**Klasterizacija primjenom simuliranog hlađenja**

**Emina Modrić**

**Irhad Halilović**

Seminarski rad

Optimizacija resursa



Elektrotehnički fakultet Sarajevo

Univerzitet u Sarajevu

Bosna i Hercegovina

Ak. godina 2017/2018.

# Sažetak

Klastering, to jeste nenadzirano razvrstavanje objekata u skupine, se široko koristi u istraživačkoj analizi podataka. Problem klasteriranja je vrlo složen, a jedna od popularnih heuristika za njegovo rješavanje je algoritam simuliranog hlađenja (SA). SA je aproksimacijski algoritam koji uključuje generisanje susjednog rješenja perturbovanjem tekućeg na malen, ali smislen način. Novo rješenje se prihvaća sa vjerojatnoćom 1 ako je kvantitativno bolje od trenutnog rješenja, a u suprotnom se prihvaća prema Metropolis kriteriju. Kvaliteta klastera se mjeri kriterijom sume kvadratnih grešaka (SSE). Većina SA algoritama za klastering direktno perturbuje jednu od koordinata podataka. S druge strane algoritam predstavljen u ovom radu perturbuje slučajno odabrani centar pomoću Gaussove mutacije, a zatim ponovo dodjeljuje točke podataka na način najbližeg susjeda. Eksperimentalni rezultati su pokazali da je ovaj algoritam takođe veoma učinkovit, a mnogo je brži zbog jednostavnosti.

# Abstract

Clustering, or unscheduled sorting of objects in a group, is widely used in research data analysis. The problem of clustering is very complex and one of the most popular heuristics for its solution is the simulated annealing algorithm (SA). SA is an approximation algorithm that involves generating an adjacent solution by perturbating the current to a small but meaningful manner. The new solution is accepted with probability 1 if it is quantitatively better than the current solution, otherwise it is accepted according to the Metropolis criterion. The cluster quality is measured by the Sum of Squared Error (SSE) criterion. Most SA clustering algorithms directly disturb one of the data coordinates. On the other hand, the algorithm presented in this work perturbes the randomly selected center by Gauss mutation, and then reassigns the data points to the nearest neighbors. Experimental results have shown that this algorithm is also very effective but much faster because of its simplicity.

# Sadržaj

[Sažetak 2](#_Toc22033)

[Abstract 3](#_Toc22034)

[Sadržaj 4](#_Toc22035)

[1. Uvod 5](#_Toc22036)

[1.1. Opis problema 5](#_Toc22037)

[1.2. Pregled literature vezane za opisani problem 6](#_Toc22038)

[1.3. Moguće aplikacije u praksi 7](#_Toc22039)

[2. Korišteni algoritam 9](#_Toc22040)

[2.1. Opis rada korištenog algoritma 9](#_Toc22041)

[2.2. Svođenje opisanog problema u formu korištenog algoritma 11](#_Toc22042)

[3. Simulacijski rezultati 14](#_Toc22043)

[3.1. Postavka simulacija 14](#_Toc22044)

[3.2. Rezultati simulacija 17](#_Toc22045)

[3.3. Zaključak 21](#_Toc22046)

[4. Zaključak i diskusija 22](#_Toc22047)

[Reference 23](#_Toc22048)

# Uvod

## Opis problema

Klastering predstavlja nenadzirano razvrstavanje objekata u grupe na način da objekti u istoj grupi imaju određene međusobne sličnosti. Na primjer, klasteringom voća prema obliku bi se mogli dobiti klasteri (grupe) okruglastog oblika (narandža, jabuka, itd), izduženog oblika (banana, itd) i sl. U opštem smislu, svrha klasteringa je grupisanje skupa tačaka podataka tako da su tačke u bilo kom klasteru sličnije jedna drugoj nego tačkama u drugim klasterima. Klasterizacija se koristi u brojnim oblastima, uključujući statistiku, ekonomiju, fiziku, psihologiju, biologiju, prepoznavanje uzoraka, inženjering i marketing. Razlika između tačaka podataka i kvaliteta klasterizacije se obično kvantifikuje korišćenjem kvadrata euklidske udaljenosti. Problem grupisanja n tačaka podataka u k klastera riješen je minimiziranjem sledeće funkcije cilja:

pri čemu je:

gdje dij označava euklidsku udaljenost između tačke i i središta klastera j, a wij ∈ {0, 1} osigurava da tačke ostaju ekskluzivne za njihove odgovarajuće klastere. [1]

