k-Chinese Postman Problem (k-CPP)

Uvod

Problem poznat kao k-Chinese Postman Problem (k-CPP) spada u oblast kombinatorne optimizacije i bavi se pronalaženjem više ciklusa u grafu tako da svi zajedno pokriju sve njegove grane.
Za razliku od klasičnog Chinese Postman problema, u kome jedan poštar obilazi sve grane grafa uz minimalnu dužinu ture, u k-CPP problemu postoji više poštara (k) koji dele posao, a cilj je da se raspodela izvrši što ravnomernije.

U varijanti Min–Max k-Chinese Postman problema, zadatak je da se minimizuje maksimalna dužina neke od k ruta, kako bi sve rute bile što ujednačenije po dužini.
Ovaj problem je NP-težak, što znači da se ne može rešiti tačno za veće instance u razumnom vremenu, pa se u praksi koriste različite heurističke i metaheurističke metode.

Zbog svoje složenosti i široke primene, problem se često koristi kao model u logistici, planiranju ruta, raspodeli zadataka u mrežama senzora i mobilnoj robotici.
U ovom radu implementirane su dve različite heuristike – Cycle-Trade algoritam i Genetski algoritam.

Genetski algoritam (GA)

Genetski algoritam (GA) predstavlja metaheurističku metodu inspirisanu procesima evolucije u prirodi.

Zasniva se na ideji da se skup mogućih rešenja problema posmatra kao populacija jedinki, koja tokom više generacija evoluira prema boljim rešenjima pomoću operatora selekcije, ukrštanja i mutacije.

Svaka jedinka u populaciji predstavlja jedno moguće rešenje problema i kodira se u obliku niza podataka koji se naziva hromozom.

Tokom svake iteracije (generacije) sprovodi se sledeći proces:

1. **Procena kvaliteta (fitnes funkcija)** – za svaku jedinku izračunava se vrednost koja pokazuje koliko je dobro to rešenje.
2. **Selekcija** – biraju se najuspešniji pojedinci koji će učestvovati u formiranju nove generacije.
3. **Ukrštanje (crossover)** – odabranim jedinkama se kombinuju delovi hromozoma kako bi nastala nova, potencijalno bolja rešenja.
4. **Mutacija** – nasumično se menja mali deo hromozoma da bi se održala raznolikost populacije i izbeglo prerano konvergiranje.
5. **Zamena generacija** – nova populacija postaje osnova za sledeći ciklus evolucije.

Ovaj proces se ponavlja dok se ne dostigne zadati broj generacija ili dok se rešenja ne stabilizuju.

Primena GA na Min-Max k-Chinese Postman problem

U ovom projektu, genetski algoritam se koristi za pronalaženje **najbolje podele grana grafa na k ruta** koje zajedno pokrivaju sve grane, pri čemu se minimizuje **maksimalna dužina** jedne rute.

- ➢ **Hromozom** predstavlja redosled obilaska svih grana grafa, zajedno sa oznakama mesta gde se prave „rezovi” između ruta.
- ➢ **Fitnes funkcija** izračunava dužine svih k ciklusa i vrednuje rešenje prema maksimalnoj dužini rute (cilj je da ona bude što manja).
- ➢ **Selekcija** se vrši turnirskim pristupom – biraju se uspešnija rešenja koja češće učestvuju u stvaranju potomaka.
- ➢ **Ukrštanje** se realizuje tako da se zadrži poredak grana, ali i deo strukture od roditelja (Ordered Crossover).
- ➢ **Mutacija** se primenjuje nasumičnom zamenom redosleda grana ili pomeranjem mesta rezova, čime se dobija nova kombinacija ruta.

Tokom više generacija algoritam poboljšava raspodelu grana i balansira dužine ruta.
Na taj način GA omogućava pronalaženje **blizu-optimalnog rešenja** čak i za složene i velike grafove, gde bi tačne metode bile vremenski neizvodljive.

