Minimum K-Center (p-center) Problem

Jovan Bjegovic 37/2021

1. Uvod

Minimum K-Center (p-center) problem predstavlja važan izazov u teoriji grafova i optimizaciji, koji se koristi za efikasno lociranje k-centara u grafu, pri čemu se minimizuje maksimalna udaljenost od bilo kog čvora do najbližeg centra. Ovaj problem ima široku primenu u oblasti transporta, rasporeda, kao i u mnogim industrijskim i tehnološkim aplikacijama gde je efikasno lociranje tačaka od kritičnog značaja. U ovom radu istražujemo različite algoritme koji rešavaju ovaj problem, sa posebnim fokusom na brzinu, tačnost i efikasnost algoritama.

2. Opis rešenja

Za rešavanje Minimum K-Center problema, implementirali smo nekoliko različitih algoritama koji variraju u pogledu računarske složenosti, preciznosti i metoda optimizacije. Algoritmi koje razmatramo uključuju:

1. **Brute-Force Algoritam**
2. **Greedy Algoritam**
3. **Local Search Algoritam**
4. **Genetski Algoritam**
5. **K-Means Heuristika**
6. **Simulirano kaljenje Algoritam**
7. **Ant Colony Optimization Algoritam**

Svi ovi algoritmi imaju za cilj da efikasno lociraju k-centre u grafu, minimizirajući maksimalnu udaljenost (radius) između centra i svih ostalih čvorova u grafu. Različite pristupe testiramo na grafovima sa različitim parametrima, kako bismo izabrali najefikasniji algoritam za rešavanje ovog problema.

2.1 Brute-Force Algoritam

Brute-force algoritam je jednostavan, ali vrlo računski zahtevan metod koji podrazumeva isprobavanje svih mogućih kombinacija k-centara u grafu. Ovaj pristup osigurava tačno rešenje jer isprobava svaku moguću konfiguraciju, međutim, brzina izvršenja drastično opada kako broj čvorova i centara raste. Ovaj metod se često koristi kao referentni model, jer pruža tačno rešenje i može se koristiti za upoređivanje sa drugim, bržim, ali manje preciznim algoritmima. U situacijama kada je broj čvorova i centara relativno mali, brute-force algoritam može biti koristan zbog svoje jednostavnosti i tačnosti.

Prednosti:

- Tačno rešenje.
- Jednostavna implementacija.

Nedostaci:

- Veoma sporo za velike grafove.
- Računarska složenost eksponencijalno raste sa brojem čvorova i centara.

2.2 Greedy Algoritam

Greedy algoritam je pristup koji rešava problem iterativnim odabirom centra na osnovu trenutne konfiguracije grafova. U svakom koraku, algoritam bira čvor koji čini najviše smanjenje maksimalne udaljenosti od bilo kojeg čvora do najbližeg centra. Iako ovaj pristup ne garantuje uvek globalno optimalno rešenje, često daje vrlo dobra rešenja u praktičnim slučajevima i značajno je brži od brute-force algoritma. Greedy pristup je jednostavan i efikasan za probleme sa velikim brojem čvorova i centara, ali ponekad može da se zaglavi u lokalnim optimumima.

Prednosti:

- Brz.
- Laka implementacija.
- Često daje dobre rezultate za većinu malih problema.

Nedostaci:

- Ne garantuje optimalno rešenje.
- Može se zaglaviti u lokalnim minimumima.

2.3 Local Search Algoritam

Local Search algoritam koristi pristup zasnovan na lokalnoj optimizaciji. Algoritam započinje sa inicijalnom konfiguracijom k-centara i zatim iterativno pokušava da poboljša ovu konfiguraciju tako što menja pozicije centara. Promene mogu biti zasnovane na susednim konfiguracijama, tj. konfiguracijama koje su samo malo drugačije od trenutne. Iako ovaj pristup može značajno poboljšati rešenje, postoji rizik da se algoritam zaglavi u lokalnim minimumima, tj. da ne pronađe globalno optimalno rešenje. Algoritam je efikasan, ali njegova uspešnost zavisi od početnih uslova i načina pretrage.

Prednosti:

- Efikasan za mnoge vrste problema.
- Dobro funkcioniše kada je problem manji ili kada su početni uslovi dobri.

Nedostaci:

- Može da se zaglavi u lokalnim minimumima.
- Nema garanciju za globalno optimalno rešenje.

2.4 Genetski Algoritam

Genetski algoritam je evolucioni algoritam koji koristi principe selekcije, ukrštanja i mutacije za pronalaženje optimalnih rešenja. U ovom algoritmu, populacija potencijalnih rešenja

evoluira kroz nekoliko generacija. Svaka generacija proizvodi nove, "bolje" pojedince (rešenja) koji imaju manji maksimalni radijus, bazirano na kriterijumima selekcije. Genetski algoritam je vrlo fleksibilan i sposoban je da pronađe dobra rešenja čak i za velike grafove. Iako genetski algoritam obično ne garantuje globalno optimalno rešenje, on je efikasan u pronalaženju vrlo dobrih rešenja za složene probleme.

Prednosti:

- Dobro funkcioniše za velike grafove.
- Može pronaći dobra rešenja u problemima sa velikim brojem varijabli.

Nedostaci:

- Nema garanciju za globalno optimalno rešenje.
- Može biti računarski intenzivan.

2.5 K-Means Heuristika

K-Means algoritam je metoda za grupisanje podataka koja koristi princip minimizacije srednje kvadratne udaljenosti između tačaka i njihovih centara (klastera). U kontekstu Minimum K-Center problema, K-Means koristi grupisanje čvorova u k-klastera, gde su centri tih klastera korišćeni kao k-centri. Ovaj algoritam je jednostavan za implementaciju i daje brze rezultate, ali može imati problema sa postavkama koje nisu dobro raspoređene. K-Means se oslanja na pretpostavku da su podaci (ili čvorovi) u relativno ravnomerno raspoređenim grupama.

Prednosti:

- Brz.
- Jednostavan za implementaciju.

Nedostaci:

- Ne može da se nosi sa nepravilno raspoređenim podacima.
- Može se zaglaviti u lokalnim minimumima.

2.6 Simulirano Kaljenje Algoritam

Simulirano kaljenje (Simulated Annealing) je stohastički algoritam koji koristi nasumične promene u konfiguraciji kako bi istražio prostor rešenja. Ovaj algoritam ima mehanizam za balansiranje između istraživanja novih rešenja i iskorišćavanja postojećih, uz pomoć parametra temperature koji se postepeno smanjuje tokom vremena. Kada je temperatura visoka, algoritam dozvoljava veće promene, dok sa smanjenjem temperature algoritam postaje precizniji i fokusira se na lokalna poboljšanja. Simulirano kaljenje može pronaći globalno optimalno rešenje, ali je njegov uspeh zavistan od parametara temperature i strategije pretrage.

Prednosti:

- Može da pronađe globalno optimalno rešenje.

- Fleksibilan u pretrazi prostora rešenja.

Nedostaci:

- Potrebno je pažljivo podešavanje parametara.
- Može biti sporo.

2.7 Ant Colony Optimization Algoritam

Ant Colony Optimization (ACO) je inspirisan ponašanjem mrava koji traže najkraće puteve do izvora hrane. Algoritam koristi kolektivnu inteligenciju kolonije mrava koja komunicira putem feromona na putu. Mravi koriste feromone da označe putove koji vode do dobrih rešenja, i tokom vremena, algoritam koristi informacije o feromonima kako bi navigirao kroz prostor rešenja. ACO je vrlo efikasan za rešavanje složenih optimizacijskih problema, posebno onih sa velikim brojem varijabli. U kontekstu Minimum K-Center problema, ACO se koristi za pronalaženje optimalnih k-centara na osnovu ponašanja kolonije mrava.

Prednosti:

- Efikasan za složene probleme.
- Koristi kolektivnu inteligenciju za pronalaženje optimalnih rešenja.

Nedostaci:

- Potrebno je pažljivo podešavanje parametara.
- Računarski intenzivan za vrlo velike grafove.

3. Rezultati

Napomena: Parametri za testiranje algoritama

Testiranje svih sedam algoritama implementiranih u ovom radu izvršeno je na grafovima sa različitim brojem čvorova, brojem grafova, vrednostima za k (broj centara) i opsegom pozicija čvorova. Parametri korišćeni za generisanje testnih grafova su sledeći:

- **Prvi graf**:
  - Broj čvorova: 10
  - Broj centara: 4
  - Opseg pozicija: (0, 10)
- **Drugi graf**:
  - Broj čvorova: 15
  - Broj centara: 6
  - Opseg pozicija: (0, 20)
- **Treći graf**:
  - Broj čvorova: 20
  - Broj centara: 8
  - Opseg pozicija: (0, 30)

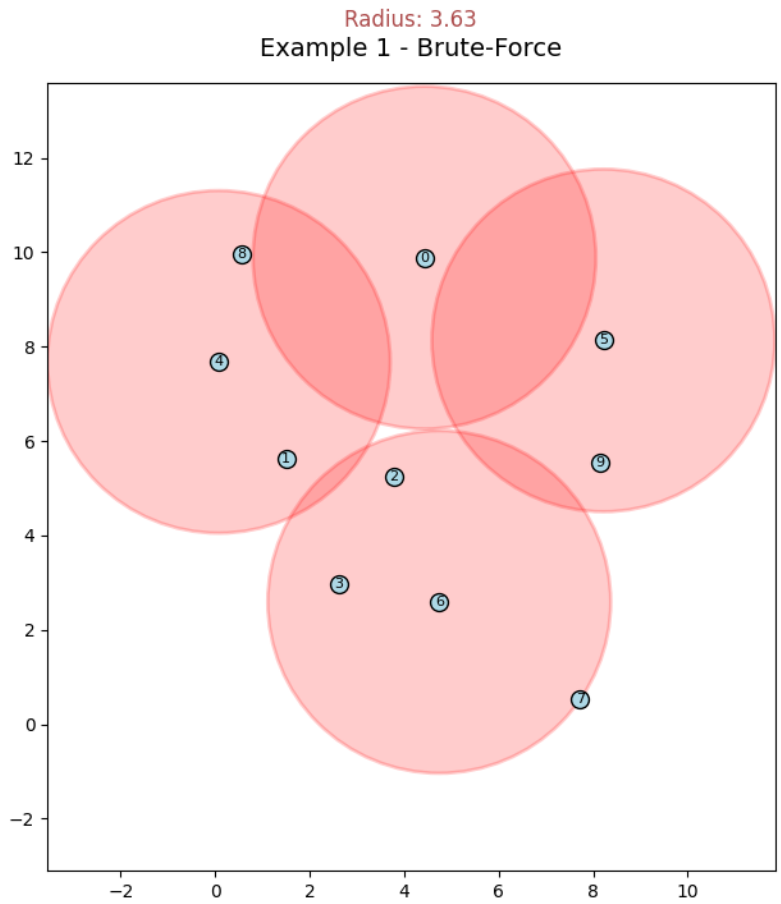
Svaki graf je generisan sa slučajnim pozicijama čvorova unutar datih opsega pozicija. Sledeći primeri prikazuju konkretne pozicije čvorova za svaki skup:

- **Prvi skup pozicija**:
  - [(4.428, 9.886), (1.499, 5.612), (3.778, 5.241), (2.626, 2.982), (0.078, 7.678), (8.214, 8.132), (4.742, 2.592), (7.724, 0.524), (0.561, 9.949), (8.132, 5.529)]
- **Drugi skup pozicija**:
  - [(9.877, 15.019), (8.461, 0.267), (7.928, 5.218), (1.693, 17.310), (18.529, 4.300), (4.056, 2.441), (7.274, 14.513), (12.056, 4.176), (6.776, 3.256), (7.023, 11.880)]
- **Treći skup pozicija**:
  - [(24.613, 4.480), (10.056, 27.268), (17.605, 27.556), (19.818, 4.807), (11.632, 16.746), (19.048, 29.152), (8.293, 13.965), (20.602, 19.268), (22.415, 3.592), (21.563, 23.743)]

Ovi parametri predstavljaju primere sa kojima cemo ćemo raditi radi testiranja efikasnosti i tačnosti algoritama pri rešavanju problema Minimum K-Center.

