optimalnog rešenja.

Stav 1. $\lim_{t\to\infty} \tau_i(t) \leqslant \frac{1}{\rho} \cdot \frac{1}{f^{opt}}$

Dokaz. Pošto je f^{opt} najmanja vrednost funkcije cilja koju u programu možemo postići, $1/f^{opt}$ je najveća količina feromona koja se može dodati posle neke iteracije, pa pomoću (9)

Algorithm 3 Pseudocode of Application of Local Search

```
function localSearchApplication
    switch true do
        case indicator1 == 1 ∧ (indicator3 == 1 ∨ (indicator4 == 1 ∧ indicator2 == 0))
            if globalBestAnt.solution == localSearch(globalBestAnt).solution then
                counter++
                indicator1 = 0
            else
                globalBestAnt = localSearch(globalBestAnt)
                counter = 0
            end if
        case indicator1 == 0 ∧ (indicator3 == 1 ∨ (indicator4 == 1 ∧ indicator2 == 0))
            counter++
        case (indicator4 == 1 ∧ indicator2 == 1)
            if previousAntFixed.solution == localSearch(previousAntFixed).solution then
                indicator1 = 0
            else
                previousAntFixed.solution = localSearch(previousAntFixed).solution
                if previousAntFixed.fitness > globalBestAnt.fitness then
                    globalBestAnt = previousAntFixed
                    indicator1 = 1
                    counter = 0
                end if
            end if
        case indicator3 == 0 ∧ indicator4 == 0
            previousAnt = localBestAnt
            previousAntFixed = localSearch(previousAnt)
            if previousAntFixed.fitness > previousAnt.fitness then
                indicator2 = 1
            else
                indicator2 = 0
            end if
            if prevAntFix.fitness > globalBestAnt.fitness then
                globalBestAnt = previousAntFix
                indicator1 = 1
                counter = 0
            end if
    return
end function
```

Algorithm 4 Pseudocode of MMAS For Clustering

```
repeat
    for all ant do
        repeat
            ExtendPartialSolutionProbabilistically()
        until solution is complete
    end for
    indicator5 = 1
    for all ant do
        if ant.solution == globalBestAnt.solution then
            indicator5 = 0
            break
        end if
    end for
    localSearchApplication()
    if counter == 5 ∨ indicator5 == 1 then
        reinitializePheromoneVector()
        counter = 0
    else
        EvaporatePheromones()
        DepositPheromones()
    end if
until termination criteria met
```

dolazimo do nejdnakosti koja će važiti za svako $i \in \{1, \ldots, N\}$:

$$\tau_i(t) \leqslant \tau_i(t-1) \cdot (1-\rho) + \frac{1}{f^{opt}}. \tag{10}$$

Koristeći (10) iterativno, dolazimo do:

$$\tau_i(t) \leqslant \sum_{j=1}^{t} (1-\rho)^{t-j} \frac{1}{f^{opt}} + (1-\rho)^t \cdot \tau_i(0).$$

Budući da $1-\rho \in (0,1)$, vidimo da je gornji red geometrijski, što nas dovodi do:

$$\begin{aligned}\tau_i(t) &\leqslant \frac{1}{f^{opt}} \sum_{j=1}^{t} (1-\rho)^{t-j} + (1-\rho)^t \cdot \tau_i(0) \leqslant \frac{1}{f^{opt}} \sum_{j=0}^{\infty} (1-\rho)^j + (1-\rho)^t \cdot \tau_i(0) \\ &= \frac{1}{f^{opt}} \cdot \frac{1}{\rho} + (1-\rho)^t \cdot \tau_i(0).\end{aligned}$$

Kada pustimo limes na prethodnu nejednakost, dolazimo do kraja dokaza:

$$\lim_{t \to \infty} \tau_i(t) \leqslant \frac{1}{\rho} \cdot \frac{1}{f^{opt}}.$$

□

Ovaj stav možemo koristiti za dinamičko menjanje vrednosti gornje granice feromona tako što ćemo je procenjivati baš sa $\frac{1}{\rho} \cdot \frac{1}{f^{opt}}$. Međutim, pošto nama f^{opt} nije poznata vrednost, umesto nje ćemo koristiti f^{gb}, tj. vrednost funkcije cilja trenutno globalno najboljeg rešenja.