Cycle-Trade algoritam

Cycle-Trade algoritam je heuristička metoda zasnovana na konceptu lokalne pretrage i razmene delova rešenja između više ciklusa.
Cilj algoritma je da poboljša postojeću podelu grafa na više ciklusa tako što će se postupnim razmenama („trade-ovima“) između ciklusa smanjiti najveća dužina rute i postići uravnoteženije opterećenje između svih ruta.

Osnovna ideja Cycle-Trade pristupa potiče iz realnog scenarija u kome više vozila (ili radnika) obavlja zadatke na različitim delovima mreže. Ako neko vozilo ima predugačku rutu, a drugo kraću, deo posla se može „razmeniti“ da bi se ukupni zadatak obavio ravnomernije.
Na sličan način algoritam pokušava da pronađe bolju raspodelu grana između ciklusa, bez potrebe za globalnom optimizacijom.

Proces rada algoritma obično se odvija kroz sledeće faze:

1. **Formiranje početnog rešenja** – kreira se početni skup ciklusa (ruta) koji zajedno pokrivaju sve grane grafa.
2. **Izračunavanje dužina** – za svaku rutu se računa ukupna dužina.
3. **Identifikacija nebalansa** – pronalazi se najduža ruta (kritična ruta) i najkraća ruta.
4. **Trgovanje (razmena)** – pokušava se prebacivanje jedne ili više grana sa duže rute na kraću, uz očuvanje povezivosti i pokrivenosti grafa.
5. **Provera poboljšanja** – ako nova raspodela dovodi do smanjenja maksimalne dužine, promena se prihvata; u suprotnom se odbacuje.
6. **Iterativno ponavljanje** – proces se nastavlja sve dok više nema poboljšanja ili dok se ne dostigne maksimalan broj iteracija.

Cycle-Trade je deterministička i lokalno orijentisana metoda: poboljšanja se postižu kroz niz malih, smislenih promena na već postojećim ciklusima. Prednost ovog pristupa je **brzina i jednostavnost implementacije**, dok mu je osnovno ograničenje to što može da zapadne u lokalni minimum i ne garantuje globalno optimalno rešenje.

Primena Cycle-Trade algoritma na Min–Max k-Chinese Postman problem

U okviru ovog projekta, Cycle-Trade algoritam je korišćen kao **heuristička metoda** za rešavanje **Min–Max k-Chinese Postman problema**, čiji je cilj da se sve grane grafa pokriju uz što ravnomerniju raspodelu dužina k ciklusa.

Na početku se formira **početno rešenje** tako što se izgradi jedna tura koja obilazi sve grane (na primer rešenje klasičnog Chinese Postman problema), a zatim se ta tura **podeli na k ciklusa**.
Svaki ciklus predstavlja rutu jednog „poštara“ odnosno jednog agenta.

Zatim Cycle-Trade algoritam prolazi kroz niz **lokalnih razmena (trade-ova)** između ciklusa s ciljem da se **smanji maksimalna dužina** među rutama.
U svakoj iteraciji:

- ➢ određuje se **najduža ruta** (ona sa najvećom ukupnom dužinom),
- ➢ traži se mogućnost da se deo njenog puta prenese u neku kraću rutu,
- ➢ proverava se da li takva razmena smanjuje maksimalnu dužinu ili poboljšava balans između ruta.

Ako dođe do poboljšanja, nova raspodela se **usvaja** i algoritam nastavlja sa narednim iteracijama.
U suprotnom, razmena se **odbacuje**, a pretraga se nastavlja na nekom drugom paru ciklusa.
Ovaj proces se ponavlja sve dok se više ne mogu ostvariti nova poboljšanja, odnosno dok se sistem ne „stabilizuje“.

Cycle-Trade algoritam u ovom kontekstu ima zadatak da **izjednači opterećenje između k ruta** i time smanji vrednost ciljne funkcije (maksimalnu dužinu rute).
Iako ne pronalazi globalno optimalno rešenje, pokazuje se kao **vrlo efikasan pristup** za veće grafove jer brzo konvergira i daje uravnotežene rezultate uz nisku računarsku složenost.