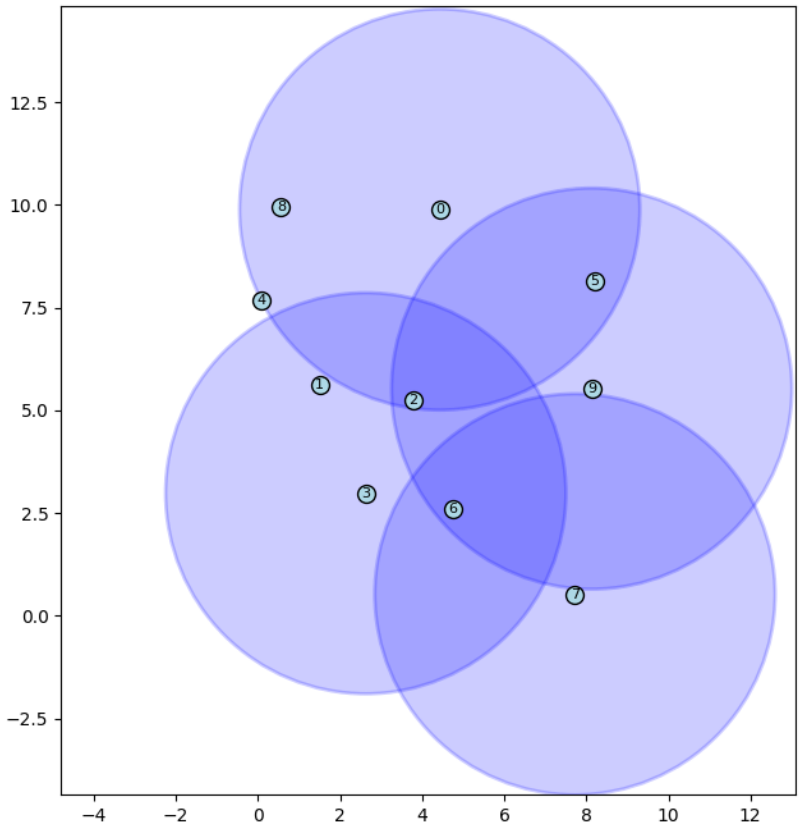
Testirali smo sve algoritme na različitim grafovima sa različitim parametrima. Algoritmi su ocenjeni na osnovu vremena izvršenja i preciznosti (minimalnog radijusa).

3.1 Testiranje prvog grafa



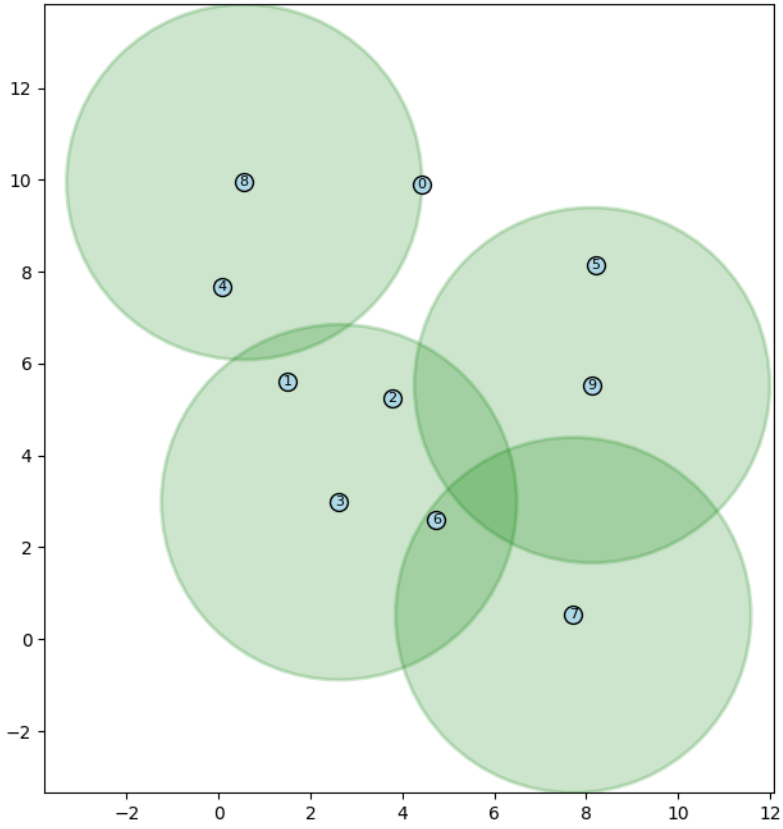
Radius: 4.88

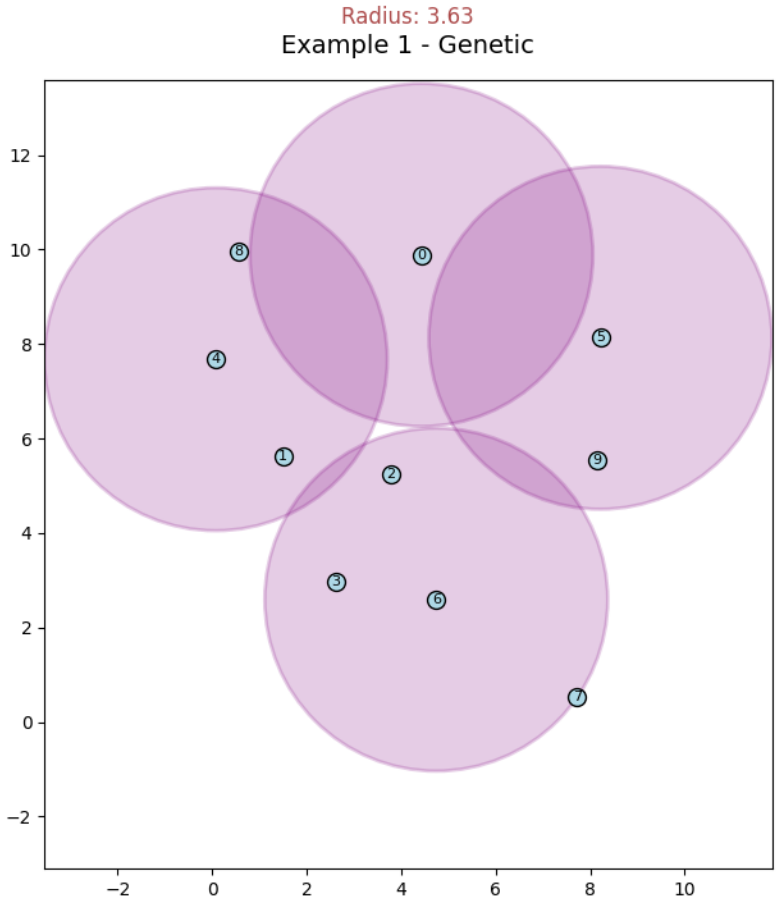
Example 1 - Greedy



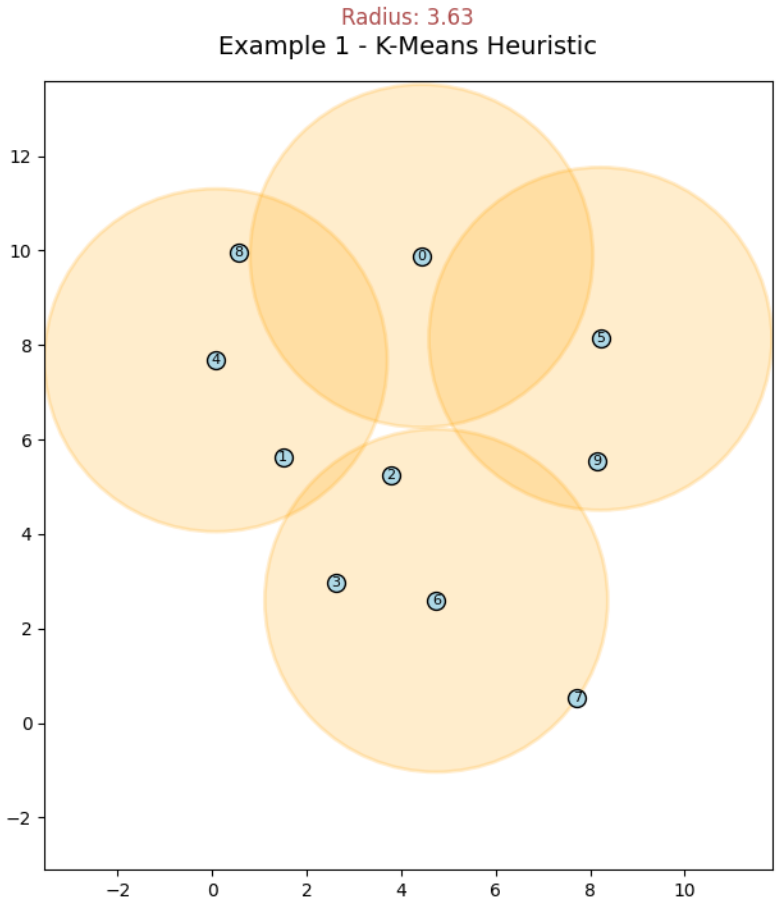
Radius: 3.87

Example 1 - Local Search



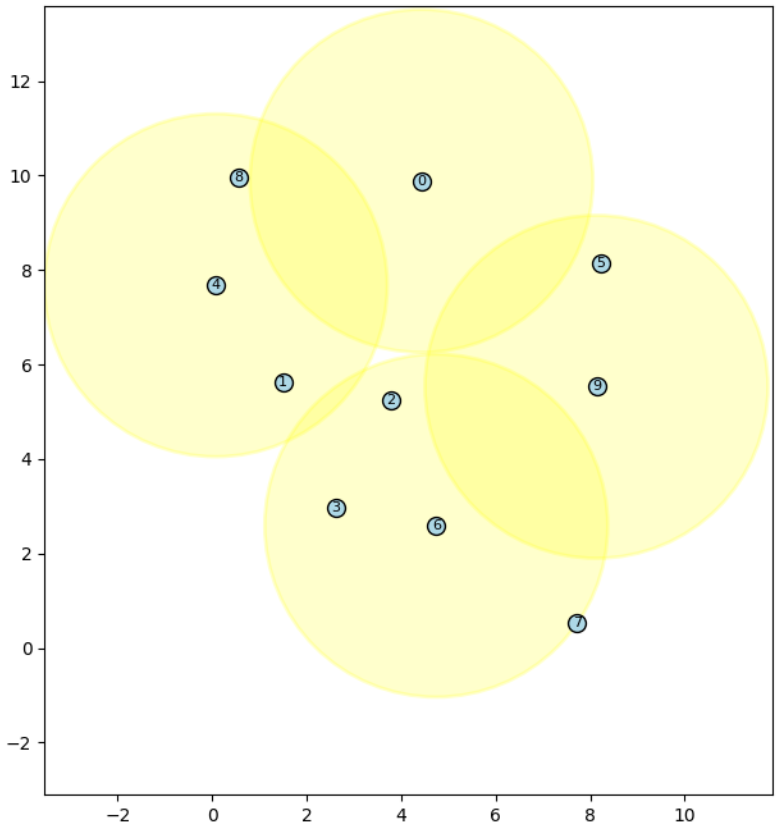

Radius: 3.63
Example 1 - Genetic
12
10
8
6
4
2
0
−2
−2
0
2
4
6
8
10
0
1
2
3
4
5
6
7
8
9

a


Radius: 3.63
Example 1 - K-Means Heuristic
12
10
8
6
4
2
0
−2
−2
0
2
4
6
8
10
0
1
2
3
4
5
6
7
8
9