Ponovo po uzoru na [12], dinamički ćemo menjati i τ_{min}. Pretpostavimo da je naš algoritam iskonvergirao i da se najbolje rešenje konstruiše sa verovatnoćom p_{best}. Takođe, uzmimo i da se sa verovatnoćom p_{dec} bira komponenta (tj. centroid) ovog najboljeg rešenja. To bi značilo

da K puta sa verovatoćom p_{dec} program treba da izabere komponentu najboljeg rešenja. Tako dolazimo do jednakosti

$$p_{best} = p_{dec}^K, \text{ tj. } p_{dec} = \sqrt[K]{p_{best}}. \tag{11}$$

S druge strane, p_{dec} možemo računati i na sledeći način. Pošto je algoritam iskonvergirao, možemo uzeti da je količina feromona na komponentama najboljeg rešenja baš τ_{max}, a na ostalim τ_{min}. Kako se kod MMAS algoritma često uzima mala vrednost za β, zanemarićemo uticaj ovog parametra i koristiti činjenicu da je u toku konstrukcije rešenja program izabrao prosečno $K/2$ centroida, pa pomoću (8) doći do jednakosti

$$p_{dec} = \frac{\tau_{max}}{\tau_{max} \cdot K/2 + (avg - K/2) \cdot \tau_{min}}, \tag{12}$$

gde avg predstavlja prosečan broj objekata među kojima algoritam bira narednu komponentu rešenja. Ovaj broj možemo računati na sledeći način:

$$avg = \frac{N + (N-1) + \cdots + (N-K+1)}{K} = \frac{\frac{(N+N-K+1)\cdot K}{2}}{K} = \frac{2N - K + 1}{2} \tag{13}$$

Izrazimo τ_{min} iz formule (12), a zatim zamenimo (13) i (11):

$$\tau_{min} = \frac{\tau_{max}(2 - Kp_{dec})}{(2avg - K)p_{dec}} = \frac{\tau_{max}(2 - K\sqrt[K]{p_{best}})}{(2N - 2K + 1)\sqrt[K]{p_{best}}}$$

Za kraj, kako ćemo smatrati da postoji jedinstveno najbolje rešenje, možemo računati

$$p_{best} = \frac{1}{\binom{N}{K}} = \frac{(N-K)!K!}{N!}.$$

Da rezimiramo, posle svakog ažuriranja globalno najboljeg mrava, prvo ćemo izračunati novo τ_{max} po formuli

$$\tau_{max} = \frac{1}{\rho} \cdot \frac{1}{f^{gb}}, \tag{14}$$

a zatim i τ_{min}:

$$\tau_{min} = \frac{\tau_{max}\left(2 - K\sqrt[K]{\frac{(N-K)!K!}{N!}}\right)}{(2N - 2K + 1)\sqrt[K]{\frac{(N-K)!K!}{N!}}}. \tag{15}$$

Napomenimo još da ako želimo da na početku programa svi objekti imaju iste šanse da budu izabrani kao centoridi, najbolje bi bilo uzeti za inicijalnu vrednost svih feromona veliki broj da bi nakon prve iteracije svi feromoni bili postavljeni na τ_{max}.

4.4.2 Broj iteracija s i broj mrava A

Iako veći broj iteracija daje mravima više vremena da iskonvergiraju ka boljem rešenju, predugo vreme izvršavanja je ono što uvek može predstavljati problem. Tabela 3, koja prikazuje rezultate kod kojih je A podešeno na 30, potvrđuje tu pretpostavku. Možemo primetiti da 200 iteracija u odnosu na 100 ne daje značajno bolju uspešnost, ali se prosečno vreme znatno povećava u ovom slučaju.

Zbog toga je odlučeno da se za broj iteracija uzme vrednost $s = 100$, dok je dalje podešavanje A urađeno u skladu s tim. Tabela 4 prikazuje prosečnu uspešnost dobijenu za različite vrednosti A.

