Minimum k-clustering

Seminarski rad u okviru kursa  
Računarska inteligencija  
Matematički fakultet

Nikola Mićić, Tamara Miković

[**1 Uvod**](#_7g8hncbnpvki) **2**

[**2 Metodologija**](#_ycoc9hns0imq) **2**

[2.1 Metoda particionisanja](#_gan6y51to1pf) 2

[**3 Poređenje algoritama**](#_azlaa391m0yy) **3**

[3.1 K-means (K-sredina)](#_9drvhmg95hzt) 3

[3.2 K-medoids](#_ra8xi5vvxxzy) 4

[3.3 Clara](#_rlcesevk4efq) 5

[3.4 Clarans](#_v8dwuwmulren) 6

[3.5 PSO - Particle swarm optimization](#_ujy2cwkjwxmk) 8

[**4 Rezultati**](#_kiucr6z3bt9e) **10**

[4.1 Skup podataka - G2 sets (g2-2)](#_wftmy49ij6wb) 10

[4.2 Skup podataka - G2 sets (g2-4)](#_va8f189iiawj) 11

[4.3 Skup podataka - Birch sets (b2-sub-10)](#_scib44qxky38) 12

[4.4 Vizuelni prikaz za skup podataka - G2 sets (g2-4)](#_j3jk6s49cx8h) 13

[**4 Zaključak**](#_i286zm66i8a5) **14**

[**5 Literatura**](#_w961lkvgt3zw) **14**

## 

# 1 Uvod

Minimum K-Clustering predstavlja NP težak problem čije rešenje ima veliku primenu u oblastima kao što su istraživanje podataka, veštačka inteligencija i mašinsko učenje. Suština ovog problema jeste podela elemenata skupa X na K klastera - grupa elemenata takvih da su elementi u grupi međusobno slični (ili povezani), i da su elementi u različitim grupama međusobno različiti (ili nepovezani).

Zadatak je minimizovati maksimalno rastojanje između dva elementa iz istog klastera[1].

Najveći doprinos i uticaj na razvoj klasterovanja imaju trojica istraživača: Robert Choate Tryon, Joe H. Ward Jr., Stephen C. Johnson. Ovi autori su imali različite pristupe u vezi sa prirodom klaster analize. Brzi razvoj računara i temeljni značaj grupisanja doprineli su popularnosti ove metode.

# 2 Metodologija

Kao što naziv projekta govori, potrebno je izvršiti podelu svih objekata na k klastera, i zbog toga smo koristili metodu koja zahteva broj klastera (k) kao početnu tačku. Jedna od takvih metoda je metoda particionisanja koju ćemo sada predstaviti.

## 2.1 Metoda particionisanja

Metode particioniranja su najosnovniji tip klaster analize, oni organizuju objekte skupa u nekoliko ekskluzivnih grupa klastera (tj. svaki objekat može biti prisutan samo u jednoj grupi).

Algoritmi za particioniranje zahtevaju broj klastera (k) kao početnu tačku. Tako je dat skup podataka D, koji se sastoji od n tačaka i k (k << n), algoritam particioniranja organizuje objekte u k particija (klastera).

Klasteri se formiraju optimizacijom objektivnog kriterijuma za particioniranje, kao što je funkcija različitosti na osnovu udaljenosti, tako da su objekti u klasteru „slični“ jedni drugima i „različiti“ od objekata u drugim klasterima u smislu atributa skupa podataka.

# 

# 3 Poređenje algoritama

Tokom rada na ovom projektu smo implementirali više algoritama za klasterovanje podataka i kasnije ih testirali nad različitim skupom podataka koji se međusobno razlikuju u veličini i dimenziji. Rezultate ovih testiranja možete pronaći u odeljku *Rezultati.* Pre toga ćemo predstaviti algoritme koje smo implementirali i uz pomoć kojih smo izvršili testiranja i minimizaciju maksimalne distance između elemenata istog klastera.