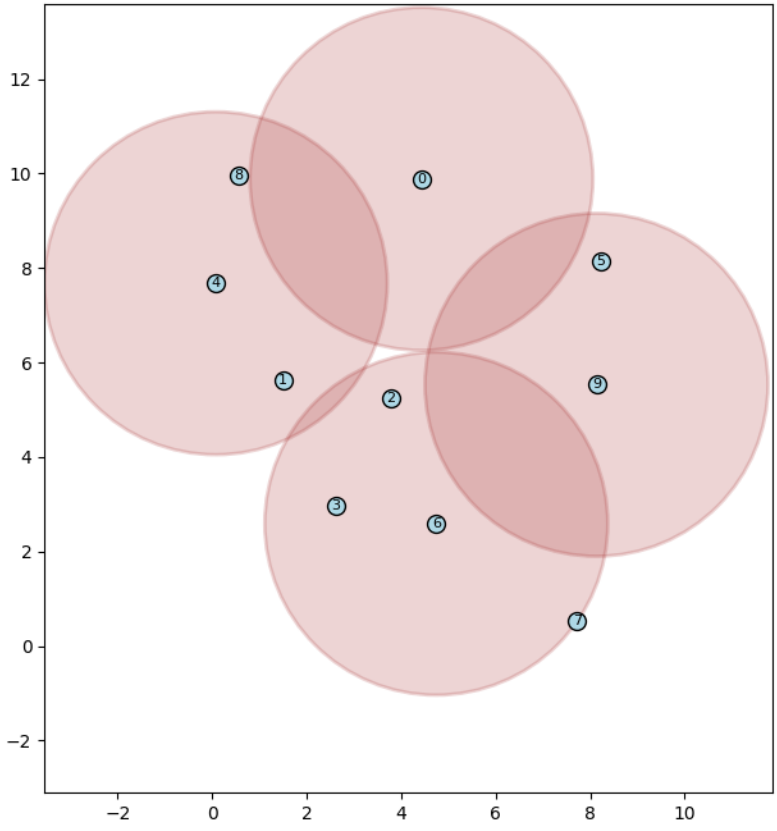
Radius: 3.63

Example 1 - Simulated Annealing

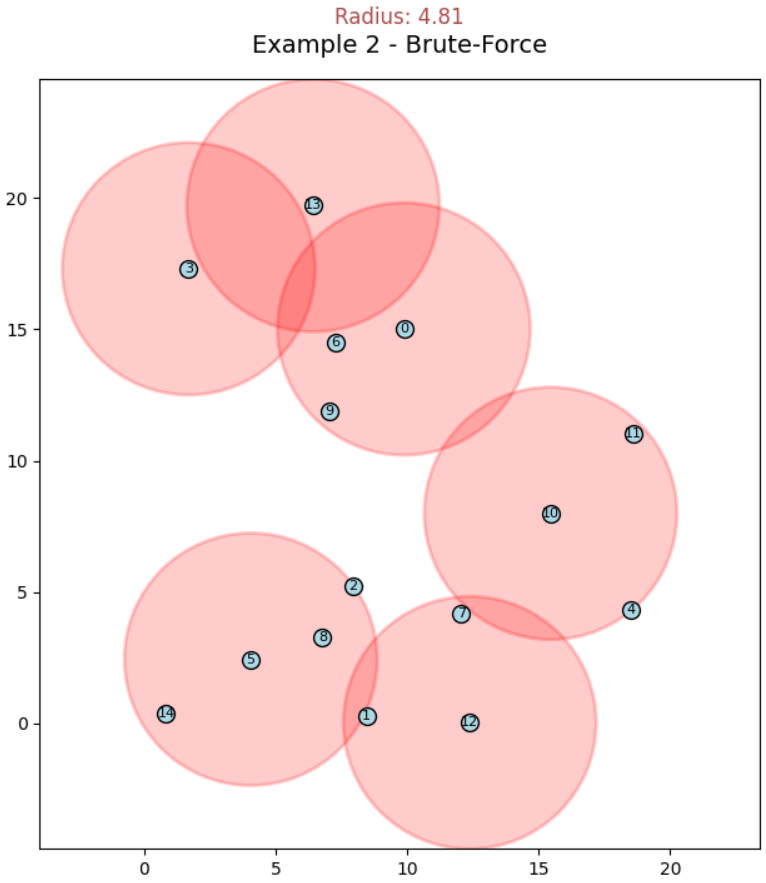


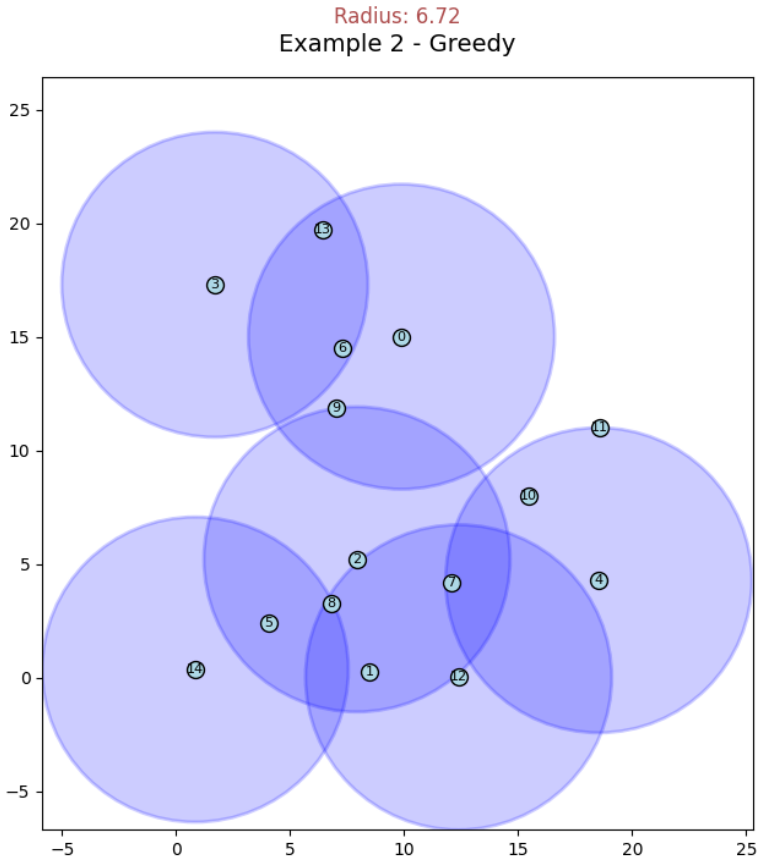
Radius: 3.63

Example 1 - Ant Colony



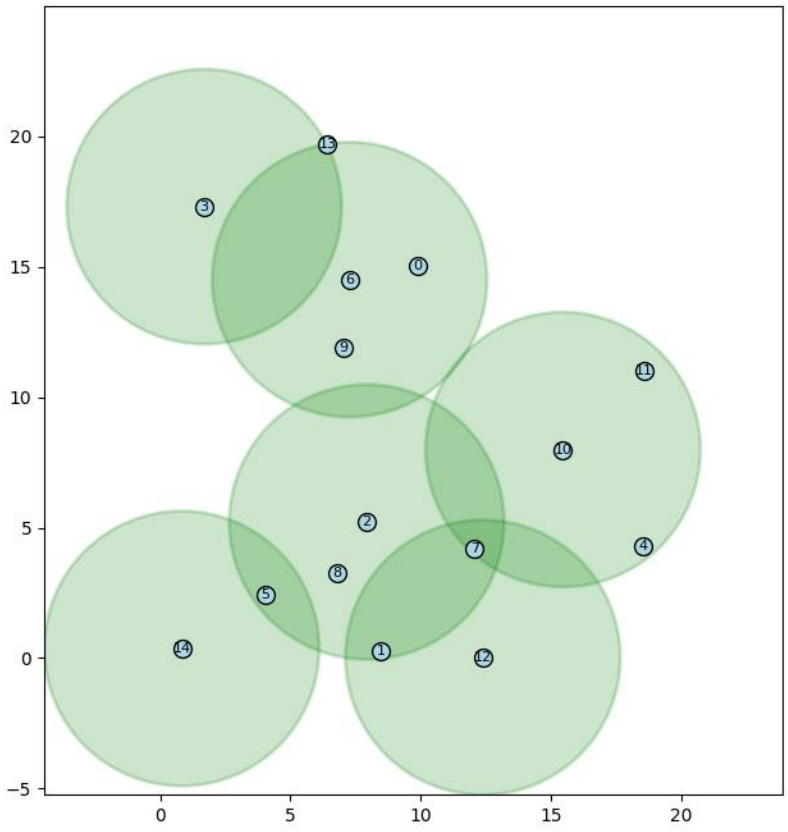
3.2 Testiranje drugog grafa





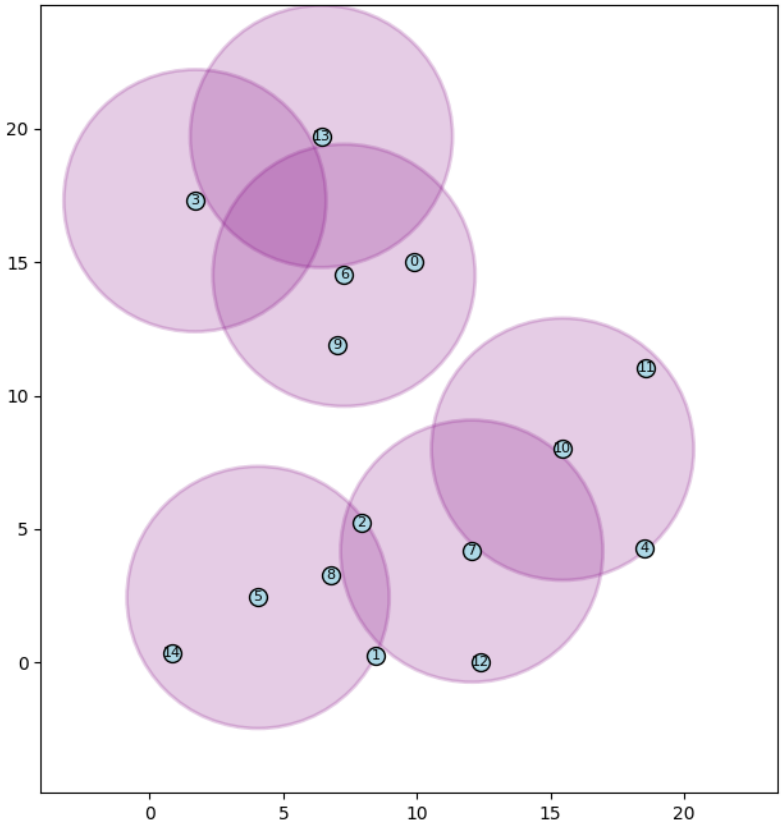
Radius: 5.27

Example 2 - Local Search



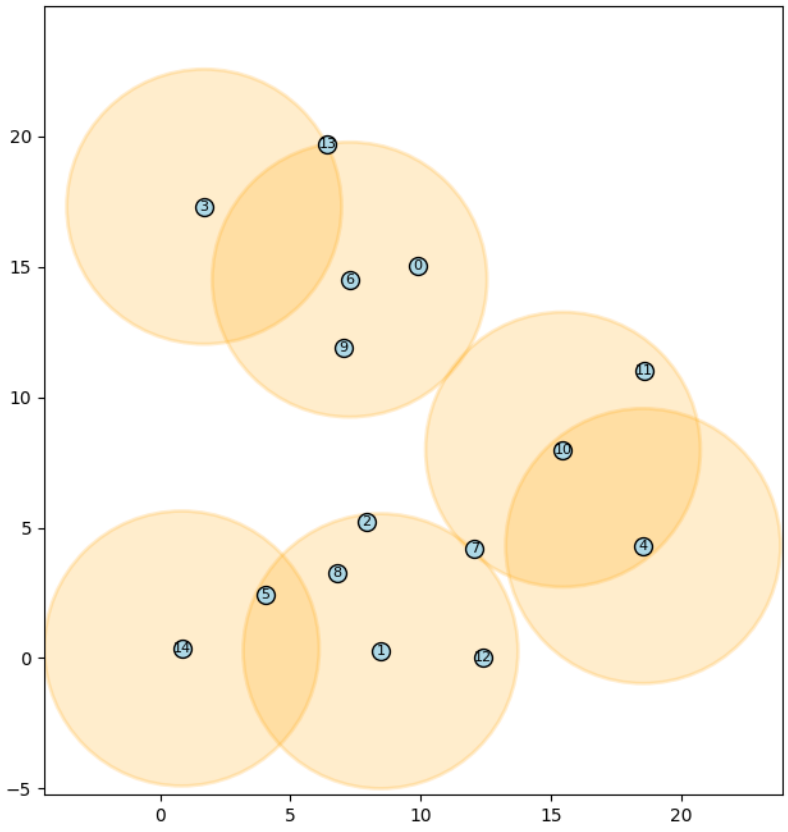