Iako $A = 10$ u proseku daje za nijansu bolji prosečni rezultat, odlučeno je da se prednost ipak da onim vrednostima A koje su se pokazale najbolje na najvećem broju baza. Iz priloženog vidimo da 20, 25 i 30 mrava dovodi šest baza do najboljeg klasterovanja. Pošto $A = 25$ daje bolje rezultate u proseku, odlučujemo se za taj broj mrava.

Tabela 3: Poređenje prosečne uspešnosti za 100 i 200 iteracija. Vrednosti uzete za ovo testiranje su $A = 30$, $\alpha = 3$, $\beta = 2$ i $\rho = 0.01$, dok je prosečno vreme izraženo u milisekundama. Najveća promena u uspešnosti iznosi 0.35730930549% što je veoma malo u poređenju sa najmanjim porastom u prosečnom vremenu koji iznosi 110.1977%.

baza	# objekata (N)	# klastera (K)	# iteracija (s)	prosečna uspešnost	prosečno vreme (ms)
B.cancer	699	2	100	0.95565092989986	55312.5
			200	**0.95579399141631**	116265.625
H.colic	368	2	100	**0.9375**	5764.0625
			200	**0.9375**	15709.375
MM	961	2	100	0.669823100936524	61215.625
			200	**0.672216441207075**	132923.4375

Tabela 4: Poređenje prosečne uspešnosti za različite vrednosti A. Pri ovom testiranju su korišćenje vrednosti $s = 100$, $\alpha = 3$, $\beta = 2$, $\rho = 0.01$ za 10 seed-ova. Za broj mrava koji je dao najbolje rezultate je uzeta vrednost $A = 25$.

	# mrava (A)				
	10	15	20	25	30
B. cancer	0.954506437768	0.954506437768	**0.955937052933**	0.955364806867	0.955793991416
CVR	**0.875862968997**	**0.875862968997**	**0.875862968997**	**0.875862968997**	**0.875862968997**
Heart - h	0.57823129251701	0.58469387755102	**0.58843537414966**	**0.58843537414966**	**0.58843537414966**
Heart - c	0.29966996699670	0.29933993399340	0.29801980198020	0.29834983498350	**0.30363036303630**
dermatology	**0.329234972678**	0.306830601093	0.308196721311	0.313387978142	0.307377049180
Hepatitis	**0.70967741935**	**0.70967741935**	**0.70967741935**	**0.70967741935**	**0.70967741935**
H.colic	**0.9375**	**0.9375**	**0.9375**	**0.9375**	**0.9375**
L.cancer	**0.625**	**0.625**	**0.625**	**0.625**	**0.625**
MM	0.6687825182102	0.6691987513007	0.6690946930280	**0.6706555671173**	0.6698231009365
prosečno	**0.66427385294334**	0.66251212111407	0.66308034796917	0.66380367217557	0.66367770744883

4.5 Parametri α, β i ρ

Kao poslednji korak u podešavanju MMAS algoritma je ostalo određivanje parametara α, β i ρ. Za skupove vrednosti nad kojima su izvšena testiranja su uzeti $\{0, 2, 4, 6, 8\}$ za α i β, i $\{0.01, 0.05, 0.1, 0.3, 0.5, 0.7\}$ za ρ . Program je izvršen 100 puta za svaku kombinaciju ova tri parametara, na svakoj od devet baza. Rezultati testiranja su prikazani na Slici 1.

Iz priloženih trodimenzionalnih grafika možemo uočiti da su različite baze pokazale drugačije preference parametera. Sličan šablon se može primetiti kod baza Heart-h i MM, kao i kod baza Dermatology i Hepatitis. Samo se kod baze Heart-c dobila jedinstvena najbolja kombinacija parametara $\alpha = 4$, $\beta = 2$ i $\rho = 0.05$.

Takođe, primećujemo da izmene u ova tri parametra nisu dovela do značajnih promena u uspešnosti. Najveća razlika između najbolje i najgore uspešnosti je dostignuta za bazu Dermatology i iznosi 0.02694, dok je najmanja kod baze Hepatitis i iznosi 0.001677.

