Uvod

Rešenja mnogih logističkih problema mogu se naći u primeni teorije grafova. Često se nalaze na mestima gde se retko očekuju. Skoro svi problemi vezani za navigaciju i nalaženje optimalnog puta mogu se na lak način rešiti pomoću određenog grafovskog pristupa.

Mi ćemo u ovom radu diskutovati o algoritmu za pronalaženje minimalnog k povezujućeg stabla (stablo minimalne težine koje sadrži tačno k čvorova, k <= n, gde je n ukupan broj čvorova u grafu) unutar nekog neusmerenog težinskog grafa.

Ovaj problem pripada grupi NP teških problema, što znači da ne postoji algoritam koji ovaj problem rešava u složenosti manjoj od eksponencijalne. Zbog toga se primenjuju različiti heuristički pristupi, u cilju dobijanja rešenja koja su u nekoj meri bliska optimalnom, ali za znatno kraće vreme.

Nalaženje minimalnog povezujućeg stabla koje obuhvata sve čvorove grafa je izvodljivo pomoću Kruskalovog algoritma (O(E logV)). Kod daljeg smanjivanja ovog stabla na k-stablo nailazimo na problem. Nije moguće efikasno izabrati k čvorova koji će obrazvati stablo minimalne težine, već se ovde moramo upustiti u rešenja eksponencijalne složenosti, što nam svakako nije cilj.

Potencijalna rešenja ovakvog problema su, u našem iskustvu, slabo dokumentovana. Jedan od faktora koji doprinose manjku literature na ovu temu je nepostojanje standardizovanog naziva za ovaj tip problema.

Postavka problema

Problem glasi:

“Za graf G = (V, E), k <= n i težinsku f-ju w: E -> N,

pronaći k-stablo tako da je njegova težina minimalna”

Osim što je NP težak, problem je takođe APX kompletan, što znači da je APX i APX težak. APX problemi su problemi za koje postoje efikasni algoritmi koji nalaze rešenje u nekom fiksnom rasponu u odnosu na optimalno rešenje (što će se videti u odeljku o VNS-u). APX težak problem je problem za koji postoji redukcija koja ga pretvara na APX problem.

Opis rešenja

Kao osnovno merilo kvaliteta koristili smo osnovni algoritam grube sile, kao i pohlepni algoritam. Ovakvi pristupi podrazumevaju koja su jako vremenski zahtevna. Što se tiče rešenja koja koriste heuristiku, imamo dve varijante algoritma kolonije mrava, kao i VNS algoritam. Ova rešenja ćemo dalje diskutovati u njihovim odeljcima.

Gruba sila

Ovaj algoritam je po prirodi naivan, ali služi kao dobra osnova sa kojom možemo da uporedimo rešenja koja ćemo dobijati u daljem radu. Algoritam radi tako što proverava svaku mogucu kombinaciju od k čvorova iz početnog skupa od n čvorova. Kada dobijemo tražene kombinacije, u petlji ćemo za svaku napraviti podgraf koji sadrži sve čvorove u toj kombinaciji, kao i sve grane koje povezuju te čvorove. Ovim smo efektivno smanjili nas graf koji je u početku imao n čvorova, na manji graf koji ima samo k čvorova. Sada kada imamo graf od k čvorova i tražimo minimalno k povežujuće stablo, rešenje možemo dobiti već pomenutim Kruskalovim algoritmom.

Kruskalov algoritam radi tako što prvo sortira listu grana u neopadajućem poretku u odnosu na njihovu težinu. Svaki čvor se posmatra kao svoje sopstveno stablo. Prolazimo kroz sortiranu listu grana i redom proveravamo da li bi se dodavanjem te grane napravio ciklus u nasem ukupnom minimalnom povezujućem stablu, pa ako ne, ta grana se dodaje u ukupno stablo i spaja dva stabla na svojim krajevima. Ovaj postupak se ponavlja sve dok se svi cvorovi grafa ne povežu u jedno minimalno povezujuće stablo. Kruskalov algoritam garantuje da će rešenje biti stablo koje sadrži sve čvorove grafa, takvo da je njegova težina minimalna.