Problem klasteringa se može rješavati različitim algoritmima koji se značajno razlikuju jedan u odnosu na drugi u razumijevanju onoga šta zapravo sačinjava klaster i kako učinkovito pronaći te klastere. Popularni pojmovi klastera uključuju skupine s malim udaljenostima između članova klastera, gustim područjima prostora podataka, intervalima ili određenim statističkim distribucijama. Grupisanje se stoga može formulisati kao problem višestrukog cilja. Odgovarajući algoritam klasterizacije i postavke parametara (uključujući parametre kao što su funkcija udaljenosti, nivo gustoće ili broj očekivanih klastera) ovise o pojedinačnom skupu podataka i namjeravanoj upotrebi rezultata. Podjela na klastere kao takva nije automatski zadatak, već iterativan proces otkrivanja znanja ili interaktivne multi-objektivne optimizacije koja uključuje metodu pokušaja i pogreške. Često je potrebno modifikovati predobradu podataka i parametre modela sve dok rezultat ne postigne željena svojstva.

Klastering je kombinatorni problem, koji se bavi pronalaženjem optimuma funkcija diskretnih varijabli. Postoje dva načina rješavanja kombinatornog optimizacijskog problema: optimizacijski algoritmi i aproksimacijski algoritmi. Optimizacijski algoritmi vraćaju globalno optimalno rješenje, dok aproksimacijski algoritmi (koji se nazivaju i “heuristički”) vraćaju rješenje koje je “blisko” globalnom optimumu. U teoriji, algoritmi optimizacije se čine boljim izborom jer vraćaju najbolje rješenje, ali u praksi oni mogu biti veoma teški za dizajniranje i implementaciju, a oni su obično problemski specifični. Što je još važnije, oni mogu biti izuzetno spori pri rješavanju problema veće dimenzionalnosti. Heuristički algoritmi, s druge strane, često se lakše implementiraju, generički su, i brži. Simulirano hlađenje (SA) je popularni heuristički algoritam koji se primijenjuje na razne optimizacijske probleme, uključujući klastering. [2] U ovom radu bit će korišten algoritam simuliranog hlađenja.

Problem klasteringa nastao je u antropologiji od strane Driver i Kroebera 1932. i uvedena je u psihologiju od strane Zubina 1938. i Roberta Tryona 1939. te korištena od strane Cattella početkom 1943. za klasifikaciju teorije osobina u psihologiji osobnosti.

## Pregled literature vezane za opisani problem

Za rješavanje postavljenog problema korišteni su radovi mnogi radovi koji su nabrojani u odsječku Literatura. U uvodnom dijelu, za definisanje problema projektnog zadatka, korišteni su radovi od Stephen Merendino, M. Emre Celebi - A Simulated Annealing Clustering Algorithm Based on Center Perturbation Using Gaussian Mutation i Samima Konjicije - Predavanja na predmetu: Optimizacija resursa, s tim da je prvi rad korišten za precizno definisanje problema, dok je uz pomoć drugog rada dat osvrt na optimizacijske i aproksimacijske aloritme. Poglavlje u kojima su diskutovanje moguće aplikacije u praksi je napisano uz pomoć informacija preuzetih sa wikipedije na temu klasteringa.

Za opis rada korištenog algoritma te za pisanje koda za implementaciju algoritma korištena su tri rada: Stephen Merendino, M. Emre Celebi - A Simulated Annealing Clustering Algorithm Based on Center Perturbation Using Gaussian Mutation, Donald E Brown, Christopher L. Huntley - A Practical Application of Simulated Annealing to Clustering, te Mustahsan Mir - Application of Simulated Annealing to Cluster-Boundary Search Algorithm for Macrocell Placement Optimization, pri čemu je najveću pomoć pružio prvi rad, te je ovaj projektni zadatak većinski urađen uz pomoć njega.

## Moguće aplikacije u praksi

Područje primjene klasteringa je veoma široko. Potreba za klasteringom se javlja u različitim domenima, a u nastavku teksta će biti spomenuti neki od njih.

