KMESA algoritam klasterovanja
(K-Means with Simulated Annealing)

1. Uvod

Algoritam klasterovanja K-sredina (eng. K-Means clustering [1]) je jedan od najpopularnijih algoritama klasterovanja zbog svoje jednostavnosti, interpretabilnosti i efikasnosti nad određenim skupovima podataka. Jedan je od najvažnijih predstavnika modela sa prototipom, tj. modela zasnovanog na reprezentativnim predstavnicima [2] koji se u ovom slučaju nazivaju centroidi. Performanse algoritma u velikoj meri zavise od početnog koraka, tj. od inicijalizacije [3] početnih centroida. U slučaju nepogodne inicijalizacije algoritam može da se „zaglavi“ u lokalnim optimumima [4] i završi sa suboptimalnim klasterovanjem [5], što je neželjena pojava koju je potrebno izbeći. U prevazilaženju ovog problema razvijene su mnogobrojne modifikacije algoritma, a jedna od njih biće detaljno razrađena u daljem tekstu.

1.1 Detalji

Za rad algoritma neophodno je navesti tačan broj klastera k i inicijalizovati početne centroide, koji predstavljaju odgovarajuće klastere u skupu podataka. Zatim se algoritam izvršava iterativno: u svakoj iteraciji centroidi se ažuriraju na pozicije srednjih tačaka tekućih klastera koje predstavljaju, a zatim se vrši dodela tačaka iz skupa podataka najbližim centroidima. Dodela tačaka klasterima (tj. reprezentativnim centroidima) zasniva se na funkciji sličnosti (eng. similarity function [6]), odnosno funkciji različitosti, koja računa rastojanje tačaka do svih centroida – tačka se dodeljuje onom klasteru čijem centroidu je najbliža. Ova funkcija rastojanja je najčešće euklidska [7], ali može biti i neka druga. Iterativni postupak se ponavlja sve dok nije dostignut maksimalan broj iteracija ili dok nije ispunjen kriterijum konvergencije.

Iako algoritam nenadgledanog učenja [8], algoritam K-sredina ima svoj optimizacioni objektiv [9] koji pokušava da minimizuje. Neka je $x_1, x_2, \ldots, x_m$ $n-$ dimenzionalni skup tačaka koje je potrebno klasterovati a $A_1, A_2, \ldots, A_k$ oznake klastera sa odgovarajućim centroidima $c_1, c_2, \ldots, c_k$. Optimizacioni objektiv zasniva se na sumi srednjekvadratnih grešaka (eng. Sum of Squared Error, SSE [10]) definisane sa

$$\mathrm{SSE} := \sum_{\mathrm{x}\,\epsilon\,\mathrm{A_i}} \|\mathrm{x} - \mathrm{c_i}\|^2, \tag{1}$$

koja predstavlja sumu kvadrata euklidskog rastojanja tačaka iz posmatranog klastera A_i od svog reprezentativnog predstavnika, tj. centroida c_i. Ovaj izraz definiše grešku klastera A_i. Na osnovu ovog izraza optimizacioni izraz postaje minimizacija ove greške po svim klasterima A_i:

$$\min_{A_i, c_i} \sum_{i=1}^{k} \sum_{x \, \epsilon \, A_i} \|x - \, c_i\|^2 \tag{2}$$

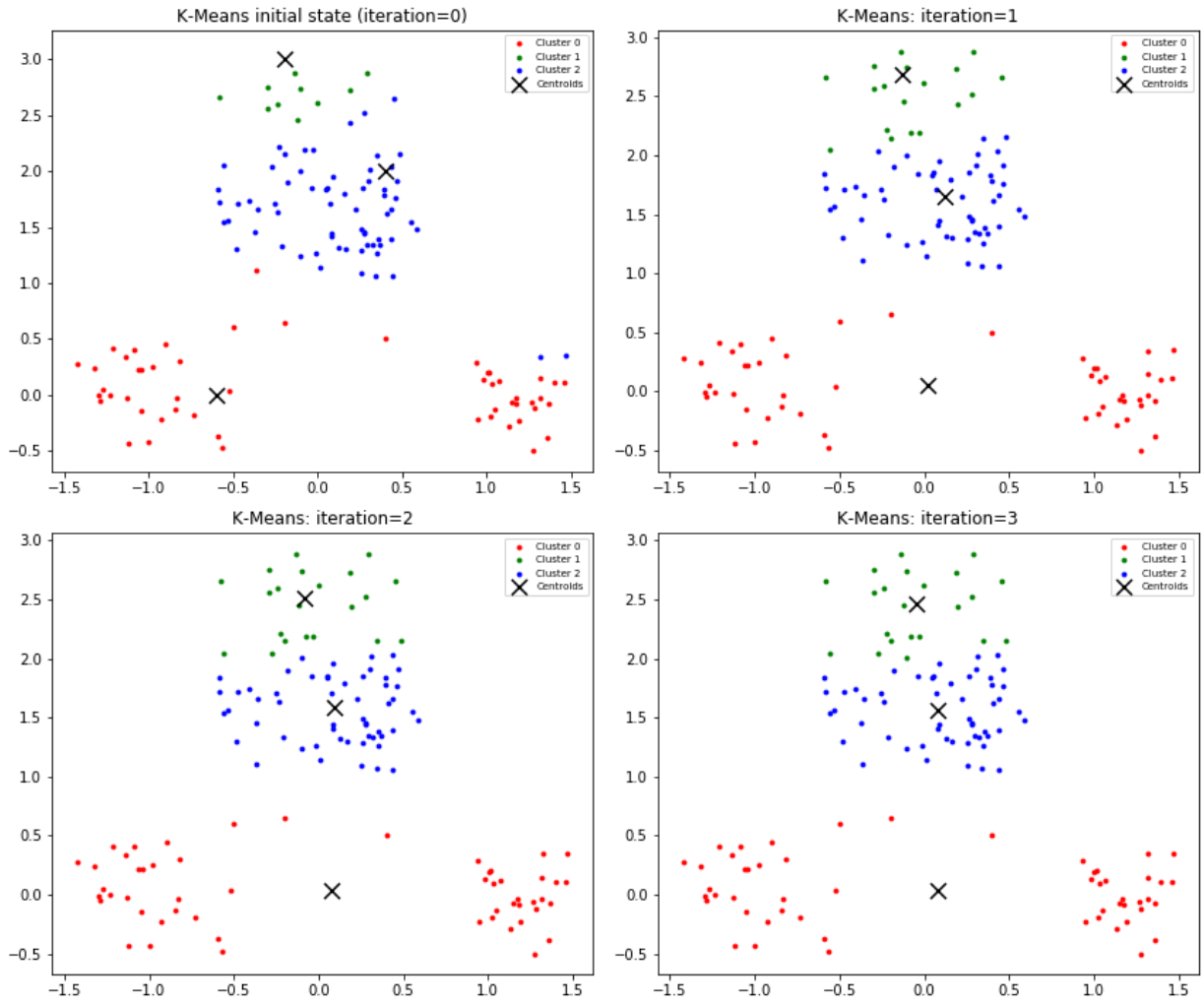
Sada je moguće jednostavno predstaviti pseudokod algoritma:

Algoritam: K-sredina
Ulaz: Skup tačaka x_1, x_2, ..., x_m za klasterovanje Izlaz: Skup centroida c_1, c_2, ..., c_k kao reprezentativnih predstavnika pronađenih klastera
Inicijalizuj početni skup centroida c_1, c_2, ..., c_k; Dok nije ispunjen kriterijum zaustavljanja ponavljaj: Izvrši dodelu tačaka x_1, x_2, ..., x_m odgovarajućim klasterima na osnovu predefinisane funkcije rastojanja; Ažuriraj pozicije tekućih centroida c_1, c_2, ..., c_k u cilju minimizacije optimizacionog izraza; Vrati c_1, c_2, ..., c_k kao rešenje;

Jasno je da korak ažuriranja centroida u cilju minimizacije (2) povlači postavljanje vrednosti tekućeg centroida c_i na središnju tačku klastera A_i:

$$c'_{ij} = \frac{1}{l} \sum_{j=1}^{n} \sum_{x \epsilon A_i} x_j \tag{3}$$

gde je l broj tačaka u klasteru A_i, a x_j predstavlja $j - ti$ atribut tačke iz skupa podataka po kome se računa srednja vrednost. Po ovom principu ažuriranja centroida algoritam je i dobio ime K-sredina. Međutim, na osnovu iterativnog pristupa ovaj postupak ne garantuje pronalazak globalno najamnje vrednosti (2), u zavisnosti od početne inicijalizacije centroida, zbog čega je moguće da algoritam ovu vrednost minimizuje samo lokalno (tačnije, minimizacijom pojedinačne greške SSE u okviru tekućih klastera) čime se dobija suboptimalno klasterovanje, što ilustruje sledeći jednostavan primer:



Slika 1: Problem suboptimalnog klasterovanja kod algoritma K-sredina

Već u trećoj iteraciji algoritam je dostigao kriterijum konvergencije (u primeru, uslov konvergencije je da razlika u distanci odgovarajućih centroida bude manja od unapred zadatog praga tolerancije, tj. $dist\left(c_i^{(it)} - c_i^{(it-1)}\right) < tol, \forall i \, \epsilon \, \{1, \dots, k\}$). Očigledno je da izvršeno klasterovanje nije zadovoljavajuće i da nije dostignut globalni minimum izraza (2) već lokalni, odnosno došlo je do suboptimalnog klasterovanja.

1.2 Modifikacija algoritma

Za rešenje problema prikazanog u primeru 1 razvijene su mnogobrojne modifikacije algoritma K-sredina, a neki od njih su: K-sredina sa primenom Lojdovih metoda [11], K-sredina sa filterom [12], Dvoslojni algoritam K-sredina [13], Brzi modifikovani globalni algoritam K-sredina sa inkrementalnom konstrukcijom klastera [14], Efikasni *pojačani algoritam K*-sredina [15],

Primena simuliranog kaljenja nad algoritmom K-sredina u kombinovanom modelovanju [16], itd. Ovaj rad koristi ideju simuliranog kaljenja za modifikaciju algoritma, spomenutu u poslednjem navedenom radu. Detalji ove modifikacije biće izložene u narednoj sekciji.

2. Simulirano kaljenje i K-sredina

2.1 Uvod

Simulirano kaljenje (eng. Simulated Annealing [17]) je probabilistička tehnika za aproksimaciju globalnog optimuma date funkcije cilja. Specijalno, ono predstavlja metaheuristiku [18] za aproksimaciju globalne optimizacije u velikom prostoru pretrage nekog optimizacionog problema. Naziv potiče od kaljenja u metalurgiji, tehnike koja podrazumeva zagrevanje i kontrolisano hlađenje metala zarad povećanja veličine njegovih kristala i umanjivanje unutrašnjih defekata.

U algoritmu simuliranog kaljenja ideja sporog hlađenja metala interpretira se kao postepeno smanjivanje verovatnoće prihvatanja lošijih rešenja u prostoru pretrage, čime se dopušta iscrpnija pretraga za globalnim optimumom. Uopšteno, algoritam vrši lokalnu pretragu za boljim rešenjem i ažurira svoje tekuće rešenje ukoliko je bolje rešenje pronađeno, ili ako je temperatura (tj. verovatnoća) prihvatanja lošijeg rešenja veća od nekog nasumičnog broja, generisanog iz uniformne (0,1) raspodele. Temperatura vremenom opada, tj. verovatnoća prihvatanja lošijih rešenja se vremenom smanjuje i algoritam se bliži konvergenciji.

2.2 Osnovni koncept

Za implementaciju simuliranog kaljenja nad algoritmom K-sredina, osnovna ideja ovog rada je da se simulirano kaljenje primeni nad centroidima nakon njihovog standardnog ažuriranja prema izrazu (3). Na ovaj način se zapravo prihvataju centroidi koji su lošiji u smislu minimizacije optimizacionog cilja (2), što je i osnovni koncept simuliranog kaljenja u smislu pretrage prostora rešenja za globalnim optimumom. Na ovaj način kaljeni centroidi „lutaju“ u potrazi za globalnim minimumom optimizacionog izraza. Izbor novog centroida, odnosno dodatna modifikacija centroida, zavisi od broja $p \epsilon (0,1)$ koji predstavlja verovatnoću prihvatanja kaljenog centroida u odnosu na standardno ažurirani centroid preko središnje tačke u klasteru. Ovaj broj se računa preko predefinisane opadajuće funkcije f koja zavisi od tekuće iteracije algoritma. Pri velikom broju iteracija ova verovatnoća postaje sve manja pa se i centroidi sve manje „kale“, odnosno algoritam postaje sve bliskiji standardnom algoritmu K-sredina, što dovodi do skore konvergencije.

Dakle, modifikacija algoritma K-sredina u tekućoj iteraciji it sastoji se u sledećim koracima (pri čemu se smatra da su centroidi c_1, c_2, …, c_k već ažurirani prema izrazu (3)):

1. Izračunaj verovatnoću p kaljenja svakog centroida c_i na osnovu predefinisane funkcije f u zavisnosti od tekuće iteracije algoritma it;
2. Za svaki centroid c_i generiši broj q_i iz uniformne $(0,1)$ raspodele;

3. Ukoliko za centroid c_i važi $p > q_i$, generiši $n-$ dimenzionalni vektor kaljenja a_i i ažuriraj tekući centroid prema formuli:

$$c_i' = c_i + wa_i \tag{4}$$

gde je w *težina kaljenja* koja može biti unapred fiksirana, ili biti izražena preko predefinisane funkcije g koja zavisi od tekuće iteracije, definisane na sličan način kao i funkcija f. Na ovaj način kaljeni centroidi c_i' mogu dovesti do pogoršanja sume srednjekvadratne greške u tekućoj iteraciji, ali se postiže fleksibilnost u izbegavanju lokalnih minimuma i povećava verovatnoća pronalaska globalnog minimuma.

Postavlja se pitanje: u slučaju primene simuliranog kaljenja, kako ažurirati tekući centroid, odnosno kako izračunati vektor kaljenja a_i? Ovaj postupak je metaheuristika koja predstavlja samu suštinu modifikovanog algoritma i biće detaljnije diskutovan u daljem tekstu.