Radius: 4.91

Example 2 - Genetic



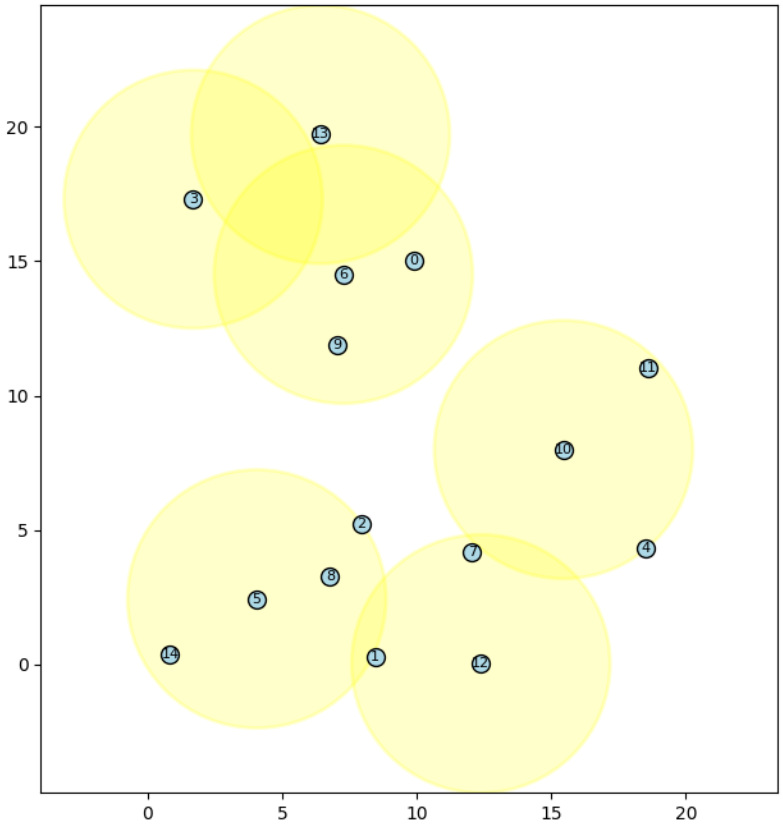
Radius: 5.27

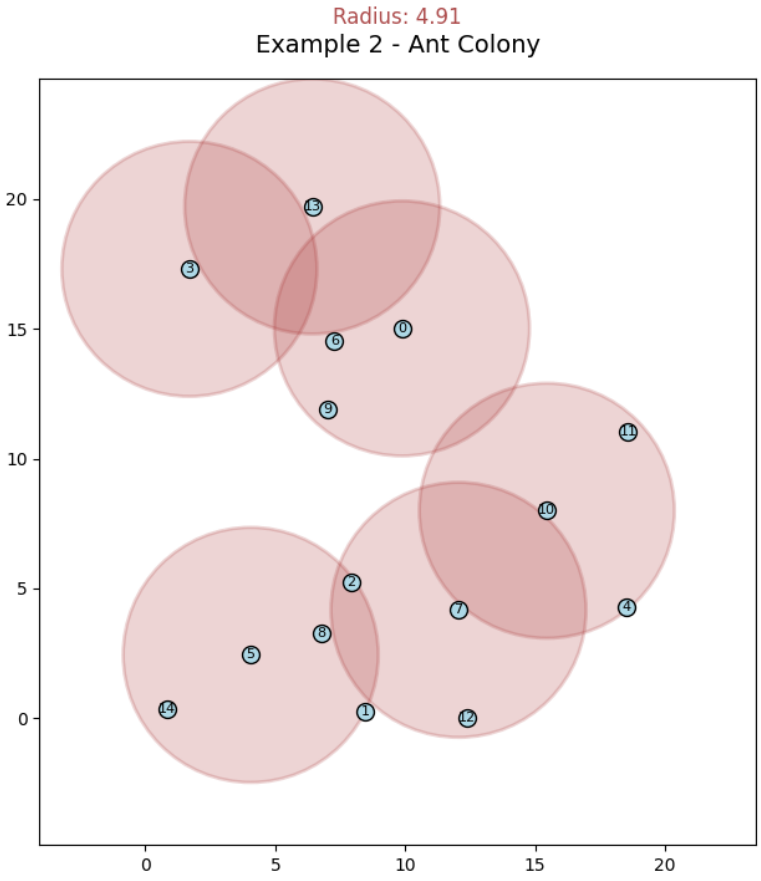
Example 2 - K-Means Heuristic



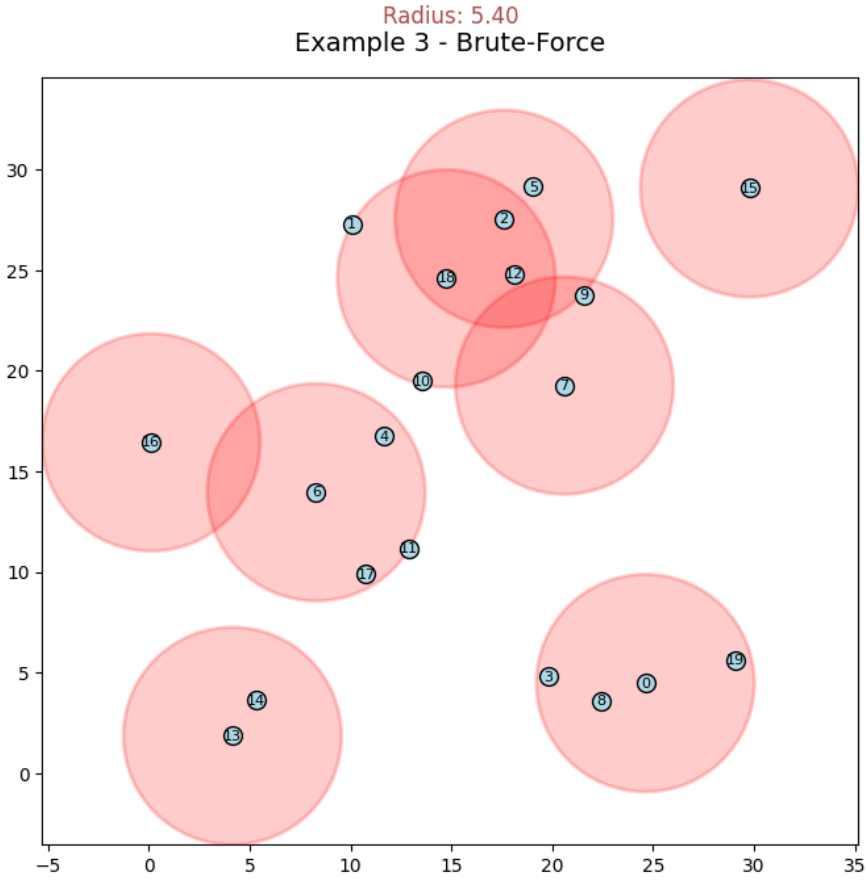
Radius: 4.81

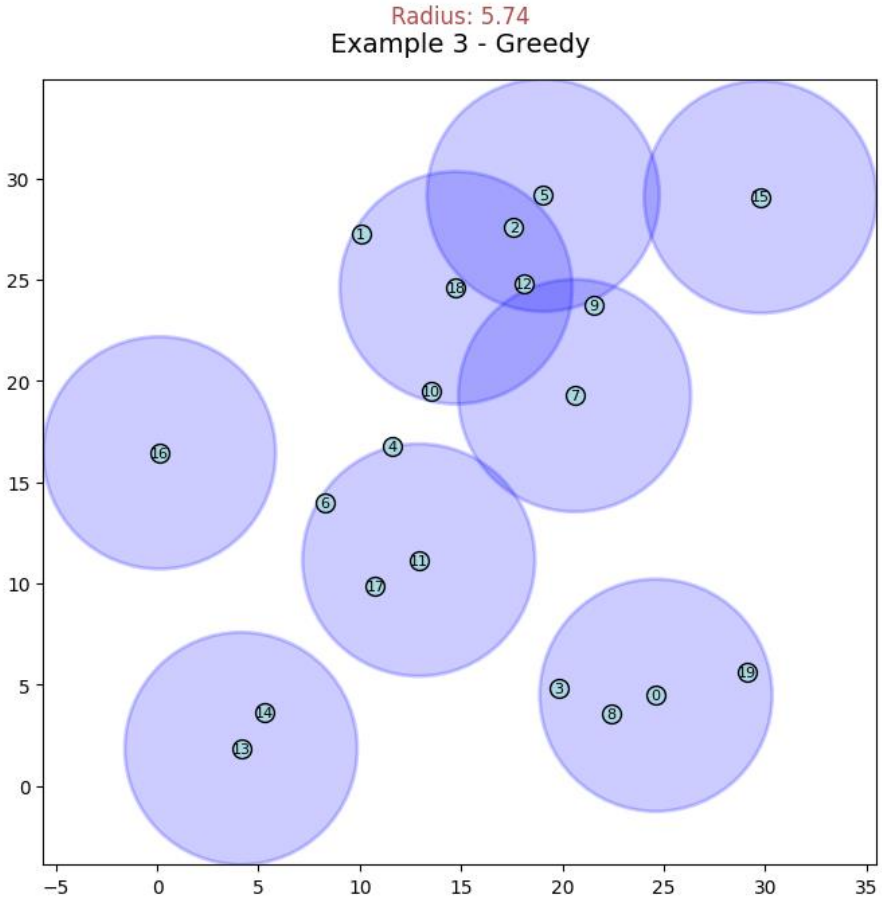
Example 2 - Simulated Annealing




Radius: 4.91
Example 2 - Ant Colony
20
15
10
5
0
0
5
10
15
20
13
3
0
6
9
11
10
2
7
4
8
5
14
1
12