Zbog toga se Cycle-Trade često koristi i kao **inicijalna faza** za druge metode, poput genetskog algoritma, koji može dodatno poboljšati već postojeće rešenje.

Eksperimentalni Rezultati

Nismo uspeli da nađemo neke dobre rezultate iz literature, pa smo generisali naše. Najslabije radi brute-force algoritam. Iz testiranja smo zaključili da trenutna verzija radi brzo za grafove sa 6/7 čvorova, preko toga računar se muči i treba mu dosta vremena da nađe rešenje.

Primeri brute-force algoritma:

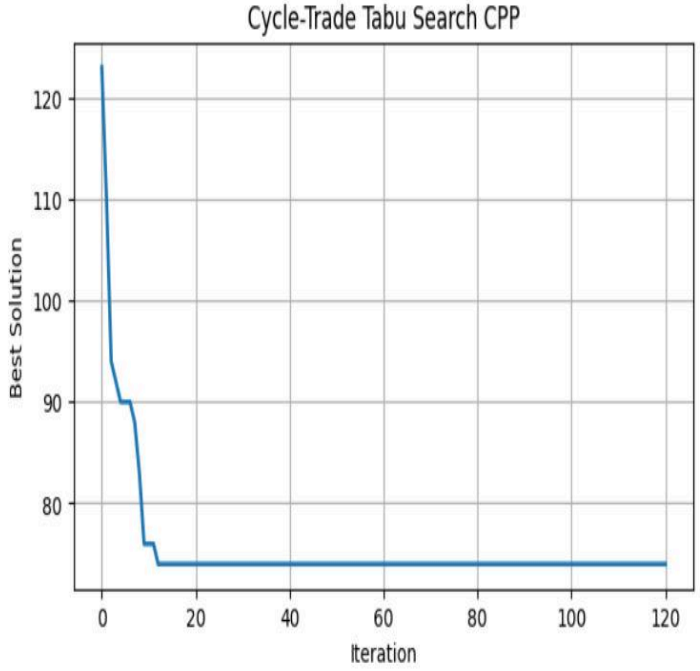
```
Graph: [(0, 1, 9), (1, 2, 8), (2, 3, 10), (3, 4, 8), (4, 5, 5), (5, 0, 1), (0, 1, 5), (0, 2, 5), (0, 4, 7), (0, 5, 6), (1, 3, 3), (1, 4, 7),
(1, 5, 10)]
Max cycle length: 56
Time: 0.20052337646484375
Postmen: 2
Nodes: 6
```

```
Graph: [(0, 1, 10), (1, 2, 4), (2, 3, 5), (3, 4, 9), (4, 5, 2), (5, 6, 1), (6, 0, 3), (0, 2, 2), (0, 3, 5), (0, 4, 6), (0, 5, 8), (1, 2, 4),
(1, 4, 9), (1, 5, 8), (2, 4, 2), (2, 5, 7), (2, 6, 9), (3, 5, 7), (4, 6, 3), (5, 6, 8)]
Max cycle length: 63
Time: 14.438583374023438
Postmen: 2
Nodes: 7
```