2.3 Vektor kaljenja

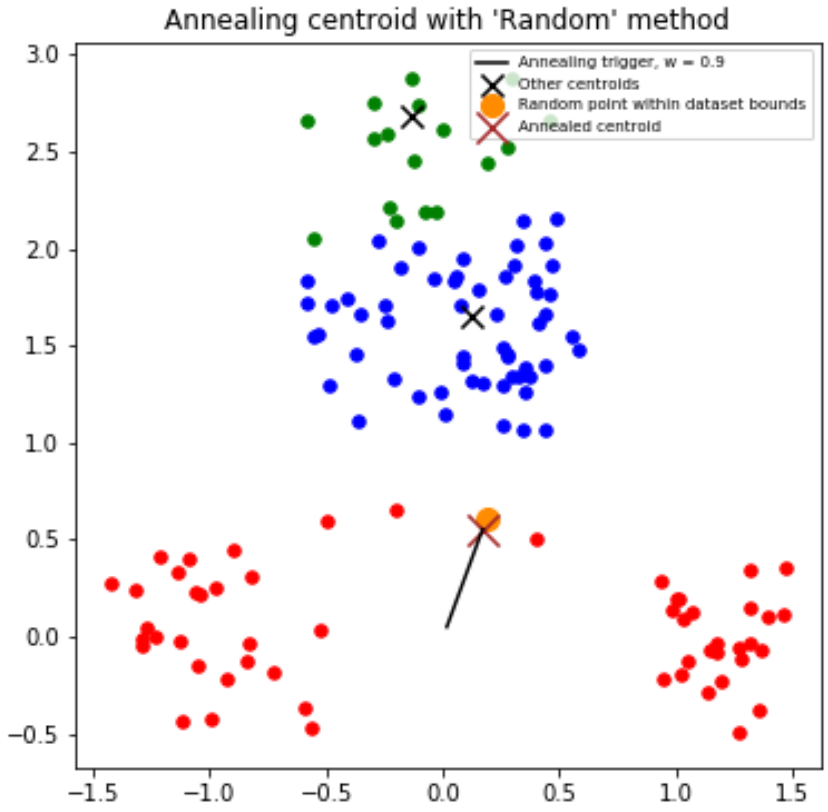
Vektor a_i iz izraza (4) nazvan je ovde vektor kaljenja jer se računa samo u slučaju kada se simulirano kaljenje primenjuje na tekući centroid i jer upravo on utiče na promenu položaja centroida c_i u cilju potrage za optimalnim klasterovanjem. Ovaj vektor se izračunava preko direkcione tačke (vektora) u koja se generiše u $n-$ dimenzionalnom prostoru, koja zapravo određuje pravac u kome će se centroid kretati. Tako za centroid c_i vektor kaljenja se jednostavno računa kao razlika direkcione tačke i tekuće vrednosti centroida:

$$a_i = u_i - c_i \tag{5}$$

U $n-$ dimenzionalnom prostoru pretrage postoji beskonačno mnogo ovakvih vektora (tačaka u_i) i u slučaju njegovog proizvoljnog odabira algoritam bi više ličio na slučajnu pretragu. Stoga je vrlo važno da ovaj vektor bude pogodno izabran i da u obzir uzme relevantne informacije algoritma, kao što su npr. tekuće vrednosti centroida, vrednosti tačaka iz skupa podataka, i sl. Ovde će biti diskutovane neke ideje za generisanje vektora kaljenja koje su implementirane u radu. Metode će biti prikazane u sledećem slučaju. Posmatrajmo primer 1 i pretpostavimo da se nalazimo u prvoj iteraciji. Posmatrajmo donji centroid na pozicji $(0.017, 0.049)$ koji je već ažuriran prema izrazu (3). Iz primera znamo da će se algoritam zaglaviti već u naredne dve iteracije (za zadati prag tolerancije $tol = 0.1$). Primenićemo različite metode kaljenja ovog centroida radi prikaza same ideje, dok će kompletan rad algoritma sa simuliranim kaljenjem nad istim skupom podataka biti detaljnije prikazan u sekciji 3.

2.3.1 Nasumično kaljenje

Najjednostavnije rešenje za generisanje vektora kaljenja je nasumično kaljenje (eng. Random annealing) – generiše se nasumična direkciona tačka iz prostora pretrage u okviru granica datog skupa podataka, izračuna vektor kaljenja prema (5) a zatim centroid pomeri u pravcu direkcione tačke, u zavisnosti od težine kaljenja w, kao što je ilustrovano na sledećoj slici:



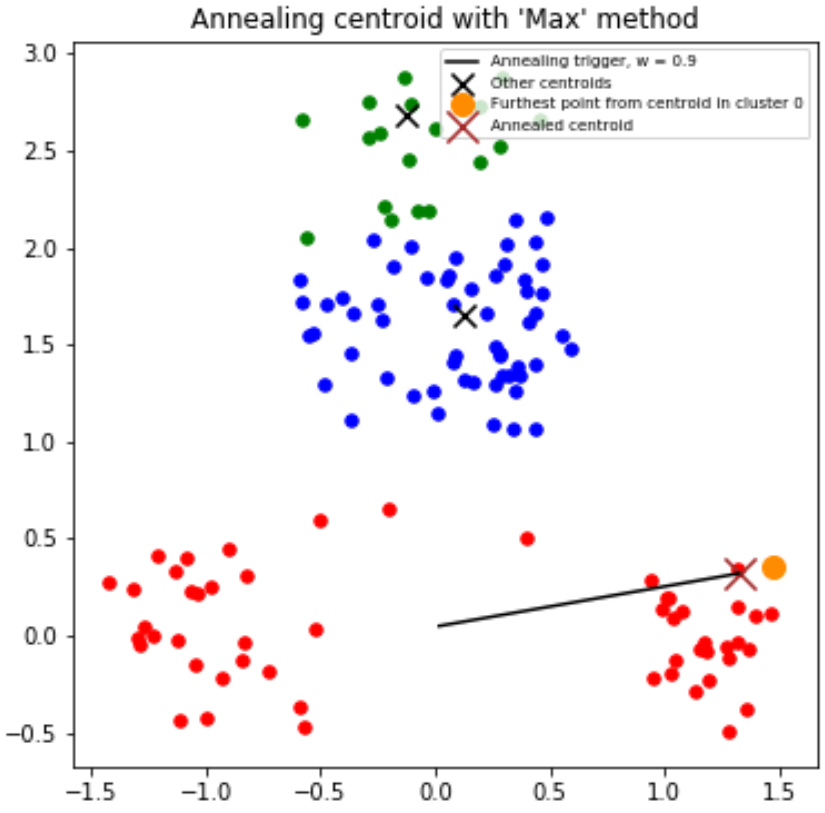
Slika 2: Nasumično kaljenje centroida

Narandžasta tačka na slici je nasumično generisana tačka prema kojoj se centroid „kreće" sa unapred fiksiranom težinom $w = 0.9$. Da li nasumično kaljenje stvarno pomaže videćemo u sekciji 3.

Napomena. Nakon ažuriranja centroida pomoću simuliranog kaljenja oznake klastera bi trebalo da se promene, što nije prikazano na slici. U prikazu ostalih metoda kaljenja ovo će takođe biti izostavljeno. Promena oznaka klastera (ponovna dodela tačaka klasterima) nakon simuliranog kaljenja će biti detaljnije obrađena u sekciji 3.

2.3.2 Maksimalno kaljenje

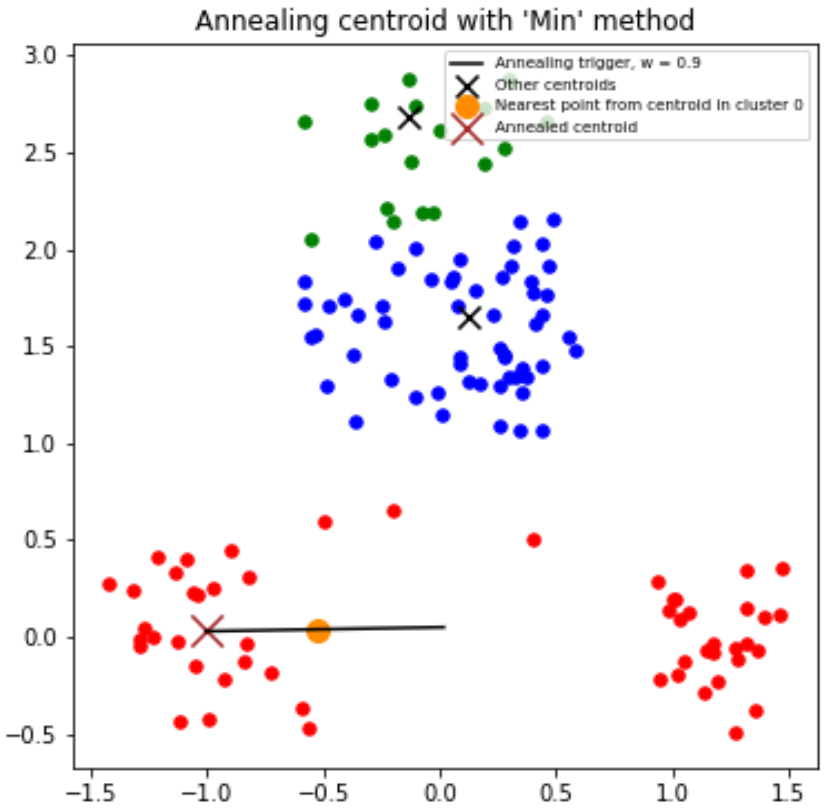
U maksimalnom kaljenju (eng. Max annealing) direkciona tačka je najudaljenija tačka iz klastera od centroida koji predstavlja taj klaster.



Slika 3: Maksimalno kaljenje centroida

2.3.3 Minimalno kaljenje

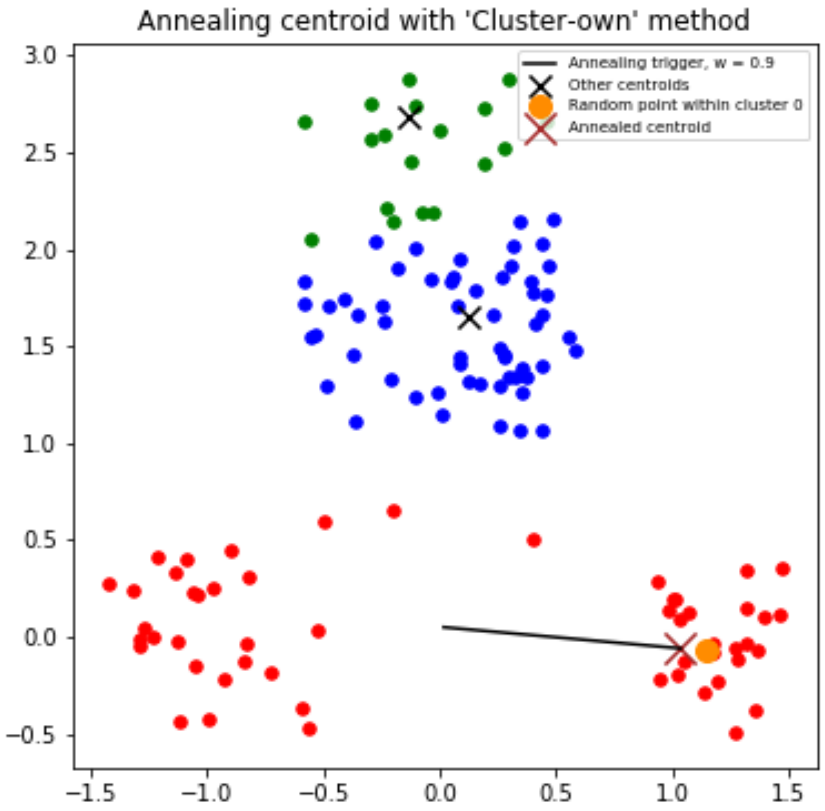
U minimalnom kaljenju (eng. Min annealing) direkciona tačka je najbliža tačka iz klastera u odnosu na centroid koji predstavlja taj klaster. S obzirom na to da se radi o najbližoj tački, u ovom slučaju minimalno kaljenje se razlikuje od ostalih metoda po tome što se centroid ne kreće prema direkcionoj tački za vektor wa_i, već za vektor $(w+1)a_i$ čime centroid "preskače" direkcionu tačku, kao što je prikazano na slici:



Slika 4: Minimalno kaljenje centroida

2.3.4 Kaljenje sopstvenog klastera

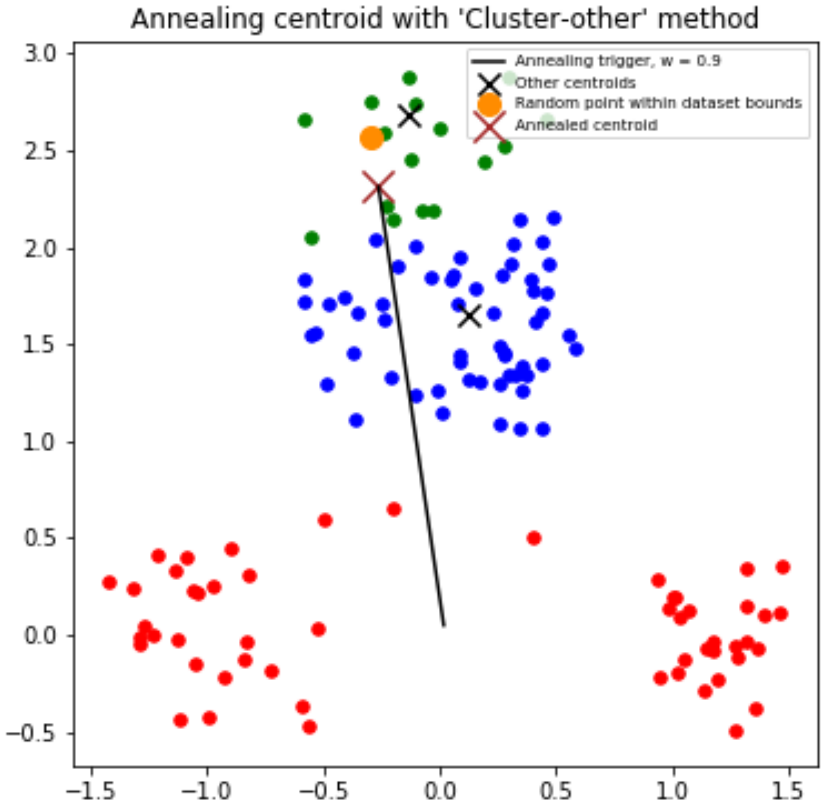
Kod kaljenja sopstvenog klastera (eng. Cluster-own annealing) direkciona tačka je nasumična tačka iz klastera čiji se centroid kali. Dakle, radi se o tački iz stvarnog skupa podataka koja pripada o tom klasteru, a ne o bilo kojoj tački iz tog opsega vrednosti za odgovarajuće atribute.



Slika 5: Kaljenje sopstvenog klastera

2.3.5 Kaljenje drugog klastera

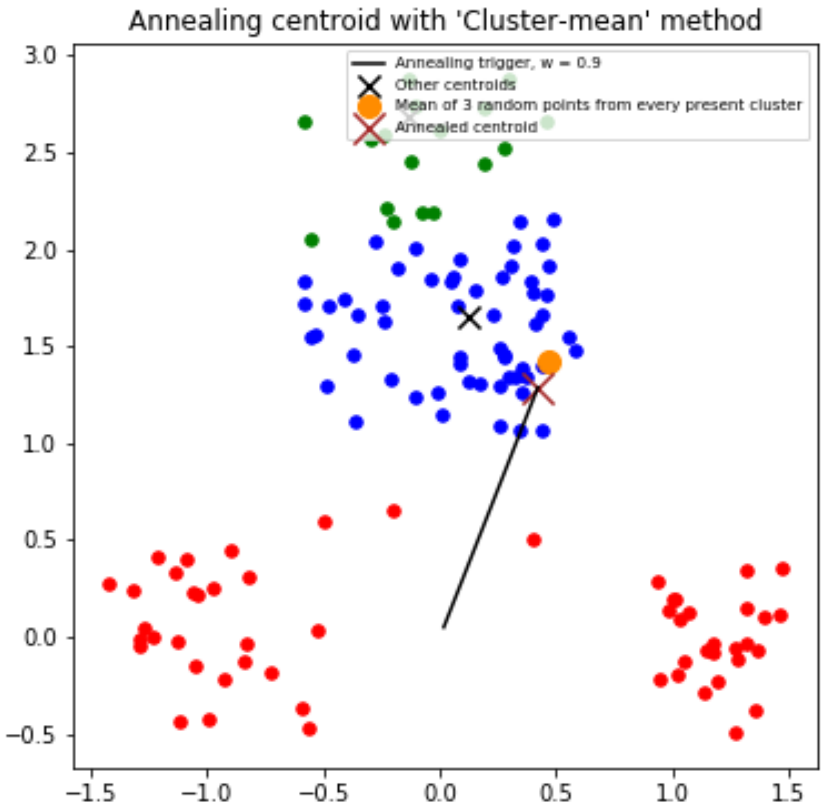
Nasuprot metodu kaljenja sopstvenog klastera, u metodu kaljenja drugog klastera (eng. Cluster-other annealing) direkciona tačka je nasumično odabrana tačka iz nekog drugog klastera, tj. iz klastera koji ne predstavlja centroid koji se kali. U sledećem primeru tačke kandidati za direkcionu tačku su plave i zelene tačke, ali ne i crvene.



Slika 6: Kaljenje drugog klastera

2.3.6 Kaljenje sredine klastera

Metod kaljenja sredine klastera (eng. Cluster-mean annealing) podrazumeva odabir nasumične tačke iz svakog klastera. Direkciona tačka se računa kao sredina ovih nasumično odabranih tačaka.