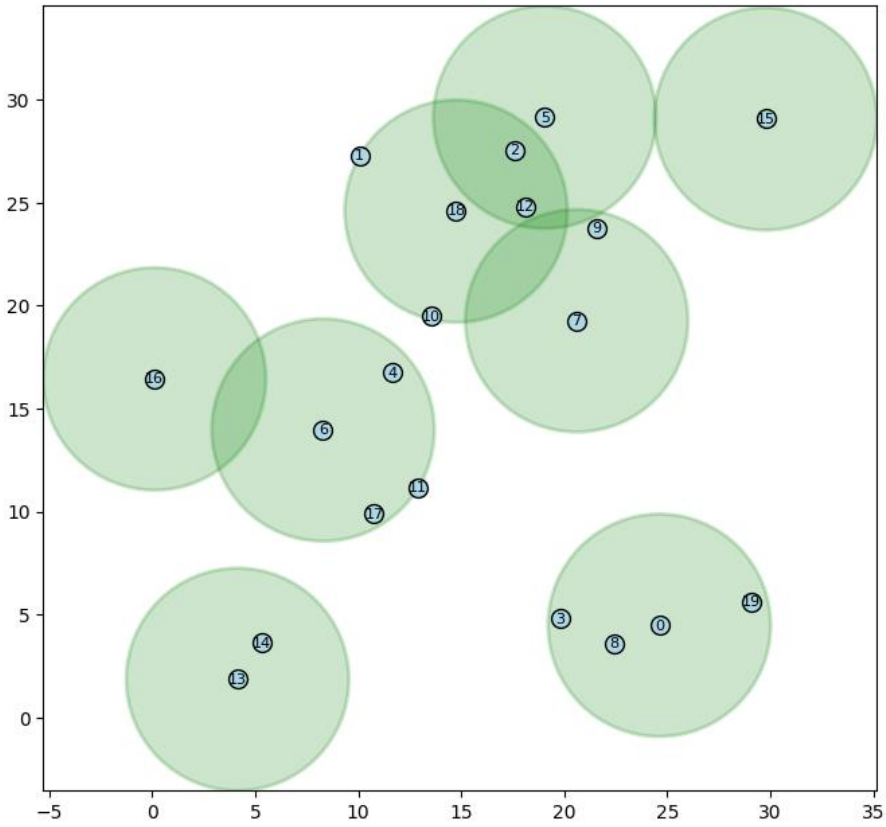
3.3 Testiranje treceg grafa





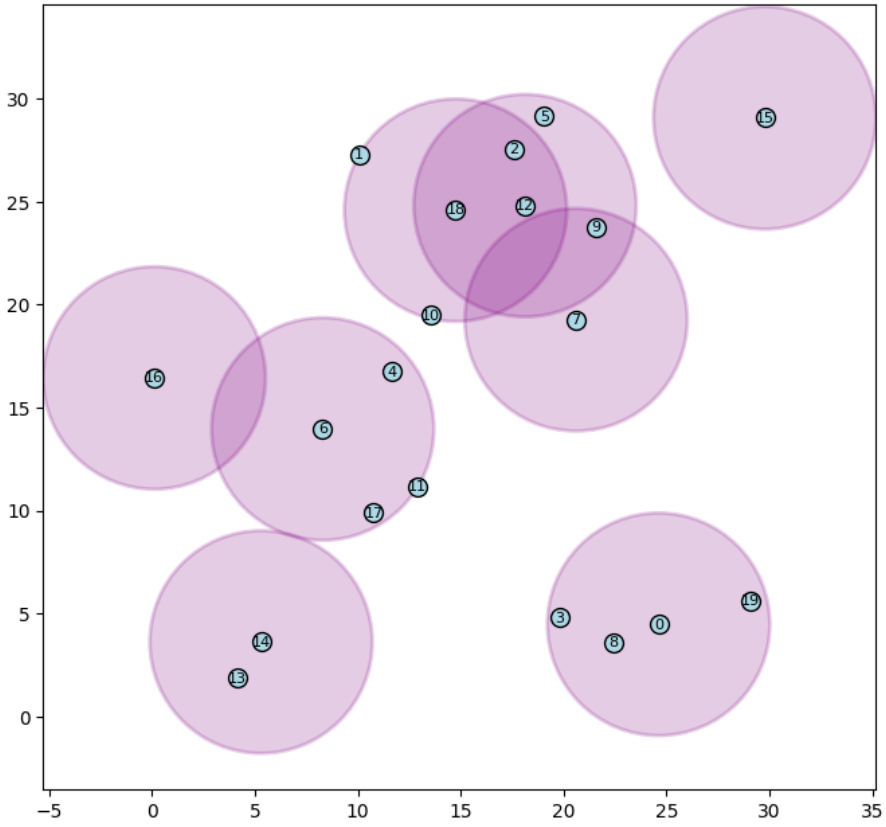
Radius: 5.40

Example 3 - Local Search



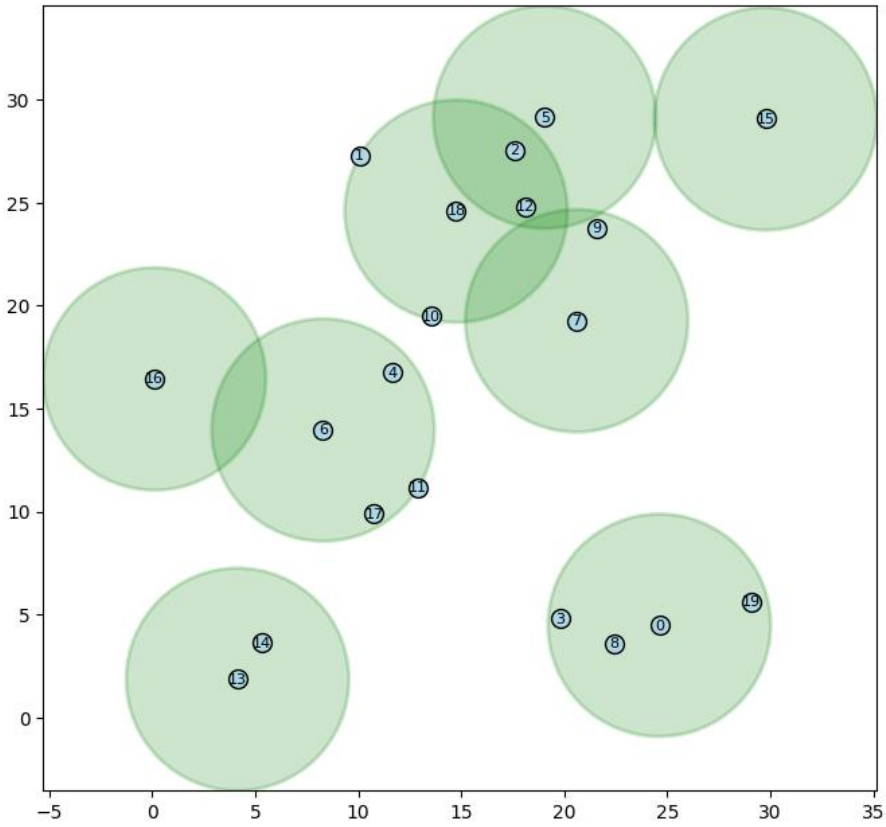
Radius: 5.40

Example 3 - Genetic



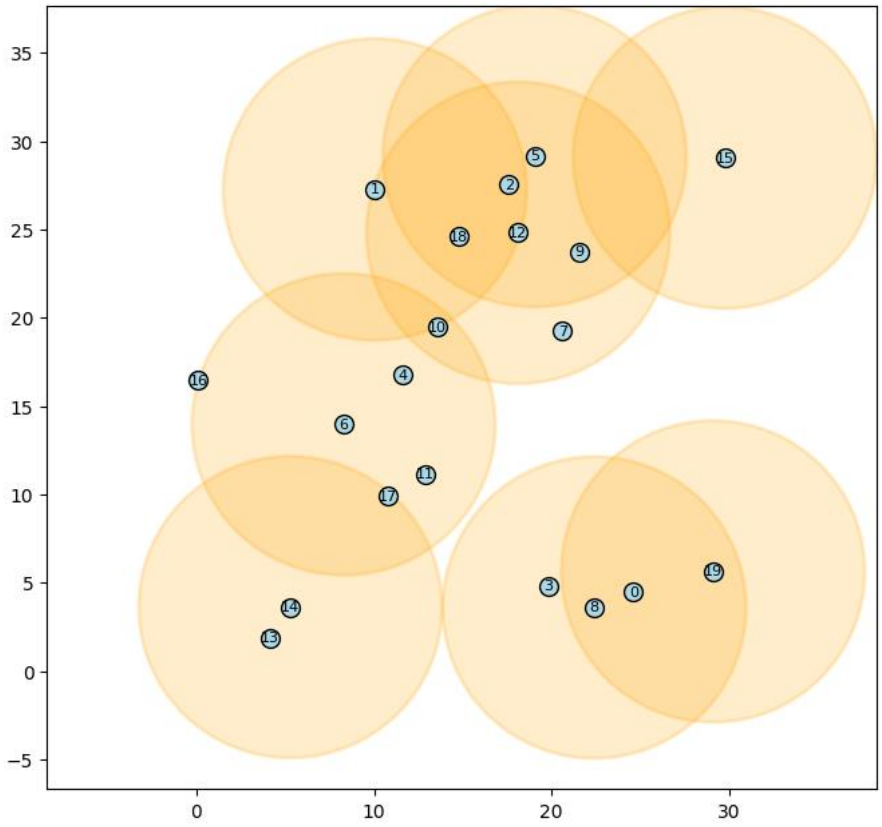
Radius: 5.40

Example 3 - Local Search



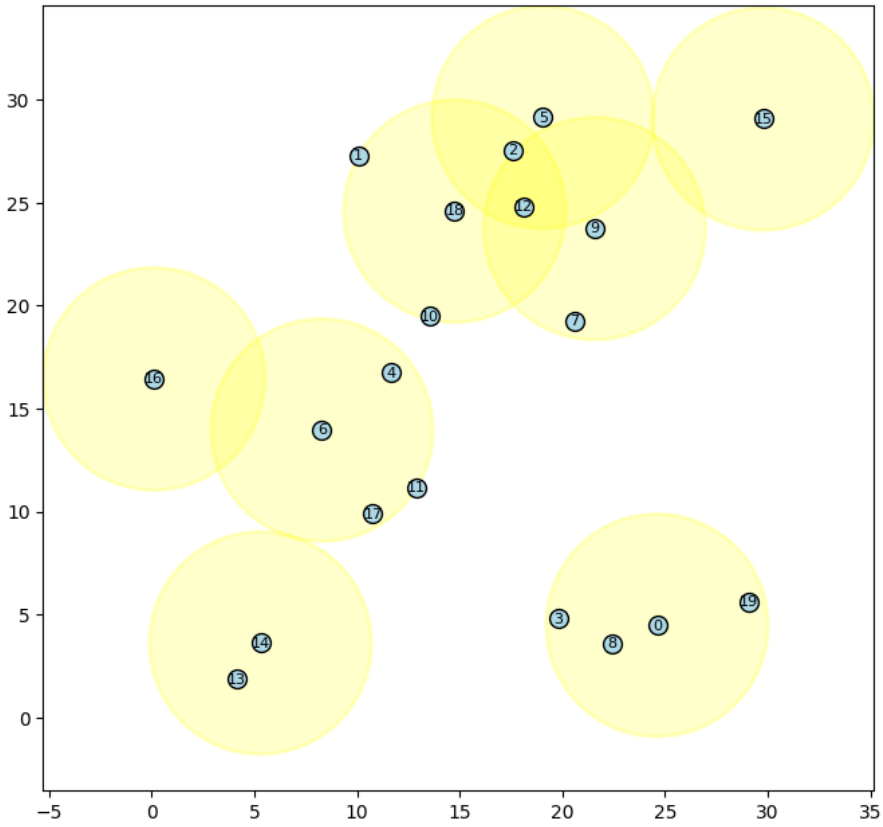
Radius: 8.55

Example 3 - K-Means Heuristic



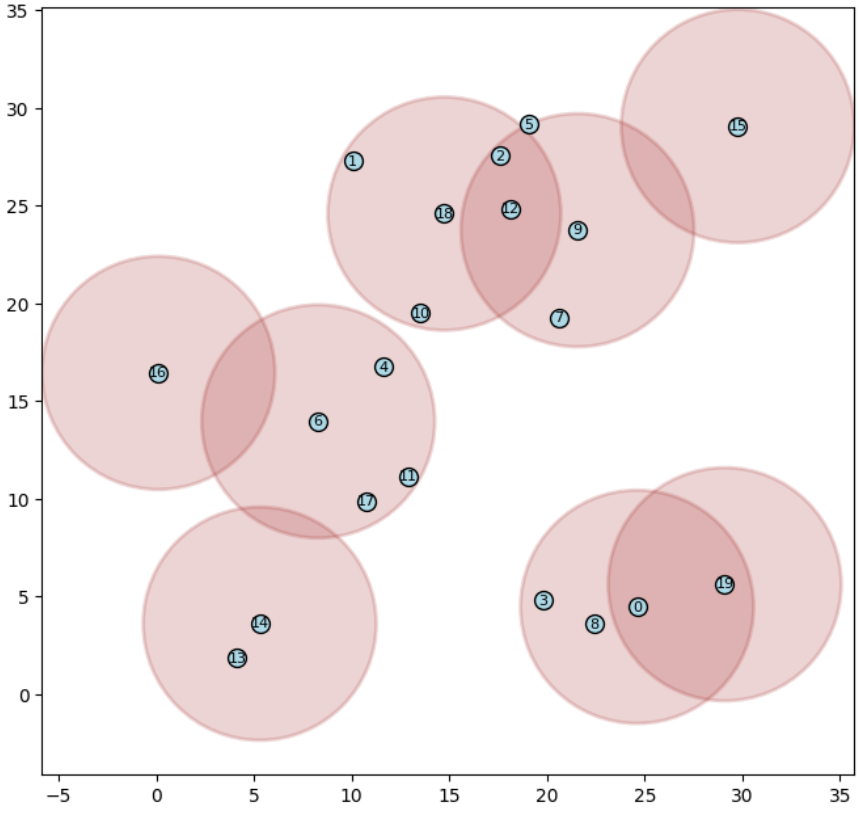
Radius: 5.40

Example 3 - Simulated Annealing

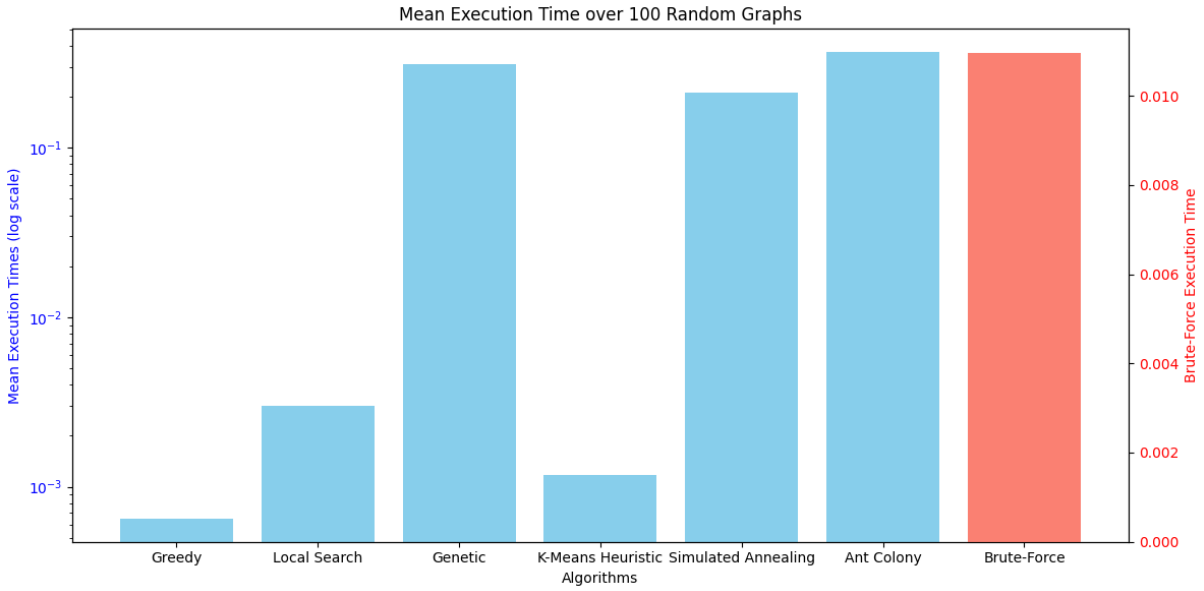