U biologiji, računarskoj biologiji i bioinformatici klastering se koristi za opisivanje i izradu prostornih i vremenskih usporedbi zajednica organizama u heterogenim sredinama. Također se koristi u sistematici biljaka za stvaranje umjetnih filogenija ili skupina organizama kao što su vrsta, rod ili skupine organizama koji dijele niz atributa prema višem nivou. Klastering se također koristi za izgradnju skupina gena sa srodnim uzorcima ekspresije (također poznatim kao koeksprimirani geni) kao u HCS algoritmu klasteriranja. Često takve skupine sadrže funkcionalno povezane proteine, kao što su enzimi za specifični put, ili geni koji su koregulirani. Zatim, klastering sekvenci se koristi za grupisanje homolognih sekvenci u obitelji gena, što je vrlo važan pojam u bioinformatici i evolucijskoj biologiji općenito. Također, u klasteringu se koristi sličnost genetskih podataka kod ljudi kako bi se zaključile populacijske strukture.

Klastering je našao velike primjene u medicini. Kod PET skeniranja, klaster analiza može se koristiti za razlikovanje tipova tkiva u trodimenzionalnoj slici što se koristi za različite svrhe. Zatim, može se koristiti za analizu obrazaca otpornosti na antibiotike, za klasificiranje antimikrobnih spojeva prema njihovom mehanizmu djelovanja, za klasificiranje antibiotika prema njihovoj antibakterijskoj aktivnosti.

U marketingu, klastering se široko koristi u istraživanju tržišta pri radu s multivarijatnim podacima iz anketa i ispitnih panela. Istraživači tržišta koriste klastering kako bi podijelili opću populaciju potrošača na tržišne segmente i bolje razumjeli odnose između različitih skupina potrošača i potencijalnih kupaca, te za korištenje u segmentaciji tržišta, pozicioniranju proizvoda, razvoju novih proizvoda i odabiru testnih tržišta. Klastering se može koristiti za grupisanje svih stavki koje su dostupne u određenom *online shop-u* u skupove jedinstvenih proizvoda.

U proučavanju društvenih mreža, klastering se koristi za prepoznavanje zajednica unutar velikih skupina ljudi, a u procesu inteligentnog grupisanja datoteka i web-mjesta, klasteriranje se može koristiti za stvaranje relevantnijeg skupa rezultata pretraživanja u usporedbi s uobičajenim pretraživačem, poput Googlea. Trenutno postoji niz alata za klasteriranje na webu, kao što je Clusty. Također se može koristiti za vraćanje sveobuhvatnijeg skupa rezultata u slučajevima gdje se pojam za pretraživanje može odnositi na znatno različite stvari. Svaka zasebna upotreba izraza odgovara jedinstvenom skupu rezultata, dopuštajući algoritmu rangiranja da vrati sveobuhvatne rezultate odabirom najboljeg rezultata iz svakog klastera.

Flickr-ova mapa fotografija i karti koristi klastering kako bi se smanjio broj oznaka na karti. To ga čini bržim i smanjuje količinu vizualnog nereda.

Klastering je koristan u evoluciji softvera jer pomaže smanjiti naslijeđena svojstva u kodu reformiranjem funkcionalnosti koja je postala raspršena.

Klastering se može koristiti za dijeljenje digitalne slike u različite regije za otkrivanje granica ili prepoznavanje objekta.

Još jedna od primjena su Markovljevi lanci Monte Carlo metode koji omogućavaju klastering za lociranje i karakterizaciju ekstrema u ciljnoj distribuciji.

Zatim, može se koristiti za otkrivanje anomalija u nekom sistemu. [3]

Naravno, pored navedenog, postoje još mnogostruke primjene i aplikacije u praksi.

# Korišteni algoritam

## Opis rada korištenog algoritma

Simulirano hlađenje (Simulated anealing – SA) je algoritam koji se zasniva na fizičkom procesu zagrijavanja u kojem se materijal zagrijava u peći koja omogućuje da se atomi slobodno kreću. Nakon zagrijavanja, materijal se hladi pažljivo i polako tako da se atomi mogu zaustaviti u kristalu ili niskoenergetskom stanju. Porijeklo SA algoritma potječe iz 1953., kada je skupina istraživača pod vodstvom Nicholasa Metropolisa razvila algoritam za modeliranje ovog procesa fizičkog zagrijavanja. [4] Pseudokod Metropolis algoritma dat je ispod. KB i t označavaju Boltzmannu konstantu i temperaturu tijela, respektivno.