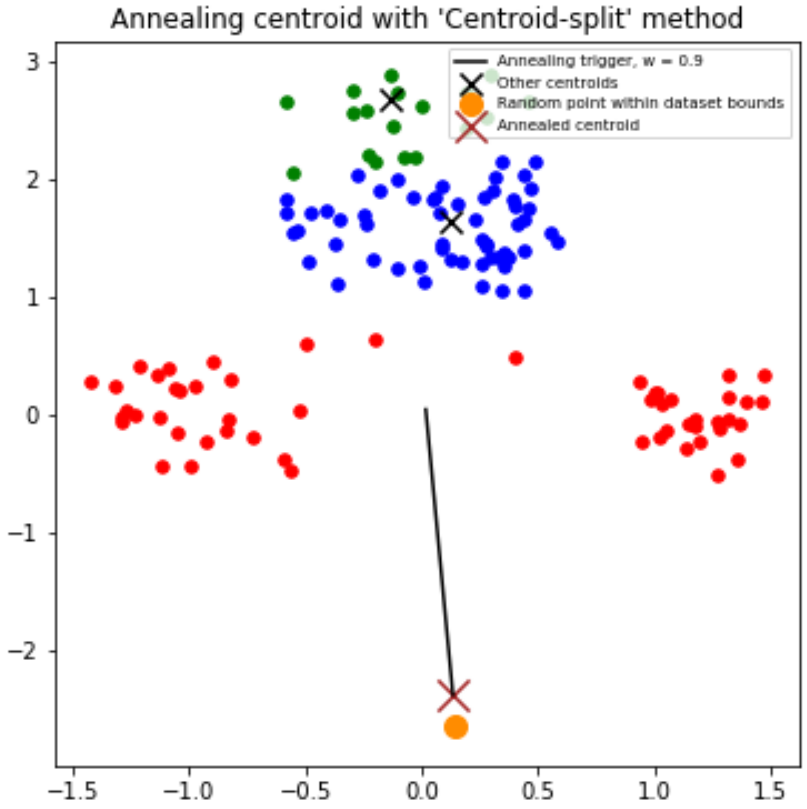


Slika 7: Kaljenje sredine klastera

Ovaj metod ima tendeciju da približuje centroide sredini skupa podataka.

2.3.7 Kaljenje razdvajanja centroida

U pristupu kaljenja koje razdvaja centroide (eng. Centroid-split annealing) računaju se distance od centroida koji se kali do svih ostalih centroida. Bira se centroid c_j koji je najbliži tekućem centroidu koji se kali, a direkciona tačka je na suprotnoj strani od centroida c_j :

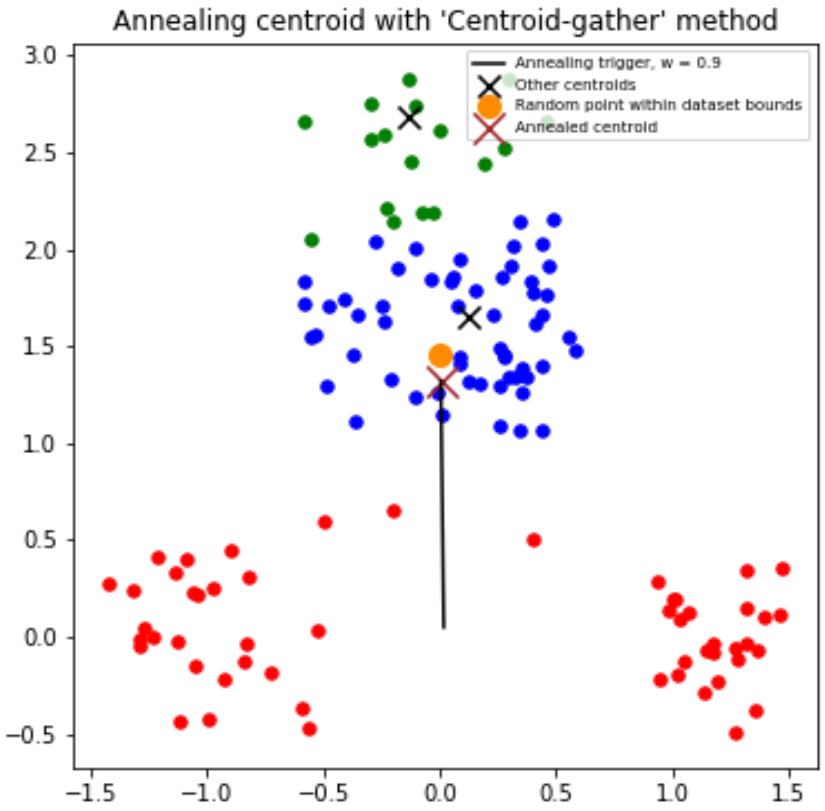


Slika 8: Kaljenje razdvajanja centroida

Kod ovog metoda postoji rizik da se neki centroidi znatno udalje od skupa tačaka, pa im u narednoj iteraciji neće biti dodeljena nijedna tačka iz skupa podataka, što dovodi do stvaranja praznog klastera. O ovom problemu biće više reči kasnije.

2.3.8 Kaljenje sakupljanja centroida

Kod kaljenja sakupljanja centroida (eng. Centroid-gather annealing) direkciona tačka je središnja tačka tekućih centroida.



Slika 9: Kaljenje sakupljanja centroida

Ovaj metod ima tendenciju, kao što je i očekivano, da centroide približuje jedan drugom. Za vrednost $w = 1$ ovi centroidi se stapaju u jednu tačku, što nije poželjna situacija.

2.3.9 Ostali metodi

Pored navedenih, u ovom radu implementirana su još dva metoda: maksmin kaljenje (eng. Maxmin annealing) koje kod neparnih iteracija primenjuje maksimalno kaljenje, a kod parnih minimalno kaljenje, i k*aljenje vrteške* (eng. Carousel annealing) koje u svakoj iteraciji nasumično bira jedan od navedenih metoda a zatim ga primenjuje nad centroidima.

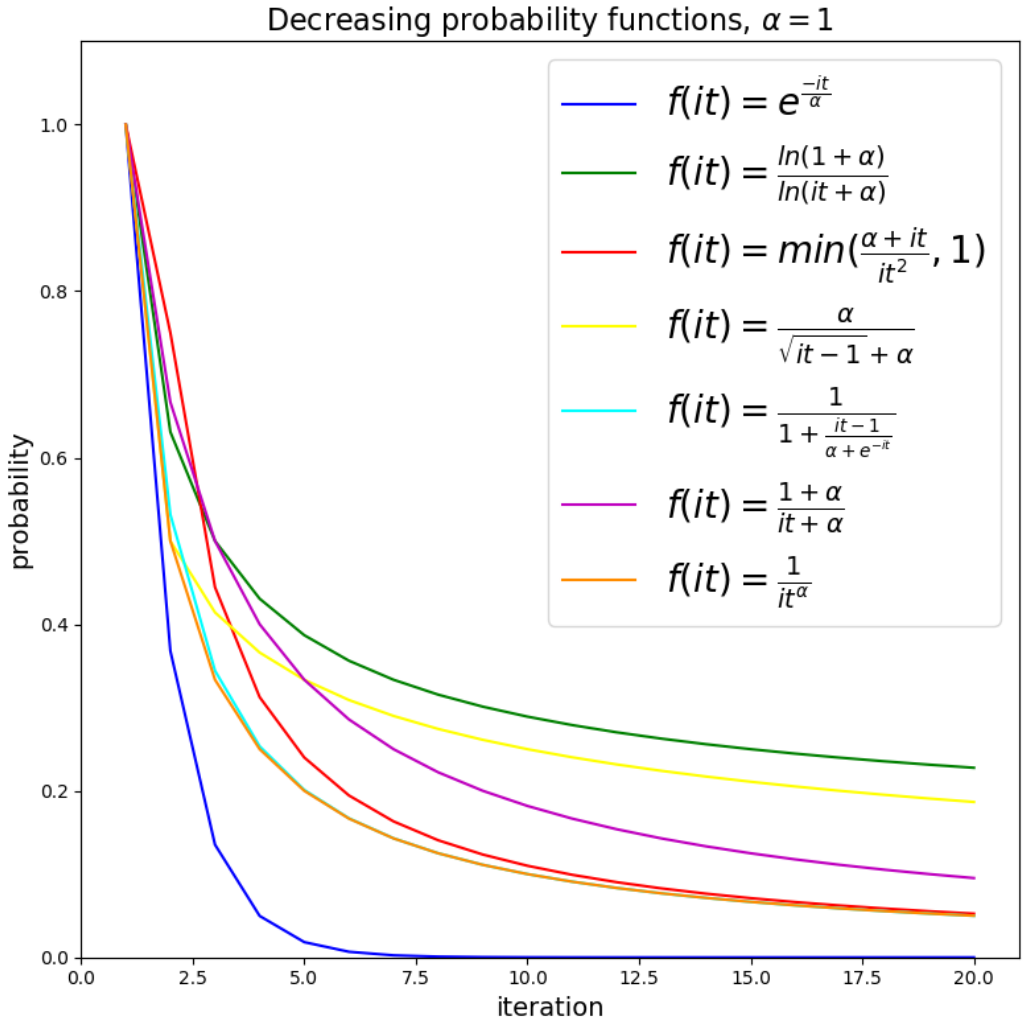
Pored navedenih, sigurno je moguće izvesti i neke kompleksnije metode kaljenja centroida koje bi u obzir uzimale i prethodne iteracije, promenu centroida u distancama, interklasterno rastojanje [19], promene u varijansi klastera itd. Ipak, ograničićemo se na navedene metode koje su dobre za testiranje same ideje simuliranog kaljenja u njenoj primeni nad algoritmom K-sredina.

Ukoliko jednostavnije metode pokažu pozitivne indikacije, onda se isplati uložiti dodatni napor u implementaciju kompleksnijih metoda i u unapređenje samog algoritma.

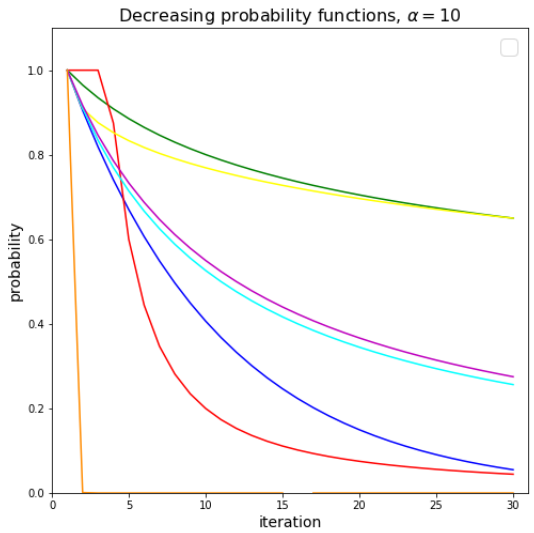
U narednoj sekciji, sve ove metode biće najpre primenjene nad primerom 1, a zatim ćemo testirati algoritam nad kompleksnijim skupovima podataka. Ali pre toga ostaje da prodiskutujemo još dva važna tehnička detalja.

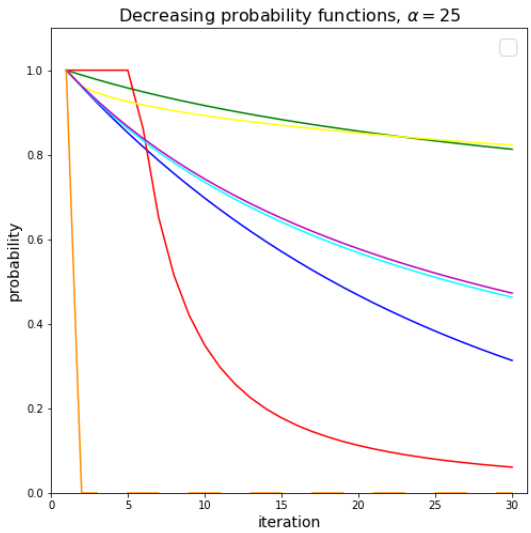
2.4 Verovatnoća kaljenja

Kao što je ranije spomenuto, verovatnoća kaljenja izražena je preko opadajuće funkcije $f\colon \mathbb{N} \to [0,1]$ koja u zavisnosti od tekuće iteracije algoritma izračunava verovatnoću kaljenja centroida. Na narednoj slici prikazane su implementirane funkcije sa dodatnim hiperparametrom α koji funkciji daje određenu fleksibilnost i reguliše njen nagib:



Za veće vrednosti hiperparametra α funkcije sporije opadaju:

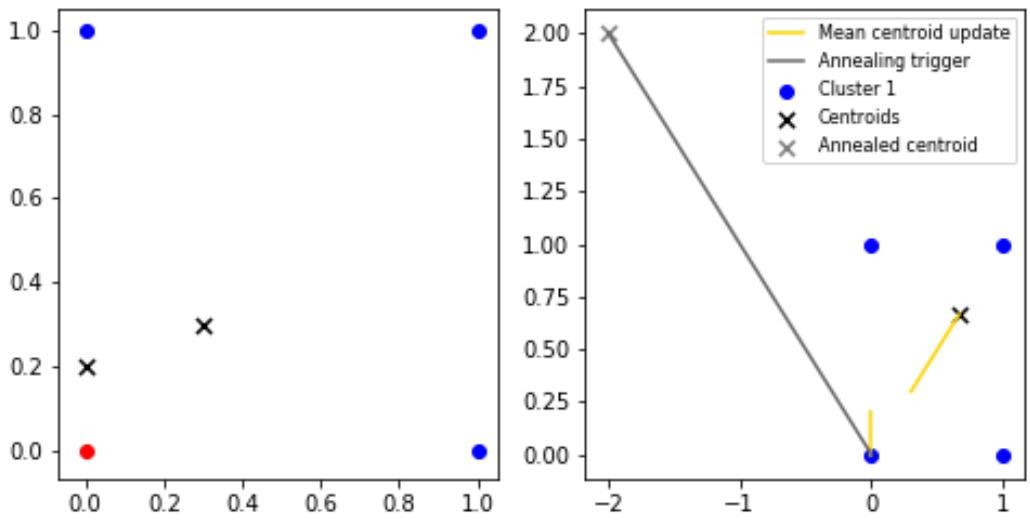




Od izbora ove funkcije i odgovarajućeg hiperparametra α zavisiće učestalost kaljenja centroida a samim tim i rad čitavog algoritma.

2.5 Problem praznog klastera

Podsetimo se da se korak dodele tačaka klasterima u algoritmu K-sredina zasniva na funkciji rastojanja – tačka se dodeljuje onom klasteru čijem je centroidu najbliža. Centroidi se potom ažuriraju na novu vrednost tako što se računaju kao središnja tačka klastera koga predstavljaju, a zatim se postupak ponavlja. Međutim, šta ako se dogodi da je neki centroid suviše daleko te mu nijedna tačka ne bude dodeljena, odnosno sve tačke budu dodeljene nekim drugim, bližim centroidima? Ovaj slučaj dovodi do praznog klastera pa se centroid bez dodeljenih tačaka ne može ažurirati, što potpuno obustavlja normalno funkcionisanje algoritma. Do ovog problema najčešće dolazi prilikom početne inicijalizacije centroida. Ukoliko su početni centroidi dobro inicijalizovani, kod standardnog algoritma K-sredina neće doći do ovakvog slučaja. Međutim, u modifikovanog algoritmu neki centroidi zbog kaljenja mogu da zauzmu nepogodnu poziciju u prostoru pretrage pri čemu im u sledećoj iteraciji neće biti dodeljena nijedna tačka. To znači da u modifikovanom algoritmu verovatnoća da dođe do problema praznog klastera je znatno veća. Ilustrujmo ovaj slučaj na jednostavnom primeru.