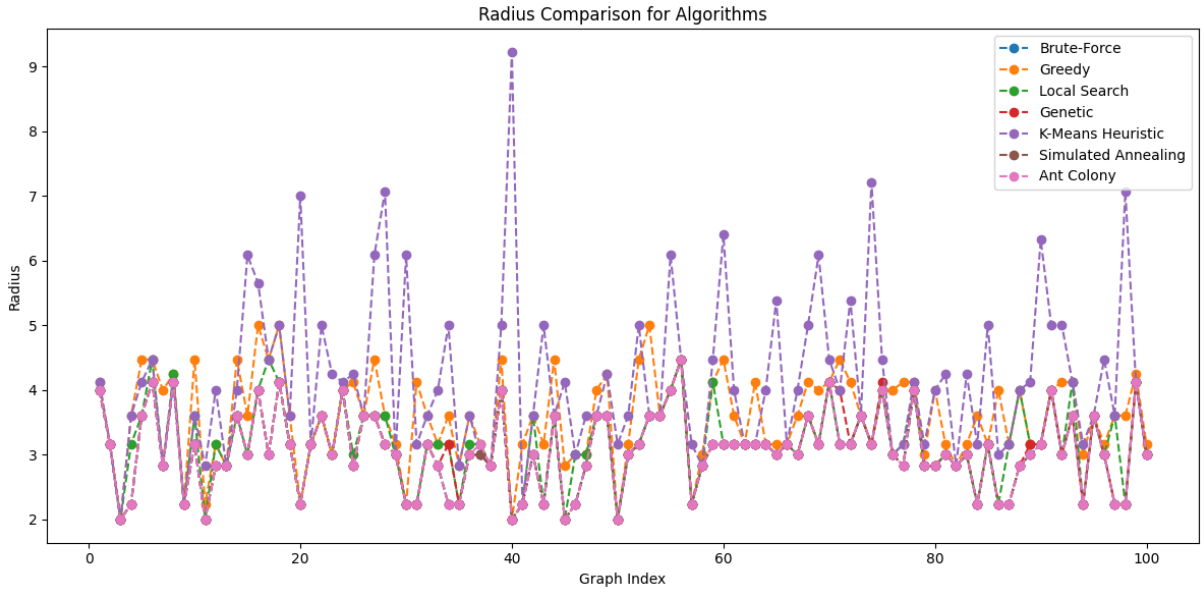
Radius: 5.97

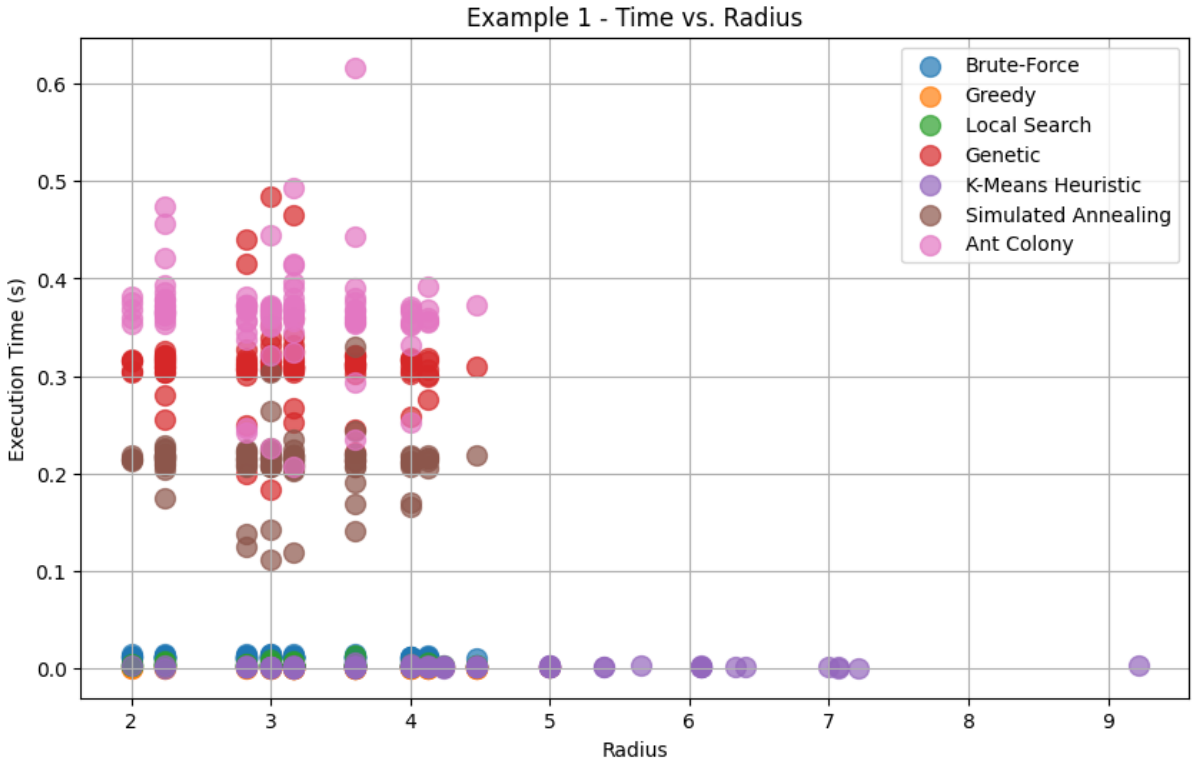
Example 3 - Ant Colony



3.4 Testiranje 100 različitih grafova prvog tipa sa svim algoritmima



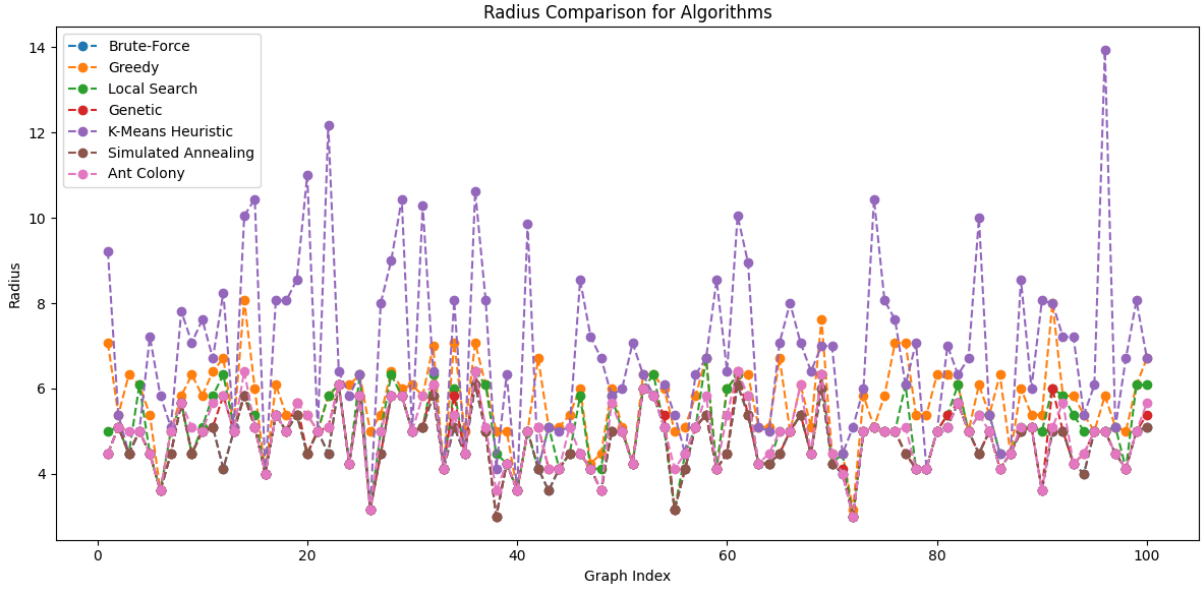


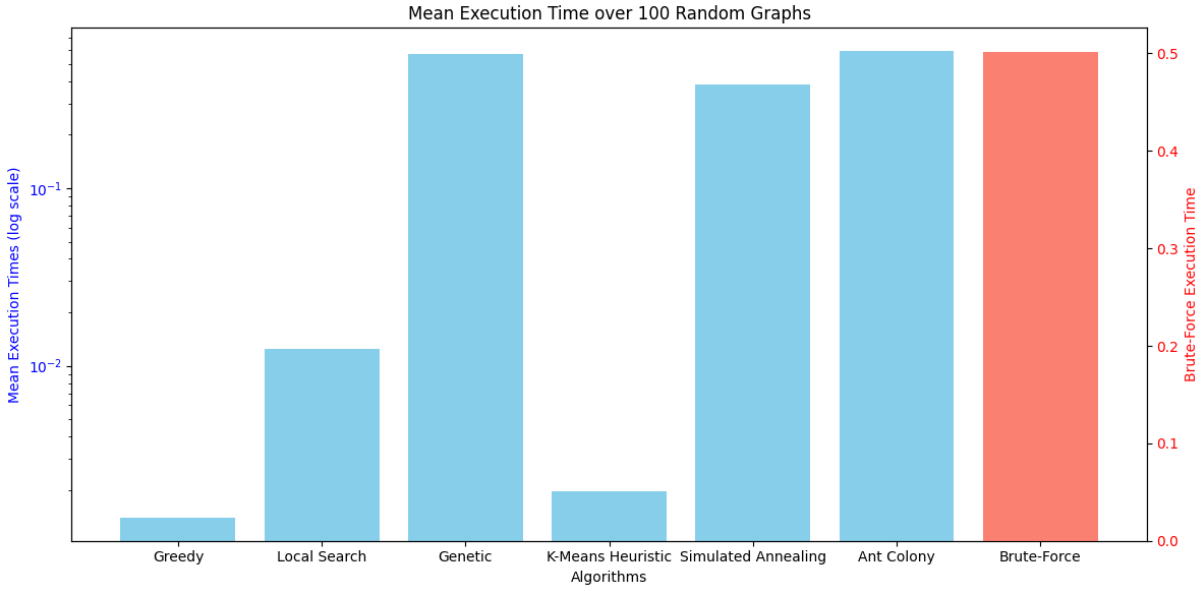


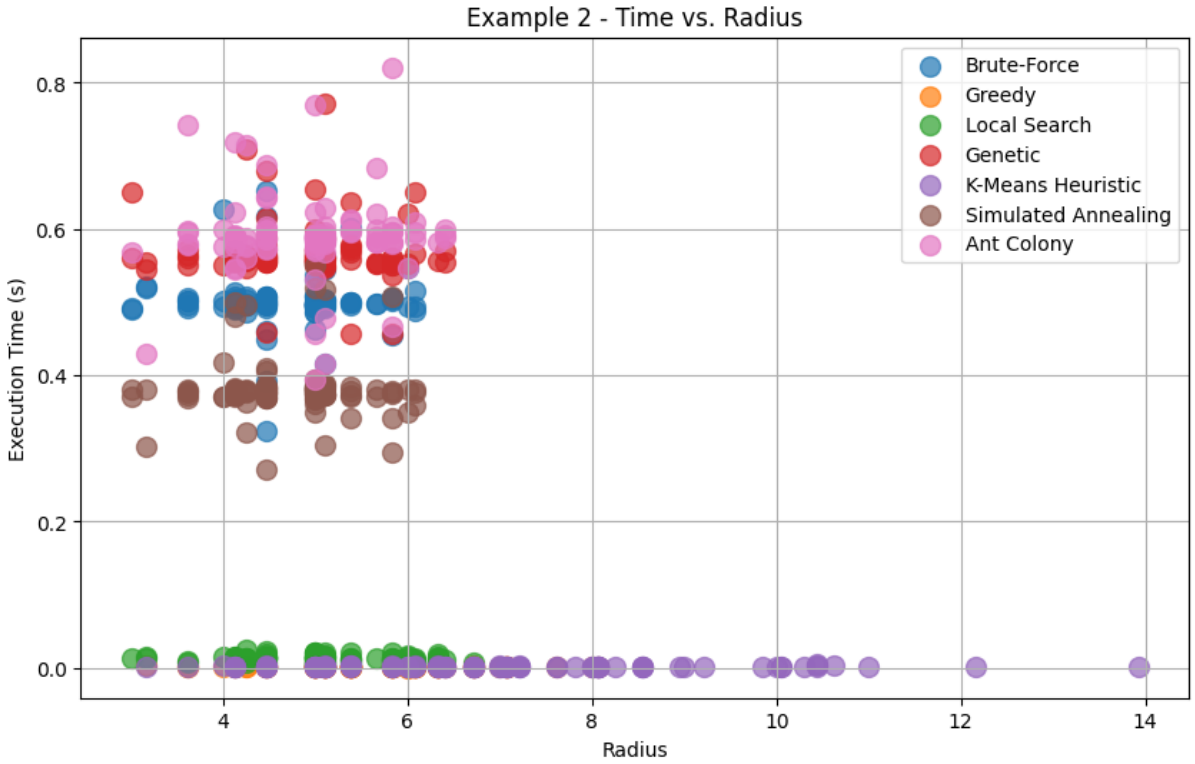
Example 1 Algorithm Performance Summary:
Brute-Force: Min Radius = 2.0000, Mean Radius = 3.0659, Mean Execution Time = 0.0110s, Score = 0.0774
Greedy: Min Radius = 2.0000, Mean Radius = 3.5866, Mean Execution Time = 0.0007s, Score = 0.4803
Local Search: Min Radius = 2.0000, Mean Radius = 3.1942, Mean Execution Time = 0.0030s, Score = 0.1766
Genetic: Min Radius = 2.0000, Mean Radius = 3.0864, Mean Execution Time = 0.3103s, Score = 0.0932
K-Means Heuristic: Min Radius = 2.0000, Mean Radius = 4.1584, Mean Execution Time = 0.0012s, Score = 0.9226
Simulated Annealing: Min Radius = 2.0000, Mean Radius = 3.0659, Mean Execution Time = 0.2104s, Score = 0.0874
Ant Colony: Min Radius = 2.0000, Mean Radius = 3.0676, Mean Execution Time = 0.3656s, Score = 0.0786