1: Odabrati početno stanje i ∈ S;

2: Generirati novo stanje primjenom male slučajno generirane perturbacije

3: Izračunati promjenu energije ΔE

4: if ∆E < 0 then

5: Novo stanje se prihvaća kao početna tačka za sljedeću iteraciju

6: else if ∆E > 0 then

7: Novo stanje se prihvaća sa vjerojatnoćom exp (-KBΔE / t)

8: end if

Za implementaciju SA algoritma, moraju se definisati sljeđeća četiri parametra:

• Rješenje – predstavlja prihvatljivo rješenje problema, može se uporediti sa "stanjem" čvrstog tijela u Metropolis algoritmu.

• Funkcija perturbacije - je metoda koja neznatno malo, ali smisleno mijenja trenutno rješenje, obično putem zamjene ili permutacije, kako bi se generisalo sljedeće potencijalno rješenje u odnosu na trenutno, uporedivo je sa atomima koji se slobodno kreću u Metropolis algoritmu.

• Funkcija troška –je funkcija koja kvantificira kvalitetu rješenja, uporedivo je s “energijom” tijela u Metropolis algoritmu.

• Funkcija hlađenja – svrha mu je odrediti početnu temperaturu, odrediti kada i koliko treba smanjiti temperaturu i kada je proces hlađenja završen.

U fizičkom zagrijavanju, temperatura ima pravo značenje, dok s druge strane u SA algoritmu temperatura funkcioniše kao kontrolni parametar. Svrha temperature je da kontroliše vjerovatnoću prihvaćanja povećanja troškova. [5] Prilikom pokretanja algoritma temperatura bi trebala biti dovoljno visoka, tako da vjerojatnoća prihvaćanja novog rješenja koje je gore od trnutnog bude blizu 1. Temperaturni parametar se monotono smanjuje tako da vjerovatnoća prihvaćanja na kraju, kako algoritam napreduje, dostigne nulu. Na svakoj temperaturi mora se dopustiti da sistem dostigne "temperaturni ekvilibrium". To znači da se na svakoj temperaturi mora napraviti dovoljno perturbacija kako bi se dobilo dobro uzorkovanje okoline za tu temperaturu. Time se osigurava da je do trenutka kada algoritam dostigne završnu fazu izbjegao sve lokalne ekstreme i da je blizu globalno optimalnog rješenja. Nakon završetka, konačna konfiguracija se uzima kao rješenje problema.

Prilikom implementacije SA algoritma moraju se donijeti dvije vrste odluka, a to su generičke i specifične. Generički izbor se odnosi na postavljanje parametara funkcije hlađenja, dok se specifični izbori problema fokusiraju na funkciju troškova, strukturu okoline i operatora preturbacije.

Algoritam koji je implementiran u ovom seminarskom radu je SA klastering algoritam koji se temelji na središnjoj perturbaciji pomoću Gaussove mutacije, u daljnjem tekstu SAGM algoritam. Osnovna struktura SAGM algoritma jednaka je standardnom SA algoritmu čiji je pseudokod prikazan iznad.

SAGM se od ostalih SA algoritama razlikuje, u načinu perturbovanja trenutnog rješenje kako bi se stvorilo susjedno rješenje. Pseudokod za funkciju perturbovanja dat je ispod:

1: Neka je C skup klastera u trenutnom rješenju.

2: Neka je c centar nasumice odabranog klastera iz skupa C.

3: Neka je δ faktor mutacije koji je blizak nuli.

4: Neka je G generator Gaussovog slučajnog broja.

5: for svaki atribut i od c do

6: Postavi i = i + δG;

7: end if

8: Ponovo dodijeliti sve tačke najbližem klasteru čiji centar ima najmanju kvadratnu euklidsku udaljenost od određene tačke.

9: Izračunajte novu sumu kvadratnih grešaka.

Za funkciju opadanja temperature koristi se geometrijska funkcija. Vrijednosti t0 i α preporučljivo je postaviti na 100, odnosno 0,95, respektivno. Funkcija opadanja temperature ima oblik:

## Svođenje opisanog problema u formu korištenog algoritma

U nastavku je prikazano jedno rješenje problema klasteringa primjenom SA algoritma.

