Michael Ovelgonne and Andreas Geyer-Schulz on Claster Cores and Modularity Maximization

Petar Tešić i Andrija Urošević

Formulacija problema

Neka je $G = (V, E)$ neusmereni graf, bez petlji i neka je $\mathcal{C} = \{C_1, \ldots, C_p\}$ nepreklapajuće klasterovanje, tj. klasterovanje čvorova grafa u grupe C_i tako da $\forall i, j : i \neq j \implies C_i \cap C_j = \emptyset$ i $\cup_i C_i = V$. Neka je M matrica susedstva grafa G (važi $m_{ij} = m_{ji} = 1$ ako $(v_i, v_j) \in E$, inače $m_{ij} = m_{ji} = 0$).

Definišemo odnos broja grana koje povezuju klastere C_i i C_j i ukupnog broja grana kao

$$e_{ij} = \frac{\sum_{v_x \in C_i} \sum_{v_y \in C_j} m_{xy}}{\sum_{v_x \in V} \sum_{v_y \in V} m_x y}.$$

Odnos broja grana u klasteru C_i i ukupnog broja grana je e_{ii}. Odnos stepen klastera C_i (broj grana koje su incidentne sa bar jedanim čvorom iz klastera C_i) i ukupnog broja grana je dat sa

$$a_i = \sum_j e_{ij}.$$

Neka je $\mathcal{G}$ skup grafova sa istim čvorovima kao i graf G, pri čemu je stepen svakog čvora isti kao i u grafu G. Verovatnoća da se nasumična grana (v_s, v_e) grafa iz $\mathcal{G}$ nađe u C_i je data sa

$$P((v_s \in C_i) \wedge (v_e \in C_i)) = P(v_s \in C_i) P(v_e \in C_i) = a_i a_i = a_i^2.$$

Mera modularnosti nenasumičnosti klasterovanja može biti data kao

$$Q = \sum_i e_{ii} - a_i^2,$$

čiji je optimizacioni problem

$$\max_{\mathcal{C}} Q.$$

Ovaj problem je NP-težak.

Drugi radovi

- Agarwal and Kempe: ILP (engl. Integer Linear Programming) formulacija
- Brendes et al.: ILP formulacija
- Agarwal and Kempe: LP (engl. Linear Programming) formulacija
  - Korisna za pronalaženje gornjih granica.
- Metaheuristike: Extremal optimization, simulated annealing, mean field annealing, tabu search, ground-state energy of spin system, stectral algorithms.
- Sakupljajuće klasterovanje: Počinjemo od pojedinačnih klastera (svaki klaster sadriži po jedan čvor). Iterativno spajamo po dva klastera sve dok ne ostane samo jedan klaster (koji sadriži sve čvorove). Ovim postupkom nasteje dendrogram (dijagram predstavlja drvo, gde svaki čvor predstavlja po jedno spajanje, a listovi pojedinačne elemente). Iz dendrograma biramo klasterovanje koje ima najveću modularnost.
  - Newman: U svakoj iteraciji spajamo dva klastera koja najviše utiču na prmenu mere modularnosti.
  - Clauset et al.: Koriste Newman-ovu ideju ali je implementiraju na optimizovaniji način (PG (engl. Plain Greedy) algoritam).
  - Schuetz and Caflisch: MSG algritam (engl. MultiStep Greedy)
    - Mnogi parovi klaster su ekvivalentni u odnosu na promenu modularnosti pri njihovom spajanju.
    - U svakoj iteraciji spajamo sve parove klastera čije je promena modularnosti u l najvećih.
    - Teško je odrediti parametar l.
  - Zhu et al.: MOME algoritam (TODO: Ne znam šta su coarsened graphs???)
  - Blondel et al.: MSG algoritam BGLL koji ima dve faze u svakoj iteraciji:
    1. Prolazi kroz čvorove grafa gde svaki čvor pomera u susedni klaster tako da se poboljša modularnost sve dok ne dostignem lokalni optimum.
    2. Svaki klaster se transformiše u jedan čvor sa odgovarajućim težinskim granama.

RG algoritam

Gramizivi algoritam uvek prati smer najvećeg pozitivnog gradijenta. Spajanje dva klaster C_i i C_j rezultuje u promeni modularnosit za

$$\Delta Q(i,j) = e_{ij} + e_{ji} - 2a_i a_j = 2(e_{ij} - a_i a_j).$$

Povezivanjem dva nepovezana klaster C_i i C_j dobijamo negativnu promenu modularnosti, tj. $\Delta Q(i,j) < 0$.