Best Algorithm for Example 1 (Weighted by Time and Radius): **Brute-Force**

3.5 Testiranje 100 različitih grafova drugog tipa sa svim algoritmima



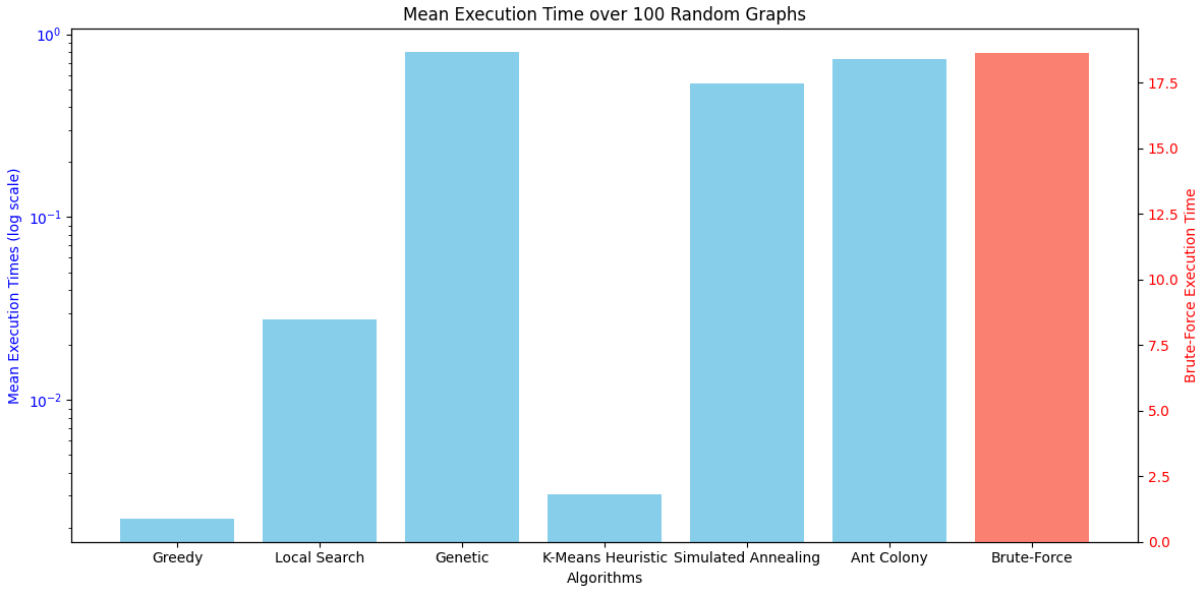


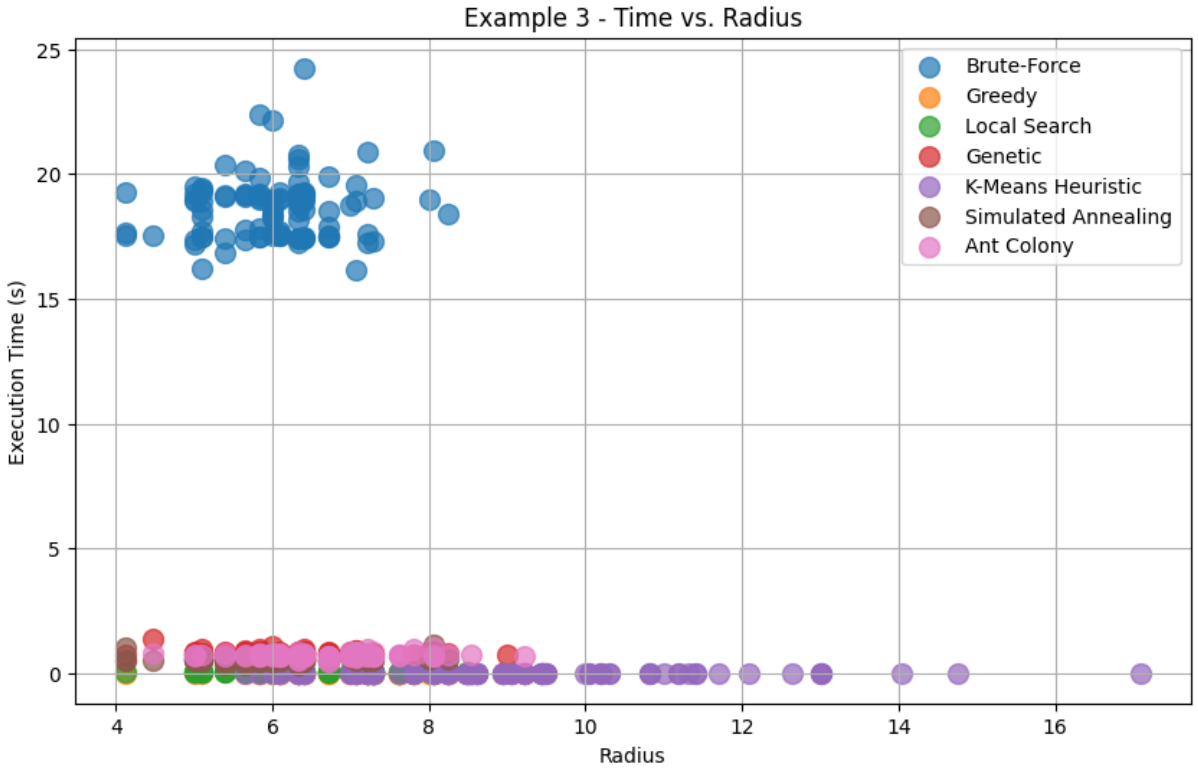


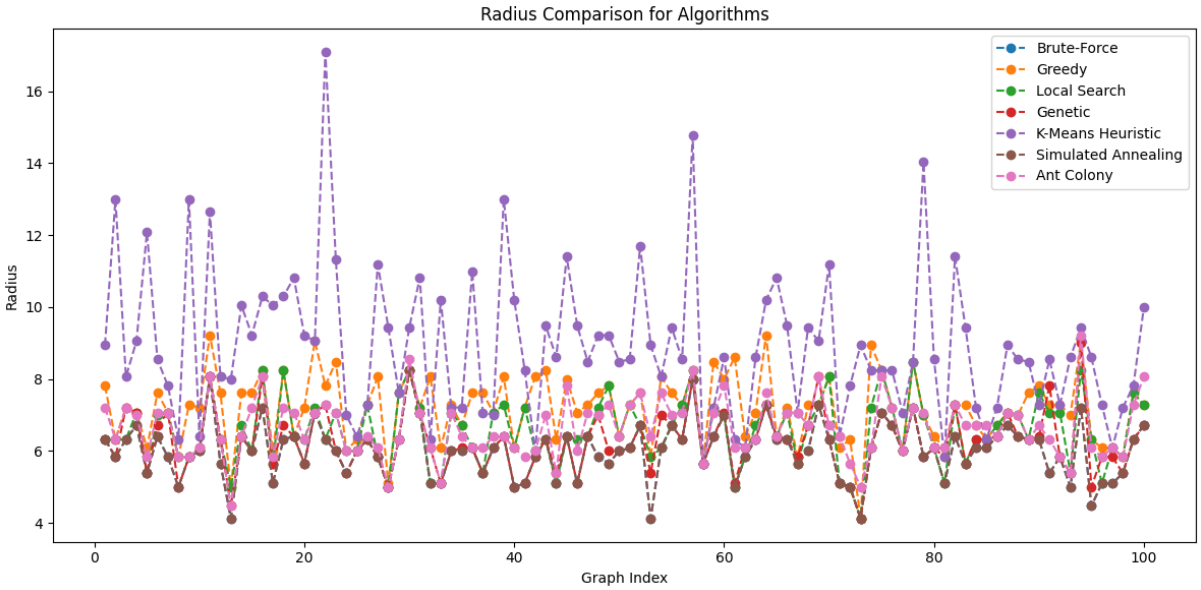
Example 2 Algorithm Performance Summary:
Brute-Force: Min Radius = 3.0000, Mean Radius = 4.7247, Mean Execution Time = 0.5011s, Score = 0.0528
Greedy: Min Radius = 3.1623, Mean Radius = 5.6675, Mean Execution Time = 0.0014s, Score = 0.4467
Local Search: Min Radius = 3.0000, Mean Radius = 5.0720, Mean Execution Time = 0.0125s, Score = 0.1916
Genetic: Min Radius = 3.0000, Mean Radius = 4.8097, Mean Execution Time = 0.5671s, Score = 0.0793
K-Means Heuristic: Min Radius = 3.1623, Mean Radius = 6.8594, Mean Execution Time = 0.0020s, **Score = 0.9572**
Simulated Annealing: Min Radius = 3.0000, Mean Radius = 4.7247, Mean Execution Time = 0.3822s, Score = 0.0428
Ant Colony: Min Radius = 3.0000, Mean Radius = 4.9327, Mean Execution Time = 0.5898s, Score = 0.1319

Best Algorithm for Example 2 (Weighted by Time and Radius): **Simulated Annealing**

3.6 Testiranje 100 različitih grafova trećeg tipa sa svim algoritmima







Example 3 Algorithm Performance Summary:
Brute-Force: Min Radius = 4.1231, Mean Radius = 6.0170, Mean Execution Time = 18.6281s
Greedy: Min Radius = 4.1231, Mean Radius = 7.1845, Mean Execution Time = 0.0022s
Local Search: Min Radius = 4.1231, Mean Radius = 6.5487, Mean Execution Time = 0.0277s
Genetic: Min Radius = 4.1231, Mean Radius = 6.2468, Mean Execution Time = 0.7982s
K-Means Heuristic: Min Radius = 5.6569, Mean Radius = 8.9362, Mean Execution Time = 0.0031s
Simulated Annealing: Min Radius = 4.1231, Mean Radius = 6.0170, Mean Execution Time = 0.5378s
Ant Colony: Min Radius = 4.4721, Mean Radius = 6.6438, Mean Execution Time = 0.7281s

Best Algorithm for Example 3 (Weighted by Time and Radius): **Simulated Annealing**

4. Zaključak

U ovom radu sam testirao sve algoritme na 3 vrste grafova i rezultati su sledeći:

- Brute-force algoritam koji prolazi kroz sva rešenja i među njima traži najbolje nalazi najmanji radijus u najkraćem roku nad malim grafovima(u mom slučaju nad slučajnim grafovima sa 10 čvorova i 4 centra). Ali kada su u pitanju veći grafovi, algoritam će uvek naći najbolje rešenje ali po cenu mnogo većeg vremena od ostalih algoritama, što ga čini neefikasnim.
- Algoritmi Greedy i K-means heuristic su ubedljivo najbrži algoritmi, ali iako sa malim grafovima daju jako dobre rezultate, kada se broj čvorova i centara samo malo poveća nisu u stanju da nađu minimalni radijus ni blizu kao ostali algoritmi.
- Genetski i Ant colony algoritam su jako efikasni algoritmi kako za male tako i za ogromne grafove radeći za veoma kratko vreme. Genetski algoritam je malo bolji i nekad približno dostiže brute-force algoritam po efikasnom nalaženju minimalnog radijusa, a s obzirom na malo vreme izvršavanja, mnogo je korisniji.
- Definitivno najbolji i najefikasniji algoritam je simulated anealing algoritam, odnosno algoritam simuliranog kaljenja. Sposoban i za male i za velike grafove, brži i efikasniji od genetskog i ant colonz algoritama, algoritam simuliranog kaljenja je najbolji algoritam za pronalaženje minimalonog broja k-centra.

5. Literatura

- **Lima, A., Rodrigues, B., Wang, F., & Xu, Z.** (2004). k-Center problems with minimum coverage. *Computers & Operations Research* https://www.researchgate.net/publication/220149263_k-Center_problems_with_minimum_coverage
- **Zhang, X., & Wang, Z.** (2017). *Solving the p-Center Problem Using Simulated Annealing Algorithm*. Computers & Operations Research, https://www.researchgate.net/publication/220857373_A_Simulated_Annealing_Algorithm_for_the_Circles_Packing_Problem
- **Cornell University.** (2009). *CS 1114: Introduction to Computer Science: Lecture 23 - Graph Algorithms* [PDF]. Cornell University. https://www.cs.cornell.edu/courses/cs1114/2009sp/lectures/CS1114-lec23.pdf
- **Dorigo, M., & Stützle, T.** (2004). *Ant Colony Optimization* [PDF]. https://people.idsia.ch/~luca/aco2004.pdf