Slika 10: Problem praznog klastera

U inicijalnom stanju prvi centroid (centroid na poziciji $(0, 0.2)$) predstavlja crveni klaster, a drugi plavi klaster. Nakon koraka ažuriranja prema središnjoj tački klastera (na drugom grafiku prikazano žutom linijom) oznake klastera se ne bi promenile. Međutim, u slučaju da se prvi centroid kali, na primer, nasumičnim kaljenjem i završi u tački $(-2, 2)$, u datom koraku neće mu biti dodeljena nijedna tačka jer je svakoj tački bliži drugi centroid, pa su sve pridružene plavom klasteru. Na ovaj način se gubi crveni klaster i došlo je do problema praznog klastera.

Zbog navedenog neophodno je da se ovaj problem adresira i u projektu implementiran je pod nazivom *razrešavanje praznih klastera* (eng. Empty Clusters Resolution, ECR). Konkretno, implementirano je najjednostavnije rešenje: ažuriraj "prazni" centroid na neku nasumičnu tačku iz prostora pretrage sve dok mu ne bude dodeljena bar jedna tačka iz skupa. Naravno, postoje i mnogo povoljnije metode, kao što su zamena centroida tačkom koja najviše učestvuje u SSE, nasumičnom tačkom iz klastera sa najvećim SSE, najudaljenijom tačkom, itd.

Napomena. Ovi metodi su u radu implementirani ali se algoritam iz nekog razloga zaglavljuje u beskonačnoj petlji. Zbog toga je primenjen najjednostavniji pristup.

Prikaz razrešavanja praznih klastera biće prikazan usput, u celokupnom radu algoritma.

3. Testiranje algoritma

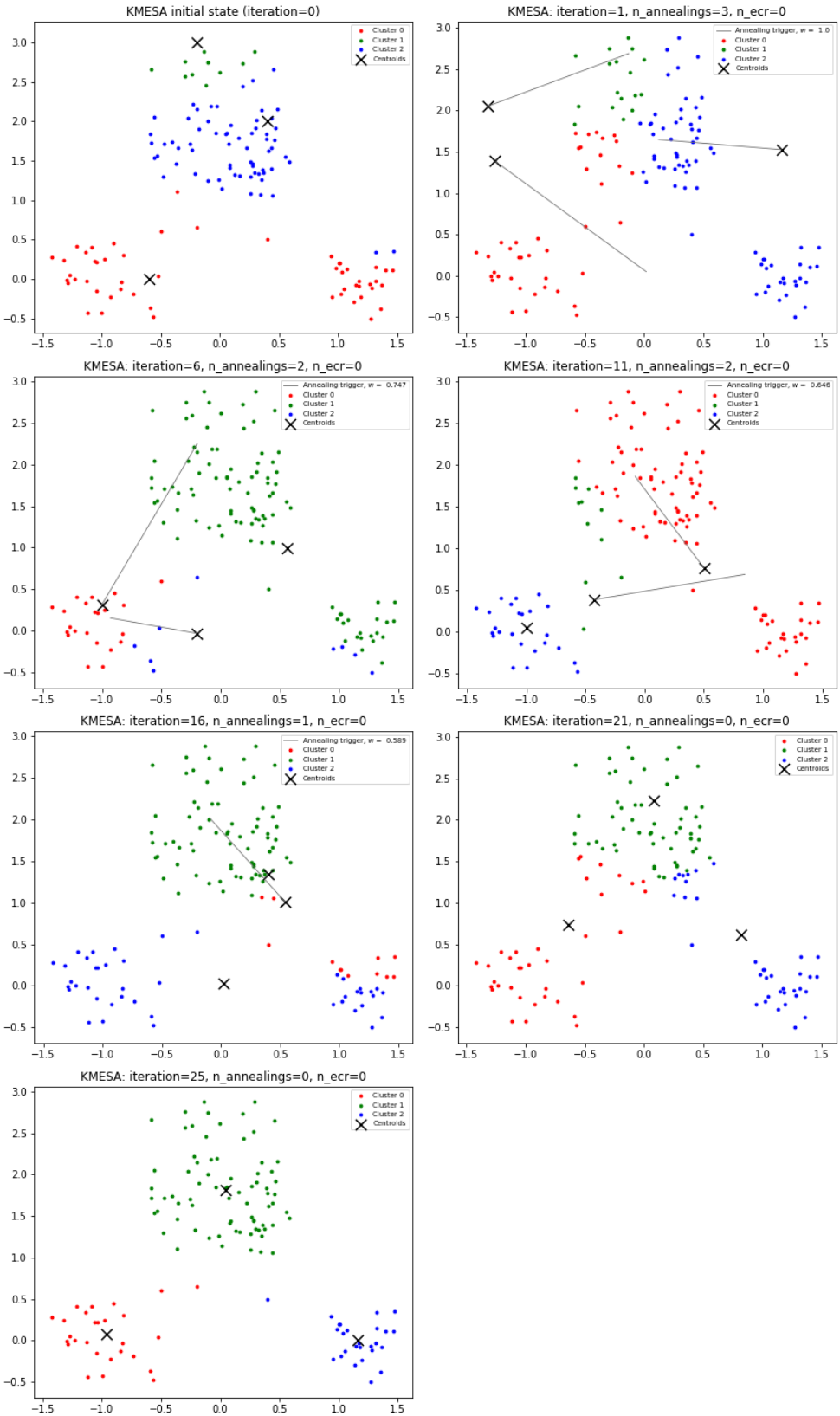
Sada pređimo na testiranje algoritma na različitim primerima. Svi detalji o parametrima, hiperparametrima, ukupnom broju izvršenih kaljenja centroida, razrešavanja praznih klastera,

proteklom vremenu izvršavanja algoritma i sl. mogu se pogledati u odgovarajućim Jupyter datotekama na Github repozitorijumu https://github.com/alexein777/K-MESA.

3.1 Jednostavan skup

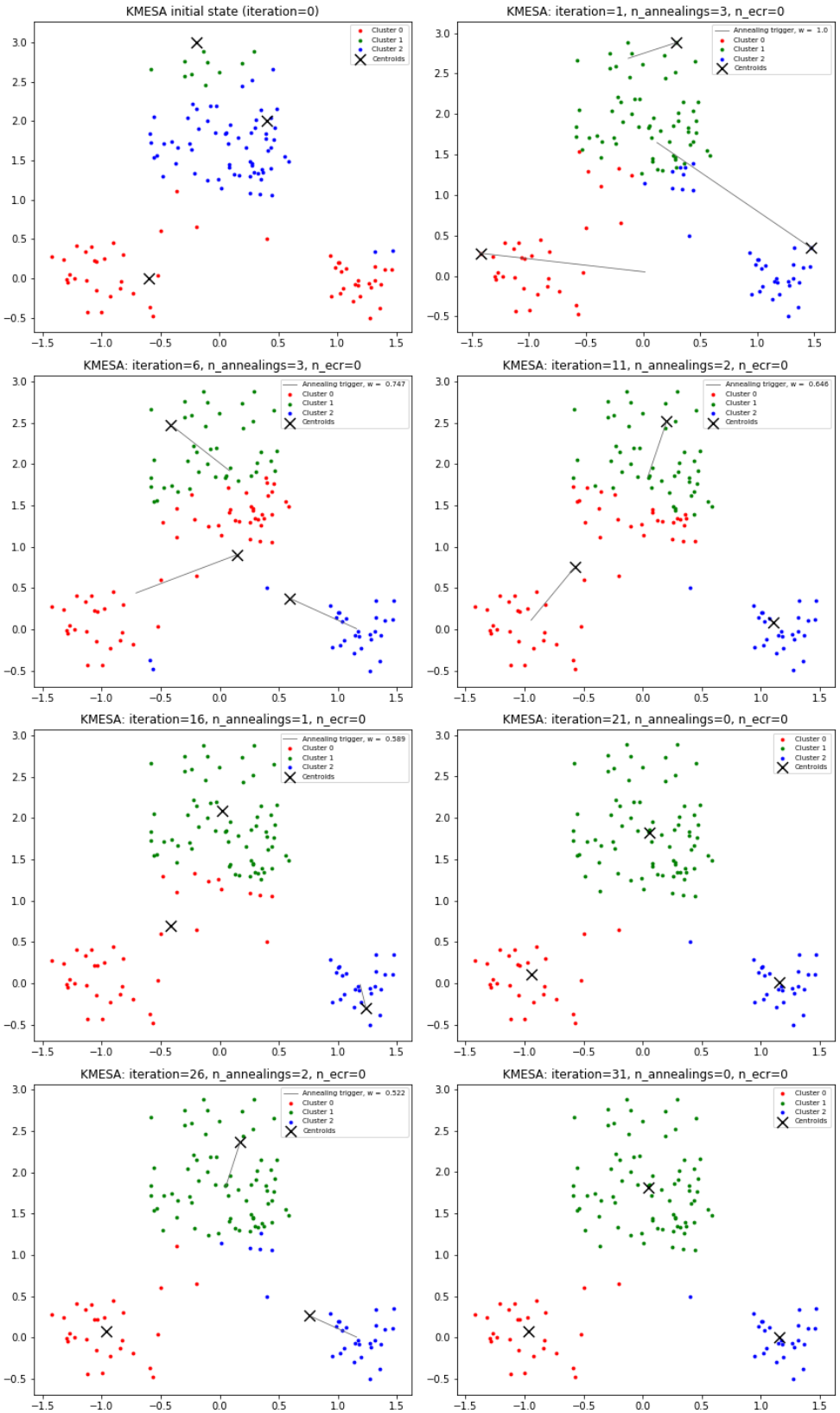
Primenićemo modifikovani algoritam najpre na jednostavnom skupu podataka iz primera 1. Primenićemo sve metode kaljenja centroida kako bismo posmatrali njihove rezultate i pogodno odabrali najbolje za primenu nad kompleksnijim podacima. Izabraćemo početne centroide baš kao u primeru 1 da bismo najpre na najjednostavnijem primeru utvrdili da li simulirano kaljenje uopšte pomaže. Radi ilustracije, ovde će biti prikazani samo metodi nasumičnog kaljenja, maksimalnog kaljenja i kaljenja sa sakupljanjem centroida. Rad algoritma sa ostalim metodima može se pogledati na zvaničnom repozitorijumu projekta na putanji img/testing/simple_3_clusters.

3.1.1 Klasterovanje sa nasumičnim kaljenjem



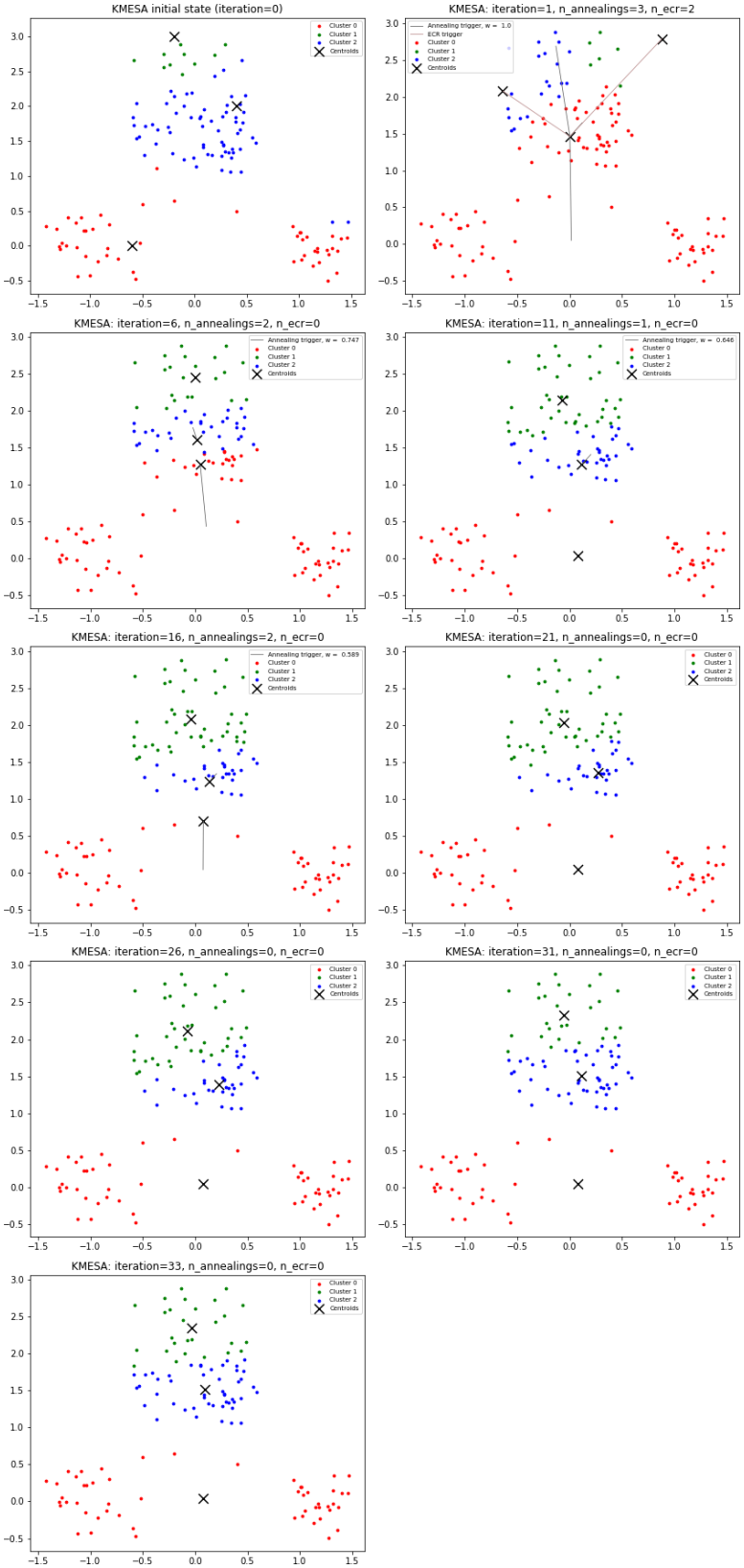
Slika 11: Klasterovanje jednostavnog skupa sa nasumičnim kaljenjem

3.1.2 Klasterovanje sa maksimalnim kaljenjem



Slika 12: Klasterovanje jednostavnog skupa sa maksimalnim kaljenjem

3.1.3 Klasterovanje sa kaljenjem sakupljajućih centroida



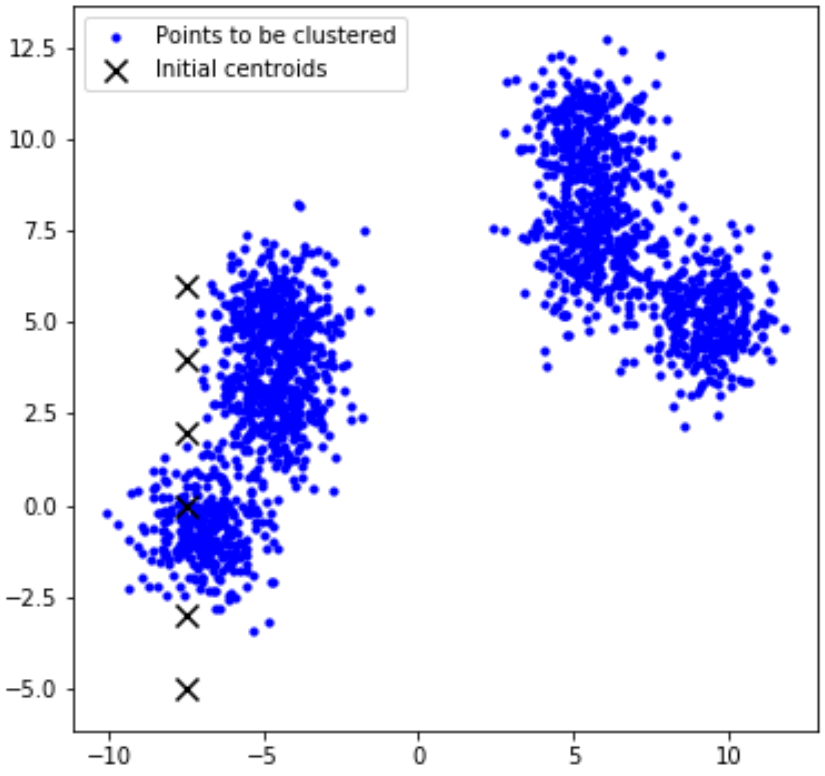
Slika 13: Klasterovanje jednostavnog skupa sa kaljenjem sakupljanja centroida

3.1.4 Zaključak

Ispostavilo se da su metodi nasumičnog i maksimalnog kaljenja doveli algoritam do globalnog minimuma, ali kaljenje sa sakupljajućim centroidima nije. Na pomenutoj putanji se mogu pogledati i ostali metodi – svi su uspeli da pronađu globalni minimum u vrlo malom broju iteracija!