S obzirom da je SA algoritam za pronalazak optimuma funkcije, potrebno je definisati funkciju koja svakom rješenju(rasporedu vektora ulaza u klastere) dodjeljuje određenu vrijednost. Na isječku koda je prikazana implementacija funkcije koja računa zbir udaljenosti svake tačke od centra njenog klastera.

def value(state, points):

cp = AssignCluster(points, state)

sum=0

for i in range(0, len(points)):

sum += distance(points[i], state[cp[i]])\*\*2

return sum

Ako se svaka tačka svrstava u klaster od kojeg je najmanje udaljena, optimalan raspored klastera je onaj u kojem je vrijednost funkcije value minimalna, odnosno zbir udaljenosti svake tačke od centra njenog klastera je minimalna.

Funkcija value prima parametar state koji sadrži koordinate svakog od klastera i parametar points sa koordinatama svih tačaka. Prvo se svakoj tački dodjeljuje najbliži klaster, a zatim se računa suma udaljenosti tačaka od njihovog klastera.

SA algoritam za optimizaciju je implementiran u funkciji SimulatedAnnealing, čija implementacija je prikazana na isječku koda ispod.

def SimulatedAnnealing(max\_iter, initial\_temperature, alpha, final\_temperature, initial\_state, points):

t = initial\_temperature

current\_state = initial\_state[:]

while(t >= final\_temperature):

for i in range(1, max\_iter):

next\_state = perturb(current\_state[:])

energy\_delta = value(next\_state,points) - value(current\_state,points)

if ((energy\_delta < 0) or (math.exp( -energy\_delta / t) >= random.randint(0,10))):

current\_state = next\_state

t = alpha \* t

return current\_state

Funkcija kao parametre prima početnu i finalnu temperaturu te vrijednost alfa (parametri initial\_temperature, final\_temperature, alpa respektivno). Algoritam se izvršava dok se ne dostigne vrijednost finalne temperature. Funkcija opadanja temperature ima oblik

Za svaku vrijednost temperature, algoritam se izvršama max\_iter puta, što predstavlja prvi parametar funkcije.

Funkcija prima parametar points, što predstavlja niz koordinata svih tačaka koje se razvrstavaju u klastere.

Parametar initial\_state sadrži početne koordinate svih klastera. U toku izvršavanja algoritma će se mijenjati koordinate klastera.

Za poboljšavanje trenutnog rasporeda klastera potrebno je generisati novi raspored, za šta se koristi funkcija perturb:

def perturb(state):

c = random.randint(0, len(state) - 1)

d = 0.1

G = np.random.normal()

#for i in range(0, len(state[c])):

#state[c] = (state[c][0] + d\*G, state[c][1] + d\*G)

state[c] = [x + d\*G for x in state[c]]

return state

Funkcija perturb kao parametar prima trenutni raspored klastera - state. Slučajnim odabirom se odabere jedan od klastera te se vrijednost njegovih koordinata mijenja za veličinu 0.1 \* G. G je slučajno generisani broj koji može biti i pozitivan i negativan.

# Simulacijski rezultati

Za potpuni prikaz rada algoritma u nastavku će biti prikazane tri simulacije. Jedna simulacija predstavlja klasterizaciju tačaka u dvodimenzionalnom prostoru, druga klasterizaciju tačaka u trodimenzionalnom prostoru dok treća predstavlja klasterizaciju objekata u petodimenzionalnom i šestodimenzionalnom prostoru (objekti imaju 5/6 atributa – koordinate).

## Postavka simulacija

1. Postavka simulacije u dvodimenzionalnom prostoru

Potrebno je izvršiti klasterizaciju tačaka u grupe u dvodimenzionalnom prostoru pa je poziv funkcije dat ispod. Points je niz tačaka u dvodimenzionalnom prostoru, a initial\_state su početni centri klastera, a nakon toga slijedi poziv kreirane funkcije SimulatedAnnealing.

initial\_state = [(2,6), (6,6), (9,3)]

points = [(3,2), (3.5,2), (3.5, 2.5), (4,3), (7,4), (7.5,3.5), (8,5), (1.5,5)]

SimulatedAnnealing(340, 100, 0.95, 0.0001, initial\_state, points)