Na početku inicijalizujemo globalnu minimalnu težinu na beskonacno veliki broj i globalno minimalno povezujuće stablo kao praznu listu. Na kraju svake iteracije ćemo kao rezultat dobijati tekuću minimalnu težinu kao i tekuće minimalno stablo, te je dovoljno samo proveriti da li je ta tekuća minimalna težina zapravo manja od naše globalne minimalne težine i ako jeste promeniti vrednost globalne minimalne težine na tekuću kao i globalnog minimalnog stabla na tekuce. Ukoliko nam je na kraju izvršavanja vrednost globalno minimalne težine jos uvek beskonačno veliki broj, to nam je znak da algoritam nije uspeo da nađe zadovoljavajuće stablo (moguće je da je k zadato da bude veće od n). Algoritam kao konacno rešenje vraća par koji sadrži najbolje nađeno minimalno drvo i najbolja nađena minimalna težina.

Pohlepna pretraga

Ovo je još jedan naivan pristup rešavanju ovog problema koji nam neće garantovati globalno optimalno rešenje, ali u velikom broju slučajeva mu prilazi jako blizu za mnogo manje vremena.

Kao i u algoritmu grube sile, inicijalizujemo globalnu minimalnu težinu na beskonacno veliki broj i globalno minimalno povezujuće stablo kao praznu listu. Za svaki cvor u grafu radimo sledeće:

1. Definišemo listu posećenih cvorova (koja za sada sadrži samo polazni čvor), listu suseda, dodajemo sve susede naseg trenutno posmatranog čvora, definišemo tekuće stablo kao praznu listu i tekuću tezinu kao nula

2. U petlji koja ide od 0 do k-1 (jer smo polazni cvor već dodali) tražimo prvog suseda u listi suseda takvog da on nije već u listi posećenih cvorova i da grana ka njemu ima manju težinu od bilo koje druge grane koja vodi ka ostalim susedima

3. Ukoliko smo uspeli da nađemo ovakvu granu, njenu težinu dodajemo na tekuću tezinu, a par cvorova koje povezuje nađena grana dodajemo u tekuće stablo

4. Nakon petlje, proveravamo da li smo zaista našli stablo koje sadrži tačno k čvorova i ako jesmo, proveravamo da li je njegova težina manja od globalno minimalne težine, pa ako jeste cuvamo je kao novu globalno minimalnu težinu i stablo kao globalno minimalno stablo

Ovaj algoritam, ako uspe da nađe minimalno k stablo, daje rešenje u vidu para najbolje minimalno stablo i najbolja minimalna težina. U suprotnom će biti ispisana adekvatna poruka obaveštenja da nije moguće naći stablo tražene veličine.

Kolonija mrava iz svakog čvora

Ovaj algoritam (KMSČ) pokušava da obuhvati najbolje moguće rešenje tako što postavlja koloniju mrava u m čvorova tokom iteracije. Tokom jedne iteracije izvršava se m kolonija mrava iz različitih čvorova, pa se pre sledeće iteracije feromoni postavljaju na nulu.

Postoje određene prednosti i mane u menjaju broja kolonija po iteraciji; što je manje m, to je algoritam brži i nepreciznji, zbog smanjenog obima pretrage, dok veće m daje približnije rešenje po cenu utrošenog vremena.

Najveća mana algoritma je vreme izvršavanja (ako se zahteva veoma približno rešenje). Izvršavanje algoritma zahteva značajno vreme i ne donosi poboljšanja u odnosu na grubu silu. Takođe, kvalitet rezultata zavisi od postavke broja kolonija po iteraciji, kao i odnosa između broja iteracija i broja kolonija po iteraciji. Taj izbor pada na korisnika, koji onda mora uložiti vreme u pronalaženje adekvatnog odnosa za određen problem koji želi da reši.

Radi jednostavnosti, u radu je prikazana samo verzija algoritma koja pravi koloniju iz svakog čvora.