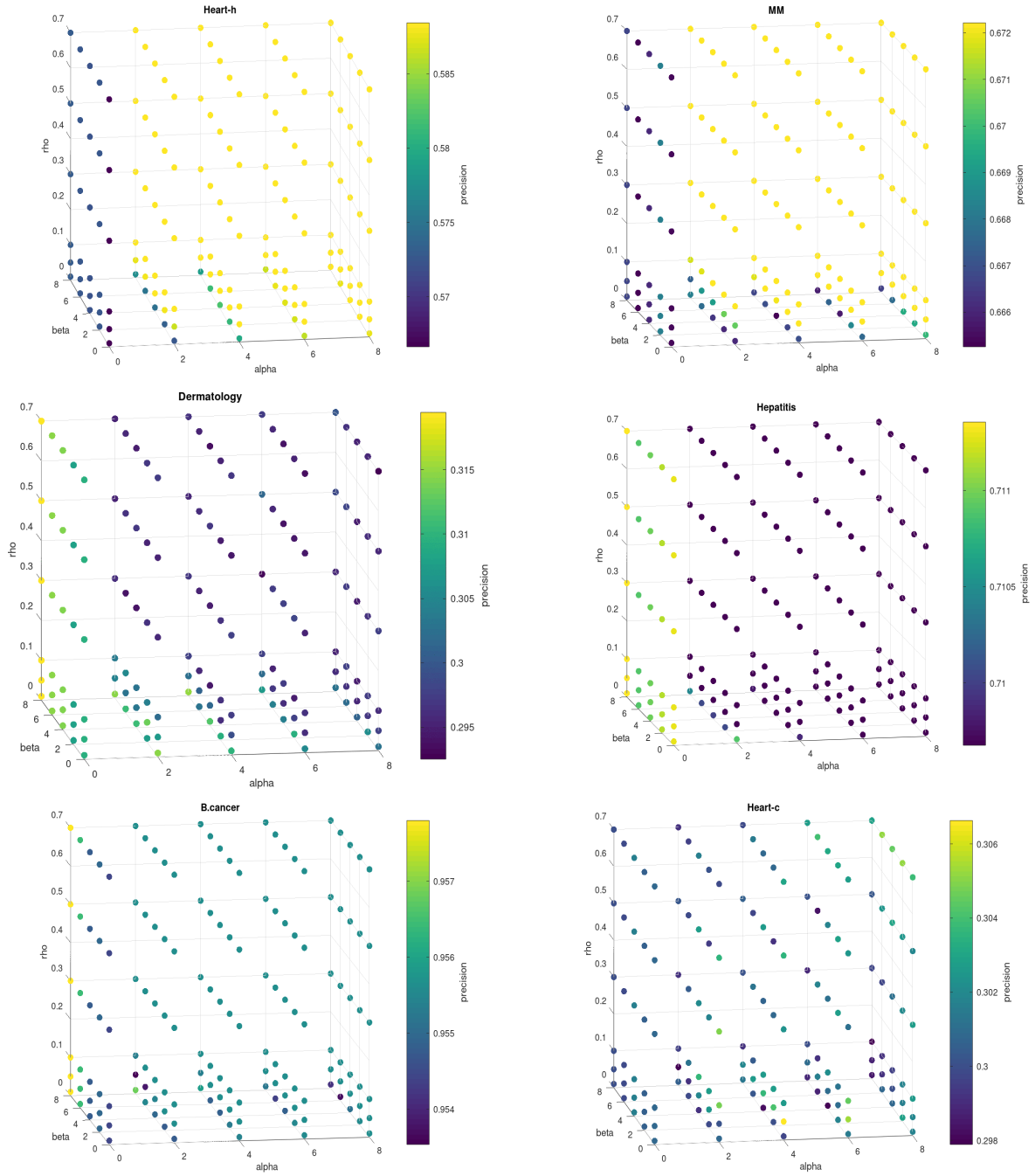
Iterarinjem kroz sve moguće kombinacije α, β i ρ, utvrđeno je da $\alpha = 2$, $\beta = 2$ i $\rho = 0.05$ maksimizuje zbir uspešnosti svih baza, pa je zato ova trojka uzeta za najbolju i iskorišćena u finalnom testiranju.