1. Postavka simulacije u trodimenzionalnom prostoru

U ovom dijelu je potrebno izvršiti klasterizaciju tačaka u grupe u trodimenzionalnom prostoru. Ispod je prikazan isječak koda kojim je ovo obavljeno koristeći kreiranu funkciju SimulatedAnnealing. Initial\_state i points imaju ista značenja kao i u dijelu pod b)

initial\_state = [(2,6,0), (6,6,6), (9,3,2)]

points = [(3,2, 2), (3,2.5,3), (3.5, 2.5, 2.5), (4,3,4), (7,4,4), (7.5,3.5,2), (8,5,0), (1.5,5,4), (-3,-1,-4), (-4,0,-3.5), (-2,-2,-3)]

SimulatedAnnealing(240, 100, 0.95, 0.0001, initial\_state, points)

1. Postavka simulacije u peto/šestodimenzionalnom prostoru

Sada će se izvršiti klasterizacija objekata sa šest, odnosno pet atributa (koordinata). Ovo predstavlja klasterizaciju u višedimenzionalnom prostoru.

Objekti koji se klasteriziraju su polovni automobili koje korisnici postavljaju na prodajnu web stranicu pik.ba uz specificiranje karakteristika automobila i tražene cijene. U tabeli 3.1. su prikazane specifikacije ovih auta, zajedno sa cijenom. U prvoj simulaciji cijena predstavlja jednu od koordinata u prostoru, dok u drugoj simulaciji cijena neće biti korištena, već će problem imati jednu dimenziju manje.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Šifra | Godište | Gorivo | Kubikaža | Tip | Kilovata | Cijena |
| 1.p | 2000 | dizel | 1.9 | Malo auto | 51 | 3000 |
| 2.p | 2005 | dizel | 1.6 | Malo auto | 66 | 4000 |
| 3.r | 2005 | Dizel | 1.5 | miniven | 63 | 4600 |
| 4.r | 2009 | benzin | 1.2 | limuzina | 55 | 7800 |
| 5.g | 2007 | Dizel | 1.9 | limuzina | 77 | 7850 |
| 6.g | 2005 | dizel | 1.9 | limuzina | 77 | 9300 |
| 7.a | 2011 | dizel | 1.6 | Malo auto | 77 | 17000 |
| 8.a | 1994 | plin | 2.0 | limuzina | 66 | 2800 |

Tabela 3.1. Specifikacije testnog skupa automobila

Specifikacijama Gorivo i Tip če biti dodijeljene brojčane vrijednosti da bi se mogle koristiti u pozivu implementacije algoritma, što je prikazano u tabelama 3.2. i 3.3., a vrijednost kubikaže je pomnožena sa 10, da bi se povećao njen uticaj na klastering.

|  |  |
| --- | --- |
| Gorivo | Vrijednost |
| Dizel | 10 |
| Benzin | 20 |
| Plin | 30 |

Tabela 3.2. Dodijeljene vrijednosti podatku gorivo

|  |  |
| --- | --- |
| Tip | Vrijednost |
| Malo auto | 10 |
| Limuzina | 20 |
| Miniven | 30 |

Tabela 3.3. Dodijeljene vrijednosti podatku tip

Sa ovim prilagodbama je moguće kreirati ulazne vektore u algoritam, čiji je poziv prikazan u isječku koda ispod. Points je niz tačaka koje predstavljaju automobile u prostoru, a initial\_state su početni centri klastera.

initial\_state = [

(1990, 10, 12, 10, 50, 3000),

(2000, 20, 16, 20, 60, 6000),

(2010, 30, 19, 30, 70, 10000)

]

points = [

(2000, 10, 19, 10, 51, 3000),

(2005, 10, 16, 10, 66, 4000),

(2005, 10, 15, 10, 63, 4600),

(2009, 20, 12, 10, 55, 7800),

(2007, 10, 19, 10, 77, 7850),

(2005, 10, 19, 10, 77, 9300),

(2011, 10, 16, 10, 77, 17000),

(1994, 30, 20, 10, 66, 2800)

]

final\_state = SimulatedAnnealing(240, 100, 0.95, 0.0001, initial\_state, points)

Poziv algoritma bez informacije o cijeni je prikazan ispod.

initial\_state = [

(1990, 10, 12, 10, 50),

(2000, 20, 16, 20, 60),

(2010, 30, 19, 30, 70)