Algoritmi na kojima smo radili su sledeći:

1. K-means
2. K-medoids
3. CLARA
4. CLARANS
5. PSO - optimizacija K-means

## 3.1 K-means (K-sredina)

Jedan od najpoznatijih i najviše korišćenih nehijerarhijskih metoda klasterovanja je algoritam K-means (K-sredina) (Ball and Hall 1965.godina; MacQueen 1967. godina; Anderberg 1973.godina). Ovaj algoritam, zajedno sa njegovim varijacijama je poznat kao brz algoritam (u smislu vremenske složenosti) i primenjiv je na velike skupove podataka. Algoritam K-means se primenjuje u radu sa neprekidnim tipovima podataka. Klasteri su opisani pomoću centroida koji predstavlja aritmetičku sredinu objekata koji se nalaze u klasteru. Počinje se od k klastera (određuju se proizvoljno, ili na osnovu prethodnog klasterovanja), a objekti se razvrstavaju u one klastere čiji centroid im je najbliži.

Označimo sa *A* konačan skup tačaka n - dimenzionalnog prostora

gde je i=1,…,m.

***Implementacija K-means algoritma***

Korak 1. Izabrati početno rešenje koje se sastoji od k centara (ne moraju pripadati skupu A)

Korak 2. Dodeliti tačke i najbližem centru čime je dobijena k-podela skupa A.

Korak 3. Ponovo odrediti centre za ovu novu podelu i vratiti se na korak 2, sve dok se ne poklope centri klastera u poslednje dve iteracije.

***Razlozi velike primene algoritma K-means su sledeći:***

* Vremenska složenost je  *O(m × k × l)* , gde je m broj slučajeva, k je broj klastera, a l broj iteracija algoritma. Kako su k i l unapred fiksirani, vremenska složenost ovog algoritma je linearna u odnosu na veličinu uzorka.
* Prostorna složenost je  *O(k+m)*.
* Algoritam je nezavisan od redosleda. Za dati inicijalni skup centara klastera, generiše istu podelu bez obzira na redosled slučajeva.

***Nedostaci ovog algoritma su sledeći:***

* Mora se unapred odrediti (zadati) broj klastera k moraju se pronaći početni centroidi da bi startovao algoritam.
* *Hartigan and Wong* (1979) sugerišu korišćenje aktuelnih objekata kao početnih centara za klastere. Oni mogu biti izabrani na slučajan način
* Često konvergira ka lokalnom optimumu
* Osetljivost na šum i autlajere

## 3.2 K-medoids

Ovaj algoritam je vrlo sličan algoritmu K-means, a razlika je uglavnom po načinu predstavljanja različitih klastera. Svaki klaster predstavljen je najcentriranijim objektom u klasteru, a ne implicitnom sredinom koja možda ne pripada skupu podataka. Metoda K-medoids je robusnija od algoritma K-means u prisustvu šuma i autlajera, jer na medoide manje utiču odstupanja ili druge ekstremne vrednosti od aritmetičke sredine. Međutim, njegova obrada je skuplja od metode K-means. Obe metode zahtevaju od korisnika da navede k, broj klastera.

Algoritam bira k medoide da predstavljaju k klastera. Zatim se stvaraju klasteri dodeljivanjem svakog od preostalih objekata najbližem medoidu. PAM je jedan od najkorišćenijih k-medoids algoritama, gde je PAM skraćenica za particioniranje oko medoida (engleski - partition around medoids).

***Implementacija K-medoids algoritma***

Korak 1. Proizvoljno odabrati iz skupa k elemenata za pocetne medoide.

Korak 2. Preostale elemente raspodeliti po klasterima tako da udaljenost elementa prema medoidu dodeljenog klastera bude manje nego prema drugim medoidima

Korak 3. Zameni svaki od medoida s jednim od preostalih objekata sve dok se kvalitet klasterovanja ne poveća

Korak 4. Iteriraj do konvergencije

***Pozitivne strane algoritma:***

* Najatraktivnije svojstvo ove metode je njena robusnost.
* Upotreba medoida za definisanje klastera čini ovu metodu vrlo otpornom na odstupanja u podacima. Ne mora da pohranjuje veliku količinu informacija pored originalnih podataka u memoriju. Sve što je potrebno je oznaka odabranog objekta.