Slobodna kolonija mrava

Slobodna kolonija mrava (SKM) pristupa problemu na drugačiji način od prethodnog algoritma. Umesto uvođenja m kolonija u svaku iteraciju, algoritam bira nasumične čvorove za broj mrava koji će učestvovati u iteraciji.

Svaki mrav se kreće iz nasumičnog čvora i pravi svoje stablo. Po završetku kretanja svih mrava i usklađivanju feromona za sledeću iteraciju, bira se najbolje mravlje stablo (minimalno stablo iz grupe stabala pojedinačnih mrava). To minimalno stablo se onda upoređuje sa globalnim minimalnim stablom.

Algoritam ima tendenciju ka konvergenciji, pa je potrebno uvesti zaštitu od iste. To se postiže računanjem faktora konvergencije i vraćanjem feromona na grafu na početne u slučaju visokog faktora. Nedostatak ove zaštite dovodi do preference algoritma ka dobrim lokalnim rešenjima.

Slobodna kolonija mrava rešava nedostatke prethodnog algoritma. SKM je brži i jednostavniji za korišćenje; broj mrava može se vezati za odnos broja čvorova i broja iteracija, dok izbor konstanti u tom odnosu daje predvidiv kvalitet rešenja.

```
num_ants = int(n/2*(num_iters))
```

Primer formule za broj mrava

VNS algoritam

Ovaj algoritam predstavlja stohastički pristup rešavanju postavljenog problema. Za početak, generišemo nasumično k povezujuće stablo unutar našeg grafa, koje će da služi kao početno rešenje koje ćemo da pokušavamo da unapredimo sve dok nam ne istekne unapred odobreno vreme.

Pokušavaćemo da unapredimo rešenje ponovljenim izvršavanjem sledećih operacija:

1. Napravimo potpuno novo nasumično rešenje, na koje staro rešenje nije uticalo
2. Sve dok nam promene daju bolje rešenje radimo sledeće:
   a. Izbacujemo nasumičan list čvor iz stabla
   b. Dodajemo novi nasumičan list čvor umesto onog kog smo izbacili
   c. Ukoliko je novo rešenje bolje od rešenja iz 1, pamtimo samo novo rešenje, u suprotnom zaboravljamo novo rešenje
3. Ukoliko je rešenje dobijeno u ovoj iteraciji bolje od našeg početnog rešenje koje smo generisali pre petlje, čuvamo rešenje dobijeno u petlji, a u suprotnom ga zaboravljamo

Algoritam konstantno ima jedno potencijalno rešenje, koje u svakom trenutku moze da vrati kao najbolje pronađeno u slučaju isteka vremena. Pored ovog rešenja, uvek imamo i jedno dodatno rešenje na kom vršimo nasumične promene u nadi da će one rezultovati boljim rešenjem od trenutnog kandidat. Dobra strana celog ovog postupka je to sto ovaj algoritam u svakom trenutku moze da vrati svoje trenutno najbolje rešenje.

Rezultati i poređenja efikasnosti

Radi poređenja, generisani su, gusto povezani, grafovi sa po 25, 50, 100, 300 i 500 čvorova. Metrike koje su korišćene izabrane su proizvoljno i prate dosadašnje iskustvo i teoriju grafova.

Veličina minimalnog stabla (k) varira između 1/6 i 1/2 broja čvorova. Iz iskustva se taj raspon odnosa pokazao kao dovoljno dobar za testiranje algoritama. Druge postavke uključuju i postavku broja slobodnih mrava i vremensku granicu za VNS.

Za svaki od generisanih grafova pokreće se algoritam sa predefinisanim metrikama. U svaki algoritam uključeno je merenje vremena i ono se prikazuje posle rezultata. Nakon svakog algoritma, pravi se slika grafa. Pošto se vns pokreće više puta za svaki graf, njegova sekcija je naglašena niskom na početku i kraju.

Algoritmi se izvršavaju od najsporijeg do najbržeg, s time da se sporiji algoritmi ne izvršavaju nad većim grafovima.