]

points = [

(2000, 10, 19, 10, 51),

(2005, 10, 16, 10, 66),

(2005, 10, 15, 10, 63),

(2009, 20, 12, 10, 55),

(2007, 10, 19, 10, 77),

(2005, 10, 19, 10, 77),

(2011, 10, 16, 10, 77),

(1994, 30, 20, 10, 66)

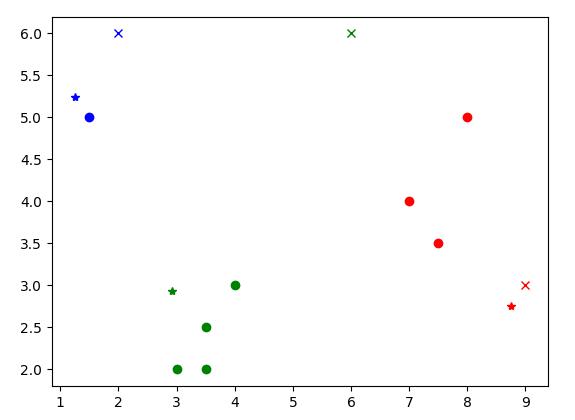
]

final\_state = SimulatedAnnealing(240, 100, 0.95, 0.0001, initial\_state, points)

## Rezultati simulacija

1. Rezultati simulacije u dvodimenzionalnom prostoru

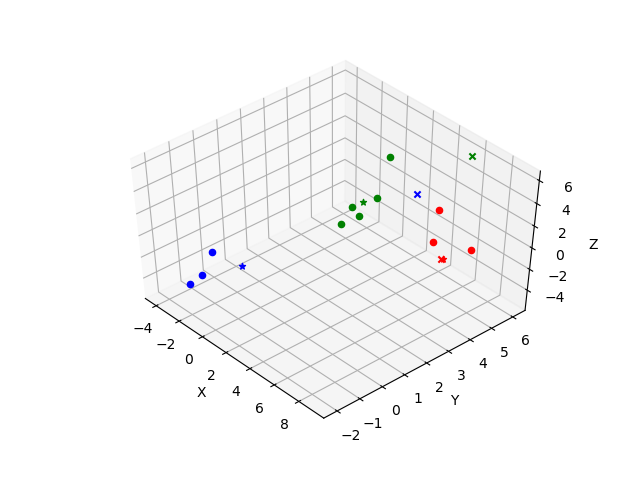
Pošto je klasterizacija vršena u dvodimenzionalnom prostoru, rezultati simulacije mogu se vizuaizirati. Na slici 3.1. ispod prikazani rezultati simulacije, pri čemu x predstavljaju početne centre klastera, \* su krajnji centri klastera a tačke su tačke u prostoru. Tačke koje pripadaju određenom klasteru imaju boju kao centar tog klastera.



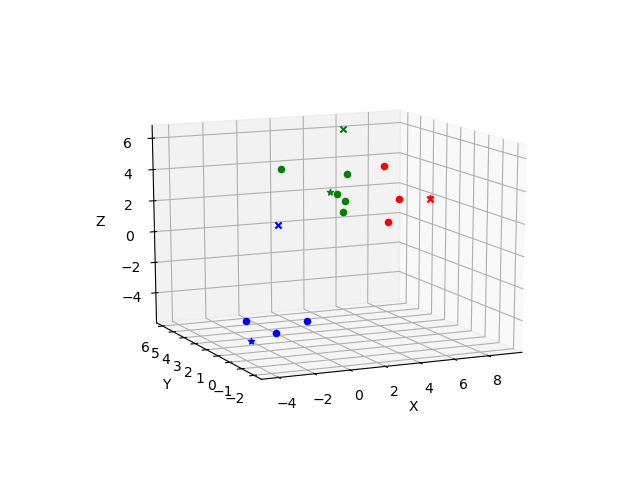
Slika 3.1. Rezultati simulacije u dvodimenzionalnom prostoru

1. Rezultati simulacije u trodimenzionalnom prostoru

U ovom dijelu je dat rezultat simulacije u trodimenzionalnom prostoru. I trodimenzionalni prostor je moguće vizualizirati, stoga će rezultati i ove simulacije bizi prikazani na slici. Ispod su prikazane dvije slike, slika 3.2. i slika 3.3. koje predstavljaju prikaz istog problema, ali iz različitih uglova, da bi se bolje mogao steću uvid u rezultate simulacije ovog problema. Ponovno x predstavljaju početne centre klastera, \* su krajnji centri klastera a tačke su tačke u prostoru. Tačke koje pripadaju određenom klasteru imaju boju kao centar tog klastera.