***K-medoids algoritam ima niz nedostataka, a to su:***

* potrebno je algoritmu proslediti broj klastera k kao ulazni parametar. Ovo zahteva dobro nagađanje od korisnika, koje možda nije dostupno
* vremenska složenost po iteraciji algoritma je O (n*2*).
* koristeći medoide, algoritam ne pruža nikakav način opisivanja klastera, osim u smislu detalja o članstvu.
* K-medoids algoritmi su veoma skupi kada su skup podataka i k vrednost velike

## 3.3 Clara

Problem koji se javlja kod primene različitih algoritama klasterovanja je rad sa velikim brojem podataka i velikim brojem obeležja. Navešćemo dva pristupa, bez obzira na vrstu promenljivih uključenih u analizu. Kaufman i Rousseeuw (1990) su predložili CLARA (Clustering LARge Applications) algoritam za klasterovanje velikih skupova podataka, koji predstavlja kombinaciju postupka uzorkovanja i PAM (Partitioning Around Medoids) algoritma klasterovanja. CLARA algoritam primenjuje mali uzorak iz velikog skupa podataka, koristi PAM za generisanje k-medoida iz uzorka, koji dalje služe za klasterovanje ostatka skupa. Računska složenost CLARA algoritma u pojedinačnoj iteraciji iznosi: *O(kmu2+k(m-k))*, gde je *m* veličina skupa, *k* broj klastera, a *mu=40+2k* obim uzorka.

CLARA algoritam proširuje k-medoids algoritam izborom potencijalno optimalnih tačaka podataka iz malih uzoraka skupa podataka. Ovo predračunavanje potencijalnih centara omogućava smanjenje složenosti medoidne pretrage i algoritmu pruža i brzinu i skalabilnost.

***Implementacija Clara algoritma***

Korak 1. Prvo se iz ukupnog skupa podataka izdvaja S uzoraka kako bi se stvorio podskup uzoraka.

Korak 2. Zatim algoritam grupiše uzorak pomoću PAM algoritma da bi dobio k centralnih tačaka.

Korak 3. Zatim primenjuje ove centralne tačke na ceo skup podataka, izračunavajući ukupne troškove trenutnog klastera i upoređujući ih sa ukupnim troškovima prethodnog klastera.

Korak 4. Zatim algoritam, zasnovan na rezultatima poređenja za odabir optimalnih tačaka klastera, odluuje o nastavku iteriranja.

***Pozitivne strane Clara algoritma***

* Veoma dobar algoritam za klasterovanje nad velikom bazom podataka
* Pruža bolje performanse prilikom klasterovanja od prethodno navedenih algoritama

***Nedostaci Clara algoritma***

* Ne pruža najbolje performanse prilikom klasterovanja nad malom bazom podataka
* Sa povećanjem broja klastera, performanse CLARA algoritma brzo slabe

## 3.4 Clarans

CLARANS (Clustering Large Applications based on RANdom Search) je algoritam koji su predložili Ng i Han (1994) kao način da poboljšaju CLARA metod. Predstavlja kompromis između troškova i efikasnosti upotrebe uzoraka za dobijanje klastera. Ovaj metod identifikuje kandidate za centroide klastera korišćenjem ponovljenih slučajnih uzoraka iz originalnog skupa podataka. Autori tvrde da on obezbeđuje bolje klastere pomoću malog broja “traženja” (poređenje k alternativnih objekata kao predstavnika klastera).

***Implementacija Clarans algoritma***

Korak 1. Nasumično bira **k** objekata u skupu podataka kao trenutne medoide

Korak 2. Zatim nasumično bira trenutni medoid ***x*** i objekat ***y*** koji nije jedan od trenutnih medoida

Korak 3. Zatim proverava sledeće stanje: Može li zamena ***x*** sa ***y*** poboljšati kriterijum apsolutne greške? Ako je odgovor da, zamena je izvršena.

Korak 4. CLARANS vrši takvu nasumičnu pretragu ***I*** puta.

Korak 5. Skup trenutnih medoida posle ***I*** koraka smatra se lokalnim optimumom.

Korak 6. CLARANS ponavlja ovaj slučajni postupak ***m*** puta i vraća najbolji lokalni optimum kao konačni rezultat.

***Objašnjenje***

Algoritam zahteva ***numlocal*** (količina iteracija za rešavanje problema), ***maxneighbor*** (maksimalan broj ispitanih suseda) i broj klastera (k) koji će se formirati kao ulaz.

Tada započinje iteracija, ***i*** se postavlja na 1, pre čega se ***mincost*** (što je optimalni trošak) postavlja na beskonačno, a ***bestnode*** (optimalni medoidi) postavlja na prazan korpicu.

