Optimizacija problema K kineskih poštara

Seminarski rad u okviru kursa
Računarska inteligencija
Matematički fakultet

Lazar Ristić, Dimitrije Stamenić
mi16150@alas.matf.bg.ac.rs, mi16260@alas.matf.bg.ac.rs

24. septembar 2021.

Sažetak

Problem k-kineskih poštara (Minimum k-Chinese Postman Problem, k-CPP) je poznati problem kombinatorne optimizacije, izveden je iz problema kineskog poštara (The Chinese Postman Problem - CPP) koji je definisao kineski matematičar Guan još 1960. godine.

Sadržaj

1 Uvod

Problem k-Kineskih poštara adresira potrebu iz stvarnog sveta da poštanska ustanova želi da na jedan teren pošalje više od jednog poštara. Cilj je pronaći k ciklusa (ruta koje počinju i završavaju se u početnom čvoru) i minimizovati najduži od tih ciklusa. Ovakav problem je NP-težak. U ovom radu će biti predstavljen algoritam grube sile koji rešava ovaj problem, a zatim će algoritam biti optimizovan pomoću tehnike simuliranog kaljenja.

2 Pristup i implementacija

Kao poznate parametre ćemo imati neusmeren graf $G=(V, E)$, težine grana w, polazni čvor s i broj poštara k. Cilj je da se svaka grana obiđe bar jednom, ali tako da svaki poštar ima približno jednaku rutu. S obzirom da rešavamo NP-tešku varijantu problema, potrebno je minimizovati najdužu od k ruta.

Za predstavljanje grafa koristićemo rečnike *nodes* i *edges*:

- *nodes* - rečnik oblika čvor: lista suseda
- *edges* - rečnik oblika čvor: cena puta do suseda

Oba rečnika su organizovana tako da i-ti član rečnika *nodes* odgovara i-tom članu rečnika *edges*. Tačnije, cena do suseda n_i. (iz rečnika *nodes*) je e_i (iz rečnika *edges*).

2.1 Algoritam grube sile

Koraci u rešavavnju problema:

1. Pronaći sve cikluse u grafu
2. Izdvojiti samo one cikluse koji počinju i završavaju se u početnom čvoru
3. Iz takvih ciklusa izdvojiti sve osim onih koji obilaze svaki čvor
4. Izračunati cenu svakog izdvojenog ciklusa
5. Izdvojiti k putanja
6. Iz tih k putanja, minimizovati najdužu

2.1.1 Pronaći sve cikluse u grafu

Funkcija *find-all-cycles(graph, start, end)* pronalazi sve cikluse u zadatom grafu.

- *graph* - početni graf predstavljen kao rečnik
- *start* - početni čvor
- *end* - krajnji čvor

Izlaz funkcije su svi ciklusi koji postoje u grafu.

2.1.2 Izdvojiti samo one cikluse koji počinju i završavaju se u početnom čvoru i kraći su od broja čvorova

S obzirom da funkcija *find-all-cycles* vraća sve cikluse u grafu, pomoću funkcije *same-start-end(cycles)* izolujemo samo one koji počinju i završavaju se čvorom 1.
Izlaz iz funkcije je lista ruta koje počinju i završavaju se čvorom 1.

Funkcjiom *short-cycles(depot-node-cycles)* dobijamo sve cikluse osim onih koji obilaze svaki čvor u grafu. Poenta problema je da svaki poštar obilazi otprilike podjednake rute, a ne svaki poštar svaki čvor.

2.1.3 Izračunati cenu svakog izdvojenog ciklusa

Nakon toga, pomoću funkcije *calculate-paths* računamo ukupnu cenu svih tih ciklusa. Izlaz iz funkcije je lista parova (putanja, cena).
Sa ovako definisanom strukturom možemo lakše nastaviti rešavanje problema.

2.1.4 Izdvojiti k putanja

Implementiramo funkciju *k-paths(calculated-paths, k-postmen)* koja prima listu parova (putanja, cena) i broj poštara koji treba da obiđu graf. Rezultat je lista k putanja i njihovih cena, lista čvorova koje su poštari obišli (potrebno je da na ovom spisku budu svi čvorovi), kao i ukupna cena ruta k poštara.
Kada dobijemo k putanja, potrebno je da odredimo najdužu i nju minimizujemo.

2.1.5 Minimizacija najduže putanje

Konačno, pomoću funkcije *kcpp-bf(k-paths)* koja prima k putanja, određujemo najskuplju od ruta i pronalazimo novu koja je najmanja moguća. Uslovi za novu rutu:

- Cena je manja od cene najduže od k ruta
- Postoje čvorovi u novoj ruti koji do sad nisu obiđeni

Izlaz iz ove funkcije je novih k putanja, čvorovi koje su poštari obišli i nova ukupna cena svih tura.

Evidencija o ukupnoj ceni tura je vođena zbog poređenja rezultata.