Gruba sila daje predvidive rezultate i ne koristi se posle najmanjeg grafa. Ona je tokom testiranja davala optimalno rešenje, kom ostali algoritmi treba da se približavaju, kao i vreme trajanja, koje ostali algoritmi treba značajno da smanje. Mravi iz svakog čvora se takođe ne koriste posle grafa sa 25 čvorova. Razlog je isti, algoritam je spor i ne doprinosi cilju nalaženja algoritma koji daje najbolju kombinaciju približno optimalnog rešenja za najmanje utrošeno vreme.

Pohlepna pretraga se pokazuje kao jako brz i poprilično korektan algoritam nad manjim grafovima. Iz manjih grafova moglo bi se pogrešno

zaključiti da je pohlepna pretraga pouzdanija i brža od slobodne kolonije mrava. Veći grafovi obrću sliku o ovim algoritmima, zato što se povećanjem grafa povećava verovatnoća da će da dođe do neoptimalnog pohlepnog izbora čvora. Drugim rečima, ako imamo graf sa velikim brojem čvorova, postoji velika verovatnoća da će traženo k biti dosta manje od n. U ovom slučaju, n-k čvorova se neće naći u finalnom rezultatu, pa je svaki pokušaj pretrage iz tih čvorova rezultovao isključivo trošenjem vremena. Naravno, isti problem postoji i kod malih grafova, ali u tom slučaju, kada je veća verovatnoća da će razlika između n i k biti manja, ovaj problem je manje evidentan.

Za razliku, slobodna kolonija mrava nema takve nedostatke, a po brzini kreće da premašuje pohlepnu pretragu zbog modularnog nameštanja broja mrava po iteraciji. Ovakvo smanjenje broja mrava zanemarljivo utiče na optimalnost rešenja. Algoritam je dosta brži u odnosu na ostale.

Za kraj imamo VNS algoritam. Ovaj pristup ima potpuno drugačiji princip rada u odnosu na ostale, u smislu da se u ovom algoritmu ne čeka neko rešenje koje on traži sve dok ga ne pronađe, već u svakom trenutku imamo neko rešenje koje može biti vraćeno. Uz dovoljno vremena, što moze biti jako dugo, u teoriji ovaj algoritam uvek može da vrati globalno optimalno rešenje, ali to nije ni malo realan slučaj na bilo kom malo većem grafu. Dakle, ovom algoritmu mi zadajemo vremensku granicu, tj. unapred mu određujemo koliko dugo mu je dozvoljeno da traži rešenje, nakon čega mora da vrati svoje trenutno najbolje, ma kakvo ono bilo. Evidentno je da ovaj algoritam ne možemo porediti sa ostalima po vremenu potrebnom za izvršavanje, već moramo smisliti neki manje konkretan način poređenja. Odlučili smo da pokrećemo ovaj algoritam više puta, sa različitim vremenskim ograničenjem, na istom grafu kako bi videli kako se ponaša u odnosu na ostale algoritme. Zaključak ovakvog testiranja je da ovaj algoritam, na velikim i kompleksnim grafovima, za veoma kratko vreme, generiše poprilično dobra rešenja. Ta rešenja umeju da budu i do duplo gora (mada prate teorijska shvatanja APX teških problema) od rešenja dobijenih slobodnim mravima, ali ako uzmemo u obzir da slobodni mravi, u proseku, vraćaju rešenje koje je jako blisko globalnom optimumu, malo smanjenje optimalnosti rešenja, u pravim uslovima je niska cena za ovoliki dobitak na brzini izvršavanja.

Konkretno, posmatrajmo najveći i najkompleksniji graf koji smo koristili u testiranju, tj. graf sa 500 čvorova i oko 87 000 grana, gde svaka grana moze da ima težinu između 1 i 1000 i traženom veličinom k postavljenom na 100. Ako tražimo stablo od 100 čvorova, znamo da to stablo mora imati 99

grana, koje u grafu u proseku imaju težinu 500. Drugim rečima, kada bi generisali nasumično k stablo u ovakvom grafu, očekivali bi da mu tezina u proseku bude oko 49500.