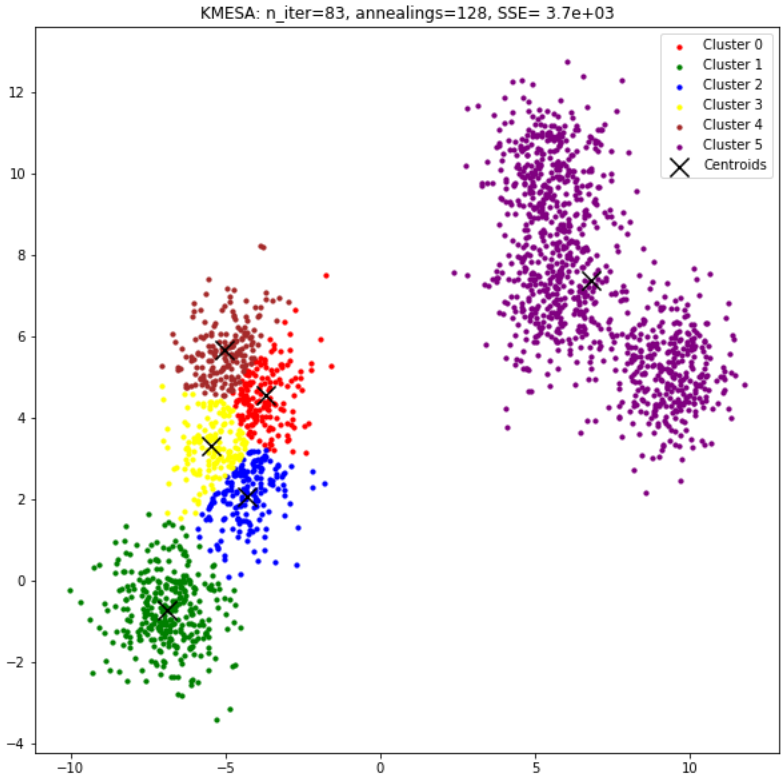
3.2 Blisko povezani klasteri

Neka je dat skup od 2000 tačaka sa fiksno postavljenim centroidima kao na slici:



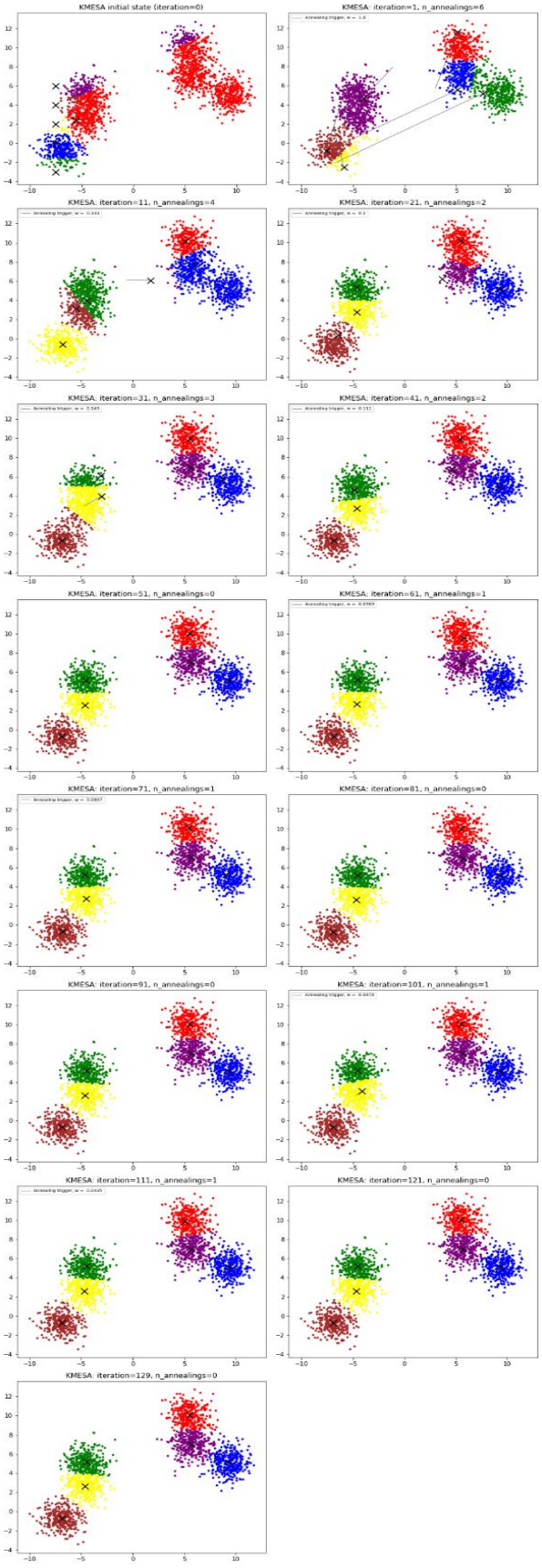
Slika 14: Skup podataka sa blisko povezanim klasterima

Ispostavlja se da algoritam K-sredina ne uspeva da pronađe optimalno klasterovanje:



Slika 15: Rešenje skupa podataka sa blisko povezanim klasterima pomoću algoritma K-sredina

Primenimo sada modifikovani algoritam sa metodom kaljenja drugog klastera:


KMESA initial state (iteration=0)
KMESA: iteration=1, n_annealings=6
KMESA: iteration=11, n_annealings=4
KMESA: iteration=21, n_annealings=2
KMESA: iteration=31, n_annealings=3
KMESA: iteration=41, n_annealings=2
KMESA: iteration=51, n_annealings=0
KMESA: iteration=61, n_annealings=1
KMESA: iteration=71, n_annealings=1
KMESA: iteration=81, n_annealings=0
KMESA: iteration=91, n_annealings=0
KMESA: iteration=101, n_annealings=1
KMESA: iteration=111, n_annealings=1
KMESA: iteration=121, n_annealings=0
KMESA: iteration=129, n_annealings=0

Slika 16: Rešenje skupa podataka sa blisko poveznaim klaterima pomoću KMESA algortima

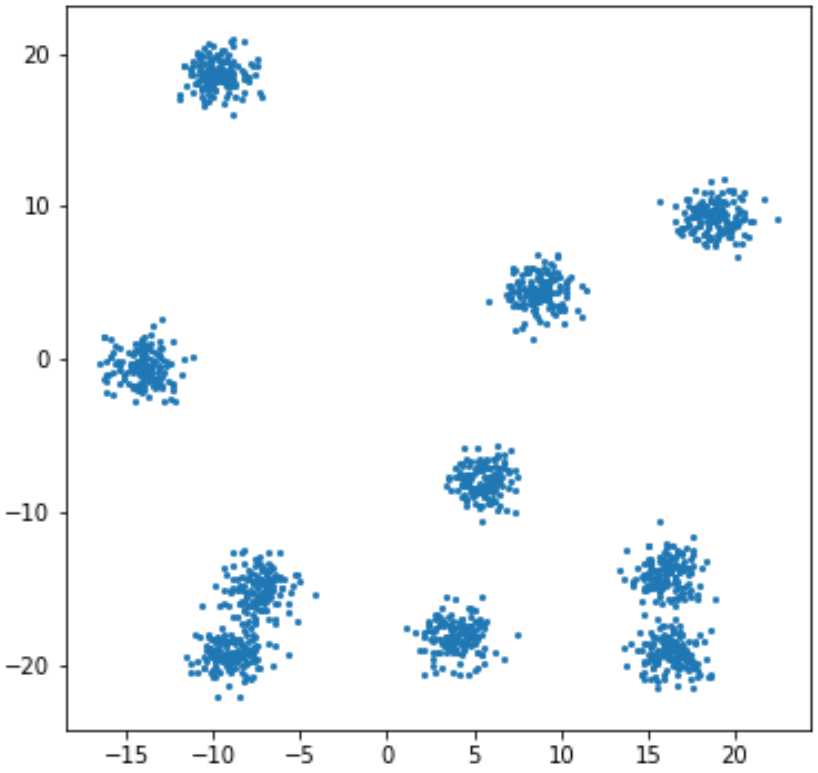
Simulirano kaljenje je uspelo da „iščupa“ algoritam iz lokalnih minimuma i na kraju ga dovode do optimalnog klasterovanja. Ipak, metoda Maxmin kaljenja to ne uspeva i završava sa istim klasterovanjem kao i običan algoritam K-sredina.

Na putanji img/testing/6_blobs nalazi se još jedan skup podataka sa 6 klastera. Na njemu se običan algoritam K-sredina takođe zaglavljuje, a modifikovani algoritam sa kaljenjem drugog klastera uspeva da pronađe globalni minimum optimizacionog izraza.

3.3 Dobro razdvojeni klasteri

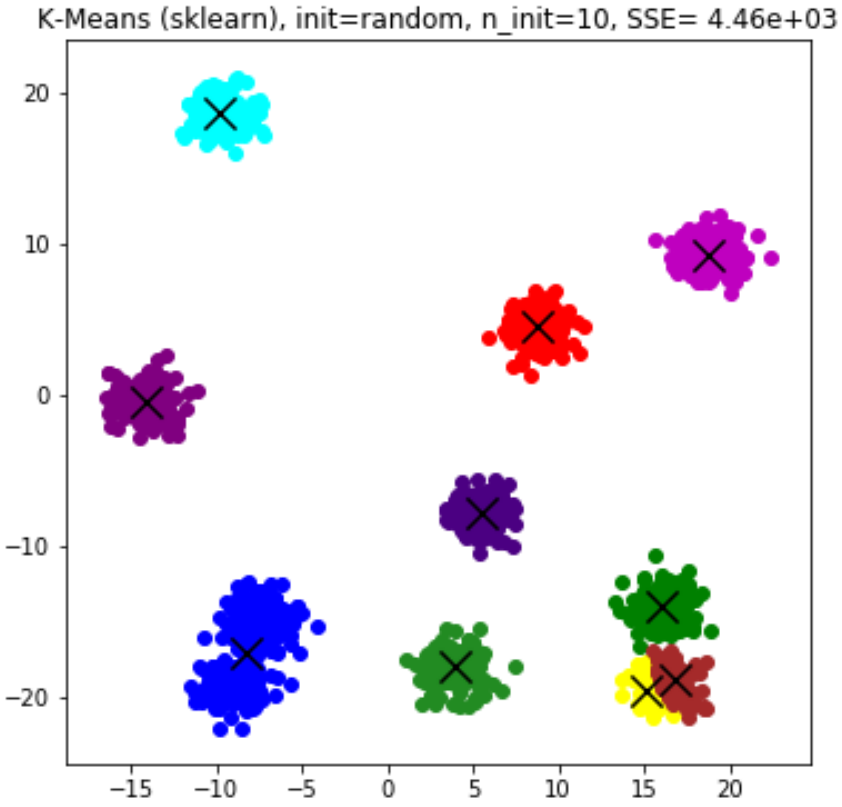
Sledeći skup podataka sastoji se od 1500 tačaka koje su nešto rasprostranjenije i koje čine dobro razdvojenih 10 klastera. Radi provere efektivnosti simuliranog kaljenja metoda za incijalizaciju centroida je nasumična. Razlog za to je sledeći: ako je ideja simuliranog kaljenja zaista efektivna, onda bismo želeli da ona pomogne algoritmu da pronađe globalni minimum bez obzira na početne centroide. Iz tog razloga nije korišćen napredniji pristup u inicijalizaciji početnih centroida (u biblioteci sklearn pod nazivom k-means++).

Pogledajmo najpre kako izgleda skup podataka:



Slika 17: Skup tačaka sa dobro razdvojenim klasterima

Da bismo proverili prednost modifikovanog algoritma u odnosu na standardni algoritam K-sredina, nad datim skupom pokrenut je standardni algoritam sa nasumičnom inicijalizacijom centroida, i to 10 puta. Od svih 10 puta najbolje pronađeno rešenje (sa najmanjim SSE) je sledeće:



Slika 18: Rešenje skupa tačaka sa dobro razdvojenim klasterima pomoću algoritma K-sredina

Očigledno je da standardni algoritam nije uspeo da pronađe optimalno klasterovanje.

Zatim je pokrenut modifikovani algoritam za sve vrste kaljenja centroida, sa po dve reinicijalizacije. Ovo su dobijeni rezultati:

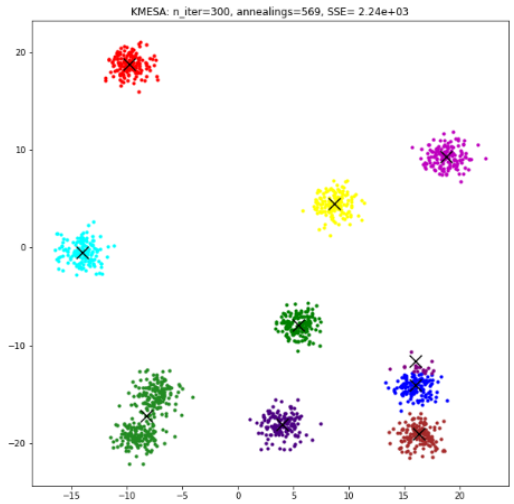


Figure 1: Random annealing

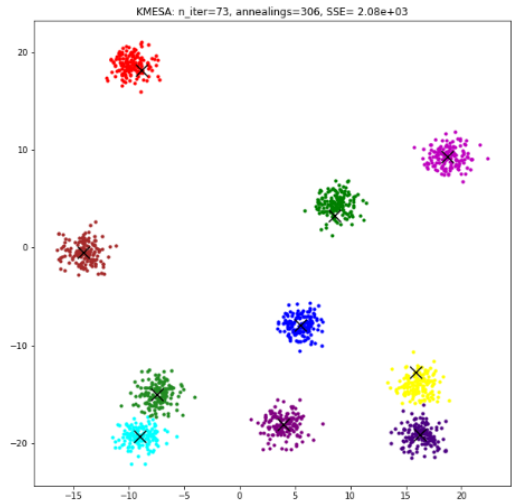


Figure 2: Max annealing

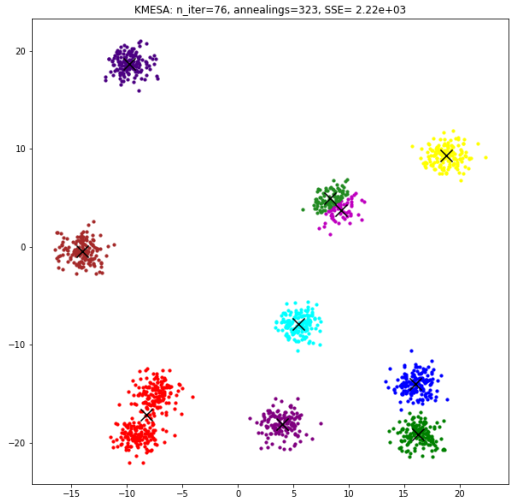


Figure 3: Min annealing

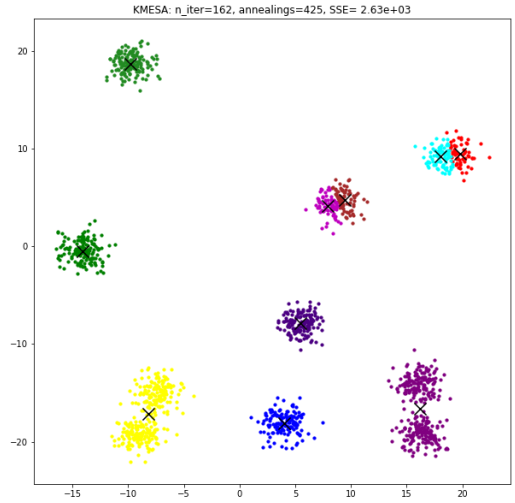


Figure 4: Cluster-own annealing

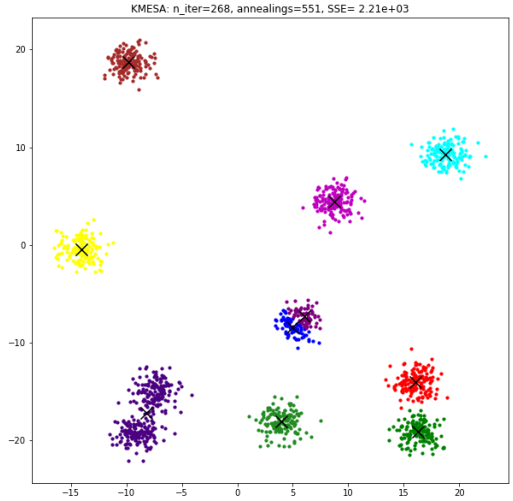


Figure 5: Cluster-other annealing

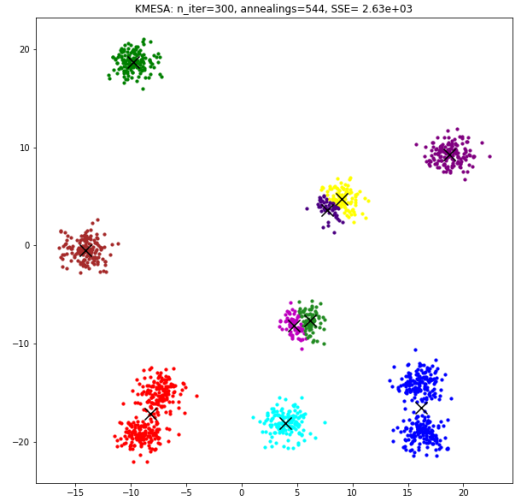


Figure 6: Cluster-mean annealing

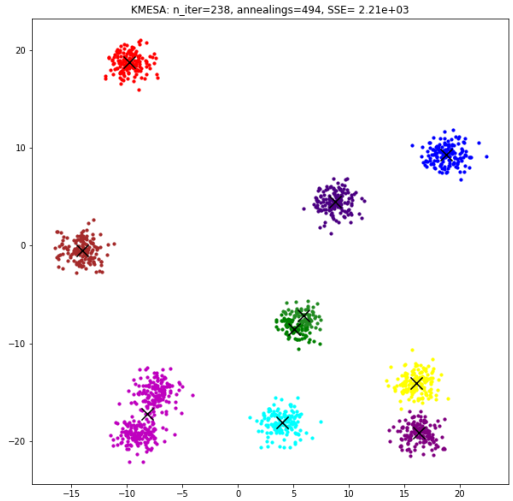


Figure 7: Centroid-split annealing

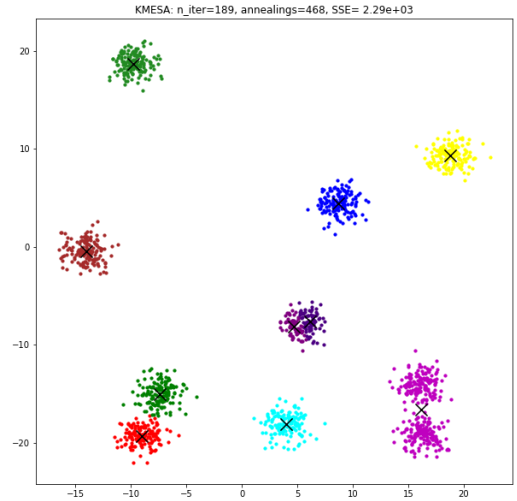


Figure 8: Centroid-gather annealing

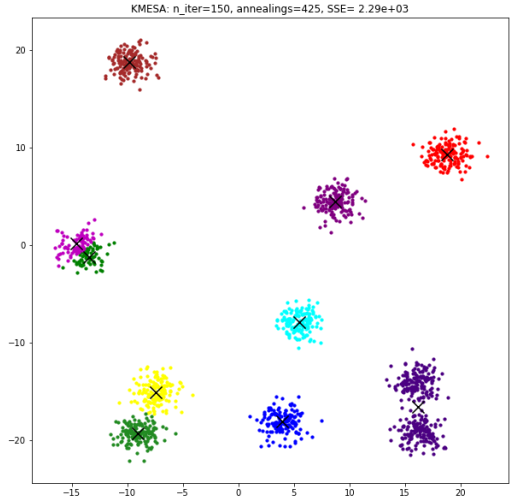


Figure 9: Maxmin annealing

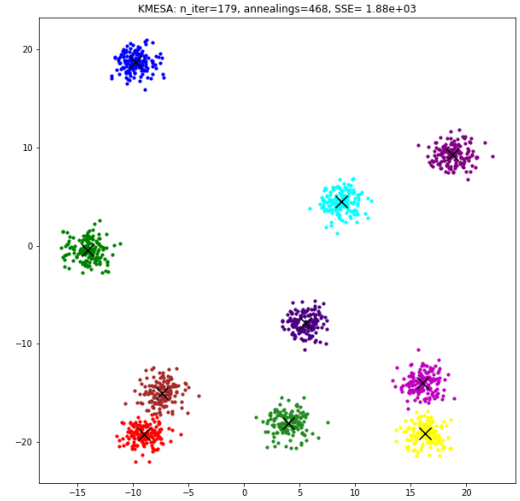


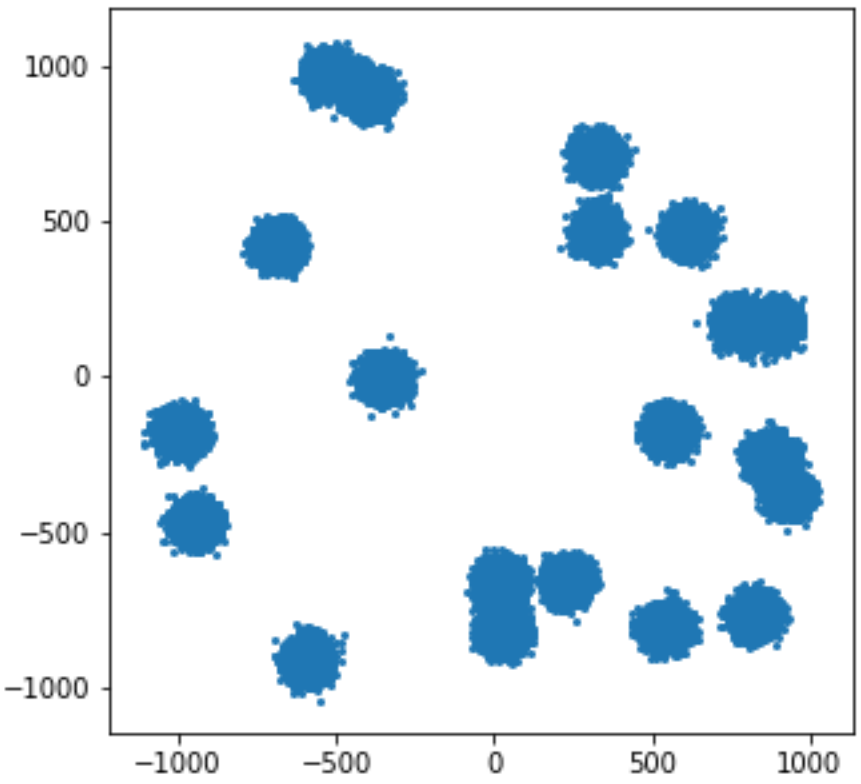
Figure 10: Carousel annealing

Slika 19: Rešenje skupa tačaka sa dobro razdvojenim klasterima pomoću KMESA algoritma sa različitim vrstama kaljenja

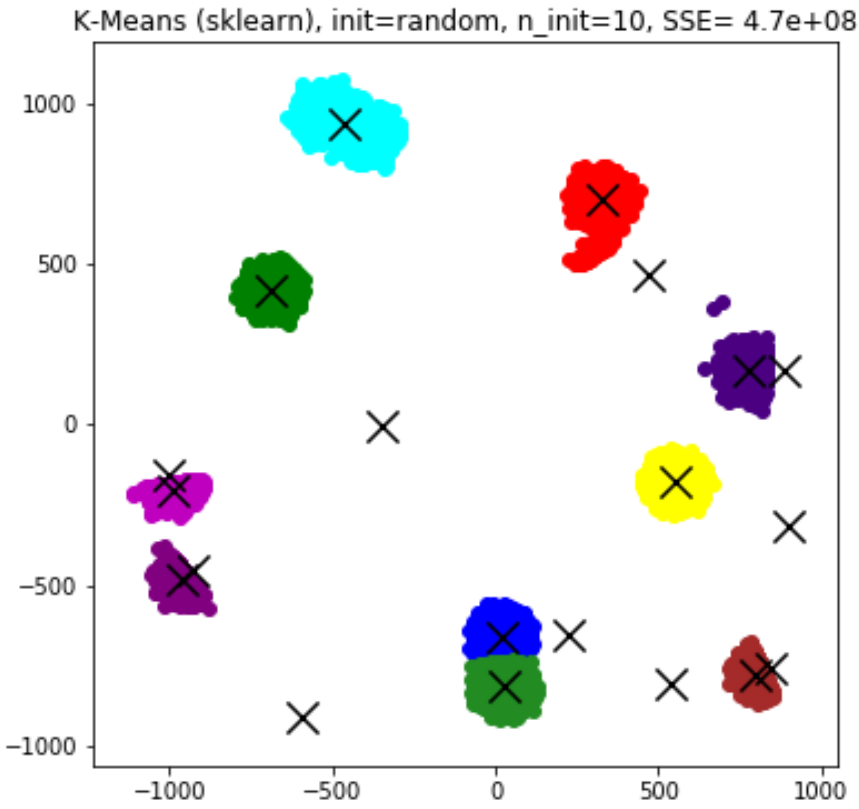
Dakle, samo maksimalno kaljenje i kaljenje vrteške uspele su da pronađu optimalno klasterovanje. Možda je bilo moguće doći do najboljeg rešenja i nekim drugim metodama pogodnim odabirom parametara kao što su verovatnoća kaljenja i težina kaljenja, ali je za to neophodno duže i pažljivije testiranje, koje će ovde biti preskočeno.

3.4 Veliki skup tačaka

Primenićemo ovde klasterovanje nad velikim skupom podataka od 100.000 tačaka razdvojenih u 20 klastera. Pogledajmo kako izgleda skup podataka a zatim primenimo na njega algoritam K-sredina iz biblioteke scikit-learn sa nasumičnom inicijalizacijom koja se ponavlja 10 puta:



Slika 20: Veliki skup tačaka



Slika 21: Rešenje velikog skupa tačaka preko algoritma K-sredina

Ne samo da standardni algoritam K-sredina nije pronašao odgovarajuće klasterovanje, već se zaglavio i zbog problema praznih klastera. Pogledajmo sada rezultate svih vrsta kaljenja centroida, za kvadratno-opadajuću funkciju verovatnoće kaljenja i korenu funkciju težine kaljenja (pogledati detalje u odgovarajućoj Jupyter datoteci KMESA_huge_dataset.ipynb):