5 Testiranje i rezultati

Nakon podešavanja parametara konačno dolazimo do finalnih testiranja MMAS algoritma. Dakle, za parametre biramo vrednosti

$$s = 100,\ A = 25,\ \alpha = 2,\ \beta = 2,\ \rho = 0.05,$$

Slika 1: Prosečna uspešnost (za 100 seed-ova) u zavisnosti od parametara $\alpha \in \{0, 2, 4, 6, 8\}$, $\beta \in \{0, 2, 4, 6, 8\}$ i $\rho \in \{0.01, 0.05, 0.1, 0.3, 0.5, 0.7\}$. Boja tačke predstavlja uspešnost klasterovanja pri konkretnoj kombinaciji parametara, s tim da su svetlijom bojom označene bolje uspešnosti. Neprikazane baze nisu pokazale varijaciju u rezultatima pri promeni navedenih parametara. Ostali parametri su podešeni na $s = 100$, $A = 25$.



dok ćemo τ_{max} i τ_{min} posle svake promene globalno najboljeg mrava ažurirati po formulama (14) i (15).

Sva testiranja i programi u radu su implementirani u programskom jeziku Python u okruženju PyCharm Community Edition 2020.1.4 x64 u Windows 10 64bit-nom operativnom sistemu, sa Intel Celeron N3350 1.1GHz procesorom i 4GB RAM-a.

Program je izvršen za svaku bazu za 100 različitih seed-ova, a rezultati testiranja su prikazani u Tabeli 5. Urađeno je i poređenje (prikazano u Tabeli 6) sa 6 metoda neuralnih mreža čiji su rezultati preuzeti iz [13]. Spomenute metode su SNNE (*Selective Neural Network Ensemble*), MGNE (*Multi-Granulation Ensemble Method*), NNNE (*Neural Network Ensemble method without any assumption about distribution*), metoda Bag1 u kojoj je primenjen *bagging* algoritam nakon što su nedostajući atributi kod objekata zamenjeni prosečnom vrednošću tog atributa, kao i Bag2 kod kog je primenjen isti algoritam, ali su objekti sa nedostajućim atributima odstranjeni i najzad, metoda NN koja takođe odstranjuje objekte sa nedostajućim atributima, a zatim klasteruje po *single classifier* principu.

Tabela 5: Rezultati klasterovanja MMAS algoritma za parametre $s = 100$, $A = 25$, $\alpha = 2$, $\beta = 2$ i $\rho = 0.05$ i 100 seed-ova. U tabeli su prikazane najbolja uspešnost (broj u zagradi označava koliko puta se u 100 seed-ova pojavila najbolja uspešnost), kao i prosečna i najgora. Takođe su prikazane i standardna devijacija i prosečno vreme izvršavanja programa (izraženo u milisekundama).

baza	najbolja preciznost	prosečna preciznost	najgora preciznost	standardna devijacija	prosečno vreme izvršavanja (ms)
B.cancer	0.9556509298998569 (100)	0.9556509298998569	0.9556509298998569	0.0	48372.8125
CVR	0.8758620689655172 (100)	0.8758620689655172	0.8758620689655172	0.0	20999.375
Heart - h	0.5884353741496599 (99)	0.5880612244897959	0.5510204081632653	0.00374149659863946	12897.34375
Heart - c	0.32673267326732675 (2)	0.30244224422442245	0.28052805280528054	0.01456835080737284	36787.8125
Dermatology	0.3743169398907104 (1)	0.31314207650273224	0.2677595628415301	0.02115990601068578	62149.375
Hepatits	0.7096774193548387 (100)	0.7096774193548387	0.7096774193548387	0.0	3232.34375
H.colic	0.9375 (100)	0.9375	0.9375	0.0	5547.34375
L.cancer	0.625 (100)	0.625	0.625	0.0	347.96875
MM	0.6722164412070759 (89)	0.670624349635796	0.6493236212278877	0.004736224896197829	57270.3125
prosečno	0.6739324274	0.6642178126	0.650258007	0.004911775368	27511.63194

Tabela 6: Poređenje prosečne uspešnosti dobijene pomoću MMAS i 6 različitih metoda neuralnih mreža.