Sada je izabrano ***k*** slučajnih tačaka podataka kao trenutni medoidi i klasteri se formiraju pomoću ovih tačaka podataka (euklidska udaljenost se može koristiti za pronalaženje najbližeg medoida za formiranje klastera).

Nakon toga započinje nova petlja, gde je ***j*** postavljeno na 1. Odabere se slučajni trenutni medoid i bira tačka podataka slučajnog kandidata (slučajni sused) za zamenu sa trenutnim medoidom.

Ako zamena kandidatske tačke podataka daje niži ***TotalCost*** (što je zbir rastojanja između svih tačaka u klasterima sa njihovim odgovarajućim medoidima) od trenutne medoide, tada se vrši zamena.

Ako se zamena izvrši, j se ne povećava, inače ***j = j +1***. Jednom kada je ***j>maxneighbor***, uzimaju se trenutni medoidi i njihov ***TotalCost*** se upoređuje sa ***mincost-om***. Ako je ukupni trošak manji od mincosta, tada se Bestnode ažurira kao trenutni medoidi.

***i*** se posle povećava, a ako je veći od ***numlocal-a***, tada se Bestnode daje kao izlaz, u suprotnom se ponavlja ceo proces.

***Pozitivne strane Clarans algoritma i poređenje sa Clara algoritmom***

* Veoma dobar algoritam za klasterovanje nad velikom bazom podataka
* CLARANS se pokazao efikasnijim nego CLARA algoritam u pogledu vremena izvršavanja, za isto vreme izvršavanja donosi bolje rezultate.
* CLARANS se pokazao efikasnijim nego CLARA pri radu nad manjom bazom podataka
* Sa povećanjem broja klastera, performanse CLARANS algoritma su bolje nego kod CLARA algoritma

***Nedostaci Clarans algoritma***

* Ne pruža najbolje performanse prilikom klasterovanja nad malom bazom podataka

## 3.5 PSO - Particle swarm optimization

Optimizacija rojem čestica (eng. Paricle Swarm Optimization – PSO) je stohastički algoritam koji se bazira na populaciji rešenja. Algoritam optimizacije rojem čestica otkriven je sasvim slučajno, pri pokušaju da se na računaru simulira kretanje jata ptica. Reynolds C.W. u svom radu 1986. godine razmatra jato ptica kao skup čestica gde svaka čestica (tj. ptica) svoj let prilagođava sledećim pravilima:

* Izbegavaj sudar
* Prilagođavanje brzine leta bliskim pticama
* Opstanak u blizini ostalih ptica

Sam algoritam je u određenoj meri inspirisan i sociološkim interakcijama između pojedinaca u populaciji. Svaka čestica je potencijalno rešenje razmatranog problema i koristi sopstveno iskustvo i iskustvo susednih čestica da bi odabrala kako da se kreće u prostoru pretrage. U fazi inicijalizacije svakoj čestici je dat slučajno odabrani položaj i početna brzina. Pozicija čestice predstavlja rešenje problema i zato ima vrednost datu funkcijom cilja. Dok se kreću u prostoru pretrage čestice pamte položaj, tj. poziciju najboljeg rešenja koje su našle. U svakoj iteraciji algoritma svaka četica se kreće brzinom koja je suma tri komponente: stare brzine, brzine koja pokreće česticu ka poziciji u prostoru pretrage gde je prethodno pronašla najbolje rešenje do tada i brzine koja pokreće česticu prema lokaciji u prostoru pretrage gde su susedne čestice našle najbolje rešenje do tada. Roj čestica je u prirodi stohastičan, on iskorišćava vektor brzine kako bi ažurirao trenutnu poziciju svake čestice u roju. Vektor brzine se ažurira na temelju pamćenja svake čestice, što koncepcijski odgovara autobiografskoj memoriji, kao i na temelju znanja koje je stekao roj kao celina.Pozicija čestice u jatu je ažurirana na temelju socijalnog

ponašanja roja koji se prilagođava svom okruženju stalnim traženjem boljih pozicija tokom vremena.