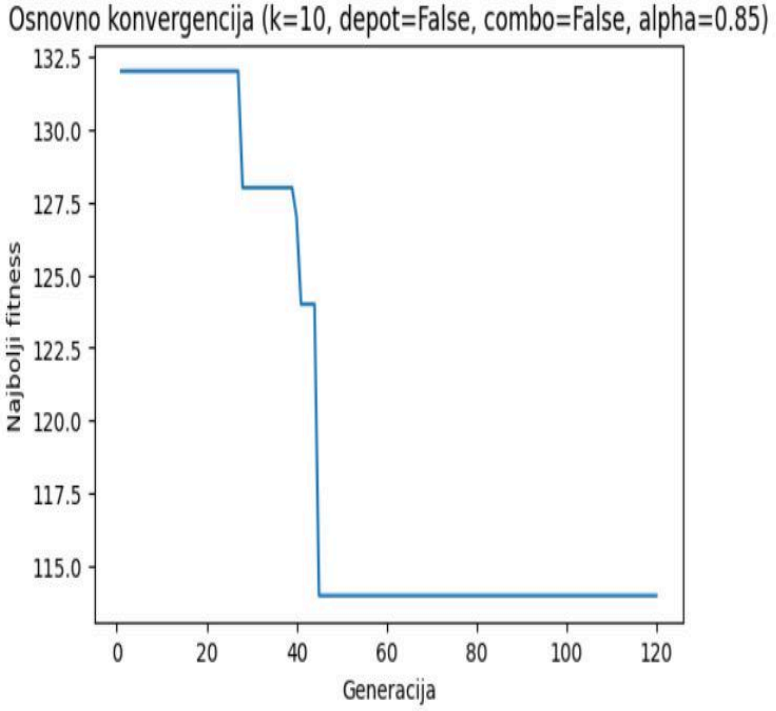
Primer cycle-trade i genetskog algoritma nad istim grafom sa 50 cvorova i 10 poštara (primer nekog kraja u gradu)

```
edges = [
    (0, 1, 7), (1, 2, 5), (2, 3, 5), (3, 4, 9), (4, 5, 2), (5, 6, 7), (6, 7, 1), (7, 8, 2), (8, 9, 6), (9, 10, 6),
    (10, 11, 10), (11, 12, 10), (12, 13, 3), (13, 14, 6), (14, 15, 2), (15, 16, 4), (16, 17, 5), (17, 18, 5),
    (18, 19, 6), (19, 20, 2), (20, 21, 10), (21, 22, 5), (22, 23, 2), (23, 24, 10), (24, 25, 7), (25, 26, 7),
    (26, 27, 1), (27, 28, 4), (28, 29, 10), (29, 30, 1), (30, 31, 6), (31, 32, 5), (32, 33, 5), (33, 34, 3),
    (34, 35, 10), (35, 36, 5), (36, 37, 5), (37, 38, 10), (38, 39, 1), (39, 40, 8), (40, 41, 6), (41, 42, 10),
    (42, 43, 4), (43, 44, 5), (44, 45, 10), (45, 46, 9), (46, 47, 2), (47, 48, 6), (48, 49, 2), (48, 37, 9),
    (3, 43, 5), (49, 1, 7), (1, 27, 5), (31, 25, 1), (8, 34, 10), (26, 7, 6), (19, 32, 8), (11, 8, 2),
    (5, 13, 3), (15, 27, 8), (18, 7, 2), (44, 35, 8), (8, 46, 2), (25, 36, 5)
]
```

```
Graph: [(0, 1, 7), (1, 2, 5), (2, 3, 5), (3, 4, 9), (4, 5, 2), (5, 6, 7), (6, 7, 1), (7, 8, 2), (8, 9, 6), (9, 10, 6), (10, 11, 10), (11, 12,
10), (12, 13, 3), (13, 14, 6), (14, 15, 2), (15, 16, 4), (16, 17, 5), (17, 18, 5), (18, 19, 6), (19, 20, 2), (20, 21, 10), (21, 22, 5), (22, 2
3, 2), (23, 24, 10), (24, 25, 7), (25, 26, 7), (26, 27, 1), (27, 28, 4), (28, 29, 10), (29, 30, 1), (30, 31, 6), (31, 32, 5), (32, 33, 5), (3
3, 34, 3), (34, 35, 10), (35, 36, 5), (36, 37, 5), (37, 38, 10), (38, 39, 1), (39, 40, 8), (40, 41, 6), (41, 42, 10), (42, 43, 4), (43, 44,
5), (44, 45, 10), (45, 46, 9), (46, 47, 2), (47, 48, 6), (48, 49, 6), (0, 49, 2), (48, 37, 9), (3, 43, 5), (49, 1, 7), (1, 27, 5), (31, 25,
1), (8, 34, 10), (26, 7, 6), (19, 32, 8), (11, 8, 2), (5, 13, 3), (15, 27, 8), (18, 7, 2), (44, 35, 8), (8, 46, 2), (25, 36, 5)]
Best solution cycle-trade: [3, 8, 8, 8, 6, 1, 7, 2, 2, 5, 8, 4, 5, 1, 5, 6, 6, 4, 9, 3, 5, 0, 2, 6, 0, 1, 6, 2, 4, 1, 5, 7, 1, 0, 4, 7, 9, 2,
7, 3, 2, 5, 8, 9, 0, 7, 4, 3, 9, 8, 6, 9, 3, 7, 2, 7, 6, 9, 8, 3, 3, 6, 1, 1, 2]
Best result: 74
Time: 28.1215 s
```




```
>>> Osnovno: k=1,2,3 bez depota (MinMax)
[k=10] fitness=114.000  lengths=114.000, 111.000, 101.000, 95.000, 102.000, 105.000, 114.000, 80.000, 107.000, 106.000  time=0.28s
```