baza	SNNE	MGNE	NNNE	Bag1	Bag2	NN	MMAS
B.cancer	0.935	0.938	0.936	0.938	0.939	0.65	**0.9556509298998569**
CVR	0.942	0.945	0.942	**0.965**	0.964	0.513	0.8758620689655172
Heart - h	**0.816**	0.806	0.806	0.812	0.76	0.656	0.5880612244897959
Heart - c	**0.526**	0.519	0.516	0.52	0.524	0.437	0.30244224422442245
Dermatology	**0.886**	0.879	0.861	0.849	0.844	0.277	0.31314207650273224
Hepatits	0.676	0.676	0.682	0.663	0.681	0.551	**0.7096774193548387**
H.colic	0.735	0.723	0.701	0.778	0.633	0.583	**0.9375**
L.cancer	0.524	0.522	0.498	0.503	0.517	0.347	**0.625**
MM	0.834	0.836	0.801	0.829	**0.84**	0.504	0.670624349635796
prosečno	**0.764**	0.760	0.749	0.762	0.745	0.502	0.6642178126

Iz poređenja možemo zaključiti da iako je MMAS u proseku bolji samo od jedne (NN) metode, on dostiže najbolje rezultate na najvećem broju baza. U pitanju su 4 baze: B.cancer, Hepatitis, H.colic i L.cancer.

6 Zaključak i budući rad

MMAS, kao jednostavniji algoritam od spomenutih metoda neuralnih mreža, dao je gore prosečne rezultate od ovih metoda (sem do jedne). Međutim, postoji još prostora za njegovo poboljšanje, kao što su detaljnije podešavanje parametara i potencijalno davanje veće snage lokalnoj pretrazi. Ovo daje mogućnost da se MMAS približi rezultatima neuralnih mreža.

Jedan od ciljeva za budući rad je takođe i nameštanje parametara u skladu s bazama na kojima su testirana druga dva ACO algoritma za klasterovanje iz radova [10] i [8], pa zatim i poređenje sa istima.

Literatura

[1] *UCI Repository of machine learning databases for classification.* `https://archive.ics.uci.edu/ml/datasets.php`.

[2] Tatjana Davidovic, Nataša Glišovic, and Miodrag Raškovic. Bee colony optimization for clustering incomplete data. *Proc. The 7th International Conference on Optimization Problems and Their Applications, OPTA 2018, Omsk, Russia, July 08-14, 2018, In: S. Belim et al. (eds.): Proceedings of the School-Seminar on Optimization Problems and their Applications (OPTA-SCL 2018)*, pages 94–108, 2018.

[3] Tatjana Davidović, Dušan Teodorović, and Milica Šelmić. Bee colony optimization-part I: The algorithm overview. *Yugoslav Journal of Operations Research*, 25(1):33–56, 2015.

[4] Marco Dorigo. Optimization, learning and natural algorithms. *PhD Thesis, Politecnico di Milano*, 1992.

[5] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[6] Nataša Glišović and Miodrag Rašković. Optimization for classifying the patients using the logic measures for missing data. *Scientific Publications of the State University of Novi Pazar Series A: Applied Mathematics, Informatics and mechanics*, 9(1):91–101, 2017.

[7] Simon Goss, Serge Aron, Jean-Louis Deneubourg, and Jacques Marie Pasteels. Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76(12):579–581, 1989.

[8] Yucheng Kao and Kevin Cheng. An aco-based clustering algorithm. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 340–347. Springer, 2006.

[9] Manuel López-Ibáñez, Thomas Stützle, and Marco Dorigo. Ant colony optimization: A component-wise overview., 2018.

[10] Héctor D Menéndez, Fernando EB Otero, and David Camacho. Macoc: a medoid-based aco clustering algorithm. In *International Conference on Swarm Intelligence*, pages 122–133. Springer, 2014.

[11] Satyasai Jagannath Nanda and Ganapati Panda. A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary computation*, 16:1–18, 2014.

[12] Thomas Stützle and Holger H Hoos. Max–min ant system. *Future generation computer systems*, 16(8):889–914, 2000.

[13] Yuan-Ting Yan, Yan-Ping Zhang, Yi-Wen Zhang, and Xiu-Quan Du. A selective neural network ensemble classification for incomplete data. *International Journal of Machine Learning and Cybernetics*, 8(5):1513–1524, 2017.