Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Dokumentacija za projektni zadatak

Studentkinja: Nikolić Lenka, SV16/2023

Predmet: Nelinearno programiranje i evolutivni algoritmi

Broj projektnog zadatka: 15

Tema projektnog zadatka: Ant colony optimization algoritam, problem najkraćeg puta

Opis problema

U datoteci **data_path_nodes.txt** dat je skup dvodimenzionalnih tačaka. Za svaku tačku je navedena njena šifra, njene koordinate, kao i šifre drugih tačaka koje su susedne ovoj tački.

Primer zapisa jedne tačke u formatu $\boldsymbol{id(x,y)}: \boldsymbol{id_1, id_2, id_3}$ ...

1605763112(1552972.7826890016,5030345.496234874): 1605763129,1605763074,1605763051

Cilj je pronaći najkraći put od početne do krajnje tačke, koristeći samo susedne tačke prilikom kretanja.

Ideja je napraviti graf čiji su čvorovi date tačke i čije grane postoje između susednih tačaka. Težine grana su euklidska udaljenost dve tačke koje povezuju. Dakle, ako postoje dve tačke t_1 i t_2 sa koordinatama redom (x_1, y_1) i (x_2, y_2) koje su povezane granom, težinski faktor te grane se računa po sledećoj formuli: $d(t_1, t_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Iako može biti napravljeno da korisnik unosi početnu i krajnju tačku između kojih bismo tražili najkraći put, moguće je da tako budu izabrane dve tačke između kojih put uopšte ne postoji, zato je zadato da id početke tačke bude:3653296222, a krajnje:3653134376.

Uvod

Ant Colony Optimization (ACO) je metaheuristički algoritam inspirisan ponašanjem mrava u realnom životu. Mravi ostavljaju feromone na putevima kako bi olakšali koloniji pronalaženje najkraće putanje do izvora hrane. U algoritmu, mravi simuliraju ovo ponašanje tako što se kreću kroz graf i ostavljaju tragove feromona na putanjama koje koriste. Putevi sa više feromona postaju privlačniji za sledeće mrave, što omogućava formiranje sve kraćih i boljih putanja tokom vremena.

ACO algoritam se koristi za pronalaženje najkraće putanje između zadate dve tačke u neusmerenom težinskom grafu opisanom u prethodnom poglavlju.

Tokom svake iteracije više mrava pokušava da pronađe put od početne do krajnje tačke. Nakon svake uspešne putanje, na svim ivicama tog puta deponuje se količina feromona proporcionalna inverznoj vrednosti ukupnog troška puta prema sledećoj formuli: $\Delta\tau = \frac{Q}{L}$, gde je Q konstanta deponovanja, a L ukupna dužina puta. Na kraju iteracije, feromoni se smanjuju pomoću stope isparavanja: $\tau \leftarrow (1 - \rho) * \tau$, gde je ρ parametar koji kontroliše brzinu isparavanja.

Ove dve operacije omogućavaju da se balansira između istraživanja novih puteva i eksploatacije postojećih dobrih rešenja.

ACO pristup je pogodan za ovakve probleme jer omogućava robusno i efikasno pronalaženje optimalnih ili blizu optimalnih rešenja, čak i u grafovima sa velikim brojem čvorova i kompleksnom strukturom.

Implementacija

Rešenje je implementirano u programskom jeziku **Python** i sastoji se iz dva modula: **graph_parser.py** i **aco.py**. Prvi modul služi za obradu podataka i konstrukciju grafa, dok drugi implementira Ant Colony Optimization algoritam.