U ovom slučaju smo dobili da cycle-trade algoritam daje bolje rešenje, međutim genetski algoritam je daleko brže stigao do rešenja. Algoritam koji bismo koristili zavisi od situacije i za koje potrebe se koristi. Ako želimo bolje rešenje i nije nam problem da sačekamo neko vreme bolji je trenutni cycle-trade algoritam. Ako želimo brže rezultate bolje je da koristimo trenutni genetski algoritam.

4.2.1 Åtvidaberg

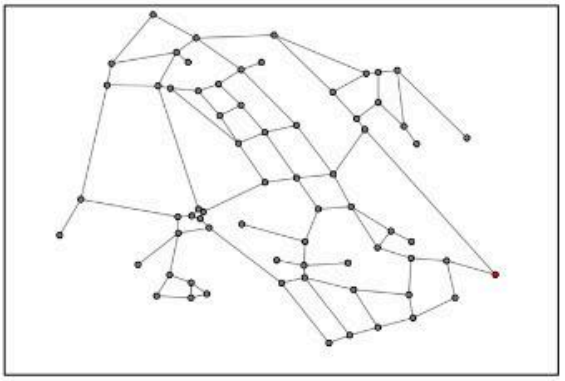
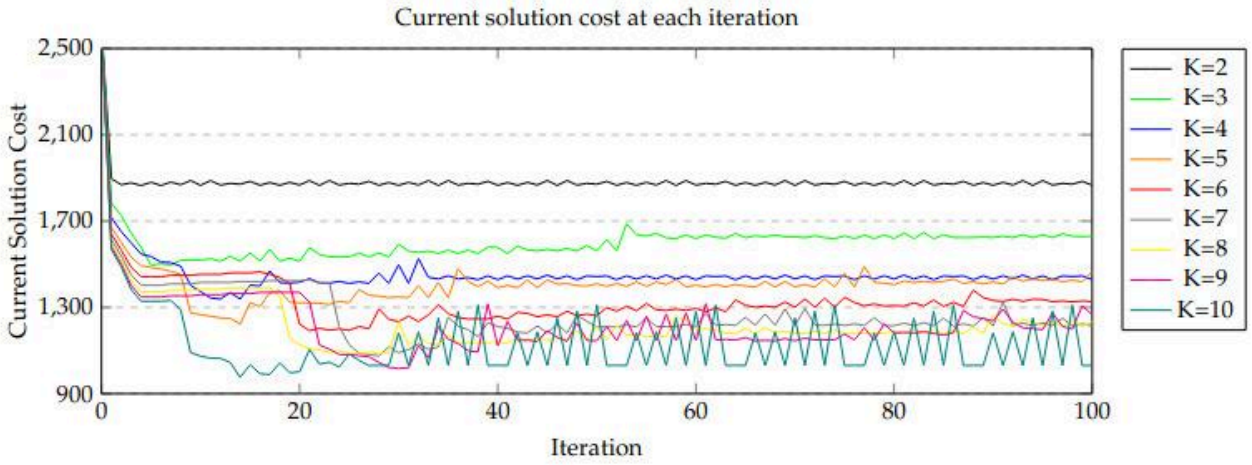