Numerički, pozicija x čestice i u iteraciji k + 1 je ažurirana na sledeći način:

pri čemu je pripradajući ažurirani vector brzine, a Δt je vrednost vremenske step funkcije. Vektor brzine svake čestice se računa kao:

pri čemu je vektor brzine u iteraciji , i predstavljaju najbolju poziciju čestice ikad i globalnu najbolju poziciju čitavog roja sve do trenutne iteracije , dok

r predstavlja nasumični broj iz intervala [0,1].

Preostali članovi su konfiguracijski parametri koji igraju važnu ulogu u konvergencijskom ponašanju PSO-a. Član (kognitivni, samospoznajni parameter) predstavlja stepen poverenja u globalnom najboljem rešenju (najbolje pronađeno rešenje od jata kao celine). Uglavnom se uzima 1,8 < = < 2,2. Poslednji član ω je iteracijska varijabla koja je iskorišćena za kontrolisanje istraživačkih sposobnosti roja tako da skalira vrednost trenutne brzine, te na taj način utiče na iznos ažuriranog vektora brzine. Većim vrednostima interacijske varijable (ω) vršimo globalno pretraživanje zbog toga što se ažurirani vektor brzine brže povećava dok zadavanjem manje vrednosti inercijske varijable vrednost ažuriranog vektora brzine postaje manja pa se tako novi položaj čestice ograničava na manje područje prostora istraživanja, tj. omogućavamo lokalno pretraživanje. [3]

***Implementacija PSO algoritma***

Korak 1. Inicijalizacija čestica

Korak 2. Za svaku česticu se računa fitness vrednost - data funkcijom cilja i ako je ona bolja od prethodne najbolje vrednosti te čestice, dolazi do zamene trenutne najbolje fitness vrednosti na novu izračunatu (pBest)

Korak 3. Bira se čestica sa najboljom fitness vrednošću (pBest) i postavlja se na trenutnu najbolju globalnu fitness vrednost (gBest)

Korak 4. Za svaku česticu se računa tj. apdejtuje brzina čestice (Jednačina za velocity) i pozicija čestice (Jednačina position)

Korak 5. Izvršavanje algoritma sve dok se ne postigne maksimalan broj iteracija ili kriterijum minimalne greške

***Optimizacija PSO algoritma***

PSO algoritam smo optimizovali tako što smo napravili hibridnu verziju ovog algoritma, u kojoj se početna inicijalizacija vrši uz pomoć K-means algoritma. Prvobitna verzija inicijalne čestice generiše nasumičnim odabirom uz pomoć funkcije random.choice iz numpy biblioteke, dok hibridna verzija inicijalne čestice generiše uz pomoć K-means algoritma i pokazalo se da ova hibridna verzija daje bolje rezultate zbog boljeg inicijalnog raspoređivanja čestica.

# 

# 

# 4 Rezultati

U ovom poglavlju će biti prikazani rezultati testiranja koja smo izveli nad implementiranim algoritmima koji su bili predstavljeni u prethodnom poglavlju. Algoritmi su testirani nad 3 različita skupa podataka koja se razlikuju u veličini i dimenziji i nad dve različite vrednosti za broj klastera (K), kako bi se što bolje uvidela razlika i efikasnost datih algoritama. Rezultati testiranja će biti predstavljeni u formi tabele, a pored toga se može videti i vizuelni prikaz izvršenog klasterovanja.

Tabela se sastoji od tri kolone, pri čemu se u prvoj koloni može videti naziv algoritma nad kojim je testiran dati skup podataka, zatim je prikazana maksimalna distanca između dva elementa koja pripadaju istom klasteru, u odnosu na svih k klastera do kojih smo došli primenom algoritama. U drugoj koloni je prikazana prosečna vrednost te maksimalne distance za 5 pozivanja programa, dok je u trećoj koloni najbolja vrednost maksimalne distance do koje smo došli nakon tih 5 pozivanja.

Maksimalnu distancu između dva elementa istog klastera smo tražili tako što smo od svih tačaka klastera nalazili tačke koje se nalaze po obodu klastera, tj. tačke koje čine konveksni omotač, a zatim tražili maksimalnu distancu između između parova tih tačaka.

## 4.1 Skup podataka - G2 sets (g2-2)

Za prvo testiranje smo uzeli manji skup podataka zbog algoritama kao što je k-medoids, koji nije efikasan i veoma je spori na velikim skupovima podataka. G2 skup g2-2 je 2-dimenzioni skup podataka, a veličina skupa podataka N je 500.