Slika 3.2. Prvi prikaz rezultata simulacije u trodimenzionalnom prostoru



Slika 3.3. Drugi prikaz rezultata simulacije u trodimenzionalnom prostoru

1. Rezultati simulacije u šestodimenzionalnom prostoru

U ovom dijelu je dat rezultat simulacije u prostoru sa pet odnosno šest dimenzija. Rezultat ove simulacije se ne može prikazati vizualno zbog broja koordinata, pa će se analiza obaviti tabelarnim prikazom. U tabeli 3.4. prikazani su rezultati obje simulacije, sa cijenom i bez cijene automobila.

|  |  |  |
| --- | --- | --- |
| Šifra | Klaster | Klaster bez cijene |
| 1.p | 0 | 0 |
| 2.p | 0 | 1 |
| 3.r | 1 | 1 |
| 4.r | 1 | 1 |
| 5.g | 1 | 1 |
| 6.g | 2 | 1 |
| 7.a | 2 | 2 |
| 8.a | 0 | 1 |

Tabela 3.4. Rezultati simulacija u prostoru sa pet i šest dimenzija

U prvom slučaju automobili sa cijenama 3000, 4000 i 2800 pripadaju nultom klasteru, automobili sa cijenama 9300 i 17000 pripadaju drugom klasteru, a ostali su u prvom. Svi automobili nultog klastera imaju manju cijenu od automobila prvog i drugog, dok svi automobili drugog klastera imaju veću cijenu od automobila iz nultog i prvog klastera. Očigledno u simulaciji u kojoj je i cijena predstavljala jednu od koordinata objekata, cijena predstavlja ključnu ulogu prilikom rasporeda u klastere, zato jer ta koordinata ima najveće vrijednosti.

U drugom slučaju, kada cijena objekata nije davala značaj na klasterizaciju objekata, rezultat simulacije bio je malo drugačiji ali veoma sličan. Automobil 1.p prešao je iz klastera 0 u klaster 1 a automobil 6.g je preašao iz klastera 2 u klaster 1. Može se zaključiti da automobil 1.p pripada grupi automobila sa najnižom cijenom, dok po specifikacijama zapravo pripada srednjoj grupi. Također, automobil 6.g pripada po cijeni najskupljoj grupi, dok po ostalim specifikacijama pripada srednjoj grupi. Sada bi se mogli izvoditi zaključci o valjanosti predloženih cijena od strane prodavača, naime automobil 1.p bi se možda mogao prodati skuplje, dok bi ponuđena cijena automobila 6.g zapravo mogla biti prevelika.

## Zaključak

Iz gore prikazanih rezultata simulacija koje su date vizualno i tabelarno može se veoma lako zaključiti da se tačke uspješno razvrstavaju u klastere te da algoritam radi zadovoljavajuće dobro.

# Zaključak i diskusija

U ovom seminarskom radu predstavljen je algoritam partijskog klasteringa baziran na SA algoritmu. Ovaj algoritam grupiranja perturbira trenutno rješenje kako bi se generisalo susjedno rješenje, pri čemu koristi Gaussovu mutaciju kako bi na kontrolirani način promijenio poziciju slučajno odabranog centra klastera. Ova metoda generisanja susjednog rješenja ima konvergenciju ka visokokvalitetnom klasteringu podataka.

# Reference

[1] Stephen Merendino, M. Emre Celebi, A Simulated Annealing Clustering Algorithm Based on Center Perturbation Using Gaussian Mutation, Louisiana State University in Shreveport, 2012

[2] Samim Konjicija, Predavanja na predmetu: Optimizacija resursa, Elektrotehnički fakultet, Univerzitet u Sarajevu, Š/K 2018/2019.

[3] <https://en.wikipedia.org/wiki/Cluster_analysis#Applications>

[4] Donald E Brown, Christopher L. Huntley - A Practical Application of Simulated Annealing to Clustering, University of Virginia, 1991

[5] Mustahsan Mir, Application of Simulated Annealing to Cluster-Boundary Search Algorithm for Macrocell Placement Optimization, Ajman University of Science & Technology, 2010