2.2 Optimizacija simuliranim kaljenjem

Kod optimizacije simuliranim kaljenjem ćemo koristiti već postojeću funkciju *calculated-paths* koja vraća putanje iz čvora 1 i njihove cene. Ova funkcija će nam koristiti da generišemo prvo nasumično rešenje problema, koje ćemo nakon toga optimizovati simuliranim kaljenjem.
Pored samog algoritma simuliranog kaljenja, koristimo i sledeće funkcije:

- Funkciju *isFeasible* - koja proverava da li je novoizabrana putanja odgovarajuća. Dodavanjem ove putanje na ostale, trebalo bi da svi čvorovi budu obiđeni. Ukoliko to nije slučaj, ne prihvatamo novu putanju
- Funkciju *change* - koja menja početno rešenje tako što menja proizvoljnu putanju za koju se potom proverava da li je ispravna pomoću funkcije *isFeasible*

- Funkciju *restore* - koja vraća izmene načinjene u funkciji *change*

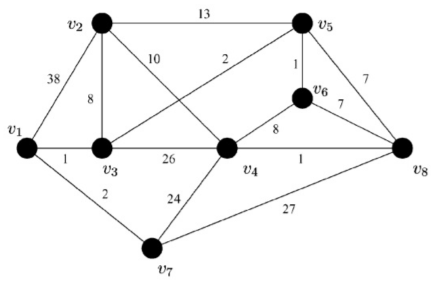
Za potrebe poređenja inicijalnih rezultata i rezultata nakon primene algoritma, čuvamo podatke o ukupnoj ceni puta, kao i sve čvorove koje su poštari obišli.

3 Eksperimentalni rezultati

Prilikom testiranja rada algoritama, postavljani su rezultati pre i posle minimazicije najduže rute, kao i ukupna cena i obiđeni čvorovi.

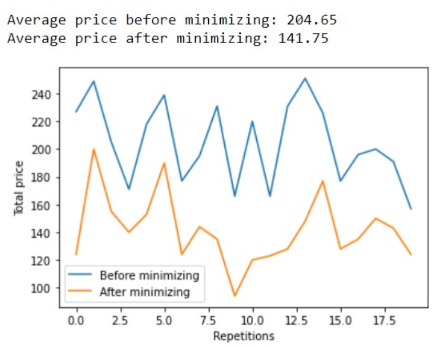
Prolazićemo redom kroz primere i prikazivati rezultate algoritma grube sile, kao i rezultate algoritma simuliranog kaljenja.

Primer 1: Graf sa 8 čvorova i 3 poštara (k=3, primer grafa preuzet iz [1])



Slika 1: Primer 1

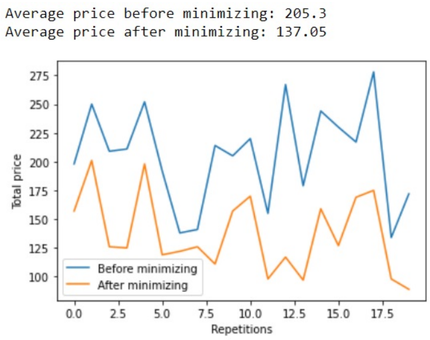
Rezultati algoritma grube sile
Nakon testiranja algoritma grube sile na grafu iz primera, rezultati su sledeći:



Slika 2: Testiranje algoritma grube sile

Zbog prirode algoritma grube sile, gotovo uvek se dolazi do najboljeg rešenja, ali ne na najbrži način. Optimizacijom algoritmom simuliranog kaljenja težimo da ovo popravimo.

Rezultati algoritma simuliranog kaljenja
Nakon testiranja algoritma simuliranog kaljenja na grafu iz primera, rezultati su sledeći:



Slika 3: Testiranje algoritma simuliranog kaljenja

Iz priloženog možemo zaključiti da se algoritmi slično ponašaju, a vreme izvršavanja nećemo porediti na ovom primeru jer je graf relativno prost.

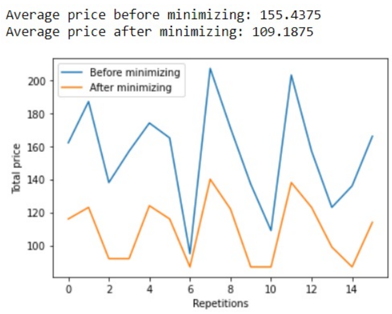
U oba slučaja, smanjivanjem najduže od putanja, ujedno je smanjena i ukupna dužina puta k poštara.

Primer 2: Isti graf, ali sa 2 poštara (k=2)

Rezultati algoritma grube sile

Slika 4: Testiranje algoritma grube sile

Rezultati algoritma simuliranog kaljenja



Slika 5: Testiranje algoritma simuliranog kaljenja

4 Zaključak

Lorem ipsum

Literatura

[1] Gerhard Reinelt Dino Ahr. A tabu search algorithm for the minmax k -Chinese postman problem. *Academia*, 2005.