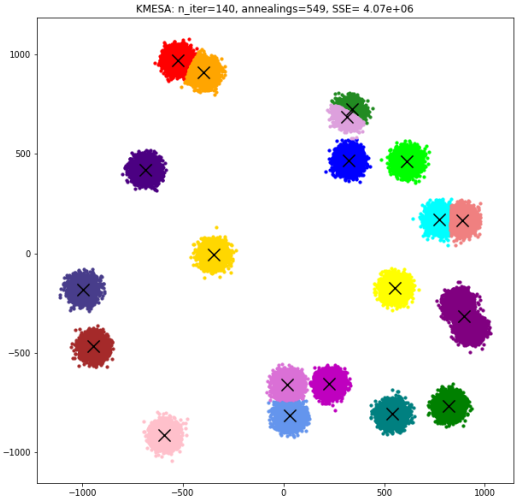


Figure 11: Random annealing

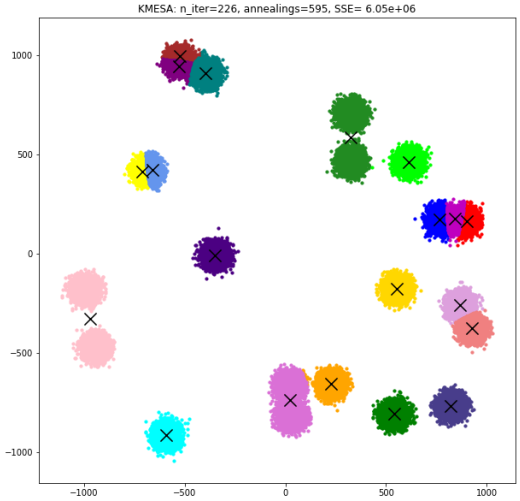


Figure 12: Max annealing

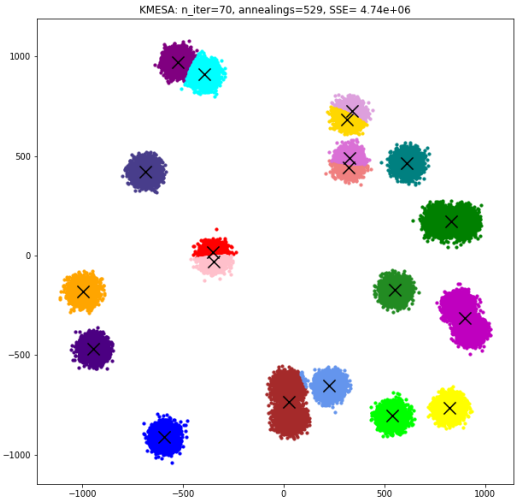


Figure 13: Min annealing

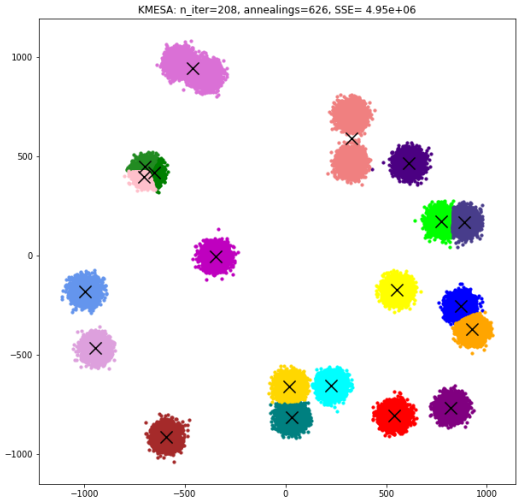


Figure 14: Cluster-own annealing

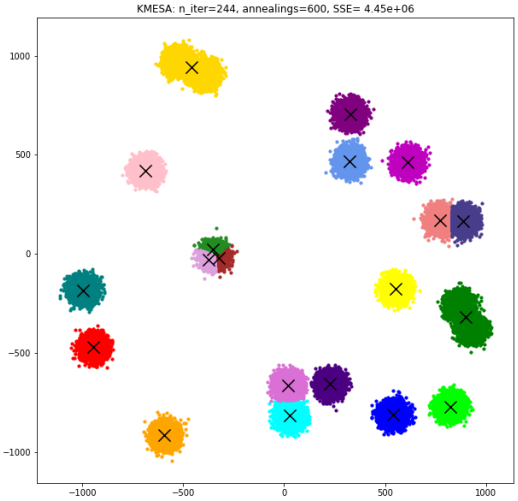


Figure 15: Cluster-other annealing

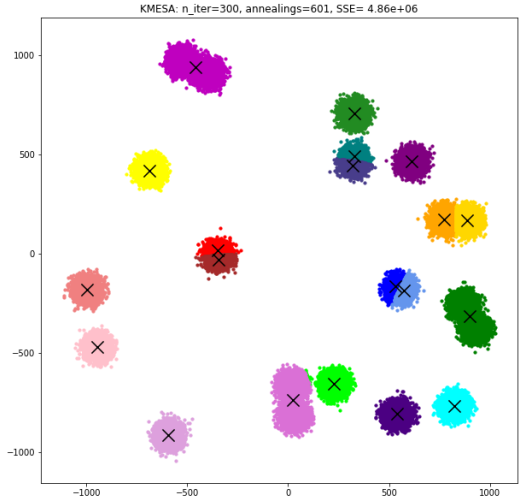


Figure 16: Cluster-mean annealing

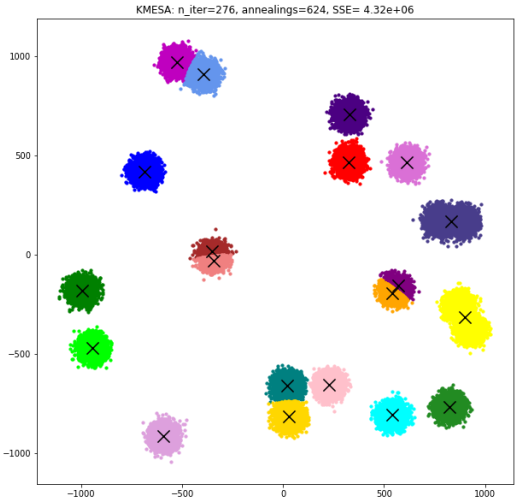


Figure 17: Centroid-split annealing

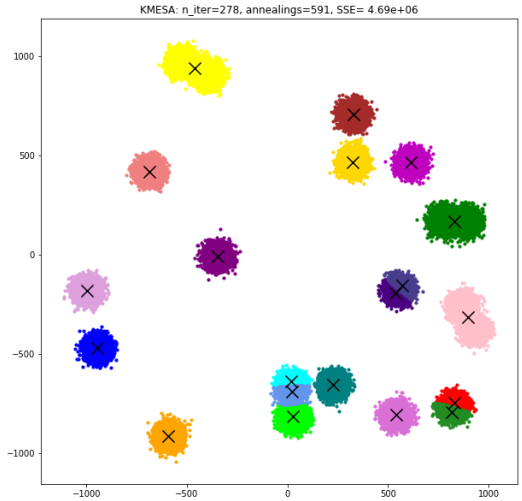


Figure 18: Centroid-gather annealing

Figure 19: Maxmin annealing

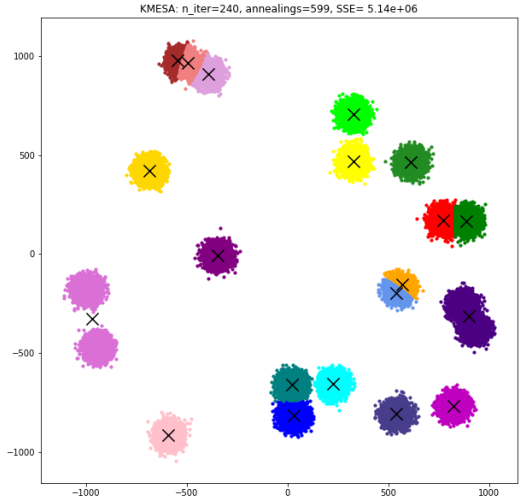
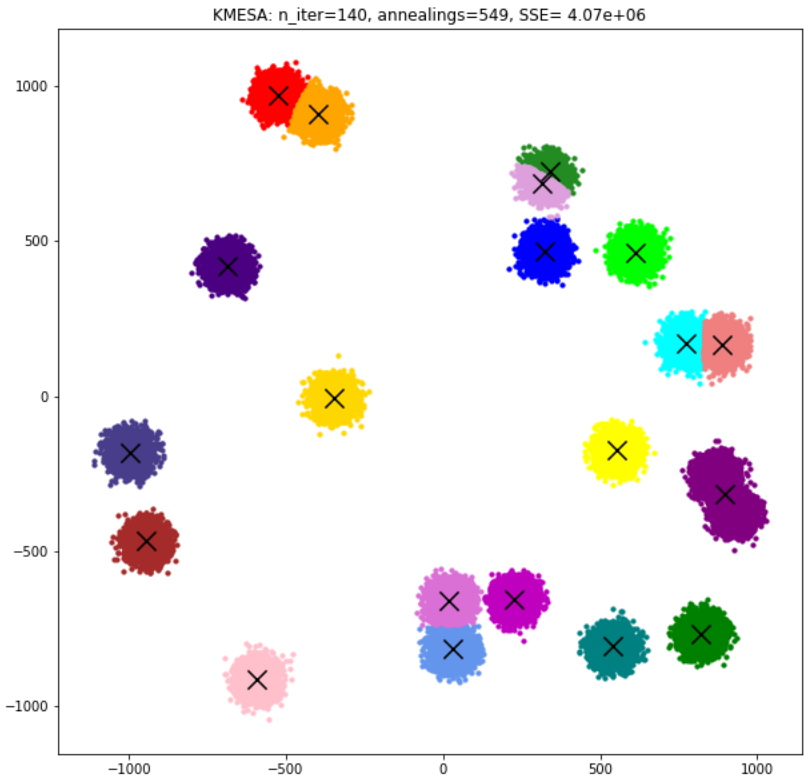


Figure 20: Carousel annealing

Slika 22: Rešenje velikog skupa tačaka preko KMESA algoritma sa različitim vrstama kaljenja

Najmanju grešku pronašao je algoritam sa nasumičnim kaljenjem (Random annealing) i iznosi $SSE = 4.07 * 10^6$. Ovo je vrlo blizu optimalnog rešenja, ali ipak izvršeno je suboptimalno klasterovanje. Naredna slika pokazuje rešenje nasumičnog kaljenja za bolje podešene parametre (4 reinicijalizacije, skup podataka je skaliran pre klasterovanja, verovatnoća i težina kaljenja su bolje podešene):

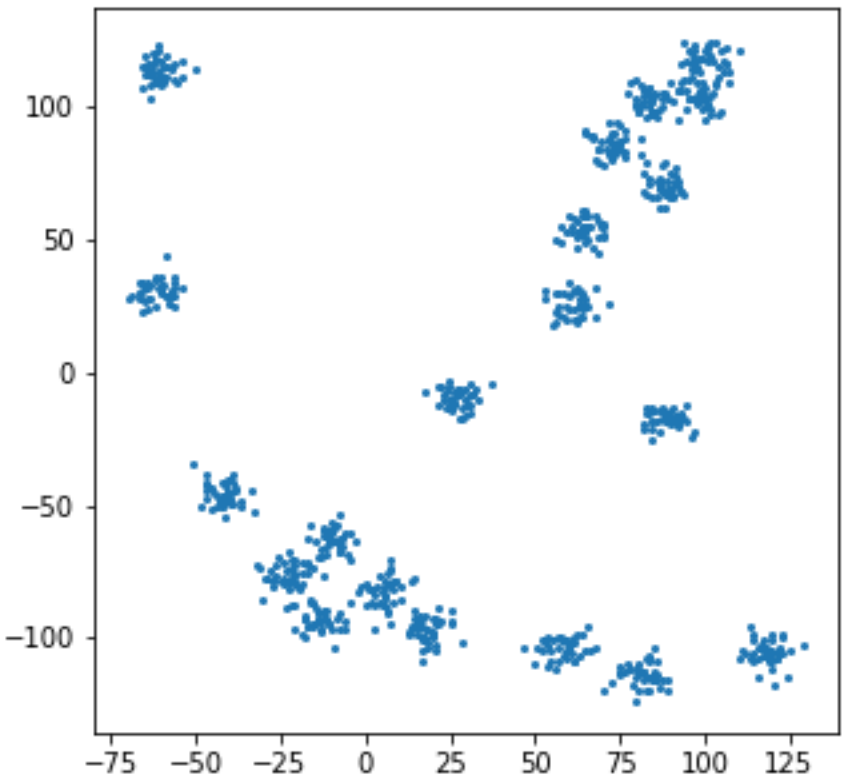


Slika 23: Rešenje velikog skupa tačaka preko KMESA algoritma sa nasumičnim kaljenjem

Rešenje se nije poboljšalo. Iz ovoga možemo da zaključimo da se, iako modifikovan, algoritam ne ponaša dobro za veliki broj klastera. Ovo će biti još snažnije u narednom primeru, zbog problema praznih klastera.

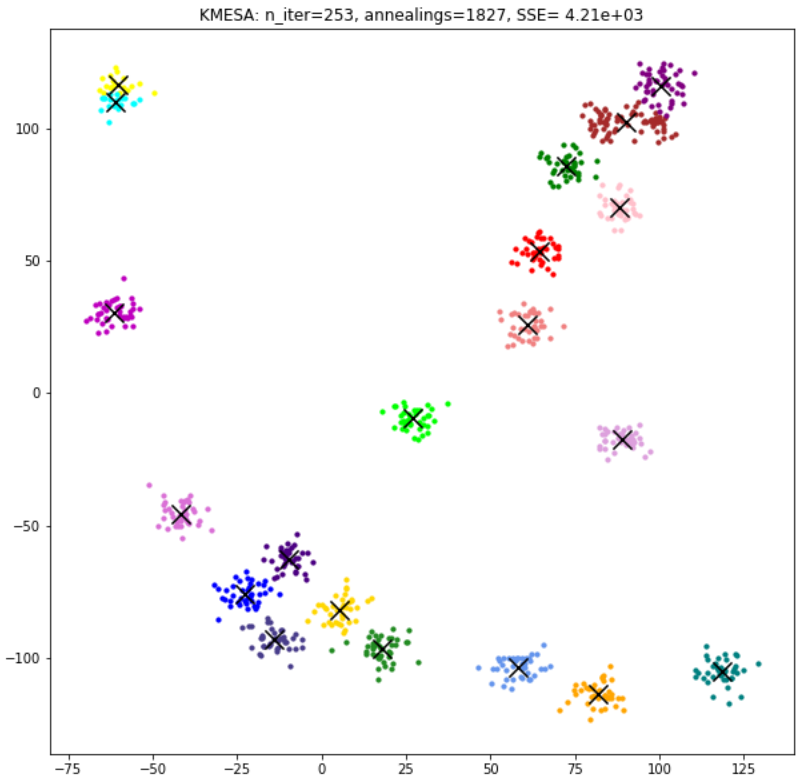
3.5 Redak skup tačaka sa puno klastera

U ovom primeru biće detaljnije prikazan problem praznih klastera. Skup se sastoji od 800 retkih tačaka sa 20 klastera, kao na slici:



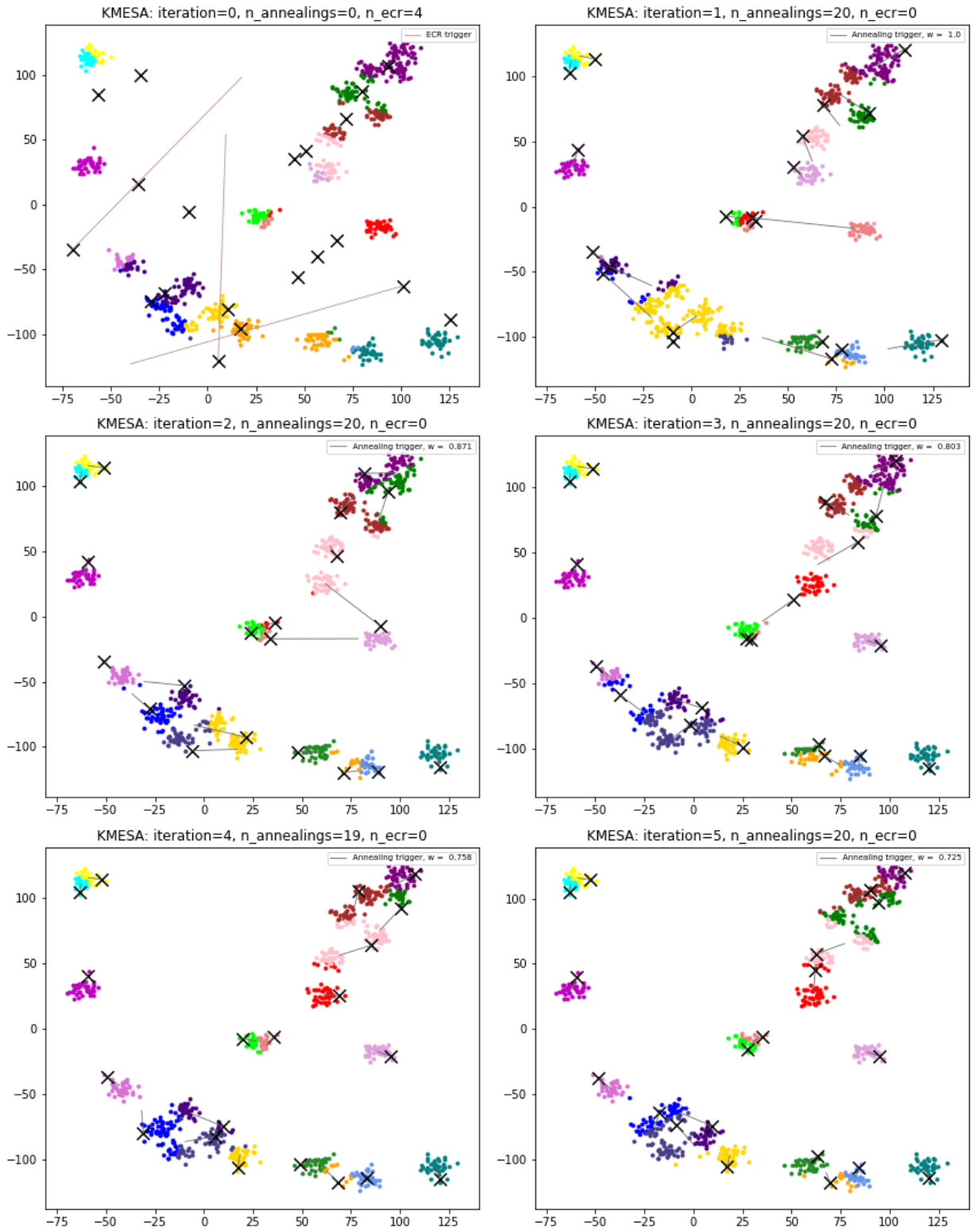
Slika 24: Redak skup tačaka sa puno klastera

Modifikovani algoritam sa maksimalnim kaljenjem (sa 5 nasumičnih reinicijalizacija) pronalazi sledeće rešenje:



Slika 25: Rešenje retkog skupa tačaka preko KMESA algoritma sa maksimalnim kaljenjem

Ovo je dosta blizu optimalnog klasterovanja, ali imajući u vidu da i pored 5 restartovanja algoritma sa sve simuliranim kaljenjem nije uspelo, rezultat nije zadovoljavajući. Ako pogledamo prvih nekoliko iteracija algoritma, videćemo da je faza inicijalizacije naišla na problem praznih klastera što je zahtevalo dosta velike ispravke:

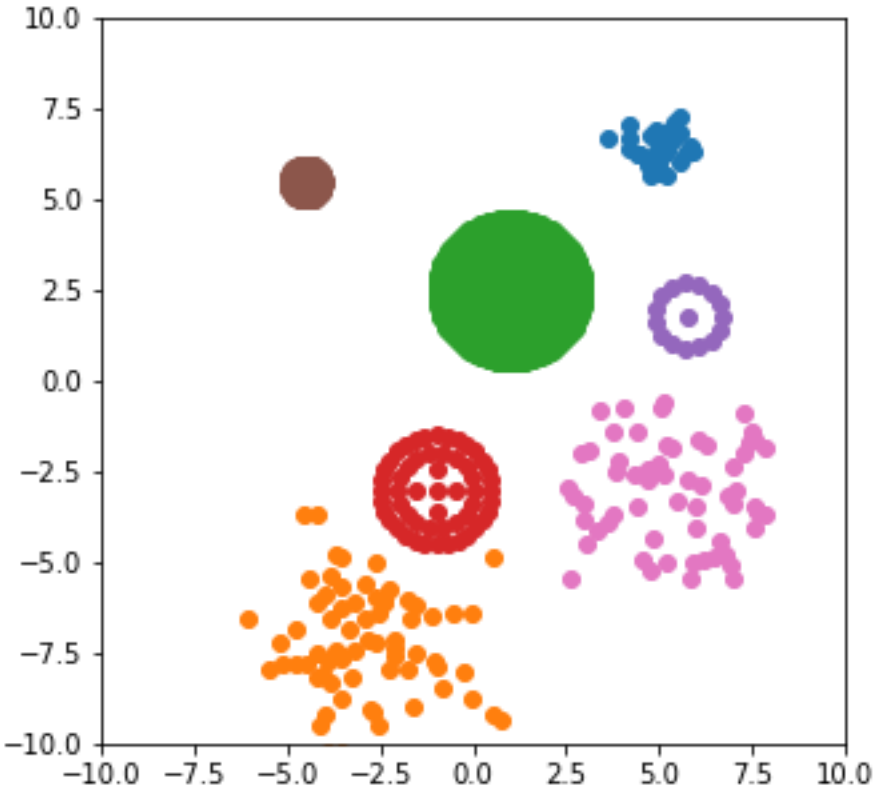


Slika 26: Problem praznih klastera kod retkog skupa tačaka

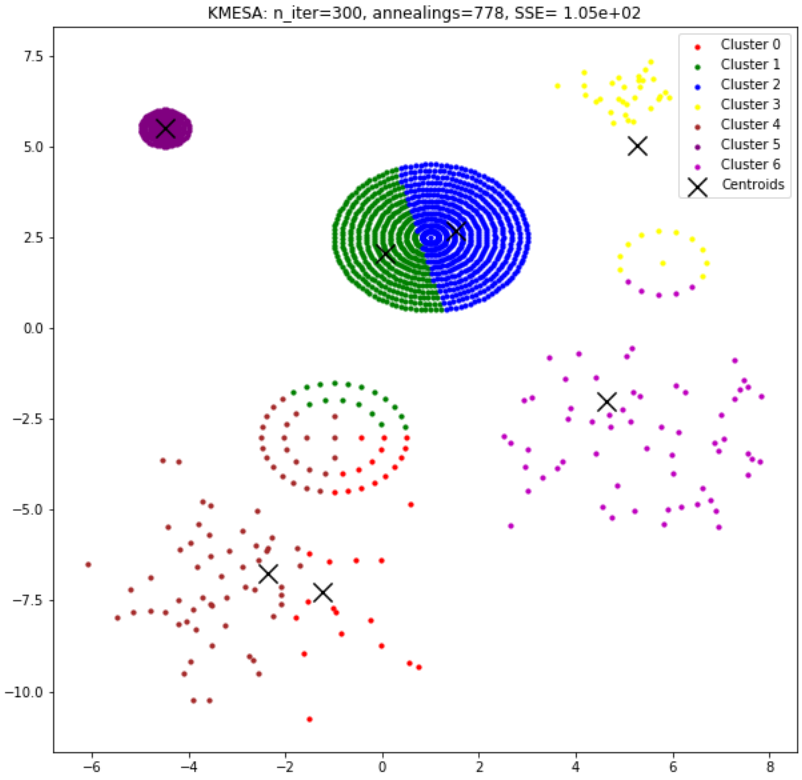
U nekim varijacijama algoritma nad istim skupom podataka ovaj problem se češće pojavljuje, čak i u nekim „središnjim“ iteracijama. Iako je u ovom radu ovaj problem rešen na najjednostavniji način, razrešavanje problema praznih klastera zahteva više pažnje i testiranja.

3.6 Klasteri različitih gustina

Naredni skup se sastoji od 1598 tačaka podeljenih u 7 klastera različitih gustina. Sa kaljenjem vrteške i čak 20 nasumičnih restartovanja algoritma, KMESA nije uspeo da pronađe optimalno klasterovanje.



Slika 27: Skup tačaka sa klasterima različitih gustina

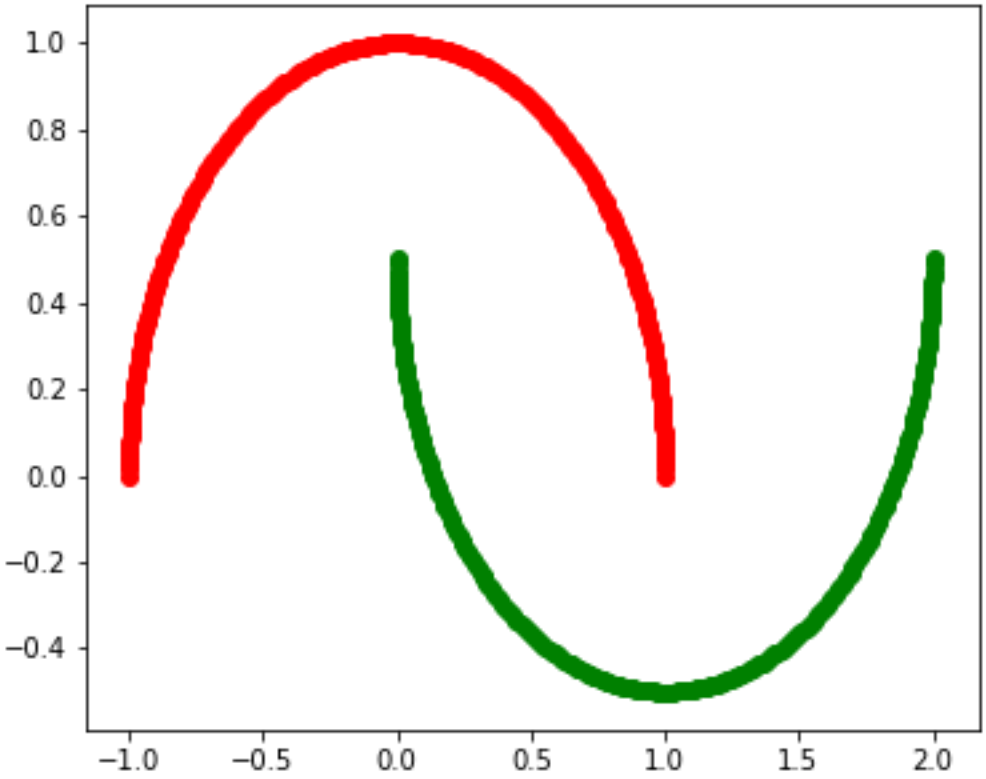


Slika 28: Rešenje skupa tačaka sa klasterima različitih gustina preko KMESA algoritma

Dakle, klasteri različitih gustina predstavljaju ozbiljan izazov za algoritam. Ovo je bio problem kod standardnog algoritma K-sredina, pa vidimo da modifikovani algoritam ne uspeva da prevaziđe ovaj problem.

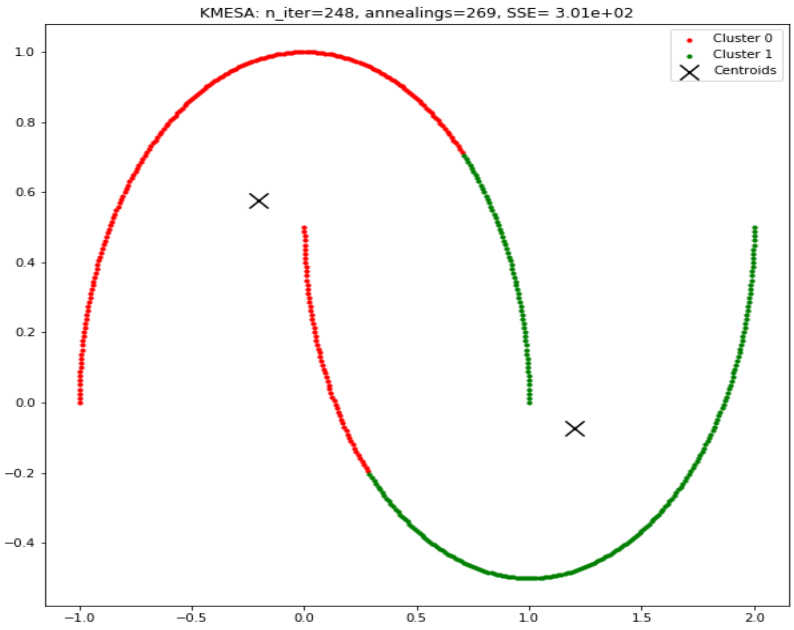
3.7 Klasteri specifičnog oblika

Ovde će biti prikazan rad algoritma nad klasterima specifičnog oblika. Preciznije, radi se o neglobularnim podacima.



Slika 29: "Mesečni" klasteri

Bez obzira na primenjeni metod kaljenja i ostale parametre algoritma, rešenje izgleda isto i optimalno klasterovanje nije pronađeno:

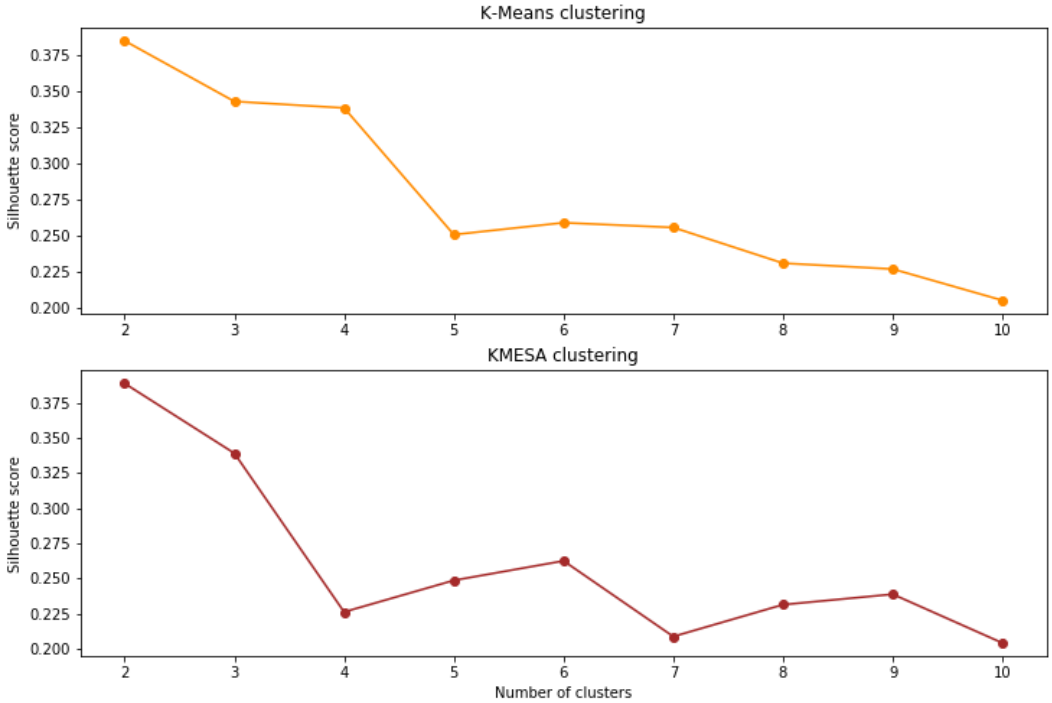


Slika 30: Rešenje mesečnih klastera preko KMESA algoritma

Modifikovani nije prevazišao ni ovaj problem koji je već postojao kod algoritma K-sredina.

3.8 Realni visokodimenzioni podaci

U skupu podataka Country-data.csv cilj je podeliti zemlje u grupe po razvoju na osnovu njihovih socijalnih, ekonomskih i zdravstvenih faktora. Skup je najpre skaliran, a zatim je za različite brojeve klastera na njega primenjeno klasterovanje K-sredina i K-sredina sa simuliranim kaljenjem za sve različite metode, od kojih je sačuvan rezultat za najbolji metod prema SSE. Upoređeni su rezultati između ova dva algoritma na osnovu silueta koeficijenta:



Slika 31: Poređenje K-Means i KMESA algoritama na osnovu silueta koeficijenta

Njihovi najbolji rezultati su sledeći:

	K-sredina	**KMESA**
Broj klastera	2	2
Silueta koeficijent	0.3845	0.3891
SSE	25.94	58.36

Naizgled KMESA je uspeo da pronađe klasterovanje koje ima nešto malo bolji silueta koeficijent u odnosu na algoritam K-sredina (tek na trećoj decimali), ali ovo poređenje nije merodavno iz prostog razloga što je silueta koeficijent u oba slučaja daleko ispod 0.5, što se smatra vrlo lošim klasterovanjem. Otuda nije jasno kakav se zaključak može izvesti, jer je moguće da u podacima postoje anomalije, poput autlajera, suvišnih atributa itd.

4. Zaključak

Kratak osvrt na jednostavne skupove podataka nad kojima je algoritam testiran može nas dovesti do određenih zaključaka: modifikovani algoritam nije prevazišao probleme algoritma K-sredina koji su postojali u odnosu na specifične skupove podataka, kao što su oni sa klasterima različitih oblika, različitih gustina, skupovima sa vrlo velikim brojem klastera i sl. Međutim, algoritam jeste doprineo boljim rezultatima nad globularnim podacima, a posebno u odnosu na korak inicijalizacije početnih centroida, što je donekle i bio cilj ovog rada. Poslednji primer klasterovanja nad realnim podacima nije pokazao mnogo, ali je ostavio nešto nade da algoritam uz dodatno poboljšanje, nove hiperparametre i naprednije heuristike u smislu simuliranog kaljenja može naći bolje klasterovanje od običnog algoritma, ali ovo ipak treba dokazati.

5. Reference

[1] https://en.wikipedia.org/wiki/K-means_clustering

[2] https://en.wikipedia.org/wiki/Agent-based_model

[3] https://en.wikipedia.org/wiki/Initialization_(programming)

[4] https://en.wikipedia.org/wiki/Local_optimum

[5] https://en.wikipedia.org/wiki/K-means%2B%2B#Example_of_a_sub-optimal_clustering

[6] https://en.wikipedia.org/wiki/Similarity_measure

[7] https://en.wikipedia.org/wiki/Euclidean_distance

[8] https://en.wikipedia.org/wiki/Unsupervised_learning

[9] https://en.wikipedia.org/wiki/Mathematical_optimization

[10] https://en.wikipedia.org/wiki/Residual_sum_of_squares

[11] https://web.archive.org/web/20151208140946/https://gautam5.cse.iitk.ac.in/opencs/sites/default/files/final.pdf; https://dl.acm.org/doi/abs/10.1145/2395116.2395117

[12] http://www.cs.umd.edu/~mount/Papers/pami02.pdf, filtrirani algoritam, sekcija 2

[13] C. C. Liu, S. W. Chu, Y. K. Chan and S. S. Yu, "A Modified K-Means Algorithm - Two-Layer K-Means Algorithm," 2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Kitakyushu, 2014, pp. 447-450, doi: 10.1109/IIH-MSP.2014.118.

[14] https://www.sciencedirect.com/science/article/abs/pii/S0031320310005029?via%3Dihub

[15] Fahim, A.M., Salem, A.M., Torkey, F.A. et al. An efficient enhanced k-means clustering algorithm. J. Zhejiang Univ. - Sci. A **7,** 1626–1633 (2006). https://doi.org/10.1631/jzus.2006.A1626...

[16] T. Dong, J. Ding, H. Yang, Y. Lei and H. Tao, "The application of simulated annealing K-means clustering algorithm in combination modeling," Proceeding of the 11th World

Congress on Intelligent Control and Automation, Shenyang, 2014, pp. 5751-5756, doi: 10.1109/WCICA.2014.7053702....
[17] https://en.wikipedia.org/wiki/Simulated_annealing
[18] https://en.wikipedia.org/wiki/Metaheuristic
[19] https://en.wikipedia.org/wiki/Cluster_analysis, sekcija 3