Figure 4.1: Åtvidaberg – 68 nodes, 95 edges



K	Weight	Obj. Func. Val	LowerBound	Rel. Error	Execution Time (s)
2	0.67	1865.01	1730.72	8%	12.70
3	0.75	1495.80	1298.04	15%	7.16
4	0.80	1338.44	1038.43	29%	5.48
5	0.83	1221.73	865.36	41%	4.21
6	0.86	1191.67	741.74	61%	3.91
7	0.88	1075.28	649.02	66%	2.76
8	0.89	1083.62	576.91	88%	2.84
9	0.90	1017.11	519.22	96%	2.45
10	0.91	975.38	472.02	107%	1.91

Ovo je primer iz literature gde je korišćen cycle-trade algoritam. Njihov algoritam dobija mnogo bolje rezultate od našeg, ali treba napomenuti da su verovatno koristili jače računare od nas i pisali kod na nekom drugom programskom jeziku koji podržava paralelizam.

Testiranja smo vršili na virtuelnoj mašini na Ubuntu operativnom sistemu. Mašina ima na raspolaganju 4 jezgra procesora i 8GM RAM-a. Koristili smo jupyter notebook za lako pokretanje pojedinačnih ćelija i lep prikaz rezultata. Kod je napisan u python programskom jeziku.

Zaključak

U ovom radu razmatran je Min–Max k-Chinese Postman problem, kao jedno od složenijih proširenja klasičnog problema poštara. Cilj je bio ravnomerna raspodela ukupnog posla između više poštara, odnosno minimizacija maksimalne dužine pojedinačne rute.
Implementirana su dva heuristička pristupa – Cycle-Trade algoritam i genetski algoritam (GA) – koji su pokazali različite prednosti.

Cycle-Trade algoritam je jednostavan, deterministički i efikasan u izjednačavanju dužina ruta, ali može zapasti u lokalni minimum. Genetski algoritam, s druge strane, pokazuje veću fleksibilnost i brzinu konvergencije, uz mogućnost pronalaženja boljih globalnih rešenja. Međutim, njegova tačnost i stabilnost zavise od parametara kao što su veličina populacije, stopa mutacije i broj generacija.

Eksperimentalni rezultati pokazali su da oba pristupa daju zadovoljavajuća rešenja za grafove srednje veličine, dok za veće instance problemi računarske složenosti postaju izraženiji. Cycle-Trade algoritam generalno proizvodi kvalitetnija rešenja, dok genetski algoritam omogućava brže pronalaženje približno optimalnih rezultata.

U budućem radu moguća su poboljšanja u vidu hibridnih pristupa – na primer, kombinovanje Cycle-Trade algoritma kao inicijalne faze sa genetskim algoritmom za dalju optimizaciju. Takođe, korišćenje jačih računara i

paralelnih tehnika moglo bi dodatno ubrzati proces i omogućiti primenu na realnim mrežama većih razmera.

Na osnovu sprovedenih eksperimenata, može se zaključiti da obe heuristike predstavljaju korisne i praktične metode za rešavanje Min–Max k-CPP problema, pružajući dobar kompromis između kvaliteta rešenja i vremena izvođenja.

Literatura

Literatura koju smo koristili pronađena je na Internetu putem Google Scholar pretraživača. Linkovi ka literaturi se nalaze ispod:

https://scholarsmine.mst.edu/cgi/viewcontent.cgi?article=1050&context=inspire-meetings&utm_source=

https://www.sciencedirect.com/science/article/abs/pii/S0305054805000663?.com

https://liu.diva-portal.org/smash/get/diva2%3A1275190/FULLTEXT01.pdf?com