Ideja za poboljšanje PG algoritma: 1. Kretanje najstrmijim gradijentom ne

garantuje pronalaženje dobrog rešenja. 2. Mnoga spajanja klastera imaju iste promene modularnosti.

RG algoritam pokušava da reši ove probleme: Interativno za k nasumičnih klastera pretražuje njihove susede i bira one sa maksimalnom promenom modularnosit ΔQ.

Gramzivo poboljšanje predstavlja postprocesiranje u kome se menja najbolje klasterovanje (ono sa najvećom merom modularnosti) iz dendrogram. Pomeranjem čvorova u susedne klastere ako rezultuje u pozitivnoj promeni modularnosit ΔQ. Postupak ponavljamo sve dok ima promene.

Problem lokalnosti

Promena modularnosti ΔQ određuje koja dva klastera spojiti u odnosu na to kako su njihovi susedi već povezani.

Postupak spajanja je balansiran ako svi klasteri rastu podjednako, tj. veličina klastera nije korelisana sa verovatnoćom da klaster bude spojen u sledećem koraku. Maksimalnu nebalansiranost dobijamo kada jedan klaster raste, dok svi ostali ostaju pojedinačni.

Posledica nebalansiranog postupka spajanja je mreža gde su neki delovi mnogo više povezani od drugih, što direktno utiče na lokalnu pretragu gramzivog algoritma. Npr, svi susedi v_2, v_3, v_4, v_5 čvora v_1 imaju isti stepen, te je zbog toga $\Delta Q(v_1, v_j)$ isto za svako $j = 2, 3, 4, 5$. Ako su v_2 i v_3 već spojeni onda $\Delta Q(\{v_1\}, \{v_2, v_3\}) > \Delta Q(\{v_1\}, \{v_4\}) = \Delta Q(\{v_1\}, \{v_5\})$. Međutim, ako su v_4 i v_5 već spojeni, onda $\Delta Q(\{v_1\}, \{v_4, v_5\}) > \Delta Q(\{v_1\}, \{v_2\}) = \Delta Q(\{v_1\}, \{v_3\})$.

RG+ algoritam

Identifikovanje ključnih tipova. Svi čvorovi koji su povezani samo čvorovima istog klastera su unutrašnji čvorovi u datom kasteru. Svi čvorovi koji su povezani sa bar jednim čvorom drugog klastera su granični čvorovu u datom klasteru.

Pretpostavka: Neki čvorovi će pripadati istoj grupi u svim mogućim klasterovanjima grafa. Identifikacija takvih čvorova i građenje odgovarajućih grupa dovodi do boljeg klasterovanja grafa.

Neka je $\mathcal{P}^* = \{C^{*1}, \ldots, C^{*x}\}$ set svih klasterovanja gde je $Q(C^{*t})$ lokalni optimum. Tražimo par čvorova v_i, v_j koji su deo nekog klastera za sva lokalno optimalna klasterovanja, tj. $\forall_x : v_i \in C_k^{*t} \implies v_j \in C_k^{*t}$. Algoritam delimo u dve faze: 1. Određivanje ključnih grupa: Naravno nije moguće izračunati P^*, jer bi tada znali klasterovanje sa maksimalnom modularnošću. Zbog toga, koristimo mali uzorak od z klasterovanja sa visokom modularnošću. Njih je moguće dobiti pokretanjem RG algoritma (za svako inicijalno klasterovanje po jednom). Od inicijalnih klasterovanja, pravimo ključne grupe, tako što uzmemo

prvo klasterovanje kao trenutni skup ključnih grupa. Iterativno delimo trenutni skup ključnih grupa pomoću novog klasterovanja. Neka je $g_{v_x,t}$ ključna grupa čvora v_x u trenutku t i $C_t(v_x)$ klaster čvora v_x u klasterovanju C_t. Čvorovi v_x i v_y su u istorj trenutnoj ključnoj grupi nakon t razdvajanja akko $g_{v_x,t-1} = g_{v_y,t-1} \wedge C_t(v_x) = C_t(v_y)$. 2. Klasterovanje ključnih grupa: Pokrenemo RG algoritam gde su ključne grupe početno klasterovanje.