Rezultati testiranja izvršeni nad datim skupom podataka, pri čemu je za broj klastera K - uzeta vrednost 3:

|  |  |  |
| --- | --- | --- |
|  | Average | The Best |
| К-means | 498.156840035164 | 484.16216291651705 |
| К-medoids | 476.415151009756 | 464.400689060643 |
| Clara | 509.546102134897 | 497.20921149954575 |
| Clarans | 457.456401217642 | 446.25665261147645 |
| PSO | 453.537958053210 | 447.5712233823797 |
| PSO - hybrid | 445.458135478621 | 443.2786933747211 |

Rezultati testiranja izvršeni nad istim skupom podataka, pri čemu je za broj klastera K - uzeta vrednost 7:

|  |  |  |
| --- | --- | --- |
|  | Average | The Best |
| К-means | 308.51456413027 | 294.24139749532185 |
| К-medoids | 289.54684102314 | 262.4576156258378 |
| Clara | 397.76432899843 | 385.8108344772085 |
| Clarans | 311.62984530264 | 301.6819517306264 |
| PSO | 449.84863910463 | 440.8004083482682 |
| PSO - hybrid | 312.40972504538 | 307.54674441456865 |

## 4.2 Skup podataka - G2 sets (g2-4)

Za drugo testiranje smo uzeli višedimenzioni skup podataka čija veličina je veća od prethodnog skupa i algoritam k-medoids nije testiran nad ovim skupom podataka jer je veoma spor.

G2 skup g2-4 skup je 4-dimenzioni skup podataka, a veličina skupa podataka N je 2048.

Rezultati testiranja izvršeni nad datim skupom podataka, pri čemu je za broj klastera K - uzeta vrednost 3:

|  |  |  |
| --- | --- | --- |
|  | Average | The Best |
| К-means | 77.82564412652021 | 75.50496672405067 |
| Clara | 78.53049622647382 | 76.47875521999558 |
| Clarans | 79.54686870213545 | 76.47875521999558 |
| PSO | 76.64786958411305 | 75.15982969645421 |
| PSO - hybrid | 74.02565486451369 | 72.178344238091 |

Rezultati testiranja izvršeni nad istim skupom podataka, pri čemu je za broj klastera K - uzeta vrednost 7:

|  |  |  |
| --- | --- | --- |
|  | Average | The Best |
| К-means | 69.4684842612214 | 67.09694478886502 |
| Clara | 75.50496672405067 | 73.25298628724975 |
| Clarans | 72.01654812654849 | 70.78841713161836 |
| PSO | 75.54687410364048 | 73.25298628724975 |
| PSO - hybrid | 68.48564563102301 | 66.63332499583073 |

## 

## 4.3 Skup podataka - Birch sets (b2-sub-10)

Za treće testiranje smo uzeli višedimenzioni skup podataka čija veličina je veća od prethodnog skupa i algoritam k-medoids nije testiran nad ovim skupom podataka, zbog istog razloga kao i malopre.

Birch skup b2-sub-10 je 2-dimenzioni skup podataka, a veličina skupa podataka N je 10000.

Rezultati testiranja izvršeni nad datim skupom podataka, pri čemu je za broj klastera K - uzeta vrednost 3:

|  |  |  |
| --- | --- | --- |
|  | Average | The Best |
| К-means | 39324.45135465131 | 37676.967778737184 |
| Clara | 43606.26220610734 | 41912.4089620246 |
| Clarans | 40937.62350210379 | 40113.07214612214 |
| PSO | 40124.5168163891 | 38381.35866537296 |
| PSO - hybrid | 38211.1589465382 | 37676.967778737184 |

Rezultati testiranja izvršeni nad istim skupom podataka, pri čemu je za broj klastera K - uzeta vrednost 7:

