Simulirano kaljenje i maksimizacija uticaja

Stevan Stojanović, Aleksandar Ćurković

Matematički fakultet u Beogradu

2. jul 2017.

Sadržaj

Uvod

- Jedna od najbitnijih funkcija socijalnih mreža je da prošire informacije od jednog korisnika do drugog
- Problem je pronaći mali podskup korisnika koji mogu da utiču na najveći broj preostalih korisnika mreže
- Formalno, ovaj problem se zove **maksimizacija uticaja**
- Postojeći metodi rešavanja ovog problema uglavnom su bili bazirani na pohlepnom pristupu i njegovoj optimizaciji

Uvod

- U ovom radu predložen je nov pristup ovom problemu: **Simulirano kaljenje**
- Da bi dodatno doprineli efikasnosti rešenja u algoritam su uvedena sledeća unapređenja:
  - EDV (Expected Diffusion Value)
  - SH (Spreading Heuristic)
- Empirijski rezultati pokazuju da algoritam baziran na SA daje bolje rezultate od najboljih pohlepnih algoritama

Nezavisni kaskadni model

- U ovom modelu stanje svakog čvora može biti aktivno ili neaktivno
- Aktivni čvorovi mogu da utiču na neaktivne (nije moguća promena stanja sa neaktivnog na aktivno)
- Model sadrži parametar **verovatnoća propagacije** koji modeluje tendenciju čvora da bude aktiviran od strane svojih suseda

Simulirano kaljenje

- Simulira kaljenje metala i optimizuje rešenja NP-teških problema

Opis rada

- Kreira inicijalno rešenje *i* i inicijalnu temperaturu i računa ocenu tog rešenja, označenu sa *f(i)*
- Nalazi susedno rešenje *j* i njegovu ocenu *f(j)*
- Ako je ocena susednog rešenja veća, ono će zameniti inicijalno rešenje
- U suprotnom, susedno rešenje će zameniti aktuelno sa verovatnoćom $\exp(\frac{-\Delta f}{T})$, gde je T trenutna temperatura sistema
- Temperatura sistema se postepeno smanjuje dok ne padne do unapred određene granice

Primena simuliranog kaljenja na problem maksimizacije uticaja

- Ocenu nekog podskupa datog grafa predstavlja broj čvorova na koje ima uticaj
- Nasumično kreiramo inicijalni podskup čvorova A i u svakoj iteraciji skup A' dobijamo tako što jedan čvor iz A zamenimo sa čvorom koji mu ne pripada
- Ako je ocena novog podskupa bolja, menjamo ga sa početnim podskupom, u suprotnom do zamene dolazi ako je $\exp(\frac{\Delta f}{T}) > \text{random}[0, 1]$

Očekivana vrednost difuzije

- Umesto da za ocenu k-podskupa čvorova koristimo simulacije koje su računski skupe, koristićemo procenu ocene na sledeći način:
  - Definišemo skup NB(A) koji predstavlja skup svih čvorova susednih čvorovima iz skupa A
  - Tada ocenu skupa možemo proceniti sledećim izrazom:

$$k + \Sigma_{v \in NB(A)-A}(1 - (1 - p)^{r(v)}$$

  - Verovatnoća da čvor iz NB(A) - A neće biti aktiviran skupom A je $(1 - p)^{r(v)}$ odn. da hoće: $1 - (1 - p)^{r(v)}$
  - k čvorova skupa A su već aktivirani
- Kada čvor w1 iz skupa A menjamo čvorom w2 iz NB(A) - A, svakom čvoru povezanom sa w1 smanjujemo vrednost r(v) za 1, a svakom povezanom sa w2 povećavamo

Ubrzavanje konvergencije

- Do sad prikazani algoritam nasumično bira čvor kojim će zameniti čvor iz skupa A
- Posmatranjem rezultata dolazi se do zaključka da je verovatnoća da top k čvorova budu na vrlo malom rastojanju zanemarljivo mala
- Vođeni time, kada se bira čvor za kreiranje skupa A' zanemaruju se čvorovi čije je najkraće rastojanje od čvorova u skupu A manje od nekog praga d

Ubrzavanje konvergencije

- Za svaki čvor izračunamo njegov uticaj
- Opet nasumično biramo čvor za kreiranje novog skupa ali ovaj put ne razmatramo čvorove na rastojanju manjem od d i uzimamo u obzir uticaj samog čvora na sledeći način:
  - kreiramo listu u kojoj svakom elementu odgovara zbir uticaja svih prethodnih čvorova i čvora na i-toj poziciji
  - nasumično izaberemo verovatnoću a zatim tražimo dva elementa u listi između kojih se nalazi
  - nasumičnost povećavamo time što zahtevamo da čvor ne bude na nekoj prethodno izabranoj nasumičnoj poziciji

Neke napomene vezane za implementaciju

- Pseudokod naveden u radu implementiran je u programskom jeziku python
- Graf je predstavljen u vidu matrica, gde vrste odgovaraju čvorovima, odn. njihovim vezama ka drugim čvorovima
- Razlika u odnosu na pseudokod: inicijalizacija nekih promenljivih koje se koriste u funkcijama vrši se van njih kako bi se izbegao prolazak kroz ceo graf pri svakom pozivu funkcija

Rezultati

Table: Vreme izvršavanja za svaki algoritam (u sekundama) za konstante parametre

k	SA	SASH	SAEDV	MSA
10	222.401	286.567	0.236	17.038
20	709.067	882.013	0.435	38.883
30	1512.045	1753.385	0.687	62.217

Table: Najveći ostvareni uticaj k-podskupa čvorova

k	SA	SASH	SAEDV	MSA
10	21	24	22	23
20	43	45	41	45
30	61	65	60	64

Komentar

- Iz navedenih rezultata može se izdvojiti nekoliko zaključaka:
  - Algoritam SAEDV radi mnogo brže od ostalih, pogotovo od SA i SASH
  - SASH i MSA daju najbolje rezultate
  - Malo veći uticaj koji se ostvaruje SASH algoritmom, u odnosu na MSA, pada u senku kada se u obzir uzme vreme izvršavanja koje je mnogo veće