1. **Parsiranje grafa (graph_parser.py)**
    - **parse_graph(filepath)**
      Ova funkcija učitava podatke iz datoteke i gradi graf u obliku rečnika gde svaki čvor sadrži koordinate i identifikatore svojih suseda. Ovo je osnovna struktura potrebna za rad algoritma.
    - **calculate_distances(graph)**
      Računa euklidske udaljenosti između svakog čvora i njegovih suseda, i popunjava ih kao težine grana. Time graf postaje težinski.
    - **make_graph_bidirectional(graph)**
      Dodaje veze u oba smera za svaku granu, čime se obezbeđuje da je graf neusmeren, što je važno za korektno funkcionisanje algoritma.
    - **get_distance_dict(graph)**
      Priprema dodatnu strukturu koja sadrži parove čvorova i njihove međusobne udaljenosti, što se koristi za brzo pristupanje informacijama tokom rada ACO algoritma.
2. **Ant Colony Optimization (aco.py)**
    - **Ant klasa**
      Predstavlja jednog mrava koji se samostalno kreće kroz graf, beleži posećene čvorove, računa ukupni trošak puta i koristi kombinaciju feromona i heuristike kako bi odlučio kojim putem da ide.
      Ključne metode u ovoj klasi su:
        - **find_path(start, end)**
          Glavna metoda koju mrav koristi da pronađe put od početnog do krajnjeg čvora. Tokom kretanja, koristi se heuristička funkcija zasnovana na feromonima i udaljenosti do cilja. Implementirana je i tzv. "escape logika" koja pokušava da mrava izvede iz začaranog kruga ukoliko ne pravi napredak duže vreme.
        - **_greedy_selection(...)**
          Bira sledeći čvor na osnovu kombinovanih faktora: količine feromona na ivici i udaljenosti do cilja. Ova metoda favorizuje čvorove koji obećavaju brži dolazak do cilja.
        - **_aggressive_escape(...)**
          Aktivira se kada mrav predugo stagnira. Fokusira se na čvorove koji nisu skoro posećeni, čime se izbegava lokalni minimum.
    - **AntColonyOptimization klasa**
      Predstavlja koloniju mrava i orkestrira njihovu aktivnost tokom više iteracija.
      Najvažnije metode su:
        - **run(start, end)**

Pokreće zadati broj iteracija, u svakoj šalje više mrava koji traže putanju. Nakon svake iteracije, vrši se ažuriranje feromona. Čuva se najbolje rešenje kroz sve iteracije.

- **_deposit_pheromones(path, cost)**
  Na svaku ivicu puta koji je mrav prešao, dodaje količinu feromona obrnuto proporcionalnu dužini tog puta. Kraći putevi ostavljaju više feromona.
- **_evaporate_pheromones()**
  Na kraju svake iteracije, svi feromoni se umanjuju fiksnom stopom isparavanja. Time se sprečava da stari tragovi zauvek dominiraju.

3. **Povezivanje modula i izvršavanje algoritma**
   Celokupan proces funkcioniše na sledeći način:
   Pomoću **graph_parser.py** fajl **data_path_nodes.txt** se pretvara u graf sa koordinatama i povezanim tačkama. Izračunavaju se sve euklidske udaljenosti između susednih čvorova. Poziva se metoda **run(start, end)** klase **AntColonyOptimization**, koja više puta pokreće virtuelne mrave da pronađu put između početnog i krajnjeg čvora. Na osnovu informacija o feromonima i heuristici, mravi kolektivno konvergiraju ka najkraćem putu.

Zaključak

ACO algoritam uspešno rešava problem pronalaženja najkraće putanje u grafovima sa lokalnim vezama između čvorova. U ovom projektu, njegova primena na realan skup podataka pokazala je otpornost na lokalne minimume i sposobnost da iterativno pronađe efikasna rešenja.

Prednost pristupa je u njegovoj robusnosti i adaptivnosti, ali algoritam može biti spor za velike grafove ukoliko parametri nisu pažljivo podešeni. Kroz eksperimente je pokazano da odgovarajuća kombinacija heuristike, feromona i mehanizama izbegavanja ponavljanja daje vrlo dobre rezultate bez potrebe za kompleksnim podešavanjima.