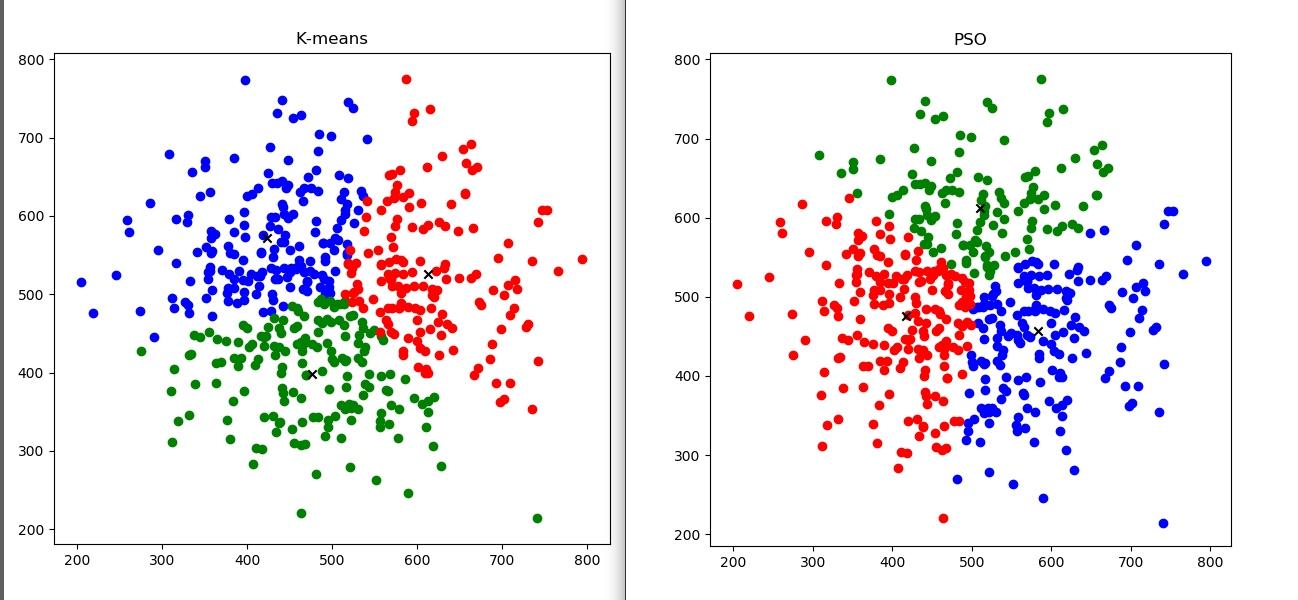
|  |  |  |
| --- | --- | --- |
|  | Average | The Best |
| К-means | 22456.45131548152 | 19919.303326170822 |
| Clara | 23531.83225696956 | 22682.89822751934 |
| Clarans | 22482.48841230942 | 19866.676647089214 |
| PSO | 22127.54601387058 | 20575.143669000223 |
| PSO - hybrid | 19548.41654810107 | 18963.281704388617 |

## 4.4 Vizuelni prikaz za skup podataka - G2 sets (g2-4)

Što se tiče vizuelnog prikaza, svako testiranje je pratilo iscrtavanje preko matplotlib-a, ali nije imalo smisla da ovde ubacimo sve slike tih testiranja, tako da smo odlučili da ubacimo jedno poređenje tog iscrtavanja za algoritme K-means i PSO, nad skupom podataka g2-4.

U ovom slučaju max distanca za K-means je iznosila: 502.3719352189383

Max distanca za PSO je iznosila: 443.85250950063664



# 

# 4 Zaključak

Na osnovu rezultata iz prethodnog poglavlja možemo videti kako smo postepeno smanjivali maksimalnu distancu između elemenata istog klastera, implementiranjem različitih algoritama i na osnovu testiranja smo zaključili da je minimizaciju maksimalne distance između dva elementa istog klastera proizveli algoritmi PSO - hybrid. Samo u jednom slučaju se desilo da je K-means imao bolje rezultate, a to je u slučaju prvog testiranja nad malom bazom podataka pri deljenju podataka na 7 klastera.

# 5 Literatura

[1] Viggo Kann, KKTH (EECS) url: <https://www.csc.kth.se/~viggo/wwwcompendium/node129.html>

[2] <https://www.pmf.ni.ac.rs/download/master/master_radovi_matematika/matematika_master_radovi/2018_2/2018-10-31-kk.pdf>

[3] Optimization of cutting parameters by nature-inspired algorithms: Anđelko Aleksić,Milenko Sekulić,Fakultet tehničkih nauka, Novi Sad

[4]

<https://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html>