Pohlepni algoritam i algoritam grube sile nisu bili u stanju da u razumnom vremenu daju rešenje, sto je i bilo za očekivati obzirom na njihovu kompleksnost i obim zadatkog problema. Sa druge strane, slobodni mravi su za 301 sekundu vratili rešenje sa težinom 213, što nije ni malo loš rezltat. U proseku, nalazi k stablo takvo da svaka grana u njemu ima težinu 2.15, jako dobar rezultat. Ako pogledamo VNS algoritam, na istom ovom problemu, za vremenske granice izmedju četvrtine sekunde i 5 sekundi, on vraća stabla tezina od 381 do 313, tj tezina grane u proseku varira od 3.84 do 3.16. Očigledno je da je optimalnost rešenja dosta ispaštala, ali ako pogledamo potrošeno vreme, ovaj algoritam se pokazuje dosta dobro.

Dakle, snaga slobodnih mrava je u blizini rešenja globalnom optimumu, a snaga VNS algoritma je u tome sto može da vrati jako dobro rešenje za frakciju vremena.

Zaključak

Problem nalaženja minimalnog povezujućeg k stabla nalazi širok spektar rešenja u domenu kolonija mrava. Eksperimentacijom dobijena su i zadovoljavajuća i nezadovoljavajuća rešenja.

Primer nezadovoljavajućeg rešenja bio bi kolonija mrava iz svakog čvora zbog prevelike koncentracije na optimalnost rešenja po cenu utrošenog vremena. Mana kolonije mrava nad ovim problemom je u osetljivosti na poziciju kolonije, što ovo rešenje zamenjuje povećanom složenošću. Ovakav pristup ne doprinosi ništa značajno u rešavanju problema.

Sa druge strane, adaptacija kolonije mrava u slobodnu koloniju, je primer zadovoljavajućeg rešenja. Originalna mana pozicije kolonije mrava se lako prevazilazi “oslobađanjem“ mrava, tj. nasumičnim odabirom početne tačke. Još jedna vrlina algoritma je mogućnost nameštanja broja mrava radi ubrzanja ili optimalnijeg rešenja.

VNS drugačije pristupa problemu, pa se time i njegova efikasnost drugačije vrednuje. Iako odstupa od optimalnog rešenja, VNS se može uspešno koristiti kao “bolje išta nego ništa“ rešenje. Preporučuje se u situacijama u kojima je problem previše velik da bi se rešio sa nekom optimalnošću i dobrim vremenom ili je zahtev takav da postoji striktno vremensko ograničenje.

Svi pristupi rešavanja problema pozitivno su uticali na razumevanje problema i mehanizama kolonije mrava. Postignuto je rešenje koje daje skoro (uvek) optimalno rešenje (slobodna kolonija), ali i rešenje koje za kratko vreme daje dobar rezultat (vns). Upoređivanjem algoritama pod različitim uslovima (veličinama grafa, broju čvorova u stablu) dobijena je dobra slika njihovog odnosa.

Iz svega gorenavedenog može se zaključiti da su varijacija kolonije mrava i VNS dobri algoritmi za rešavanje problema nalaženja minimalnog razapinjućeg k stabla. U ovom radu je to uspešno izvedeno menjanjem kolonije mrava u “slobodnu koloniju“ i primenjivanjem adaptiranog VNS-a.

Dalji rad na problemu bilo bi fokusiranje na određen algoritam i/ili širenje situacija u kojima se on testira. Iako je to u ovom radu donekle odrađeno, nepozanto je kako se algoritmi ponašaju u specifičnim situacijama, već je to ostavljeno grupama koje interesuje određen slučaj. Preporuka je da se uz proširenje prostora testiranja i koncentrisanje na domene od interesa takođe poveća kvantitet podataka radi kvalitetnijeg empirijskog istraživanja.

Rad napisali Bojan Veličković i Luka Stanković.

IX 2024.